Programiranje prilagođeno potrošaču

Škvorc, Dejan

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Dejan Škvorc

PROGRAMIRANJE PRILAGOĐENO POTROŠAČU

DOKTORSKA DISERTACIJA

Zagreb, 2010.

Doktorska disertacija je izrađena u Zavodu za elektroniku,
mikroelektroniku, računalne i inteligentne sustave
Fakulteta elektrotehnike i računarstva
Sveučilišta u Zagrebu

Mentor: Prof.dr.sc. Siniša Srbljić

Doktorska disertacija ima 347 stranica

Disertacija br.: _____

Povjerenstvo za ocjenu doktorske disertacije:

1. Dr.sc. Nikola Bogunović, redoviti profesor
   Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva

2. Dr.sc. Siniša Srbljić, redoviti profesor
   Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva

3. Dr.sc. Nikola Hadjina, naslovni izvanredni profesor
   Zavod za ispitivanje kvalitete, Zagreb

Povjerenstvo za obranu doktorske disertacije:

1. Dr.sc. Nikola Bogunović, redoviti profesor
   Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva

2. Dr.sc. Siniša Srbljić, redoviti profesor
   Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva

3. Dr.sc. Nikola Hadjina, naslovni izvanredni profesor
   Zavod za ispitivanje kvalitete, Zagreb

4. Dr.sc. Goran Martinović, izvanredni profesor
   Sveučilište u Osijeku, Elektrotehnički fakultet Osijek

5. Dr.sc. Zoran Kalafatić, izvanredni profesor
   Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva

Datum obrane disertacije: 08. srpnja 2010. godine

Ovaj rad posvećujem svom djedu Dragutinu Pristavu i baki Mariji Škvorc koji su, na sebi svojstven i meni često nedokučiv način, proživljavali i pratili me na putu prema doktoratu znanosti, a svojim me bezgraničnim ponosom i vjerom u moja nastojanja zadužili da na svojim budućim životnim putevima promičem plemenitost, poštenje i vjeru u ljude.

Zahvala

Kada je američki astronaut Neil Armstrong prvi put stupio na tlo Mjeseca, izgovorio je da je to maleni korak za njega, ali veliki za čovječanstvo. Iako je malo ljudi u životu imalo priliku učiniti maleni korak poput Neilovog koji je tako veliki za čovječanstvo, svi mi često radimo korake koji su mali za čovječanstvo, ali veliki za nas i ljude oko nas. Kada me netko upita zašto sam se odlučio za stjecanje doktorata znanosti, šaleći se odgovaram da je razlog moja djevojka Maja, doktorica medicine, zbog koje sam i ja odlučio postati doktor kako bismo u svemu u životu bili ravnopravni. Prava je istina, međutim, to što sam znao da uz sebe imam ljude u koje se mogu pouzdati kada zakoračim taj, za mene dosad možda i najveći korak. Stranica pisane riječi premalo je prostora da bih zahvalio svima koji su u taj moj korak ugradili dio sebe.

Zahvaljujem prof.dr.sc. Siniši Srbljiću na mentorstvu i stručnom vodstvu tijekom studija te nesebičnoj pomoći tijekom izrade doktorske disertacije. Zahvaljujem mu na stvaranju poticajne radne sredine i prenošenju znanja, iskustava i kontakata iz njegove bogate međunarodne karijere, čime je ovaj rad prepoznat i u inozemnim znanstvenim krugovima.

Zahvalu upućujem svim profesorima, znanstvenicima i suradnicima na čelu s akademikom Leom Budinom čijim je zalaganjem pokrenut tehnologijski projekt *CroGrid*, čime mi je omogućeno zapošljavanje i istraživački rad na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu, te istraživački projekti *Raspodijeljeni ugrađeni računalni sustavi*, *Računalne okoline za sveprisutne raspodijeljene sustave*, *Middleware Architecture in New Generation Networks*, *End-User Tool for Gadget Composition* i *Unified Translation Memory* u okviru kojih su obavljena istraživanja vezana uz izradu doktorske disertacije. Tvrtkama Google Inc., Mountain View, Kalifornija, SAD i Ericsson Nikola Tesla d.d. Zagreb iskazujem zahvalnost na uspješnoj suradnji i financijskoj potpori istraživačkom radu u okviru kojega je izrađena doktorska disertacija. Posebna zahvala Borisu Debiću, dipl.ing. na promicanju istraživanja i pomoći tijekom uspostave kontakata s tvrtkom Google Inc.

Zahvaljujem svim nastavnicima Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu te članovima Zavoda za elektroniku, mikroelektroniku, računalne i inteligentne sustave na stvaranju poticajne radne sredine, a naročito suradnicima iz *Laboratorija za potrošaču usmjereno računarstvo* mr.sc. Miroslavu Popoviću, Ivanu Žužaku, dipl.ing., Marinu Šiliću, dipl.ing., Klemi Vladimiru, dipl.ing., Jakovu Kroli, dipl.ing., Goranu Delaču, dipl.ing. i Ivanu Budiseliću, dipl.ing. na brojnim diskusijama vezanim uz istraživački rad, čitanju teksta i preporukama za poboljšanje kakvoće doktorske disertacije. Zahvaljujem vanjskim suradnicima doc.dr.sc. Andri Milanoviću iz tvrtke Synerva d.o.o. Zagreb te dr.sc. Danielu Skrobi, dr.sc. Ivanu Skuliberu, dr.sc. Ivanu Bencu, doc.dr.sc. Darku Huljeniću i doc.dr.sc. Saši Dešiću iz tvrtke Ericsson Nikola Tesla d.d. Zagreb na suradnji tijekom istraživačkog rada.

Posebnu zahvalu upućujem svojim bivšim nastavnicima iz Tehničke škole Čakovec i Osnovne škole Gornji Mihaljevec koji su u meni potaknuli istraživački duh i probudili svijest za ulaganje u vlastito obrazovanje.

Najveću zahvalnost želim izraziti djevojci Maji na riječima podrške, strpljenju i razumijevanju zbog mojih čestih odsutnosti duhom i tijelom za vrijeme istraživačkog rada i pisanja doktorske disertacije. Naposljetku, hvala roditeljima, bratu i prijateljima na moralnoj podršci i razumijevanju te Biserki Budigam, baki Mariji Budigam i pudlu Kaiu na toplom prijemu u svoj dom u Ivanjoj Reci.

Dejan Škvorc

Sadržaj

1 Uvod

Posljednjih nekoliko godina, računalo je postalo uređaj koji uz televizor, štednjak ili telefon koristi gotovo svako kućanstvo. U sprezi s globalnom računalnom mrežom Internet, računala se danas koriste za komunikaciju, informiranje, poslovanje, obrazovanje, zabavu i druženje. Od pojave mrežnih dnevnika (engl. *weblog*, *blog*) [1, 2], raspravnih središta (engl. *forum*) [3], mrežnih enciklopedija kao što je *Wikipedia* [4], središta za razmjenu višemedijskih sadržaja kao što je *YouTube* [5] i društvenih mreža kao što su *Facebook* [6], *MySpace* [7] i *Twitter* [8], računala i mreža Internet koriste se za slobodno izražavanje, ravnopravno sudjelovanje u udruženom stvaranju umreženih sadržaja te utjecaj na stvaranje društvene svijesti. Osim pojavljivanja u osnovnom obliku, računala su danas ugrađena i u veliki broj elektroničkih uređaja i tehničkih sustava široke potrošnje, od telefona, televizora i kućanskih uređaja, do automobila, zgrada i prometnica. Ključno svojstvo koje računalo razlikuje od ostalih oblika potrošačkih uređaja i omogućava njegovu sveprisutnu primjenu je svojstvo programirljivosti. Svojstvo programirljivosti čini računalo pogodnom napravom za jednostavnu preobrazbu između različitih oblika funkcionalnosti koje pruža potrošačima.

Suvremeni primjenski programi koji se danas koriste za obavljanje svakodnevnih osobnih i poslovnih potreba, uz primjenu ponuđenih sadržaja, potrošačima omogućuju i visoki stupanj uključenosti u njihovo stvaranje. Mrežni dnevnici, raspravna središta, mrežne enciklopedije i društvene mreže pretvaraju korisnike iz pasivnih potrošača u aktivne proizvođače informacija. Međutim, usprkos razmjerno izražajnim obrascima za upravljanje sadržajima i funkcionalnostima dostupnima putem računala, programiranje računala,

odnosno izrada računalnih programa, još uvijek je privilegija školovanih programera. S druge strane, sveprisutnost i veliki broj pojavnosti računala u svakodnevnim životnim i radnim okolinama zahtijevaju izražajnije oblike međudjelovanja od onih koji se postižu primjenom računalnih programa. Potpuni doživljaj i prilagodba računalima pogonjenih okolina pojedinačnim sklonostima i potrebama zahtijevaju mogućnost samostalne izgradnje računalnih programa od strane potrošača.

Za razliku od ostalih oblika primjene računala, programiranje se smatra izrazito zahtjevnom umnom aktivnošću jer zahtijeva apstraktni način razmišljanja i formalni oblik izražavanja kako bi se izrečena informacija mogla upotrijebiti u obliku računalnog programa. Usprkos stručnom znanju iz područja vlastite djelatnosti, bez poznavanja tehnike programiranja, prosječni korisnik računala nije u mogućnosti vlastito znanje samostalno pretvoriti u računalni program. Iako nerijetko sposoban osmisliti inovativne računalne programe za rješavanje problema s kojima se susreće, zbog nemogućnosti samostalnog programiranja računala, prosječni korisnik računala ograničen je na uporabu onih primjenskih programa koje gradi razmjerno mali broj programera. Jedan od istraživačkih izazova u području informacijsko-komunikacijske tehnologije u nadolazećem razdoblju je, stoga, osmišljavanje i razvoj računalnih okolina, oblikovnih postupaka i razvojnih alata koji širokom krugu potrošača, nesputanih stupnjem i vrstom obrazovanja koje su stekli, djelatnošću kojom se bave i ekonomskim uvjetima u kojima djeluju, omogućuju samostalni razvoj primjenskih programa.

Istraživanje obuhvaćeno ovom doktorskom disertacijom bavi se programiranjem prilagođenim potrošaču (engl. *consumer programming*). Programiranje prilagođeno potrošaču bavi se istraživanjem i oblikovanjem tehnologija, postupaka i razvojnih alata pogodnih za samostalnu izgradnju primjenskih programa od strane potrošača, bez potrebe za posebnim oblicima obrazovanja. U okviru doktorske disertacije uspostavljen je teorijski model za ocjenu svojstava koja programiranje čine prikladnim za najširi krug potrošača računala, predložena je metodologija za ostvarenje takve vrste programiranja te je ocjenjeno povećanje stvaralačkih mogućnosti koje nastaje uključivanjem širokog kruga korisnika računala u razvoj primjenskih programa.

U nastavku ovog poglavlja opisani su razlozi koji potiču istraživanje programiranja prilagođenog potrošaču te su definirani ciljevi istraživanja. U poglavlju 2 uvedena je definicija programiranja prilagođenog potrošaču, uspostavljena je metodologija istraživanja te je prikazana organizacija doktorske disertacije po poglavljima.

1.1 Poticaj istraživanja

Nekoliko je glavnih razloga za istraživanje područja programiranja prilagođenog potrošaču. Prvi razlog je porast oslonjenosti društva na informacijsko-komunikacijske tehnologije i s njima povezane usluge koji dovodi do sve veće potrebe za razvojem primjenskih programa za gospodarenje globalno dostupnim informacijama. S druge strane, tržište rada u području računarstva već je duže vrijeme suočeno s nedostatkom programera. Potražnja na tržištu rada znatno premašuje broj stručnjaka koje je moguće uputiti na tržište kroz sveučilišne obrazovne programe [9]. Programiranjem prilagođenim potrošaču, dio razvoja primjenskih programa prepušta se potrošačima.

Drugi razlog je potreba za izgradnjom poosobljenih primjenskih programa kojima potrošači zadovoljavaju vlastite potrebe za kakvoćom doživljaja (engl. *quality of experience – QoE*) [10]. U okolnostima u kojima su životne i radne okoline čovjeka pogonjene ili upotpunjene informacijskim sustavima, nerijetko se zahtijeva njihova prilagodba poosobljenim potrebama pojedinaca. Budući da ulaganje sredstava u razvoj poosobljenih primjenskih programa koje koristi mali broj ljudi nije ekonomski isplativo, jedno od mogućih rješenja je uključivanje potrošača u samostalni razvoj primjenskih programa za vlastite potrebe.

Treći razlog je ublažavanje digitalne podijeljenosti (engl. *digital divide*) [11, 12] između potrošača računalnih sustava i programera te između potrošača u razvijenim i nerazvijenim društvima. Uključivanjem širokog kruga potrošača u udruženi razvoj primjenskih programa nastaju nove vrijednosti koje proizlaze iz raznolikosti i širine zamisli velikog broja ljudi. Time se stvara zajednica čije stvaralačke mogućnosti i inovativni doprinos mnogostruko premašuju one kojima raspolaže razmjerno mali broj školovanih programera. Nadalje, oblikovanjem programiranja prilagođenog potrošaču, razvoj primjenskih programa približava se korisnicima tehnološki nerazvijenih područja kojima obrazovanje u području informacijsko-komunikacijske tehnologije nije dostupno.

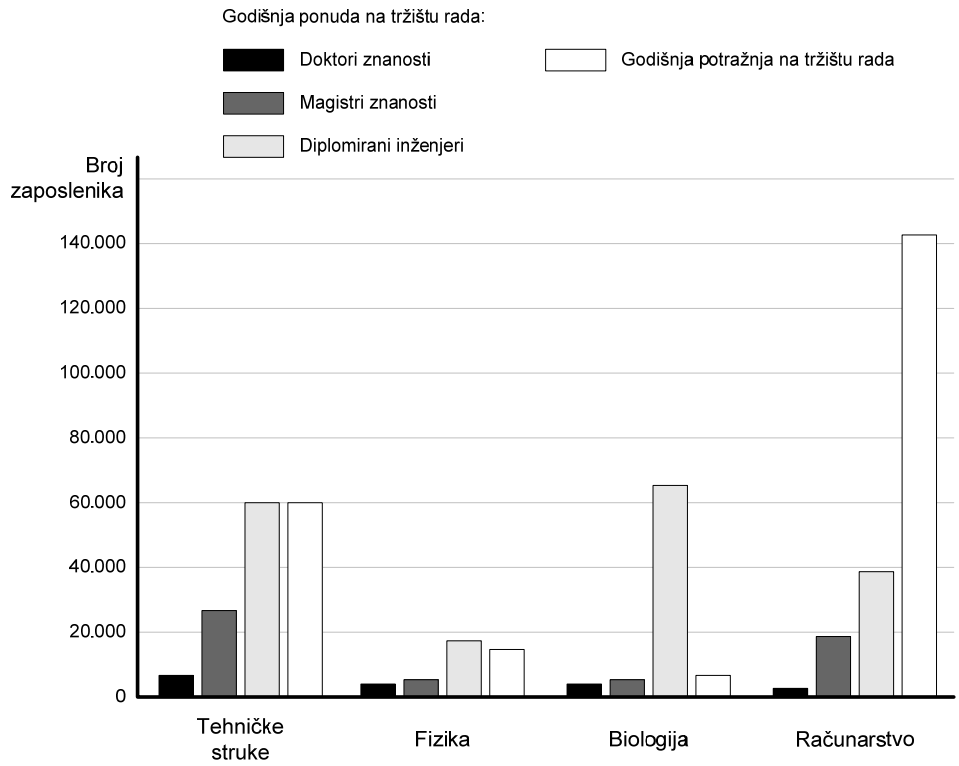
1.1.1 Nesrazmjer ponude i potražnje na tržištu rada računarstva

Jedan od najvažnijih razloga za prepuštanje dijela razvoja primjenskih programa potrošačima je nesrazmjer ponude i potražnje na tržištu rada u području računarstva. Slikom 1.1 prikazani su godišnji odnosi ponude i potražnje na tržištu rada Sjedinjenih Američkih Država za razdoblje od 2002. do 2012. godine za četiri područja djelatnosti. Prikazani podaci rezultat su statističkih ispitivanja i predviđanja budućih kretanja koje je predstavilo

Ministarstvo trgovine Sjedinjenih Američkih Država na konferenciji *CRA Computing Research Summit* održanoj 2004. godine [9].

Kao što je vidljivo iz grafičkog prikaza, iskazane potrebe za školovanim stručnjacima u području tehničkih struka tijekom jednogodišnjeg razdoblja iznose približno dvije trećine ukupno školovanih stručnjaka koji se u istom razdoblju pojavljuju na tržištu rada. Odnosi ponude i potražnje na tržištu rada u području fizike približno su isti, ali je u tom području ukupni broj iskazanih potreba i raspoloživih kadrova znatno manji. S gledišta stručnjaka raspoloživih na tržištu rada, situacija je znatno nepovoljnija u području biologije, gdje ponuda stručnih kadrova gotovo za red veličine premašuje potražnju.

S druge strane, u području računarstva primjetna je potpuno obrnuta situacija. Dok u ostalim područjima djelatnosti tržište rada uspijeva zadovoljiti, a nerijetko i značajno premašiti potražnju, u području računarstva ponuda školovanih stručnjaka uspijeva zadovoljiti tek četvrtinu iskazanih potreba za zaposlenicima. Kako bi barem djelomično namirili nedostatak školovanih kadrova, poslodavci u području računarstva primorani su zapošljavati priučene zaposlenike ili stručnjake iz drugih područja s neodgovarajućim ili površnim poznavanjem struke.



Slika 1.1. Usporedba godišnjih odnosa ponude i potražnje na tržištu rada Sjedinjenih Američkih Država u različitim područjima djelatnosti za razdoblje od 2002. do 2012. godine

Nesrazmjer između ponude i potražnje na tržištu rada u području računarstva posljedica je velike potražnje za novim primjenskim programima koja je uzrokovana sveprisutnošću računala i informacijsko-komunikacijske tehnologije. S druge strane, nedostatku graditelja programske potpore pogoduje i visoka razina tehnologijskih znanja i vještina kojima je potrebno ovladati prije upuštanja u razvoj programskih sustava. Oblikovanje i razvoj programske potpore zasniva se na primjeni programskih tehnologija niske razine apstrakcije, kao što su tekstualni programski jezici i apstraktno grafičko znakovlje, što postupke oblikovanja, izgradnje, ispitivanja i održavanja čini dugotrajnim i ograničenim na uski krug školovanih programera.

Cilj istraživanja

Jedno od rješenja za prevladavanje nesramjera između ponude i potražnje stručnjaka u području programskog inženjerstva je povećanje broja školovanih programera putem sveučilišnih obrazovnih programa. Drugi način je pronalazak novih metodologija, razvojnih postupaka i programskih apstrakcija za poboljšanje učinkovitosti graditelja suvremenih programskih sustava. Zamisao istraživanja provedenog u okviru ove doktorske disertacije je prijedlog novog programskog modela za razvoj računalnih primjenskih programa na visokim razinama apstrakcije koje skrivaju arhitekturalne i tehnologijske pojedinosti računalnih i informacijskih sustava, a naglašavaju primjenska svojstva računalnih programa. Uvođenjem novih programskih apstrakcija te postupaka oblikovanja i izgradnje primjenskih programa, predloženi programski model s jedne strane školovanim stručnjacima podiže razinu učinkovitosti, a s druge strane omogućava velikom broj krajnjih potrošača primjenskih programa aktivno i samostalno uključivanje u njihov razvoj.

1.1.2 Poosobljeni primjenski programi i kakvoća doživljaja

Porastom broja korisnika računala te količine i raznovrsnosti računalnih sadržaja, čemu je ponajviše pridonijela sveprisutnost elektroničkih usluga dostupnih u mreži Internet, ubrzano raste i raznovrsnost korisničkih zahtjeva te zahtijevana razina kakvoće korisničkog doživljaja (engl. *quality of experience – QoE*) [10]. Jedan od značajnih elemenata kakvoće korisničkog doživljaja je dostupnost usluge ili sadržaja upravo u onom obliku koji je potrošaču najpogodniji. Međutim, usprkos raznovrsnosti ponude usluga i sadržaja koju nude današnji primjenski programi, za obavljanje složenih poslova potrošači još uvijek nerijetko trebaju koristiti nekoliko različitih primjenskih programa dostupnih iz različitih izvora. Primjerice, napredno pretraživanje stanja na tržištu građevinskog zemljišta zahtijeva istovremeno korištenje usluge za elektroničko oglašavanje nekretnina, usluge elektroničkih

zemljopisnih karata s planovima gradova za pronalaženje položaja oglašene nekretnine, usluge gradske uprave s pregledom cijena pristojbi za izdavanje građevinskih dozvola za pojedine dijelove grada te usluge policijske uprave sa statističkim pokazateljima stope kriminala za pojedine dijelove grada.

Potreba za obavljanjem složenih poslova primjenom nekoliko različitih usluga dostupnih iz različitih izvora ima nepovoljni utjecaj na kakvoću korisničkog doživljaja. Problem nedostatka cjelovitih rješenja za obavljanje složenih poslova raste s porastom broja potrošača i s porastom učestalosti kojom pojedini potrošač zahtijeva već viđenu i korištenu kombinaciju sadržaja. S porastom učestalosti primjene ovog složenog postupka, raste i nezadovoljstvo potrošača zbog utroška vremena na ponavljajuće aktivnosti i veću mogućnost nastanka pogreške.

Cilj istraživanja

S obzirom na to da nije moguće unaprijed predvidjeti raznovrsne zahtjeve potrošača koji su često vrlo poosobljene prirode, razinu kakvoće korisničkog doživljaja moguće je povećati samostalnim poosobljavanjem primjenskih programa od strane potrošača. Jedan od oblika poosobljavanja je povezivanje dostupnih informacija, usluga i procesa u složene procese prilagođene potrebama potrošača. Obavljanje složenog posla za koji je u nedostatku cjelovitog rješenja potrebno pribjeći korištenju nekoliko dostupnih informacija, usluga ili procesa moguće je zamijeniti programskim povezivanjem skupa dostupnih informacija, usluga ili procesa u poosobljeni radni tijek (engl. *personalized workflow*) posebno prilagođen potrebama pojedinca. Programsko povezivanje skupa informacija, usluga i procesa u novi složeni proces putem kojeg je složena informacija ili usluga dostupna na jednom mjestu omogućava potrošaču organizaciju informacija, usluga i ostalih računalnih sadržaja na njemu svojstven, prirodan i poželjan način.

1.1.3 Uključenost sveukupne društvene zajednice u stvaranje primjenskih programa

Osim nedostatka školovanih programera i potrebe za izgradnjom poosobljenih primjenskih programa, treći značajan razlog za istraživanje područja programiranja prilagođenog potrošaču je ublažavanje digitalne podijeljenosti (engl. *digital divide*) [11, 12] između potrošača primjenskih programa i računalnih programera. U današnje vrijeme sveprisutnosti računala i mreže Internet, moguće je uočiti nekoliko oblika digitalne podijeljenosti, od podijeljenosti na tehnologijski razvijene i tehnologijski zakinute krajeve,

preko podijeljenosti u računalnoj pismenosti, odnosno vještinama i znanjima ljudi koje su potrebne za primjenu računala i internetskih usluga, do podijeljenosti u sposobnosti programiranja računala.

Ublažavanju različitih oblika digitalne podijeljenosti pristupa se na različite načine. Zahvaljujući pojedincima, tvrtkama, društvenim zajednicama i vladama zemalja koji se nalaze na razvijenoj strani digitalno podijeljenog svijeta, pokrenute su opsežne inicijative za uključivanje cjelokupnog svjetskog stanovništva u jedinstveni globalni informacijski prostor, bez obzira na njihovo obrazovanje, imovinsko stanje, nacionalnu pripadnost te ruralnu i teritorijalnu izdvojenost [13]. Djelovanje povezano sa sudjelovanjem u stvaranju informacijskog društva i uključenošću u globalni informacijski prostor naziva se e-uključenošću (engl. *e-inclusion*, *e-participation*).

Razvijeni svijet nastoji postići e-uključenost kroz više razina, od uspostave informacijske infrastrukture i školovanja ljudi za primjenu informacijskih tehnologija, preko razvoja primjenskih programa za pristup informacijama, do razvoja primjenskih programa za objavljivanje informacija. Infrastrukturna uključenost najčešće se provodi programima uvođenja širokopojasnog pristupa mreži Internet u škole i domove te osiguravanjem jeftinih računala [14] i sredstava za izgradnju komunikacijske infrastrukture [15] u nerazvijenim dijelovima svijeta. Viša razina e-uključenosti postiže se razvojem primjenskih programa za pronalazak i pristup informacijama kao što su internetske tražilice, a najviša razvojem primjenskih programa za objavljivanje informacija, kao što su mrežni dnevnici [1, 2], raspravna središta [3], društvene mreže [6, 7, 8] te središta za razmjenu višemedijskih informacija [5] i gradnju otvorenih enciklopedija [4]. Primjena ove vrste primjenskih programa omogućuje da korisnici od pasivnih potrošača postanu aktivni proizvođači informacija, čime svaki pojedinac, bez obzira na kojoj se strani digitalno podijeljenog svijeta nalazi, dobiva priliku za sudjelovanje u stvaranju društvene svijesti.

Iako su dosadašnjim nastojanjima za ublažavanje digitalne podijeljenosti postignuti zapaženi rezultati na polju globalne uključenosti sveukupne svjetske zajednice u stvaranje, razmjenu i primjenu informacija dostupnih putem mreže Internet, potpunu digitalnu ravnopravnost moguće je postići tek kada se svima pruži jednaka prilika za razvoj primjenskih programa za gospodarenje informacijama. S obzirom na pojačanu težnju suvremenog društva prema oslonjenosti na informacijsko-komunikacijske tehnologije i s njima povezane usluge, životne i radne okoline pogonjene suvremenim informacijskim sustavima nerijetko zahtijevaju prilagodbu poosobljenim potrebama pojedinaca. U takvim okolnostima, korisnicima mreže Internet nije dovoljna samo mogućnost izbora između

ponuđenih primjenskih programa koje za njih grade posebno školovani programeri. Za potpunu prilagodbu primjenskih programa vlastitim sklonostima i potrebama te za potpuni doživljaj sudjelovanja u stvaranju digitalnog društva, korisnicima mreže Internet potrebno je omogućiti samostalno stvaranje novih primjenskih programa za svoju i opću dobrobit.

Iskorištavanje stvaralačkih mogućnosti široke društvene zajednice u području informacijsko-komunikacijske tehnologije nije nepoznanica. Jedan od najpoznatijih računalnih sustava oslonjenih na stvaralački doprinos širokog kruga korisnika je otvorena internetska enciklopedija *Wikipedia* [4]. Enciklopedija *Wikipedia* omogućuje svakom korisniku mreže Internet slobodno stvaranje i uređivanje enciklopedijskih članaka. Znanje skupljeno u enciklopedijskim člancima internetskoj je zajednici izloženo u obliku koji je rezultat zajedničkog djelovanja skupine ljudi. Iako točnost informacija u člancima nije formalno provjerena i nema jedinstvenog autora pojedinog članka, točnost informacija razmjerno je velika. Štoviše, članci uređeni od strane velike skupine autora manje su podložni iznošenju jednostranih pogleda na teme oko kojih ne postoji jedinstveno mišljenje.

Cilj istraživanja

Zamisao programiranja prilagođenog potrošaču je oblikovanje metodologije programiranja koja je dovoljno jednostavna za uključivanje sveukupne internetske zajednice u razvoj računalnih programa. Spregom programiranja prilagođenog potrošaču i mreže Internet omogućuje se zajedničko djelovanje i međusobno nadopunjavanjem skupine ljudi tijekom programiranja, čime se zajednički dolazi do novih vrijednosti u obliku računalnih programa. Programiranjem prilagođenim potrošaču, pojedinci dobivaju mogućnost samostalnog razvoja primjenskih programa za vlastite potrebe, a društvo inovativna rješenja kao posljedicu neusporedivo većih stvaralačkih mogućnosti od onih kojima raspolaže razmjerno mali broj školovanih programera.

U današnje vrijeme, većina potrošača računala ovladava osnovama računalne pismenosti kojima je obuhvaćeno poznavanje osnova rada na računalu uporabom primjenskih programa. Uz poznavanje uloge i funkcije određenog primjenskog programa, za njegovu primjenu zahtijeva se i poznavanje određenog oblika međudjelovanja putem sučelja čovjek-računalo (engl. *human-computer interaction – HCI*). Najrašireniji oblik međudjelovanja čovjek-računalo u današnje je vrijeme uzajamno djelovanje korisnika putem grafičkog korisničkog sučelja koje se sastoji od tipki, slika, polja za unos znakova i padajućih izbornika (engl. *windows-icons-menus-pointers – WIMP*) [16, 17]. Naročit doprinos omasovljavanju broja potrošača koji ovladavaju osnovama računalne pismenosti i

vještinama za upravljanje radom računala putem grafičkog korisničkog sučelja dala je sveprisutnost informacijsko-komunikacijske tehnologije i primjenskih programa zasnovanih na pregledniku *World Wide Web* sadržaja.

Usprkos značajnom napretku u području podizanja razine računalne pismenosti i vještina za korištenje primjenskih programa, razvoj novih primjenskih programa još uvijek zahtijeva visoku razinu tehnologijskog znanja koje nije dostupno širokom krugu potrošača. Za razvoj primjenskih sustava zahtijeva se poznavanje programskog inženjerstva, programskih jezika i tehnika programiranja. Iako je upravljanje radom računala putem grafičkog korisničkog sučelja danas poznato i blisko većini potrošača, te vještine nisu iskorištene u programiranju. Istraživanje provedeno u okviru ove doktorske disertacije bavi se ispitivanjem svojstava međudjelovanja čovjeka i računala putem grafičkog korisničkog sučelja primjenskih programa prikazanih u pregledniku *World Wide Web* sadržaja i mogućnostima njegove primjene u svojstvu programskog jezika namijenjenog potrošaču. Cilj istraživanja je pronalazak metodologije programiranja koja pojednostavljuje razvoj primjenskih programa do mjere da su za njihovu izgradnju dovoljna znanja i vještine o uzajamnom djelovanju korisnika i računala putem grafičkog korisničkog sučelja prikazanog u pregledniku *World Wide Web* sadržaja.

2 Programiranje prilagođeno potrošaču

Programiranje prilagođeno potrošaču (engl. *consumer programming*) bavi se istraživanjem i oblikovanjem tehnologija, oblikovnih postupaka i razvojnih alata pogodnih za samostalnu izgradnju računalnih primjenskih programa od strane potrošača. Ključno svojstvo koje programiranje prilagođeno potrošaču razlikuje od ostalih oblika programiranja je mogućnost izgradnje primjenskih programa bez potrebe za posebnim oblicima obrazovanja. Slično kao što su područjem potrošačke elektronike (engl. *consumer electronics*) uvedena načela oblikovanja elektroničkih uređaja, uključujući i oblikovanje računala, kako bi ti uređaji cijenom i načinom primjene postali prikladni za široku potrošnju [18], programiranjem prilagođenim potrošaču uvode se načela kojima se širokom krugu potrošača računala približavaju postupci izgradnje računalnih primjenskih programa.

S obzirom na ciljnu skupinu korisnika, današnje programske paradigme i jezici pripadaju jednoj od dviju osnovnih grana razvoja programske potpore. U grani *profesionalnog razvoja programske potpore* koriste se programske paradigme i jezici opće namjene kojima je moguće odgovoriti na većinu zahtjeva naručitelja ili tržišta. Primjena programskih paradigmi za profesionalni razvoj programske potpore zahtijeva stručnjake obrazovane u području programskog inženjerstva. S druge strane, u grani *razvoja prilagođenog krajnjem korisniku* (engl. *end-user development – EUD*) [19, 20, 21] oblikuju se programske paradigme posebne namjene koje su prilagođene pojedinim područjima struke. Ciljna skupina ovih programskih paradigmi i jezika su stručnjaci u pojedinim granama ljudske djelatnosti koji grade primjenske programe za potporu obavljanju vlastite djelatnosti. *Programiranjem prilagođenim potrošaču* otvara se novo područje programskog

inženjerstva kojim se izrada primjenskih programa nastoji približiti svim korisnicima računala, bez obzira na stupanj stručnog obrazovanja i djelatnost kojom se bave.

Kako bi se omogućila izgradnja primjenskih programa bez potrebe za obrazovanjem u području programskog inženjerstva, programska paradigma prilagođena potrošaču zasniva se na izboru kognitivno prihvatljivih elemenata, tehnika i pravila za izgradnju primjenskih programa, na općim znanjima potrošača stečenim tijekom redovnog školovanja te na vještinama stečenim iskustvom tijekom uporabe računala i računalnih primjenskih programa. Programiranje prilagođeno potrošaču je višedisciplinarno znanstveno područje koje istraživanja u području programskog inženjerstva upotpunjuje rezultatima istraživanja u području kognitivne znanosti, psihologije te istraživanja i planiranja tržišta.

U ovom poglavlju definirane su ključne značajke programiranja prilagođenog potrošaču. Poglavlje 2.1 definira potrošače kao ciljnu skupinu korisnika programiranja prilagođenog potrošaču. Prikazana je usporedba osobina u kojima se potrošači značajno razlikuju od krajnjih korisnika koji primjenske programe grade u području vlastite stručnosti te školovanih programera koji primjenske programe grade za naručitelja ili na zahtjev tržišta.

U poglavlju 2.2 prikazan je ciljni razred primjenskih programa za čiju je izgradnju namijenjeno programiranje prilagođeno potrošaču. Primjenski programi izgrađeni ovom vrstom programiranja zasnovani su na komponentnom *World Wide Web* sustavu, a grade se usložnjavanjem udomljenika (engl. *widgets*, *gadgets*). Prikazana je sprega programiranja prilagođenog potrošaču i profesionalnog razvoja programske potpore te je pokazano da ove dvije grane razvoja programske potpore nisu isključive, već se međusobno nadopunjavaju.

U poglavlju 2.3 definirana su kognitivna svojstva programiranja prilagođenog potrošaču. Modelom ulaganja umnih sredstava opisani su procesi koji tijekom programiranja nameću najveće umno opterećenje na korisnika te je predložen model za njihovo izbjegavanje tijekom oblikovanja programiranja prilagođenog potrošaču. Kognitivne značajke programiranja prilagođenog potrošaču uspoređene su sa značajkama programskih paradigmi u području razvoja prilagođenog krajnjem korisniku i profesionalnog razvoja programske potpore.

Na osnovi kognitivnih svojstava programiranja prilagođenog potrošaču prepoznata su srodna znanstvena područja koja je potrebno proučiti kako bi se izveli zaključci o izboru elemenata, tehnika i pravila za ostvarenje potrošaču prilagođene programske paradigme. Metodologija istraživanja opisana je u poglavlju 2.4. Poglavlje završava pregledom organizacije doktorske disertacije po poglavljima.

2.1 Potrošači, krajnji korisnici i programeri

Definicija programiranja prilagođenog potrošaču zasniva se na uočljivim razlikama u ponašanju, razmišljanju, znanju, očekivanjima i potrebama između potrošača, krajnjih korisnika i školovanih programera. Razlike između ovih triju ciljnih skupina korisnika razvojnih metodologija i razvojnih alata postoje u stupnju stručne izobrazbe u području programskog inženjerstva, području primjene izgrađenih primjenskih programa, veličini i težini problema koje pripadnici pojedine skupine nastoje riješiti programskim putem te postupcima koje koriste tijekom rješavanja problema.

Programerima se smatraju stručnjaci u području programskog inženjerstva koji su školovani u svim područjima razvoja programskih sustava, od prikupljanja i obrade korisničkih zahtjeva te izrade programske specifikacije, preko ostvarenja i ispitivanja programskog sustava, do izrade programske dokumentacije i održavanja sustava. Osnovna djelatnost programera je projektiranje i izrada programskih sustava za naručitelja ili na zahtjev tržišta. Programske paradigme, jezici i alati koje koriste školovani programeri omogućuju izgradnju primjenskih programa opće namjene uz iskorištavanje svih posebnosti računalne arhitekture za koju se gradi program. Izražajnost takvih programskih paradigmi, jezika i alata najčešće se postiže na račun jednostavnosti primjene.

Krajnjim korisnicima smatraju se stručnjaci u određenom području ljudske djelatnosti izvan područja programskog inženjerstva, kao što je kemija, fizika, matematika, medicina, ekonomija, prosvjeta, elektrotehnika, strojarstvo ili građevinarstvo. Iako njihova osnovna djelatnost nije razvoj primjenskih programa, krajnji korisnici nerijetko pokazuju potrebu za samostalnom izgradnjom primjenskih programa za potporu vlastitoj djelatnosti. Najčešći razlog su posebnosti programske potpore koje zahtijevaju dobro razumijevanje struke. Kako bi se korisnicima koji nisu školovani za razvoj programskih sustava olakšao razvoj primjenskih programa, programske paradigme, jezici i alati posebno su prilagođeni ciljnom području primjene. Područje razvoja primjenskih programa prilagođenog krajnjem korisniku i njegov utjecaj na oblikovanje programiranja prilagođenog potrošaču iscrpnije su opisani u poglavlju 5.

Potrošačima se smatra najširа skupina korisnika računalnih primjenskih programa, neovisno o stupnju i vrsti stručne naobrazbe te djelatnosti kojom se bave. Najčešći oblik primjenskih programa za najširi krug potrošača računala u današnje su vrijeme primjenski programi za *World Wide Web* prikazani u pregledniku *World Wide Web* sadržaja. Zbog raznovrsnosti ponuđenih sadržaja, primjenskim programima za *World Wide Web* moguće je zadovoljiti većinu potrošačkih potreba. U području programiranja prilagođenog potrošaču,

potrošačem se smatra korisnik koji je usvojio načela primjene računala putem primjenskih programa za *World Wide Web*. Za razliku od krajnjih korisnika koji primjenske programe grade i koriste u području vlastite stručnosti, potrošači primjenske programe uglavnom grade i koriste za široki opseg svakodnevnih osobnih potreba koje obuhvaćaju široko područje primjene, od informiranja, zabave i druženja, preko kućanskih poslova, do obrazovanja i zdravstva, neovisno o djelatnosti kojom se stručno bave.

Tablica 2.1. Usporedba osobina stručnjaka i potrošača

	Stručnjaci	**Potrošači**
Postupak upoznavanja s novim tehnologijskim rješenjem	Primjena priručnika za uporabu	Metoda pokušaja i pogreške
Razina upoznavanja s novim tehnologijskim rješenjem	Funkcijska svojstva, način primjene i načela rada	Funkcijska svojstva i način primjene
Poticaj za izradu računalnih programa	Usmjereni na opća rješenja ili potrebe struke	Usmjereni osobnim potrebama
Doživljaj programiranja	Svjesna uključenost u razvoj programske potpore	Popratna pojava uobičajenih radnji

U okviru istraživanja, uspoređene su osobine školovanih programera, krajnjih korisnika i potrošača tijekom primjene tehnologijskih proizvoda i usluga. Uočeno je da u doticaju s tehnologijskim proizvodima i uslugama, prosječni potrošači pokazuju određene osobine koje ih značajno razlikuju od ljudi koji te proizvode i usluge koriste za obavljanje stručne djelatnosti. Primjerice, dok stručnjaci u području vlastite djelatnosti primjenjuju metode koje zadovoljavaju propise, standarde ili ustaljenu politiku radne organizacije, a za njihovo nepoštivanje snose odgovornost, potrošači tehnologijske proizvode i usluge koriste po vlastitoj volji, često zanemarujući čak i preporuke proizvođača. Tablicom 2.1 istaknute su ključne razlike u osobinama stručnjaka i potrošača koje je potrebno uzeti u obzir tijekom oblikovanja programiranja prilagođenog potrošaču.

Tijekom upoznavanja novih tehnologijskih rješenja, stručnjaci pokazuju sklonost proučavanju priloženih priručnika za uporabu kako bi postigli ispravan rad proizvoda i izbjegli odgovornost za nepropisno rukovanje. S druge strane, većina prosječnih potrošača sklonija je eksperimentiranju metodom pokušaja i pogreške. Ova osobina potrošača ukazuje da je programiranje prilagođeno potrošaču potrebno definirati na način da dopušta upuštanje u postupak programiranja bez potrebe za prethodnom primjenom priručnika za uporabu. Paradigmu programiranja potrebno je oblikovati na način da već i prvi pokušaj upuštanja potrošača u postupak programiranja daje vidljive rezultate, a za njegovu primjenu dovoljna su znanja i iskustva kojima većina potrošača ovladava izvan konteksta programiranja.

Upoznavajući se s novim tehnologijskim rješenjem, osim razumijevanja funkcijskih svojstava i načina primjene, stručnjaci pokazuju i sklonost upoznavanja s načelima rada proizvoda ili usluge. S druge strane, za većinu potrošača načela rada nisu bitna dok proizvod ili usluga propisno ispunjava namijenjenu funkciju i način primjene. Programiranje prilagođeno potrošaču potrebno je, stoga, oblikovati na način da je računalne programe moguće graditi od sastavnih dijelova poznavajući isključivo njihovu funkciju i način primjene, bez potrebe za poznavanjem unutarnjih načela rada.

Tijekom izrade računalnih programa, stručnjaci su usmjereni na opća rješenja koja zadovoljavaju potrebe većine potrošača ili potrebe struke kojom se bave. Primjerice, programeri su navikli voditi brigu o svim elementima koji utječu na izradu programa, od računalne arhitekture i operacijskog sustava za izvođenje programa, preko mogućih sigurnosnih rizika, do ciljne skupine korisnika programskog rješenja. Krajnji korisnici usmjereni su na svojstva programa koja su usko povezana sa zakonitostima struke kojom se bave, ali se ne opterećuju pojedinostima računalne okoline u kojoj se program koristi. S druge strane, potrošači su usmjereni vlastitim primjenskim potrebama. Programiranjem prilagođenim potrošaču poželjno je omogućiti izradu računalnih programa usmjeravajući se prvenstveno na njihova primjenska svojstva.

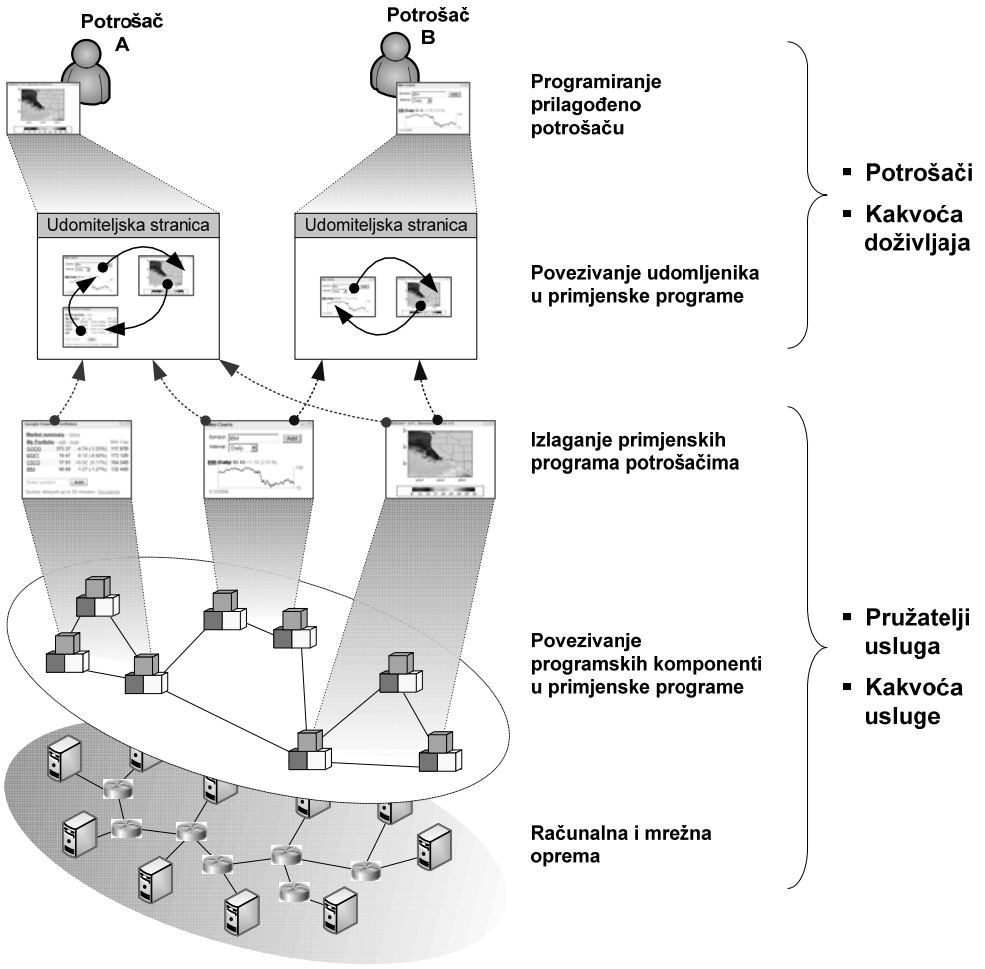
Razlika između potrošača i stručnjaka postoji i u doživljaju programiranja, odnosno u pristupu izgradnji primjenskih programa. Dok stručnjaci u izradu programske potpore svjesno ulažu dodatni napor, vrijeme i znanje kako bi u budućnosti od toga ostvarili korist, potrošači u pravilu nisu zainteresirani za takav oblik ulaganja dodatnog napora i vremena. Potrošači su usmjereni na rješavanje vlastitih zadaća primjenom računala. Programiranje prilagođeno potrošaču je, stoga, potrebno oblikovati kao nenametljivu popratnu pojavu uobičajenih radnji potrošača tijekom uporabe računala. Time se omogućuje izgradnja primjenskih programa bez potrošačeve svjesne uključenosti u postupak programiranja.

Usprkos značajnim razlikama u pristupu rješavanju problema programskim putem, između stručnjaka i potrošača postoje i brojne sličnosti. Višestruko korištenje jednom izgrađenog programskog rješenja, razmjena rješenja u zajednici korisnika sa sličnim interesima te održavanje i unaprjeđivanje programskog rješenja podjednako su važni zahtjevi za obje skupine korisnika.

2.2 Primjenski programi izgrađeni od strane potrošača

Osim uočljivih razlika u osobinama, znanju i vještinama potrošača koje ih razlikuju od školovanih programera i krajnjih korisnika, primjenski programi izgrađeni programiranjem

prilagođenim potrošaču također pokazuju posebnosti po kojima se razlikuju od primjenskih programa izgrađenih primjenom ostalih oblika programiranja. Svrha primjenskih programa izgrađenih programiranjem prilagođenim potrošaču je povezivanje *potrošaču usmjerenih primjenskih funkcija* u *poosobljene radne tijekove po volji potrošača* (engl. *consumer-specific workflow*) kojima se zadovoljavaju pojedinačne potrebe potrošača za kakvoćom doživljaja (engl. *quality of experience – QoE*).



Slika 2.1. Višerazinski model za potrošaču prilagođenu izgradnju primjenskih programa sa svojstvom kakvoće doživljaja

Slikom 2.1 prikazan je višerazinski model za potrošaču prilagođenu izgradnju primjenskih programa sa svojstvom kakvoće doživljaja. Prikazani model ističe spregu programiranja prilagođenog potrošaču s granom profesionalnog razvoja programske potpore. Programiranje prilagođeno potrošaču ne zamjenjuje profesionalni razvoj programske potpore, već ga nadopunjava u dijelovima koji se odnose na poosobljene zahtjeve pojedinaca na koje uobičajenim postupcima razvoja programske potpore nije moguće kvalitetno odgovoriti.

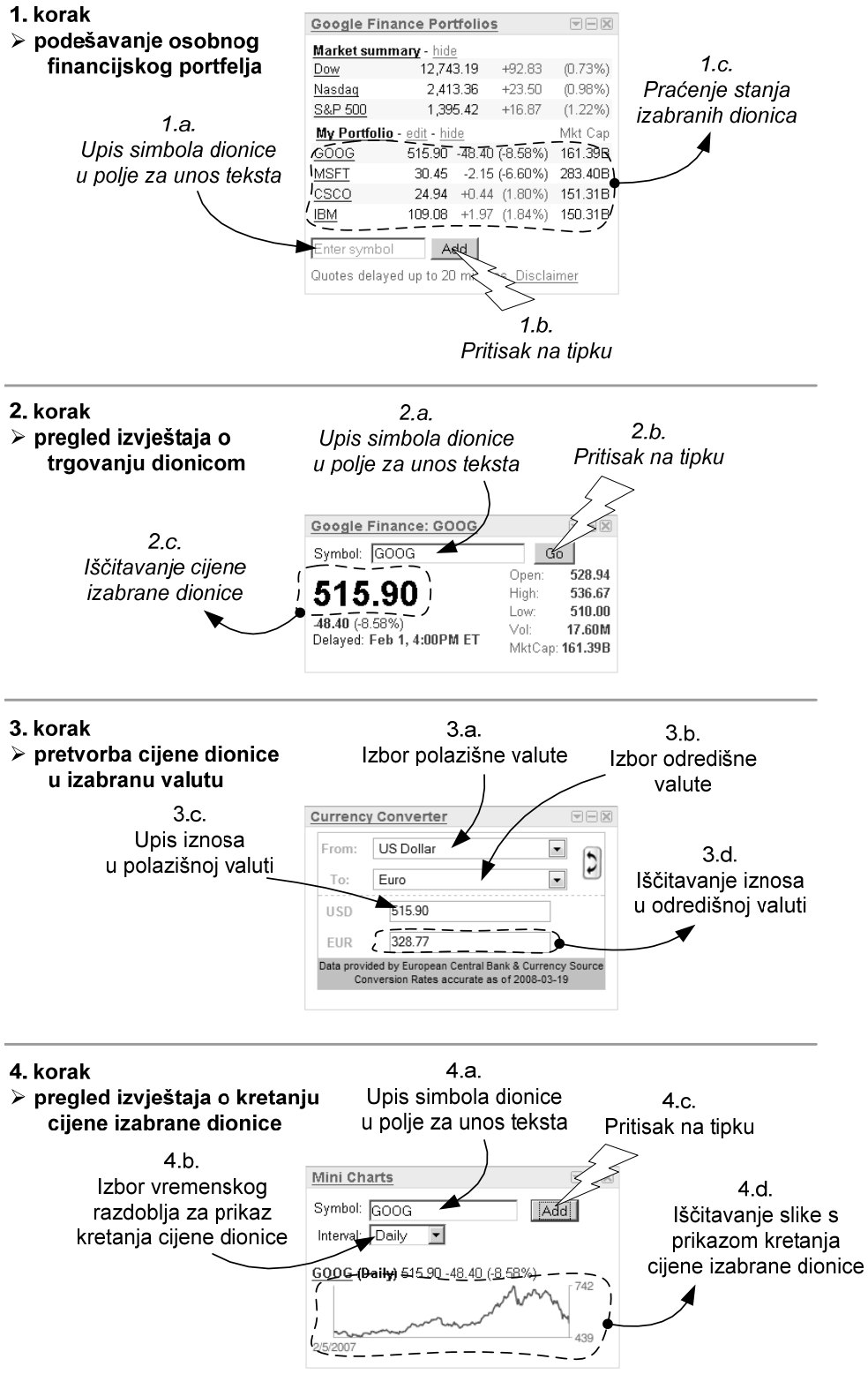
Na sklopovskoj računalnoj i mrežnoj opremi izvode se programske komponente koje obavljaju pojedine dijelove primjenske logike. Školovani programeri prema zahtjevu pružatelja usluga povezuju skup komponenata u primjenske programe na način da se zadovolje funkcijska svojstva programa i kakvoća isporučene usluge. Parametri kakvoće usluge su cijena usluge, dostupnost usluge, odzivna svojstva usluge i slično. Funkcionalnosti primjenskih programa koji se sastoje od skupa povezanih programskih komponenata potrošačima su najčešće izložene putem *World Wide Web* stranica.

Usprkos tome što su pojedini primjenski programi potrošačima izloženi u obliku *World Wide Web* stranica, cjelovita rješenja za obavljanje složenih poslova često nisu dostupna. U takvim okolnostima, korisnici su primorani pretraživati i pronalaziti pojedinačne sadržaje, koristiti ih na način na koji to pojedini pružatelji usluga zahtijevaju, prosljeđivati rezultate dobivene iz jednog izvora kao ulazne podatke za drugi izvor te prikupljati završne rezultate pojedinih koraka kako bi se oblikovalo traženo rješenje. Problem nedostatka cjelovitih rješenja za obavljanje složenih poslova ne leži u nedostatku tehnologijskih dostignuća ili nedovoljnom zanimanju pružatelja usluga za ponudu složenih usluga. Srž problema leži u činjenici da s porastom broja usluga i potrošača nije moguće predvidjeti sve moguće potrebe korisnika i sve moguće načine povezivanja pojedinačnih usluga i sadržaja u poosobljene primjenske programe.

Kako bi se potrošačima omogućilo samostalno upravljanje kakvoćom doživljaja tijekom korištenja primjenskih programa, današnji primjenski programi izlažu se u obliku udomljenika (engl. *widgets, web gadgets*) [22]. Udomljenici su primjenski programi s grafičkim korisničkim sučeljem u obliku *World Wide Web* stranica i mogućnošću udomljavanja i izvođenja unutar drugih *World Wide Web* stranica. Stranica na kojoj se udomljenik izvodi naziva se udomiteljskom stranicom. Unutar jedne udomiteljske stranice moguće je udomiti proizvoljni skup udomljenika, čime potrošači dobivaju mogućnost obavljanja složenog posla s jednog mjesta.

Pokretanjem izvođenja i prenošenjem podataka između izabranog skupa udomljenika, potrošač gradi poosobljeni radni tijek kojim postiže elemente kakvoće doživljaja prema vlastitim subjektivnim mjerilima. Izgradnjom poosobljenih radnih tijekova nad udomljenicima, potrošač objedinjuje i nadopunjuje radne tijekove sa svojstvom kakvoće usluge koji su od strane pružatelja usluga definirani nad programskim komponentama. Međutim, primjena skupa udomljenika još uvijek zahtijeva učestalo ručno upravljanje izvođenjem pojedinih udomljenika. Programiranjem prilagođenim potrošaču nastoji se

omogućiti programsko povezivanje skupa udomljenika u novi primjenski program te izlaganje objedinjene funkcionalnosti u obliku novog udomljenika.



Slika 2.2. Primjer korištenja skupa primjenskih programa u obliku udomljenika za obavljanje složenog posla

Slikom 2.2 prikazan je primjer korištenja skupa nezavisnih primjenskih programa izloženih u obliku udomljenika za napredno upravljanje osobnim financijskim portfeljom. U primjeru složenog posla prikazanog na slici, pretpostavlja se da potrošač želi pratiti stanje izabranog skupa dionica na američkoj burzi s mogućnošću dinamičkog dodavanja novih dionica u skup. Neka se za izabranu dionicu iz skupa zahtijeva prikaz cijene u eurima po kojoj se trguje dionicom te grafički prikaz kretanja cijene dionice tijekom tekućeg dana.

S obzirom na to da na tržištu nije dostupan primjenski program koji pruža cjelokupnu funkcionalnost koju potrošač zahtijeva, za obavljanje opisanog posla potrebno je koristiti četiri različita primjenska programa. Neka je svaki od četiri primjenska programa dostupan u obliku udomljenika s grafičkim korisničkim sučeljem prikazanim u pregledniku *World Wide Web* sadržaja.

U prvom koraku, potrošač koristi udomljenik za podešavanje osobnog financijskog portfelja. Korisničko sučelje tog udomljenika omogućava praćenje stanja skupa izabranih dionica na burzi te dodavanje novih dionica u skup izabranih dionica. Dodavanje nove dionice u skup izabranih dionica obavlja se upisivanjem simbola dionice u polje za unos teksta *Symbol* (1.a). Pritiskom na tipku *Add* (1.b), izabrana dionica pojavljuje se u skupu izabranih dionica čije je stanje moguće pratiti u rubrici *My Portfolio* (1.c).

U drugom koraku, potrošač koristi udomljenik za pregled izvještaja o trgovanju izabranom dionicom tijekom tekućeg dana. Korisničko sučelje tog udomljenika omogućava iscrpno praćenje podataka o trgovanju dionicom, uključujući trenutnu, najvišu i najnižu cijenu po kojoj je trgovano dionicom, ukupni promet koji je trgovima dionicom ostvarila tijekom tekućeg dana i slično. Dohvat iscrpnog izvještaja o trgovanju izabranom dionicom dobiva se upisivanjem simbola dionice u polje za unos teksta *Symbol* (2.a) i pritiskom na tipku *Go* (2.b). Uz ostale podatke koji su dobiveni iscrpnim izvještajem o trgovanju dionicom, cijena dionice prikazana je u polju za prikaz teksta pod nazivom *Price* (2.c).

Budući da je cijena dionice kojom se trguje na američkoj burzi izražena u američkim dolarima, pretvorbu cijene dionice u eure potrebno je obaviti primjenom udomljenika za pretvorbu valuta. Taj se dio ukupnog složenog posla obavlja u trećem koraku postupka. Korisničko sučelje udomljenika za pretvorbu valuta potrošaču omogućava izbor polazišne valute iz padajućeg izbornika *From* (3.a) te izbor odredišne valute iz padajućeg izbornika *To* (3.b). Upis iznosa u polazišnoj valuti obavlja se popunjavanjem polja za unos teksta *USD* (3.c), dok je iznos u odredišnoj valuti moguće pročitati iz polja za prikaz teksta *EUR* (3.d).

Nakon što je potrošač dobio informaciju o cijeni dionice u traženoj valuti, zadnji korak postupka je dohvat izvještaja o kretanju cijene izabrane dionice kroz određeno vremensko

razdoblje. U četvrtom koraku postupka, potrošač koristi udomljenik za pregled izvještaja o kretanju cijene dionice jer taj udomljenik pruža mogućnost grafičkog prikaza kretanja cijene dionice. Upisom simbola dionice u polje za unos teksta *Symbol* (4.a), izborom vremenskog razdoblja za prikaz kretanja cijene dionice iz padajućeg izbornika *Interval* (4.b) te pritiskom na tipku *Add* (4.c) dobiva se grafički prikaz kretanja cijene izabrane dionice u izabranom vremenskom razdoblju. Podaci o kretanju cijene dionice prikazani su slikovnim elementom pod nazivom *Graph* (4.d).

Primjer na slici 2.2 prikazuje složenost postupka kada je, u nedostatku cjelovitog rješenja, za obavljanje posla potrebno koristiti više različitih primjenskih programa. Tablicom 2.2 prikazana je usporedba broja potrebnih elemenata korisničkog sučelja i broja potrebnih akcija nad elementima korisničkog sučelja koje je potrebno obaviti nad korisničkim sučeljem skupa nezavisnih udomljenika u odnosu na broj stvarno korisnih elemenata i akcija. Podaci pokazuju da je za upravljanje osobnim financijskim portfeljom potrebno koristiti četrnaest različitih elemenata korisničkog sučelja razasutih po sučeljima četiriju različitih udomljenika. Nad tih četrnaest elemenata korisničkog sučelja, potrebno je obaviti deset različitih akcija.

Tablica 2.2. Usporedba broja potrebnih elemenata korisničkog sučelja i obavljenih akcija u odnosu na broj korisnih elemenata i obavljenih akcija

	Ukupno	**Korisno**	**Međukoraci**
Broj korištenih elemenata	14	5	9
Broj obavljenih akcija	10	2	8

Razmjerno veliki broj elemenata i akcija koje je potrebno obaviti nad elementima korisničkog sučelja posljedica je nepostojanja programa, odnosno udomljenika koji bi u potpunosti obavljao funkciju koja je potrošaču potrebna. Zbog potrebe za korištenjem više različitih udomljenika i prijenosom međurezultata između sučelja pojedinih udomljenika, u postupku se pojavljuje razmjerno veliki broj međukoraka. Od ukupno četrnaest korištenih elemenata korisničkog sučelja, samo njih pet se koristi za zadavanje početnih informacija i prikaz konačnih rezultata koje potrošač želi vidjeti kao rezultate izvođenja programa. Preostalih devet elemenata korisničkog sučelja koristi se za prijenos međurezultata između pojedinih udomljenika. Slična situacija vrijedi i za operacije koje je potrebno obaviti nad elementima korisničkog sučelja pojedinih udomljenika. Od ukupno deset takvih operacija, samo dvije se koriste za zadavanje početnih uvjeta i pokretanje procesa. Preostalih osam operacija koristi se za prijenos međurezultata između udomljenika.

Problem nedostatka cjelovitih rješenja za obavljanje složenih poslova raste s porastom broja potrošača i s porastom učestalosti kojom pojedini potrošač zahtijeva već viđenu i korištenu kombinaciju sadržaja. S porastom učestalosti primjene ovog složenog postupka, raste i nezadovoljstvo potrošača zbog utroška vremena na ponavljajuće aktivnosti i veću mogućnost nastanka pogreške.

Cilj programiranja prilagođenog potrošaču je oblikovanje metodologije programiranja kojom potrošači dobivaju mogućnost iskorištavanja funkcionalnosti dostupnih primjenskih programa i njihovo povezivanje u nove, poosobljene kombinacije prilagođene osobnim mjerilima kakvoće doživljaja. Slikom 2.3 prikazan je primjer poosobljenog primjenskog programa u obliku udomljenika koji obavlja složeni posao prikazan slikom 2.2, a nastaje primjenom programiranja prilagođenog potrošaču.

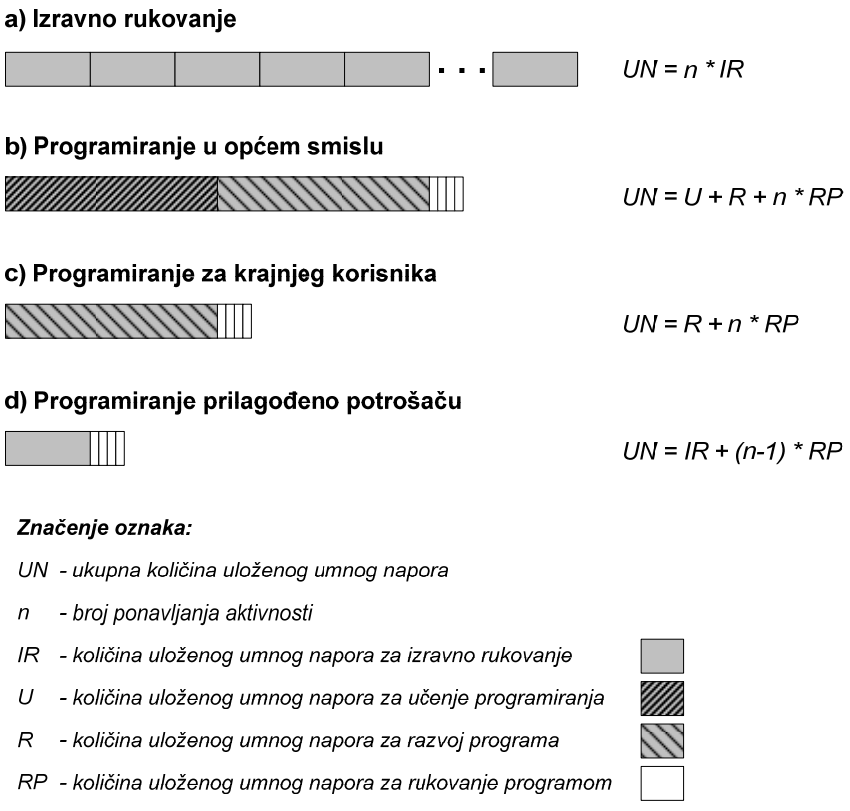
Slika 2.3. Primjer poosobljenog primjenskog programa u obliku udomljenika

Umjesto korištenja informacija razasutih po sučeljima četiriju različitih udomljenika, poosobljenim primjenskim programom cjelokupna je funkcionalnost dostupna na jednom mjestu. Korisničko sučelje poosobljenog primjenskog programa omogućava upravljanje listom izabranih dionica, praćenje cijene izabrane dionice u željenoj valuti te praćenje kretanja cijene dionice tijekom tekućeg dana. Lista izabranih dionica preuzeta je od udomljenika za podešavanje osobnog financijskog portfelja. Cijena po kojoj se trguje dionicom preuzeta je od udomljenika za pregled izvještaja o trgovanju izabranom dionicom tijekom tekućeg dana. Pretvorba cijene dionice izražene u američkim dolarima u iznos izražen u eurima obavljena je primjenom udomljenika za pretvorbu valuta. Naposljetku, grafički prikaz kretanja cijene izabrane dionice tijekom tekućeg dana dobiven je iskorištavanjem udomljenika za pregled izvještaja o kretanju cijene dionice. Cjelokupna

funkcionalnost izložena je u obliku jedinstvenog udomljenika s onim elementima korisničkog sučelja koji su nužni i dovoljni za obavljanje zahtijevane funkcije.

2.3 Kognitivna svojstva programiranja prilagođenog potrošaču

Programiranjem prilagođenim za najširi krug potrošača računala potrebno je zadovoljiti svojstva nenametljivosti i neprimjetnosti kako bi se potrošače potaknulo na upuštanje u postupak izgradnje novih primjenskih programa, ne prisiljavajući ih na usvajanje novih vještina niti promjenu uobičajenih navika tijekom korištenja računala. Poticaj potrošačima za upuštanje u samostalni razvoj primjenskih programa objašnjava se modelom ulaganja umnih sredstava (engl. *attention investment model*) [23]. Modelom ulaganja umnih sredstava opisuje se sveukupni umni napor i umna aktivnost koje je potrebno uložiti za obavljanje određene radnje. Umni napor mjeri se količinom utrošenog vremena, uloženom koncentracijom, prihvaćenim rizikom od neuspjeha i slično.



Slika 2.4. Model ulaganja umnih sredstava tijekom primjene i programiranja računala

Slikom 2.4 prikazani su odnosi količine uloženog umnog napora za nekoliko različitih oblika primjene i programiranja računala. Slučaj a) prikazuje količinu umnog napora koju je

potrebno uložiti za izravno rukovanje računalnim sustavom. Ako potrošač primjenom računala rješava određeni problem koji zahtjeva njegovu stalnu uključenost u postupak rješavanja, onda on ulaže određeni umni napor. Za ponavljajuće poslove, ukupni uloženi umni napor jednak je umnošku broja ponavljanja aktivnosti i količine umnog napora koju je potrebno uložiti za pojedinačno rukovanje sustavom.

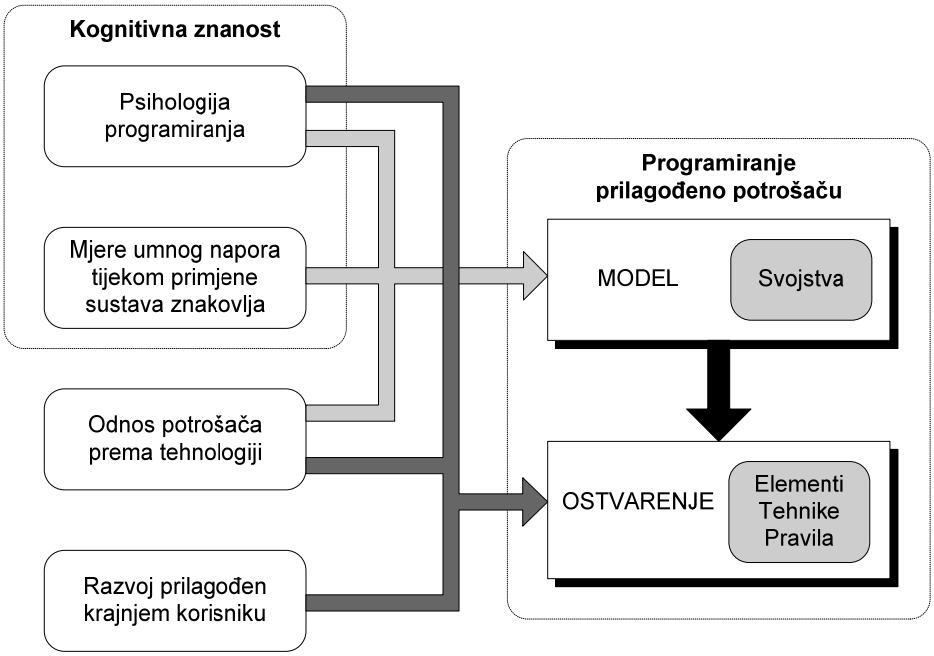
Umni napor koji je potreban za ručno upravljanje izvođenjem računalnog sustava moguće je smanjiti izgradnjom računalnog programa. Količina umnog napora potrebna za izgradnju računalnog programa prikazana je slučajem b) na slici 2.4. Izgradnjom programa, veći dio umnog napora moguće je prenijeti na računalo koje zamjenjuje aktivnosti čovjeka tijekom obavljanja posla. Međutim, za razvoj programa potrebno je prethodno uložiti dodatni umni napor, najprije za učenje programskog jezika, a nakon toga za izgradnju programa u izabranom jeziku. Nakon što je programski jezik naučen i nakon što je izgrađen računalni program, njegovom primjenom značajno se smanjuje količina umnog napora potrebna za rješavanje problema jer je količina umnog napora potrebna za pokretanje i nadzor nad izvođenjem programa značajno manja od količine umnog napora potrebne za izravno rukovanje računalom tijekom čitavog postupka.

U području razvoja prilagođenog krajnjem korisniku koje je prikazano slučajem c) na slici 2.4, umni napor potreban za učenje programiranja nastoji se izbjeći oblikovanjem specijaliziranih programskih jezika za pojedina područja primjene. Smanjivanje umnog napora potrebnog za učenje programskog jezika postiže se primjenom posebno oblikovanih programskih apstrakcija koje su bliske korisnicima u zadanom području primjene. Takvi korisnici, međutim, još uvijek ulažu značajan umni napor u razvoj programa prije nego što počnu ostvarivati korist od njegova razvoja.

Zamisao programiranja prilagođenog potrošaču je da se, osim napora za učenje programskog jezika, u potpunosti izbjegne i umni napor potreban za izgradnju programa. To je moguće ostvariti metodologijom koja ne koristi posebne postupke koji se koriste isključivo za izgradnju programa, već je izgradnja programa posljedica, odnosno popratna pojava uobičajenih postupaka koji se koriste tijekom primjene računala postupkom izravnog rukovanja. Model ulaganja umnog napora u području programiranja prilagođenog potrošaču prikazan je slučajem d) na slici 2.4. Izgradnji programa pristupa se na jednak način kao i u slučaju izravnog rukovanja. Na osnovi radnji potrošača tijekom izravnog rješavanja problema, stvara se računalni program. Na taj je način potrošačima omogućeno da bez posebno uloženog napora grade vlastite primjenske programe koji kasnije zamjenjuju čovjeka tijekom ponovnih rješavanja istog problema.

2.4 Metodologija istraživanja

Istraživanje programiranja prilagođenog potrošaču provedeno je višedisciplinarnim aktivnostima koje su, osim računalnih, obuhvatile i područje kognitivnih znanosti te teorije marketinga i planiranja tržišta. Metodologija istraživanja koja je korištena tijekom izrade doktorske disertacije shematski je prikazana slikom 2.5. Lijeva strana slike prikazuje srodna znanstvena područja koja su istražena kako bi se postavile osnove za oblikovanje programske paradigme prikladne za široki krug potrošača računala i informacijskih usluga. Desna strana slike prikazuje postignute rezultate u području programiranja prilagođenog potrošaču koji su definirani na osnovi zaključaka proizašlih proučavanjem srodnih područja te izvođenjem vlastitih zaključaka.



Slika 2.5. Metodologija istraživanja

Ostvarenje postavljenih ciljeva doktorske disertacije obavljeno je u dva koraka. U prvom koraku uspostavljen je teorijski model programiranja prilagođenog potrošaču. Tim modelom definiran je skup svojstava za ocjenu bliskosti zadane programske paradigme umnom režimu, znanju i iskustvu potrošača. Skup svojstava definiranih teorijskim modelom moguće je iskoristiti za ocjenu i usporedbu postojećih programskih paradigmi ili za prijedlog nove paradigme.

U drugom koraku istraživanja predložena je programska paradigma za ostvarenje programiranja prilagođenog potrošaču koja zadovoljava svojstva definirana teorijskim

modelom. Ostvarenjem paradigme programiranja prilagođenog potrošaču izabran je skup pogodnih elemenata, tehnika i pravila za oblikovanje i izgradnju primjenskih programa od strane potrošača.

Tijekom definiranja modela programiranja prilagođenog potrošaču, korištene su spoznaje iz područja kognitivne znanosti te iskustvena opažanja proizašla iz analize odnosa potrošača prema tehnologiji. U okviru doktorske disertacije su, stoga, proučeni teorijski modeli kognitivne znanosti kojima se opisuju i objašnjavaju umni procesi tijekom učenja, razmišljanja i zaključivanja te njihova primjena u oblikovanju programskih jezika. Posebna grana kognitivne znanosti pod nazivom psihologija programiranja bavi se umnim procesima koji se javljaju tijekom oblikovanja i izgradnje računalnih programa. Posebna pažnja je, stoga, posvećena proučavanju teorijskih modela psihologije programiranja.

Dok se modelima psihologije programiranja objašnjava umna aktivnost čovjeka tijekom programiranja, usporedba umnog opterećenja koje nameće učenje i primjena pojedinih programskih paradigmi i programskih jezika provodi se sustavom mjera umnog napora tijekom primjene sustava znakovlja (engl. *cognitive dimensions of notations*) [24]. Sustav mjera umnog napora tijekom primjene sustava znakovlja je radni okvir za procjenu umnog opterećenja kojem je čovjek izložen tijekom učenja i primjene različitih vrsta znakovlja. U terminologiji programiranja, sustav znakovlja naziva se programskim jezikom. Sustav mjera umnog napora koristi se kako bi se prepoznala ključna svojstva koja određenu programsku paradigmu ili programski jezik čine prikladnim za ciljnu skupinu korisnika, kao i ona koja je tijekom oblikovanja programske paradigme ili jezika potrebno izbjegavati.

Proučavanjem psihologije programiranja i mjera umnog napora tijekom primjene sustava znakovlja, zaključeno je da je dio umne aktivnosti čovjeka moguće objasniti prirođenim osobinama, dok je dio posljedica iskustva. Kako bi se model programiranja prilagodio potrebama, znanjima i iskustvu potrošača, proučen je odnos potrošača prema tehnologiji. U tu svrhu proučeni su modeli iz teorije marketinga i planiranja koji se koriste tijekom plasmana novih tehnologijskih proizvoda ili usluga na tržište. Programiranje prilagođeno potrošaču moguće je smatrati oblikom nove tehnologijske usluge za koju se očekuje podvrgavanje zakonu o prihvaćanju tehnologijskih inovacija među potrošačima. Proučavanjem odnosa potrošača prema tehnologiji, utvrđeno je da je prihvaćanje novih tehnologijskih proizvoda i usluga olakšano i ubrzano ako se one na tržište plasiraju kao unaprjeđenje poznatih i usvojenih koncepata, za razliku od onih koje zahtijevaju korjenite promjene u navikama potrošača ili utjecaju na njihov svakodnevni život. U model programiranja prilagođenog potrošaču su, stoga, uključena svojstva kojima se zahtijeva

oslanjanje na znanja i vještine stečene prethodnom primjenom računalne i informacijsko-komunikacijske tehnologije.

Na osnovi svojstava definiranih teorijskim modelom, predloženo je ostvarenje programske paradigme za potrošaču prilagođeno programiranje. Izbor elemenata, tehnika i pravila koja zadovoljavaju pojedina svojstva potrošaču prilagođenog programiranja izvršen je na osnovi triju skupina podataka: izvođenjem vlastitih zaključaka na osnovi istraživanja kognitivnih znanosti i psihologije programiranja, na osnovi profila potrošača koji ističe njegove potrebe, očekivanja, znanja i vještine tijekom rukovanja računalnim i informacijskim uslugama, a izgrađen je primjenom analize odnosa potrošača prema tehnologiji te na osnovi analize i usporedbe programskih paradigmi koje se koriste u području razvoja prilagođenog krajnjem korisniku.

2.5 Organizacija doktorske disertacije

Sadržaj doktorske disertacije organiziran je u skladu s metodologijom istraživanja koja je prikazana slikom 2.5. U trećem poglavlju opisana je sprega programiranja i kognitivne znanosti. Tim su poglavljem obuhvaćeni teorijski modeli iz područja psihologije programiranja i mjere umnog napora tijekom primjene sustava znakovlja. Opisana je teorija održavanja i preispitivanja znanja, teorija kratkoročnog pamćenja privremenih činjenica, teorija višeprolaznog iščitavanja te teorija jezične predodređenosti kojom se objašnjava utjecaj govornog jezika na doživljaj vremena.

Četvrto poglavlje donosi analizu odnosa potrošača prema tehnologiji i tehnologijskim inovacijama te utjecaj tog odnosa na područje programiranja prilagođenog potrošaču. Opisana je vremenska ovisnost priljeva novih korisnika proizvoda ili usluga s obzirom na vrstu tehnologijske inovacije. Posebno su analizirani slučajevi nadovezujuće i korjenite inovacije. Nadalje, prikazano je smanjenje složenosti međudjelovanja čovjeka i računala kroz povijest razvoja računalnih sustava, uz istodobni porast vještina potrošača uvjetovan sve raširenijom primjenom računalnih programa. Zaključeno je da su tablični proračuni najčešće korištena vrsta primjenskih programa za osobna računala, da je grafičko korisničko sučelje prikazano u pregledniku *World Wide Web* sadržaja prevladavajući oblik međudjelovanja potrošača i računala te da su udomljenici potrošaču bliski elementi od kojih je moguće graditi poosobljene primjenske programe. Ovi zaključci iskorišteni su za oblikovanje modela i prijedlog ostvarenja programiranja prilagođenog potrošaču.

U petom poglavlju dan je pregled dosadašnjih dostignuća u području razvoja prilagođenog krajnjem korisniku. Opisano je programiranje primjenom grafičkih oznaka,

programiranje primjenom tabličnih proračuna, programiranje primjenom obrazaca, programiranje na pokaznom primjeru i programiranje prirodnim jezikom kao osnovni razredi postupaka koji se koriste u tom području razvoja programskih sustava. Istaknuti su razvojni alati koji su tipični predstavnici pojedinog razreda. Uspoređena su svojstva pojedinih razreda s obzirom na prikladnost za primjenu od strane potrošača. Istaknute su glavne prepreke koje programiranje za krajnjeg korisnika još uvijek čine neprikladnim za potrošača te su izdvojeni elementi i načela koji su pogodni za iskorištavanje tijekom oblikovanja programske paradigme prilagođene potrošaču.

Model programiranja prilagođenog potrošaču opisan je u šestom poglavlju. Poglavlje započinje uspostavom kvalitativnog teorijskog modela za usporedbu kognitivnih svojstava programskih elemenata i tehnika programiranja koje se koriste u području programiranja prilagođenog potrošaču s ostalim programskim paradigmama i jezicima. U nastavku poglavlja definiran je radni okvir sa skupom svojstava za ostvarenje programiranja prilagođenog potrošaču. Usporedno s definiranjem radnog okvira, izvršena je ocjena i usporedba postojećih razvojnih alata namijenjenih korisnicima koji nisu školovani za razvoj računalnih programa.

U sedmom, osmom, devetom, desetom i jedanaestom poglavlju opisana je programska paradigma za usložnjavanje udomljenika (engl. *widget composition*) kojom se ostvaruje programiranje prilagođeno potrošaču. U predloženom ostvarenju potrošaču prilagođene programske paradigme, udomljenici (engl. *widgets, web gadgets*) su iskorišteni u svojstvu programskih blokova za izgradnju primjenskih programa, dok su radnje nad elementima grafičkih korisničkih sučelja iskorištene za povezivanje skupa udomljenika u radni tijek primjenskog programa. Pokazana je istovjetnost programirljivih svojstava programskih i grafičkih korisničkih sučelja.

U dvanaestom poglavlju definiran je matematički model za formalni opis primjenskih programa izgrađenih od strane potrošača primjenom programske paradigme za usložnjavanje udomljenika. Formalni opis primjenskih programa zasnovan je na teoriji grafova. Definiran je skup grafova kojima se opisuje strukturna, podatkovna i upravljačka povezanost udomljenika u poosobljenim potrošačkim programima.

Kako bi se stvaralačke mogućnosti programske paradigme za usložnjavanje udomljenika usporedile s ostalim oblicima programskih paradigmi, u trinaestom je poglavlju definirana mjera *stvaralačkih mogućnosti programske paradigme*. Pokazano je da zbog kratkog vremena potrebnog za razvoj programa i velikog broja korisnika kojima je omogućeno sudjelovanje u njihovu razvoju, stvaralačke mogućnosti programske paradigme

za usložnjavanje udomljenika mnogostruko premašuju stvaralačke mogućnosti strojnog programiranja te objektno-orijentiranih i primjenski usmjerenih programskih paradigmi. Izvršene su procjene budućeg porasta broja udomljenika te je pokazana potpunost paradigme usporedbom s modelom Turingovog stroja. Zaključci doktorske disertacije, otvorena pitanja i smjernice za nastavak istraživanja izneseni su u četrnaestom poglavlju.

3 Programiranje i kognitivna znanost

Kognitivna znanost (engl. *cognitive science*) [25, 26, 27] je znanstvena disciplina koja se bavi istraživanjem ljudskog uma, umnih funkcija, svojstava umnih procesa i umnog djelovanja te čovjekove inteligencije. S obzirom na to da se programiranje smatra umno zahtjevnom ljudskom aktivnošću koja iziskuje visoku razinu usredotočenosti, pamćenja, zaključivanja i predviđanja budućih događaja, spoznaje iz područja kognitivne znanosti u računarstvu se primjenjuju tijekom oblikovanja programskih jezika. Grana kognitivne znanosti koja proučava umne aktivnosti svojstvene za programiranje naziva se psihologijom programiranja (engl. *psychology of programming*) [28, 29].

U ovom poglavlju opisani su teorijski modeli kognitivne znanosti kojima se objašnjava umna aktivnost čovjeka tijekom programiranja. Na osnovi teorijskih modela procjenjuje se umni napor kojem je programer izložen tijekom izrade računalnog programa. U okviru doktorske disertacije, prikazani teorijski modeli iskorišteni su za analizu i prepoznavanje ključnih svojstava koje programska paradigma i programski jezik moraju zadovoljiti kako bi bili prikladni za primjenu od strane širokog kruga potrošača računalnih sustava.

U poglavlju 3.1 opisano je područje istraživanja kognitivne znanosti te povijesni razvoj od vremena starogrčkih filozofa do suvremene znanstvene discipline. U poglavlju 3.2 opisana je primjena kognitivne znanosti u računarstvu. Poglavlja 3.3 i 3.4 donose pregled teorijskih modela psihologije programiranja kojima se objašnjava umna aktivnost tijekom oblikovanja, izgradnje i tumačenja značenja računalnih programa. U poglavlju 3.5 prikazan je sustav mjera umnog napora tijekom primjene sustava znakovlja kao širokoprihvaćeni

radni okvir za ocjenu kognitivnih svojstava programskih jezika, razvojnih alata i tehnika programiranja. U poglavlju 3.6 prikazan je utjecaj govornog jezika na način izražavanja vremenskih odnosa u računalnim programima.

3.1 Područje i metode istraživanja u kognitivnoj znanosti

Razvoj suvremene kognitivne znanosti započeo je sredinom pedesetih godina dvadesetog stoljeća kada su znanstvenici počeli primjenjivati znanstvene metode i razvijati teorijske modele o načinima ljudskog pamćenja, razmišljanja, učenja i stjecanja znanja. Primjenjujući složene načine predstavljanja činjenica i postupke izračunavanja kojima je moguće opisati misaone procese učenja i zaključivanja, kognitivna znanost gradi modele koji vjerodostojno objašnjavaju i predstavljaju umnu aktivnost čovjeka. Suvremena kognitivna znanost je višedisciplinarno znanstveno područje koje koristi znanstvene spoznaje iz područja psihologije, filozofije, neuroznanosti, lingvistike, antropologije, računalne znanosti, biologije i fizike.

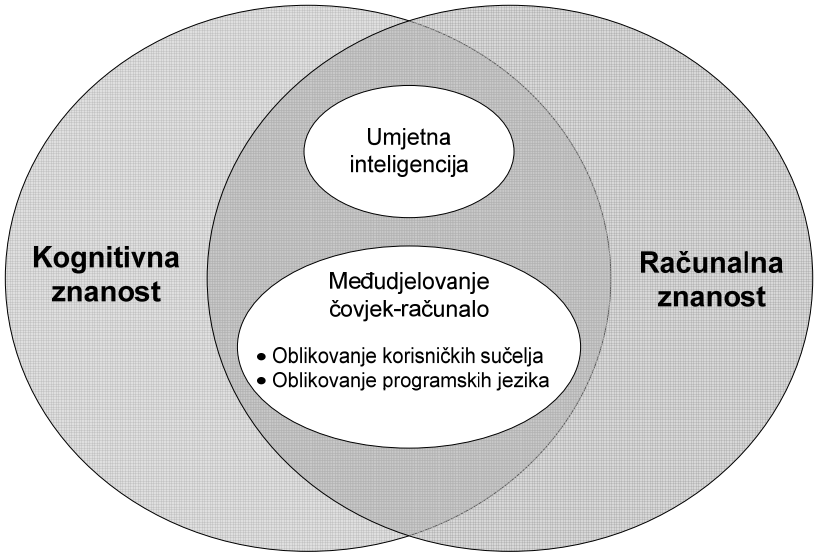
Iako je suvremena kognitivna znanost relativno mlado znanstveno područje, pokušaji razjašnjavanja umnih procesa poput pamćenja, mišljenja, učenja i zaključivanja sežu u vrijeme starogrčke civilizacije. Starogrčki filozofi Platon i Aristotel pokušavali su objasniti prirodu ljudskog znanja te postupke učenja i zaključivanja. Proučavanje umnih aktivnosti nije, međutim, doživjelo značajan znanstveni pomak sve do devetnaestog stoljeća i pojave eksperimentalne psihologije. Eksperimentalna psihologija proširila je dotada uobičajen teorijski pristup objašnjavanju psiholoških pojava kod živih bića na način da se psihologija počela promatrati kao prirodna znanost za koju je uobičajeno da se teorijske spoznaje potvrđuju pokusnim ispitivanjima. Primjenom pokusnih ispitivanja i promatranja psiholoških osobina ljudi, započelo je razdoblje znanstvenog pristupa istraživanju umnih aktivnosti čovjeka i razvoj kognitivne znanosti kao znanstvene discipline. Pojam kognitivne znanosti u široku je primjenu uveo britanski znanstvenik Christopher Longuet-Higgins 1973. godine [25]. Longuet-Higgins se bavio teorijskom kemijom, ali je najznačajnije znanstvene rezultate postigao u području umjetne inteligencije dok je radio na Sveučilištu u Edinburghu u Škotskoj.

Suvremena kognitivna znanost snažno se oslanja na spoznaje dobivene pokusnim mjerenjima i opažanjima koja se provode na ljudima primjenom psiholoških ispitivanja. Istraživanja u području kognitivne znanosti zasnivaju se na promatranju, proučavanju i analizi vanjskog doživljaja umnih aktivnosti čovjeka te na objašnjavanju procesa koji se odvijaju u mozgu. Polazište kognitivne znanosti jest pretpostavka da je umnu aktivnost

čovjeka moguće opisati i objasniti podatkovnim strukturama za predstavljanje činjenica te računskim postupcima sličnim računalnim algoritmima za rukovanje tim strukturama. Prema središnjoj i od većine znanstvenika prihvaćenoj teoriji umnog djelovanja, ljudski um se sastoji od umnih struktura za predstavljanje znanja u obliku pojmova (engl. *concepts*), slika (engl. *images*), logičkih formula (engl. *logical propositions*), pravila (engl. *rules*) i sličnosti s drugim pojmovima (engl. *analogies*), dok se za rukovanje umnim strukturama primjenjuju postupci pretraživanja (engl. *search*), uspoređivanja (engl. *matching*), razlučivanja (engl. *deduction*) te preispitivanja i popravljanja (engl. *revision and retrieval*). Na osnovi rezultata pokusnih ispitivanja, razvijeno je nekoliko teorijskih modela koji opisuju umne strukture za predstavljanje činjenica i povezuju ih s računskim postupcima koji njima upravljaju. Dva najznačajnija teorijska modela za objašnjavanje umnih struktura i umnog djelovanja su teorija umnih poveznica (engl. *connectionist theory*) [30, 31] i teorija ponašajnih modela (engl. *behaviorism*) [32, 33].

3.2 Primjena kognitivne znanosti u računarstvu

Spoznaje iz područja kognitivne znanosti već duže vrijeme pronalaze značajnu primjenu u računalnoj znanosti. Slikom 3.1 istaknuta su dva osnovna pravca primjene kognitivne znanosti u oblikovanju i izgradnji računalnih sustava. Spoznaje o načinima umnog djelovanja čovjeka u računalnoj se znanosti primjenjuju u području umjetne inteligencije (engl. *artificial intelligence*) i u području međudjelovanja čovjeka i računalnog sustava (engl. *human-computer interaction*).



Slika 3.1. Dva najznačajnija pravca primjene kognitivne znanosti u računalnoj znanosti: umjetna inteligencija i međudjelovanje čovjek-računalo

Na osnovi modela koji opisuju umne strukture za predstavljanje te umne postupke za rukovanje urođenim i stečenim ljudskim znanjem, grade se umjetni modeli ljudskog uma u obliku računalnih sustava kojima je u određenoj mjeri moguće nadomjestiti inteligenciju čovjeka. Računalni sustavi koji su sposobni igrati umno zahtjevne društvene igre poput šaha ili kartaških igara, sustavi za prepoznavanje rukopisa i govora te sustavi za samostalno djelovanje u nepredvidljivim kriznim okolnostima primjeri su sprege kognitivne znanosti i umjetne inteligencije.

Primjena spoznaja iz područja kognitivne znanosti u grani računarstva koja se bavi međudjelovanjem čovjeka i računalnog sustava koristi se za osmišljavanje postupaka koji smanjuju umni napor čovjeka tijekom korištenja računalnog sustava. Dva najznačajnija pravca istraživanja u području međudjelovanja čovjeka i računalnog sustava sa značajnim osloncem na spoznaje iz područja kognitivne znanosti su oblikovanje korisničkih sučelja (engl. *user interface design*) te oblikovanje i vrednovanje programskih jezika (engl. *programming language design and evaluation*). Svrha primjene spoznaja iz područja kognitivne znanosti u oblikovanju korisničkih sučelja i programskih jezika je približavanje doživljaja računalnog sustava prirodi umnog djelovanja čovjeka.

Grana kognitivne znanosti koja istražuje umnu aktivnost čovjeka tijekom oblikovanja računalnih programa naziva se psihologijom programiranja (engl. *psychology of programming*). S obzirom da je predmet ove doktorske disertacije oblikovanje programskog jezika za široki krug korisnika računalnih sustava, nastavak ovog poglavlja donosi pregled dosadašnjih spoznaja iz područja psihologije programiranja koje su važne u postupcima oblikovanja i vrednovanja programskih jezika.

3.3 Psihologija programiranja

Psihologija programiranja je grana kognitivne znanosti poduprta rezultatima istraživanja u području eksperimentalne psihologije koja se bavi proučavanjem utjecaja pisanja i tumačenja računalnih programa na umno opterećenje čovjeka. Znanstvenici u području psihologije programiranja bave se razvijanjem postupaka za praćenje i mjerenje umnog režima čovjeka (engl. *human performance*) za vrijeme rješavanja problema primjenom različitih vrsta znakovlja (engl. *notation*), kao što su slike, skice, dijagrami ili slovni zapisi. Budući da se značajan dio istraživanja oslanja na pokusna ispitivanja i iskustvena opažanja na nizu ispitanika uključenih u postupke programiranja, psihologija programiranja u literaturi se često navodi i pod pojmom iskustvenog istraživanja programiranja (engl. *empirical study of programming*).

Glavnina istraživanja u području psihologije programiranja zasniva se na pokusnim ispitivanjima i opažanjima koja se provode nad malim skupinama ispitanika koji se podvrgavaju rješavanju dobro definiranog problema unutar zadanih ograničenja [34]. Uobičajeno je da se različitim skupinama ciljano odabranih ispitanika zadaje sličan problem, ali s različitim vrstama ograničenja, kako bi dobiveni rezultati isticali upravo promatrani element umnog režima. Mjerilo za vrednovanje umnog režima ispitanika je vrijeme potrebno za rješavanje problema, postignuta razina točnosti prikazanog rješenja, povratno mišljenje ispitanika i slično. Rezultati pokusnih ispitivanja podvrgavaju se statističkoj obradi na način da se u obzir uzimaju zbirni rezultati ispitne skupine, dok se pojedinačni rezultati ispitanika zanemaruju. Zaključivanje na osnovi objedinjenih rezultata ispitne skupine vjerodostojnije ističe značajke umnog djelovanja koje su zajedničke ciljnoj skupini korisnika. Na osnovi sustavno obrađenih rezultata ispitivanja, osmišljavaju se modeli i postupci za vrednovanje znakovlja s obzirom na prikladnost za rješavanje zadanog problema.

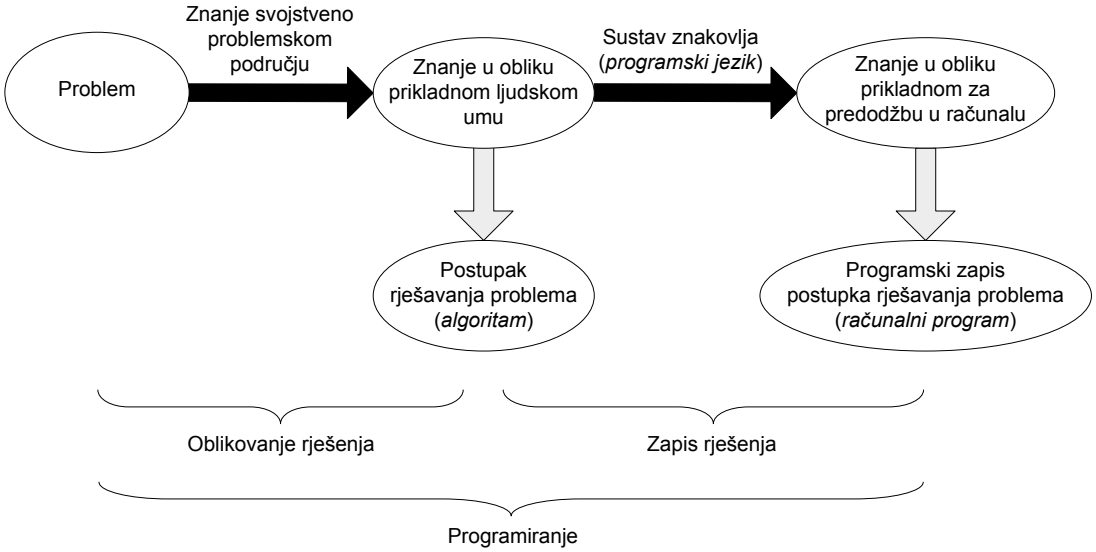
Istraživanja u području psihologije programiranja i srodnih znanstvenih područja u posljednjih tridesetak godina ukazala su na brojne važne elemente koji određuju mjeru uporabljivosti programskih jezika i razvojnih alata. Slijedeći smjernice i preporuke psihologije programiranja tijekom osmišljavanja programskog jezika i oblikovanja razvojnog alata, opis programskog rješenja zadanog problema trebao bi biti blizak prirodnom načinu razmišljanja čovjeka tijekom rješavanja istog problema. Međutim, usprkos znanstveno potvrđenim rezultatima, u osmišljavanju programskih jezika te preporuke se često zanemaruju. U novim su programskim jezicima, stoga, nerijetko iznova prisutne odavno prepoznate prepreke njihovoj uporabljivosti od strane ciljne skupine korisnika.

Predmet ove doktorske disertacije je oblikovanje programskog jezika za široki krug potrošača računalnih sustava koji razumiju prirodu problema koji rješavaju, ali nisu stručnjaci u području programskog inženjerstva. U okviru disertacije su, stoga, proučeni teorijski modeli iz područja psihologije programiranja koji su primjenjivi za najširu skupinu korisnika računalnih sustava. Te su spoznaje kasnije iskorištene za oblikovanje programske paradigme i programskog jezika prilagođenog potrošačima.

3.3.1 Umna aktivnost čovjeka tijekom programiranja

Programiranje računalnih sustava (engl. *computer programming*) definira se kao postupak prilagodbe ponašanja računalnog sustava potrebama korisnika. Ponašanje računalnog sustava određuje čovjek pisanjem računalnog programa. Programiranje je, stoga, proces pretvorbe znanja iz oblika prikladnog ljudima u oblik prikladan za predodžbu u

računalu. Slika 3.2 prikazuje model postupka programiranja kao tijek umnih aktivnosti i dobivenih međukoraka na putu od postavljenog problema do programskog zapisa rješenja.



Slika 3.2. Razlaganje postupka programiranja na fazu traženja i oblikovanja rješenja te fazu zapisa rješenja u zadanom sustavu znakovlja

Postupak programiranja čine dvije faze: faza oblikovanja rješenja i faza zapisa rješenja. Tijekom faze oblikovanja rješenja, programer računalnog sustava na osnovi znanja o problemskom području koje zahvaća postavljeni problem pronalazi prikladan način za njegovo rješavanje. Uvažavajući ograničenja unutar kojih je potrebno tražiti rješenje problema, oblikuje se postupak rješavanja koji uključuje skup elementarnih koraka te njihovo vremensko uređenje. U terminologiji programskog inženjerstva, postupak rješavanja problema naziva se algoritmom (engl. *algorithm*). Pronađeni postupak rješavanja problema odraz je umne aktivnosti čovjeka, odnosno tijeka razmišljanja programera koji rješava postavljeni problem.

Nakon faze oblikovanja rješenja, slijedi faza zapisa rješenja u zadanom sustavu znakovlja. U terminologiji programskog inženjerstva, sustav znakovlja za zapis postupka rješavanja problema naziva se programskim jezikom (engl. *programming language*). Primjenom sustava znakovlja, znanje o rješavanju problema se iz oblika prikladnog ljudskom umu prevodi u oblik prikladan za predodžbu računalnom sustavu. Zapis postupka rješavanja problema u sustavu znakovlja koji je razumljiv računalnom sustavu naziva se računalnim programom (engl. *computer program*).

Faza oblikovanja i faza zapisa rješenja iziskuju različite vrste umnog napora. Tijekom faze oblikovanja rješenja, programer koristi stečena znanja koja su svojstvena problemskom

području kojemu pripada postavljeni problem. Pronalazak algoritamskog rješenja za postavljeni problem predstavlja iznimno zahtjevan umni zadatak jer pripada skupini slabo ustrojenih problema (engl. *ill-structured problems*) za koje ne postoje strogo definirani postupci rješavanja [34]. Umno opterećenje za vrijeme programiranja, međutim, dodatno otežava činjenica da je nakon oblikovanja rješenja potrebno uložiti dodatni umni napor kako bi se to rješenje izrazilo u zadanom sustavu znakovlja, odnosno u programskom jeziku za odabrani računalni sustav. Taj dodatni umni napor moguće je smanjiti oblikovanjem prikladnog programskog jezika u kojem se tijekom faze zapisa rješenja koriste ista ili slična načela i koncepti kao i tijekom faze oblikovanja rješenja.

Programiranje i izravno rukovanje

Oblikovanje i zapis računalnog programa je izrazito zahtjevna umna aktivnost jer se tijekom programiranja gubi mogućnost izravnog rukovanja (engl. *direct manipulation*) sredstvima kojima upravlja napisani program. Napisani program izvodi se na računalnom sustavu u nekom budućem trenutku. Za razliku od programiranja, učinci izravnog rukovanja vidljivi su odmah po završetku obavljene operacije. Primjerice, ispravljanje tipkarskih pogrešaka u programu za obradu teksta moguće je provesti izravnim rukovanjem posežući za ručnim pregledavanjem dokumenta i ručnim ispravljanjem uočenih pogrešaka. Postizanje istog učinka primjenom postupka programiranja zahtijeva definiranje uvjeta za provedbu operacije pretraživanja teksta te zamjene pronađenog teksta zamjenskim uzorcima. Predviđanje svih mogućih uzroka nastanka tipkarskih pogrešaka nije uvijek jednostavno utvrditi unaprijed pa stoga ni učinak izvođenja napisanog programa ne mora nužno odgovarati očekivanim rezultatima. Prednost postupka programiranja, međutim, leži u činjenici da je jednom napisani program moguće izvoditi neograničeni broj puta bez potrebe za posredovanjem čovjeka, dok postupak izravnog rukovanja iziskuje stalnu prisutnost rukovatelja.

Usporedba postupaka izravnog rukovanja i programiranja te vrednovanje njihove prikladnosti u zadanim okolnostima prikazuje se modelom isplativosti uložene umne aktivnosti (engl. *attention investment model*) [23]. Osnovna zamisao tog modela jest vrednovanje dobivenih učinaka u odnosu na količinu i stupanj umnog napora koji je potrebno uložiti da bi se problem riješio primjenom izabranog postupka. Tablica 3.1 prikazuje usporedbu mjerljivih elemenata izravnog rukovanja i programiranja prikazane modelom isplativosti uložene umne aktivnosti.

Tablica 3.1. Usporedba mjerljivih elemenata izravnog rukovanja i programiranja primjenom modela isplativosti uložene umne aktivnosti

	Izravno rukovanje	**Programiranje**
Usredotočenost	osrednja	visoka
Rizik pogreške	nizak	visoki
Isplativost	niska	visoka

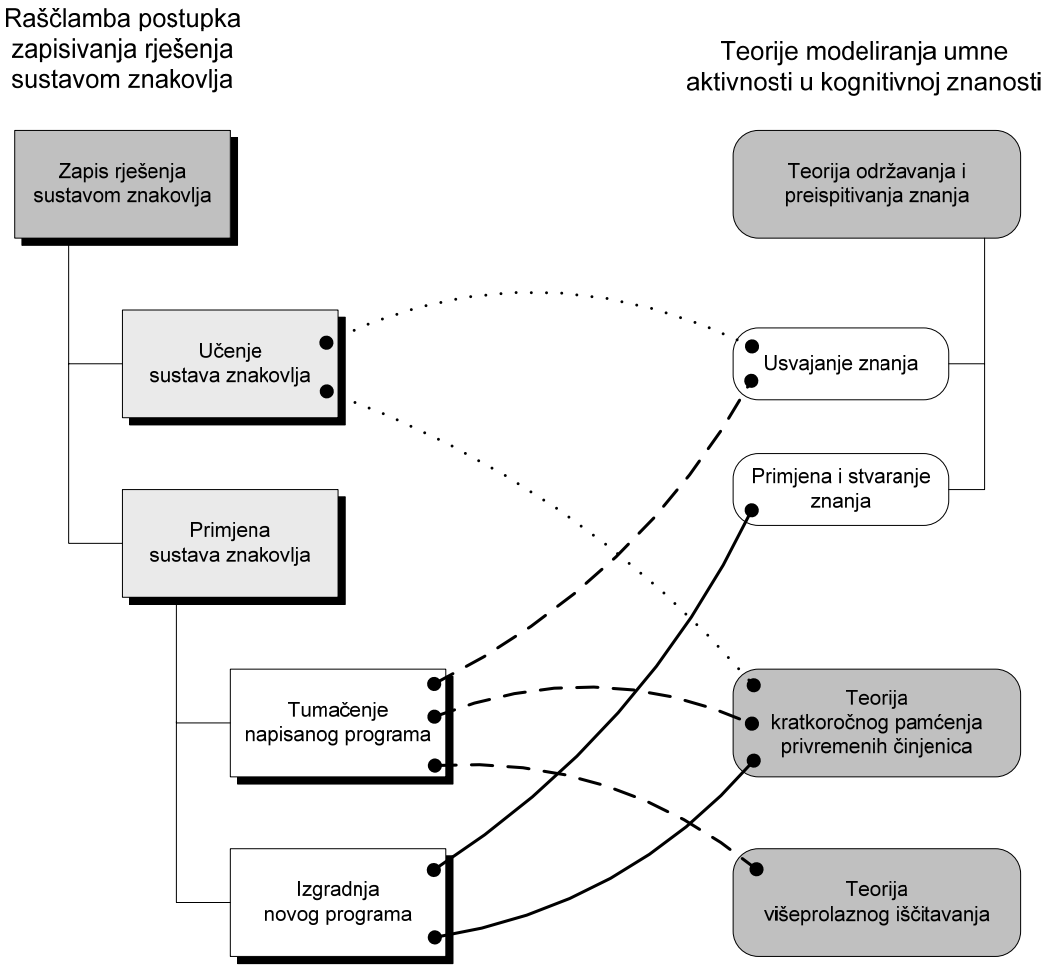
Budući da su učinci izravnog rukovanja vidljivi neposredno nakon izvršene operacije, izravno rukovanje dozvoljava smanjenu razinu usredotočenosti na postupak upravljanja računalnim sustavom, umanjuje rizik učinjene pogreške, ali i isplativost postupka upravljanja zbog nužne prisutnosti rukovatelja. Za razliku od izravnog rukovanja, programiranje pruža višu razinu isplativosti postupka upravljanja jer ne zahtijeva prisutnost rukovatelja za vrijeme izvođenja napisanog programa, ali zahtijeva unaprijedno prepoznavanje svih mogućih načina izvođenja programa i predviđanje svih mogućih vanjskih utjecaja. Zbog toga programiranje zahtijeva visoku razinu usredotočenosti programera uz neizbježan visoki rizik od nastanka pogreške.

3.4 Teorijski modeli psihologije programiranja i njihova primjena u oblikovanju programskih jezika

Istraživanjem umno najprihvatljivijih načina rješavanja problema u pojedinim problemskim područjima bave se posebne grane kognitivne znanosti. Slika 3.3 prikazuje vezu pojedinih koraka koji se javljaju tijekom postupka programiranja s teorijskim modelima psihologije programiranja kojima se objašnjava umna aktivnost programera. Psihologija programiranja pokušava odgovoriti na pitanje kako pristupiti oblikovanju programskih jezika kako bi faza zapisa algoritamskog rješenja smanjila umno opterećenje programera. U istraživanju umne aktivnosti programera tijekom faze zapisa programskog rješenja, proučavaju se umni režimi za vrijeme učenja sustava znakovlja te umni režimi za vrijeme primjene naučenog sustava znakovlja za programski zapis rješenja problema. Tijekom primjene naučenog sustava znakovlja, s kognitivnog gledišta dodatno se razlikuju postupci tumačenja napisanih programa i postupci izgradnje novih programa.

Učenje sustava znakovlja je s gledišta kognitivne znanosti jedan od pojavnih oblika općenitijeg procesa učenja u svrhu stjecanja novog znanja. Primjena naučenog sustava znakovlja tijekom izgradnje računalnih programa pojavni je oblik procesa zaključivanja na osnovi stečenog znanja u svrhu izvođenja novih činjenica. Umnu aktivnost čovjeka tijekom

učenja u svrhu stjecanja novog znanja objašnjava teorija održavanja i preispitivanja znanja (engl. *constructivist learning theory*) [35]. Pritom je za učenje sustava znakovlja značajan teorijski model koji objašnjava umnu aktivnost tijekom usvajanja novog znanja, dok je za primjenu sustava znakovlja značajan model koji objašnjava umnu aktivnost tijekom primjene stečenog znanja.



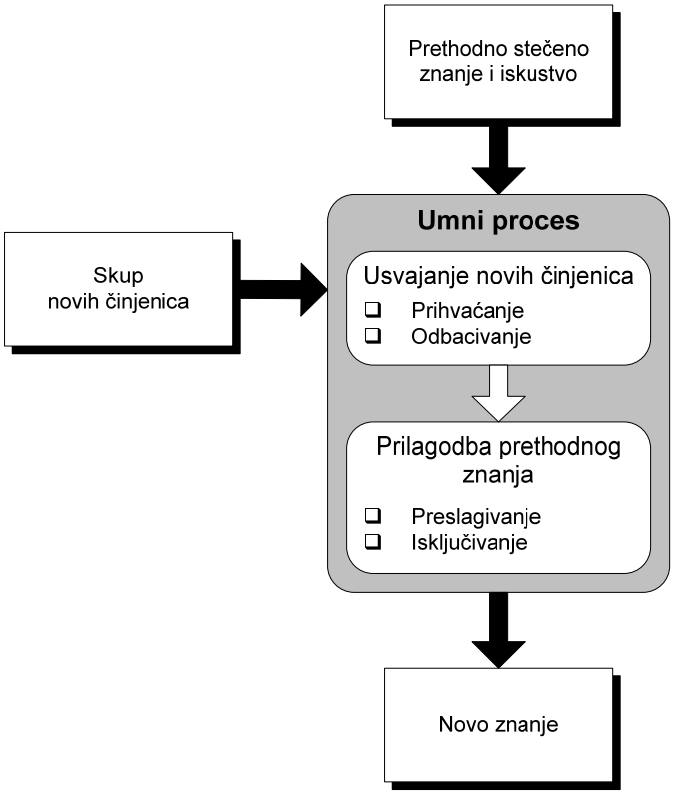
Slika 3.3. Veza pojedinih koraka u postupku zapisivanja rješenja problema sustavom znakovlja s teorijskim modelima za modeliranje umne aktivnosti programera

Tijekom tumačenja i pisanja računalnih programa za koje su svojstveni umni postupci pretraživanja, preslagivanja i zaključivanja uočena je pojava kratkoročnog pamćenja privremenih međurezultata i krajnjih rezultata. Ta se pojava objašnjava teorijom kratkoročnog pamćenja privremenih činjenica (engl. *short-term working memory*) [39]. Dodatno, tijekom tumačenja napisanog programa uočena je pojava postupnog produbljivanja znanja o značenju programa. Programer kreće od pokušaja razumijevanja međusobnih odnosa velikih programskih cjelina prema razumijevanju pojedinosti unutar pojedine cjeline.

Postupni pristup razjašnjavanju značenja napisanog programa objašnjava se teorijom višeprolaznog iščitavanja (engl. *multi-stage theory*) [41]. S obzirom da je primjena stečenog znanja umno zahtjevnija aktivnost u usporedbi sa stjecanjem novog znanja, pisanje novih programa u pravilu nameće veće umno opterećenje od tumačenja napisanih programa.

3.4.1 Teorija održavanja i preispitivanja znanja

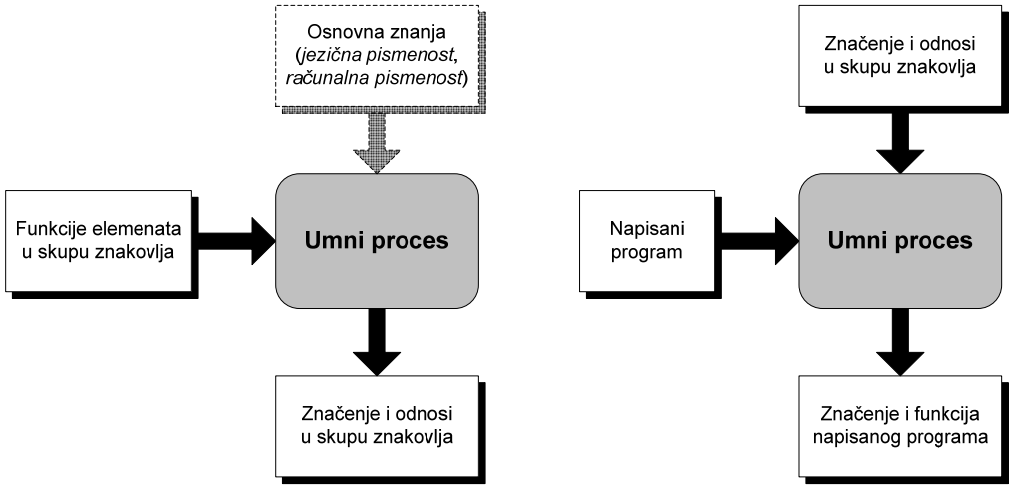
Teorija održavanja i preispitivanja znanja (engl. *constructivist learning theory*) [35] jedan je od najčešće primjenjivanih teorijskih modela za objašnjavanje umnih procesa tijekom učenja, odnosno usvajanja novih znanja. Tijekom oblikovanja programskog jezika, teorija održavanja i preispitivanja znanja koristi se za procjenu umnog napora koji je potreban za učenje oblikovanog jezika te za tumačenje značenja programa izgrađenih primjenom tog jezika. Slika 3.4 prikazuje teorijski model kojim se opisuje postupak stvaranja novog znanja na osnovi prethodno stečenog znanja.



Slika 3.4. Teorijski model stvaranja novog znanja na osnovi prethodno stečenog znanja

Umni proces tijekom stjecanja novog znanja sastoji se od faze usvajanja novih činjenica (engl. *assimilation*) i faze prilagodbe prethodno stečenog znanja novim spoznajama (engl. *accommodation*) [35]. Tijekom faze usvajanja novih činjenica, provodi se vrednovanje novih činjenica s obzirom na vjerovanje u njihovu istinitost. Ta se faza opisuje postupcima

prihvaćanja onih činjenica koje se smatraju istinitima te postupcima odbacivanja ostalih činjenica za koje se na osnovi prethodnog znanja i iskustva vjeruje da nisu istinite. Tijekom faze prilagodbe prethodnog znanja, provodi se usklađivanje prethodnog znanja s novonaučenim činjenicama. Umna aktivnost u ovoj fazi opisuje se postupcima preslagivanja i isključivanja. Prihvaćanjem novih činjenica, dio prethodnog znanja poprima novi oblik. Preslagivanjem činjenica koje čine prethodno stečeno znanje provodi se ažuriranje stečenog znanja u skladu s informacijama koje nose novostečene činjenice. Pridruživanjem novonaučenih činjenica, moguće je da dio prethodno naučenog znanja postane netočan ili zastario. Postupkom isključivanja provodi se uklanjanje netočnih ili zastarjelih činjenica iz skupa prethodno stečenih znanja.



a) učenje sustava znakovlja **b) tumačenje napisanih programa**

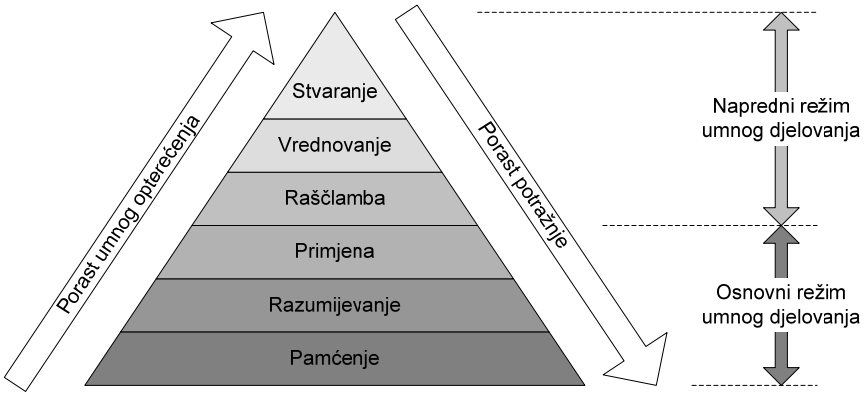
Slika 3.5. Objašnjavanje umne aktivnosti tijekom učenja sustava znakovlja i tumačenja napisanih programa primjenom teorije održavanja i preispitivanja znanja

Slikom 3.5 prikazana je primjena teorije održavanja i preispitivanja znanja u postupcima učenja sustava znakovlja te tumačenja napisanih programa. Tijekom učenja sustava znakovlja, skup novih činjenica predstavlja skup elemenata koji čine sustav znakovlja te funkcije pojedinih elemenata u skupu. Rezultat umnog djelovanja, odnosno novostečeno znanje, predstavlja naučeno značenje sustava znakovlja predočeno definicijama odnosa pojedinih elemenata u skupu. Model prikazan slikom prikazuje da za postupak učenja sustava znakovlja nije potrebno prethodno stečeno znanje, što programeru omogućava započinjanje postupka učenja novog programskog jezika. Međutim, s obzirom da su osnovna znanja koja nisu svojstvena za postupak programiranja, poput jezične ili računalne pismenosti, ipak neophodna, ta su znanja prikazana isprekidano-omeđenim pravokutnikom na slici.

Tijekom tumačenja napisanih programa, skup novih činjenica predstavlja napisani program čije značenje je potrebno objasniti. Rezultat umnog djelovanja u ovom je slučaju objašnjenje značenja i funkcije napisanog programa. Umno djelovanje tijekom postupka tumačenja programa zasnovano je na poznavanju značenja elemenata i odnosa među elementima u sustavu znakovlja. To znanje je stečeno tijekom prethodnog postupka učenja sustava znakovlja, dok tijekom postupka tumačenja napisanog programa ono predstavlja prethodno stečeno znanje.

Bloomova hijerarhija umnog djelovanja

Osim objašnjavanja umnih procesa koji se javljaju tijekom učenja, teorija održavanja i preispitivanja znanja objašnjava i one umne procese koji se javljaju tijekom zaključivanja, jer je zaključivanje ključni element u procesu stjecanja znanja. Američki znanstvenik Benjamin Bloom (1913.-1999.), koji se bavio istraživanjem obrazovne psihologije i dao značajan doprinos razumijevanju postupaka za ovladavanje učenjem i razvoj darovitosti, postavio je 1956. godine teoriju o šest razina umnog djelovanja čovjeka. Bloomovu hijerarhiju umnog djelovanja (engl. *Bloom's hierarchy*, *Bloom's levels of learning*, *Bloom's level of thinking*) koja je prikazana slikom 3.6 čine postupci pamćenja, razumijevanja, primjene, raščlambe, vrednovanja i stvaranja.



Slika 3.6. Bloomova hijerarhija umnog djelovanja

Umno djelovanje na razini pamćenja zahtijeva prizivanje u sjećanje prethodno zapamćenih činjenica. Na razini razumijevanja zahtijeva se tumačenje značenja stečenog znanja te razumijevanje pojmova povezanih sa stečenim znanjem. Razina primjene zahtijeva uporabu stečenog znanja u okolnostima koje su različite od onih pod kojima je znanje stečeno. Umno djelovanje na razini raščlambe iziskuje sposobnost razlaganja naučenih informacija na elementarne sastavne činjenice u svrhu boljeg razumijevanja njihovih

međusobnih odnosa. Za razinu vrednovanja svojstvena je sposobnost donošenja odluka na osnovi dubinskog razmatranja, kritičkog razmišljanja i procjene. Razina stvaranja zahtijeva sposobnost zaključivanja na osnovi prethodno stečenog znanja, sposobnost izvođenja novog znanja te sposobnost promatranja prethodno stečenog znanja s nove točke gledišta.

Kretanjem uzduž Bloomove hijerarhije od nižih prema višim razinama, raste stupanj umnog opterećenja koje pojedine razine iziskuju. Niže razine Bloomove hijerarhije koje uključuju pamćenje, razumijevanje i primjenu znanja smatraju se osnovnim režimom umnog djelovanja. Gornje razine umnog djelovanja koje uključuju raščlambu, vrednovanje i stvaranje smatraju se naprednim režimom umnog djelovanja. S druge strane, okolnosti u kojima je dovoljno umno djelovanje na razini pamćenja i razumijevanja znatno su brojnije od onih koje zahtijevaju vrednovanje i stvaranje. To znači da najveći dio aktivnosti u svakodnevnom životu prosječnog čovjeka pripada rutinskim i predvidljivim poslovima koji ne zahtijevaju značajno umno naprezanje. Umno zahtjevni poslovi, poput poslova sa stalnom prisutnošću nepredvidljivih elemenata i elemenata nesigurnosti, svojstveni su manjoj skupini djelatnosti, a u životu prosječnog čovjeka javljaju se razmjerno rijetko.

Tijekom postupaka programiranja, osnovni režim umnog djelovanja svojstven je učenju sustava znakovlja i tumačenju napisanih programa, dok se napredni režim umnog djelovanja zahtijeva tijekom postupka pisanja novih programa. U terminologiji teorije održavanja i preispitivanja znanja, stjecanje znanja pripada osnovnom, a primjena i stvaranje znanja naprednom režimu umnog djelovanja. S obzirom da je primjena stečenog i stvaranje novog znanja umno zahtjevnija aktivnost u usporedbi sa stjecanjem novog znanja, pisanje novih programa u pravilu nameće veće umno opterećenje od tumačenja napisanih programa. Gledano kao cjelina, programiranje nameće veliko umno opterećenje na programera jer zahtijeva umno djelovanje u oba režima, osnovnom i naprednom, na svim razinama Bloomove hijerarhije.

Primjena za oblikovanje potrošaču prilagođenog programiranja

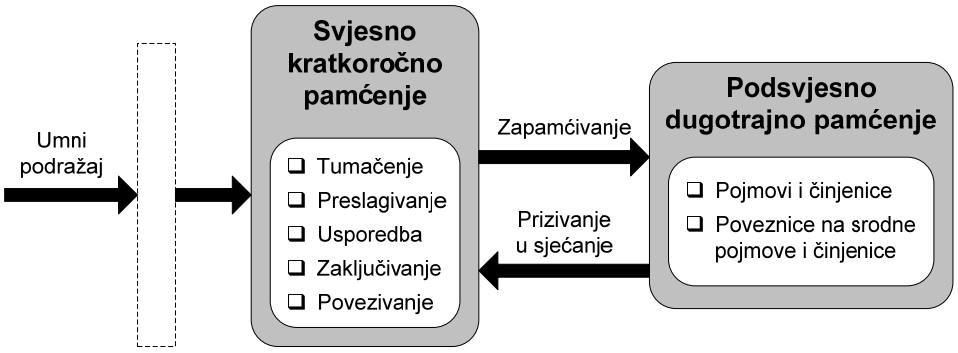
Tijekom oblikovanja potrošaču prilagođenog programiranja, teorija održavanja i preispitivanja znanja iskorištena je za izbor elemenata za izgradnju primjenskog programa i programskog jezika za povezivanje elemenata u radni tijek kako bi se umni napor za učenje i primjenu programske paradigme smanjio ili u potpunosti izbjegao. Programski elementi i tehnika izgradnje programa izjednačeni su s elementima i tehnikama koje se koriste tijekom uporabe primjenskih programa za *World Wide Web*. Iskorištavanjem poznatih načela i koncepata, izbjegava se uvođenje novih programskih apstrakcija, čime se smanjuje umni

napor potrošača i skraćuje vrijeme potrebno za učenje programske paradigme. Programska paradigma prilagođena potrošaču koja je predložena u okviru doktorske disertacije zasnovana je na programskim elementima u obliku udomljenika s grafičkim korisničkim sučeljem prikazanim u pregledniku *World Wide Web* sadržaja. Povezivanje elemenata u radni tijek zasnovano je na definiranju skupa radnji nad elementima grafičkih korisničkih sučelja udomljenika. Programska paradigma zasnovana na povezivanju udomljenika opisana je u poglavljima 7, 8, 9, 10 i 11.

3.4.2 Teorija kratkoročnog pamćenja privremenih činjenica

Ključna uloga u procesu umnog djelovanja pripada čovjekovoj sposobnosti pamćenja i zaključivanja. Tijekom programiranja, pamćenje i zaključivanje poprima ključnu ulogu u fazi zapisa rješenja primjenom programskog jezika kada je skup pojedinačnih i nezavisnih elemenata naučenog jezika potrebno povezati u usklađenu cjelinu koja predstavlja računalni program.

Iako sposobnost pamćenja igra presudnu ulogu u oba režima umnog djelovanja, uloga pamćenja tijekom osnovnog i naprednog režima značajno je različita. Teorijski modeli kognitivne znanosti razlikuju svjesno kratkoročno i podsvjesno dugotrajno pamćenje [39]. Slikom 3.7 prikazan je teorijski model za obradu informacija unutar ljudskog uma koji povezuje kratkoročno i dugotrajno pamćenje.



Slika 3.7. Model obrade informacija unutar ljudskog uma zasnovan na kratkoročnom i dugotrajnom pamćenju

Dugotrajno pamćenje (engl. *long-term memory*) omogućuje očuvanje naučenih činjenica ili zaključaka izvedenih iz naučenih činjenica za njihovo kasnije prizivanje u sjećanje i primjenu. Budući da se odvija bez potrebe za svjesnim umnim djelovanjem, dugotrajno pamćenje naziva se podsvjesnim pamćenjem. Kratkoročno pamćenje dolazi do izražaja tijekom provedbe složenih umnih procesa poput razumijevanja i zaključivanja za

koje su svojstveni višekoračni postupci tumačenja, pretraživanja, uspoređivanja, preslagivanja i odabira prethodno zapamćenih ili novodoživljenih činjenica. Kratkoročno pamćenje nužno je za rukovanje i privremeno očuvanje nepotpunog znanja u obliku međurezultata ili međukoraka na putu do izvođenja konačnih zaključaka u obliku novog znanja. Budući da postupci razumijevanja i zaključivanja na osnovi upamćenih ili novodoživljenih činjenica zahtijevaju stalnu usredotočenost i svjesno umno djelovanje, kratkoročno pamćenje naziva se svjesnim pamćenjem. S gledišta biologije, kratkoročno pamćenje posljedica je pojave kratkotrajnih električnih naboja na završecima moždanih neurona. Dugotrajno pamćenje omogućeno je stvaranjem dugotrajno održivih električkih naboja, što uzrokuje promjenu u fizičkoj strukturi neuronskih završetaka te omogućuje dugoročno održanje upamćenih činjenica.

Dva osnovna svojstva kratkoročnog pamćenja su kratko vrijeme zadržavanja sjećanja na ostvareni umni podražaj te ograničeni broj činjenica koje je moguće istovremeno pamtiti, odnosno kojima je moguće istovremeno rukovati. Istraživanja u području kognitivne znanosti pokazala su da bez stalnog osvježavanja, činjenice prikupljene kroz sustav umnih podražaja ostaju zapamćene tek dvadesetak sekundi, a nakon toga bivaju zaboravljene [39]. Kratkoročno pamćenje poprima dugotrajni oblik višestrukim osvježavanjem privremeno upamćenih činjenica kroz sustav ponavljajućih umnih podražaja ili smislenim povezivanjem privremeno upamćenih činjenica s prethodno zapamćenim činjenicama.

Osim kratkog vremena zadržavanja sjećanja na ostvareni umni podražaj, istraživanja su pokazala da postoje ograničenja u broju činjenica kojima čovjek može istovremeno rukovati. Teoriju o ograničenju kapaciteta kratkoročnog pamćenja iznio je američki znanstvenik George Armitage Miller 1956. godine pod nazivom teorija kratkoročnog pamćenja privremenih činjenica. George Armitage Miller danas je umirovljeni profesor psihologije američkog sveučilišta Princeton, a tijekom životnog vijeka radio je u svojstvu profesora na prestižnim američkim sveučilišnim središtima Rockefeller i Harvard te na američkom Institutu za tehnologiju države Massachusetts (engl. *Massachusetts Institute of Technology – MIT*). U teoriji potkrijepljenoj nizom pokusnih ispitivanja iznosi se zapažanje o broju sedam kao prosječnoj količini nepovezanih činjenica ili jedinica informacije kojima odrasle osobe mogu istovremeno rukovati [42]. Kod većine ljudi, taj broj se kreće u rasponu od pet do devet, dok prosječna vrijednost iznosi sedam. Pamtivi oblici činjenica ili jedinica informacije nisu jednaki za sve ljude, već ovise o znanju, iskustvu i vještinama pojedinca [42]. U najjednostavnijim slučajevima, pamtive jedinice informacije su jednostavni pojmovi poput brojki i slova. U slučajevima kada je elementarne pojmove moguće smisleno objediniti u složene smislene cjeline, pamtiva jedinica informacije postaje cjelokupni složeni pojam.

Primjer složenog pojma koji predstavlja pamtivu jedinicu informacije je riječ koja se sastoji od skupa slova ili rečenica koja se sastoji od skupa riječi. Znanje, iskustvo i vještine pojedinca pridonose sposobnosti objedinjavanja elementarnih pojmova u složene pojmove pa na taj način utječu i na učinkovitost kratkoročnog pamćenja.

Teorija kratkoročnog pamćenja privremenih činjenica objašnjava pojavu umnog preopterećenja koje ima za posljedicu smanjenje učinkovitosti umnog djelovanja. Učinkovitost umnog djelovanja narušena je u slučajevima kada je potrebno odjednom pamtiti preveliku količinu nepovezanih činjenica ili je nepovezane činjenice potrebno predugo držati u sjećanju. Objašnjenje umnog djelovanja teorijom kratkoročnog pamćenja privremenih činjenica ima značajnu ulogu u oblikovanju i vrednovanju sustava znakovlja s obzirom na bliskost umnom djelovanju čovjeka. Iako je funkcija kratkoročnog pamćenja nezaobilazna tijekom postupka učenja sustava znakovlja, njezina je uloga značajnija tijekom primjene naučenog sustava jer su za fazu primjene svojstvene umne aktivnosti razumijevanja i zaključivanja koji pripadaju naprednom režimu umnog djelovanja. Na osnovi opisane teorije kratkoročnog pamćenja privremenih činjenica, osmišljen je skup mjera umnog napora tijekom primjene sustava znakovlja. Skup tih mjera opisan je u poglavlju 3.5. Slijedeći preporuke za primjenu mjera umnog opterećenja u oblikovanju sustava znakovlja, elementi programskih jezika osmišljavaju se na način da primjena jezika ne zahtijeva istovremeno pamćenje većeg broja međurezultata nego što je kapacitet kratkoročnog pamćenja, da omogućuju objedinjavanje skupa jednostavnijih programskih elemenata u lako pamtive složenije elemente te da omogućuju što izravnije preslikavanje problema u elemente programskog jezika kako zapisivanje rješenja ne bi zahtijevalo nepotrebno opterećenje kratkoročnog pamćenja.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Tijekom oblikovanja potrošaču prilagođenog programiranja, teorija kratkoročnog pamćenja privremenih činjenica iskorištena je za oblikovanje tehnike izgradnje programa. Elementi za izgradnju primjenskog programa predstavljeni su blokovima koji od potrošača skrivaju pojedinosti programskog ostvarenja pojedinih funkcionalnih cjelina, a putem grafičkog korisničkog sučelja izlažu potrošaču usmjerene primjenske funkcije. Povezujući elementi kojima se skup nezavisnih programskih blokova povezuje u radni tijek prema zamislima potrošača predstavljeni su radnjama nad elementima grafičkih korisničkih sučelja. Kako bi se rasteretilo kratkoročno pamćenje potrošača, izbor povezujućih elemenata koji su u pojedinim koracima izgradnje radnog tijeka raspoloživi za primjenu obavlja se putem grafičkog korisničkog sučelja u obliku izbornika. Osim što se uklanja potreba za pamćenjem

skupa raspoloživih povezujućih elemenata, izabranom tehnikom izgradnje programa uklanja se i potreba za pamćenjem leksičkih i sintaksnih pravila programskog jezika. Postupak izgradnje programa primjenom grafičkog korisničkog sučelja u obliku izbornika opisan je u poglavljima 7, 8, 9 i 10.

3.4.3 Teorija višeprolaznog iščitavanja

Teorija višeprolaznog iščitavanja (engl. *multi-stage theory*) [41] primjenjuje se za objašnjavanje umne aktivnosti programera tijekom tumačenja značenja napisanih računalnih programa. Proučavanjem umne aktivnosti programera tijekom tumačenja značenja napisanih programa, uočena je pojava postupnog produbljivanja znanja o značenju programa. Pokusna ispitivanja pokazala su da je postupke tumačenja značenja napisanih programa moguće svrstati u jednu od tri skupine: postupci tumačenja od vrha prema dnu, postupci tumačenja od dna prema vrhu i postupci tumačenja odabirom dijela programa [35]. Za postupke tumačenja od vrha prema dnu svojstveno je da programer kreće od pokušaja razumijevanja međusobnih odnosa velikih programskih cjelina prema razumijevanju pojedinosti unutar pojedine cjeline. Za postupke tumačenja od dna prema vrhu svojstveno je da se cjelokupna slika o funkciji napisanog programa gradi na osnovi poznavanja funkcije pojedinih programskih cjelina. Postupci tumačenja odabirom dijela programa inačica su dvaju prethodno opisanih postupaka.

Rezultati pokusnih ispitivanja na ispitnim uzorcima programera pokazali su da odabir postupka za tumačenje značenja programa ovisi o poznavanju područja primjene za koje je program napisan te o poznavanju primijenjenog programskog jezika [35]. Programeri koji dobro poznaju područje primjene za koje je napisan zadani program češće se koriste postupkom tumačenja programa od vrha prema dnu, dok su manje iskusni programeri ili programeri s površnim poznavanjem programskog jezika skloni primjeni postupka tumačenja programa od dna prema vrhu.

Teorijom višeprolaznog iščitavanja smatra se da tijekom tumačenja značenja programa programer gradi dva umna modela, od kojih se svaki gradi u zasebnom prolazu iščitavanja programskog zapisa. Prvim umnim modelom predstavljena je struktura programa, za što su bitniji dijelovi programa kojima se opisuje tijek izvođenja. Drugi umni model služi za preslikavanje dijelova programskog zapisa u objekte, pojave i funkcije iz područja primjene programa, za što se pretežno koriste dijelovi kojima se opisuje tok podataka. U skladu s teorijom višeprolaznog iščitavanja, sustav znakovlja je blizak umnom djelovanju čovjeka ako su razlike u načinu predodžbe elemenata programa za opis tijeka izvođenja i za opis toka

podataka dovoljno naglašene kako bi ove dvije skupine elemenata bile međusobno dobro uočljive i jednostavno razlučive od elemenata druge skupine.

Primjena za oblikovanje potrošaču prilagođenog programiranja

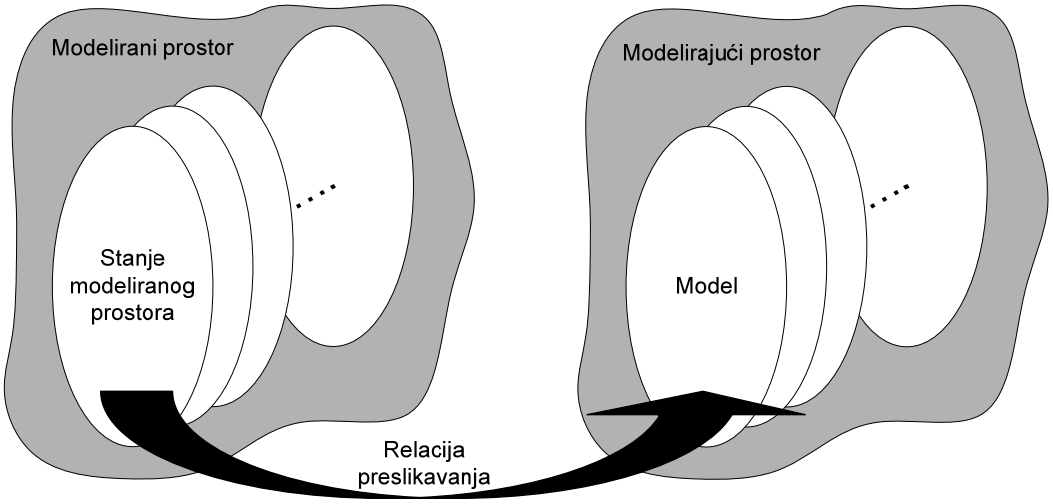
Tijekom oblikovanja potrošaču prilagođenog programiranja, teorija višeprolaznog iščitavanja iskorištena je za oblikovanje okoline za zapis i prikaz programa. Za prikaz programa koristi se dvodimenzionalna tablična ploha s dvosmjernim napredovanjem vremena. U takvoj se okolini na prvi pogled jasno raspoznaju vremenski zavisni i vremenski nezavisni dijelovi programa. Prikazom programa u obliku dvodimenzionalne tablice ističu se uzorci tijeka izvođenja programa, kao što je slijedno i istodobno izvođenje te mjesta razdvajanja i spajanja vremenskih tijekova. Podrobnijim pregledom zapisa programa, uočavaju se elementi za opis toka podataka na osnovi kojih potrošač stvara sliku cjelokupne funkcije programa. Prikaz programa primjenom dvodimenzionalne tablične plohe opisan je u poglavlju 11.

3.5 Mjere umnog napora tijekom primjene sustava znakovlja

Osim teorijskih modela opisanih u poglavlju 3.4 kojima se objašnjava umna aktivnost i procjenjuje umni napor čovjeka tijekom programiranja, značajan dio istraživanja u području psihologije programiranja bavi se istraživanjem svojstava različitih vrsta sustava znakovlja. Sustav znakovlja (engl. *notation*) je oblik vanjske predodžbe (engl. *external representation*) unutarnjeg stanja uma (engl. *mental representation*, *internal representation*). Iako se sustavom znakovlja smatra bilo koji oblik govornog, pismenog ili vidnog prenošenja informacije, u području psihologije programiranja pod pojmom sustava znakovlja uglavnom se podrazumijeva pismeni oblik prenošenja informacije. Najčešće korišteni oblici pismenog prenošenja informacije pojavljuju se u obliku teksta, slika, skica ili dijagrama. U terminologiji programskog inženjerstva, sustav znakovlja naziva se programskim jezikom.

Istraživanjem svojstava sustava znakovlja procjenjuje se stupanj umnog opterećenja koji primjena pojedine vrste znakovlja nameće na umni režim programera tijekom faze zapisa programskog rješenja. Za zapis rješenja istog problema, često je moguće primijeniti različite vrste znakovlja. Slika 3.8 prikazuje elemente koji definiraju stanje uma predočeno sustavom znakovlja. Sustavom znakovlja obuhvaćeni su modelirani prostor, modelirajući prostor, skup stanja modeliranog svijeta, skup modela i relacija preslikavanja iz modeliranog u modelirajući prostor. Modelirani prostor (engl. *represented world*) je umni doživljaj stvarnog svijeta ili postavljenog problema. Modelirajući prostor (engl. *representing world*) je

apstraktna predodžba modeliranog prostora opisana elementima sustava znakovlja. Modelirani prostor sastoji se od konačnog ili beskonačnog skupa stanja u kojima se nalazi. Model je prikaz stanja modeliranog svijeta sustavom znakovlja. Relacija preslikavanja određuje postupak stvaranja modela na osnovi stanja modeliranog svijeta.



Slika 3.8. Predodžba stanja uma sustavom znakovlja

Za procjenu bliskosti programskog jezika umnom režimu čovjeka, u području psihologije programiranja uspostavljen je sustav mjera umnog napora tijekom primjene sustava znakovlja (engl. *cognitive dimensions of notations*, *cognitive dimensions framework*) [24, 34]. Vrednovanjem sustava znakovlja s obzirom na sustav mjera umnog napora, moguće je objektivno utvrditi njegovu prikladnost za zapis rješenja postavljenog problema. U okviru doktorske disertacije proučen je skup mjera umnog napora te je iskorišten tijekom oblikovanja potrošaču prilagođenog programskog jezika. Na osnovi skupa mjera umnog napora koji daje niz općenitih smjernica za oblikovanje sustava znakovlja neovisno o području primjene, u okviru doktorske disertacije definiran je model za oblikovanje programske paradigme prilagođene potrošaču. Tim modelom definiran je skup mjera za oblikovanje i vrednovanje bliskosti programske paradigme umnom režimu, znanju i iskustvu potrošača. Definirani model opisan je u poglavlju 6.

Mjere umnog napora tijekom primjene sustava znakovlja su izravnost relacije preslikavanja (engl. *closeness of mapping*), stupanj smislenog objedinjavanja (engl. *abstraction gradient*), dosljednost (engl. *consistency*), samoopisivost (engl. *role-expresiveness*), podložnost pogreškama (engl. *error-proneness*), složenost rasuđivanja (engl. *hard-mental operations*), skrivene zavisnosti (engl. *hidden dependencies*), preuranjeno odlučivanje (engl. *premature commitment*), postupno vrednovanje (engl. *progressive*

evaluation), pozadinski sustav znakovlja (engl. *secondary notation*), prionljivost (engl. *viscosity*) te uočljivost i usporedivost (engl. *visibility and juxtaposability*).

3.5.1 Stupanj smislenog objedinjavanja

Smisleno objedinjavanje ili apstrakcija (engl. *abstraction*) je postupak objedinjavanja skupa elemenata u zajedničku cjelinu na način da ih se promatra kao jedan element. Najčešća primjena smislenog objedinjavanja je skrivanje pojedinosti modeliranog sustava, uz isticanje samo onih značajki koje su nužne za njegovo korištenje. Prema stupnju smislenog objedinjavanja, sustavi znakovlja dijele se na sustave znakovlja s obvezom smislenog objedinjavanja (engl. *abstraction-hungry notation*), sustave znakovlja s mogućnošću smislenog objedinjavanja (engl. *abstraction-tolerant notation*) i sustave znakovlja bez mogućnosti smislenog objedinjavanja (engl. *abstraction-hating notation*). Primjena sustava znakovlja s obvezom smislenog objedinjavanja od korisnika nužno zahtijeva objedinjavanje elemenata modela u složene smislene cjeline. Primjena sustava znakovlja s mogućnošću smislenog objedinjavanja korisniku dopušta smisleno objedinjavanje jednostavnijih elemenata modela u složene cjeline, ali ostavlja mogućnost modeliranja sustava bez smislenog objedinjavanja. Primjena sustava znakovlja bez mogućnosti smislenog objedinjavanja korisniku ne dopušta objedinjavanje jednostavnijih elemenata modela u složene cjeline.

S gledišta psihologije programiranja, smisleno objedinjavanje doprinosi razumijevanju sustava znakovlja i smanjenju umnog napora korisnika. Smisleno objedinjavanje omogućuje sažetiji zapis modela primjenom sustava znakovlja, čime se smanjuje opterećenje kratkoročnog pamćenja. Primjer sustava znakovlja koji dopušta primjenu smislenog objedinjavanja jest telefonski sustav s mogućnošću brzog biranja unaprijed definiranog skupa telefonskih brojeva (engl. *speed dial*). Korištenje sustava znakovlja bez primjene smislenog objedinjavanja zahtijeva pojedinačni unos svih znamenki telefonskog broja. S druge strane, korištenje sustava znakovlja uz primjenu smislenog objedinjavanja zahtijeva biranje samo jedne znamenke koja određuje mjesto na kojem je zapamćen zadani telefonski broj. Umna aktivnost tijekom primjene sustava znakovlja bez smislenog objedinjavanja zahtijeva pamćenje svih znamenki telefonskog broja, redoslijed njihova pojavljivanja i povezivanje telefonskog broja s osobom kojoj broj pripada. Umna aktivnost tijekom primjene tog istog sustava znakovlja, ali uz primjenu smislenog objedinjavanja, zahtijeva pamćenje samo jedne znamenke koja određuje na kojem se mjestu nalazi broj tražene osobe.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Tijekom oblikovanja modela potrošaču prilagođenog programiranja, mjera smislenog objedinjavanja iskorištena je za definiranje svojstva *blokovske izgradivosti primjenskog programa* koje je opisano u poglavlju 6.2.1 i svojstva *višerazinskog usložnjavanja blokova* koje je opisano u poglavlju 6.2.4. Svojstvom blokovske izgradivosti primjenskog programa, skup povezanih programskih funkcionalnosti smisleno se objedinjava i izlaže potrošaču u obliku jedinstvenog programskog bloka. Svojstvom višerazinskog usložnjavanja blokova omogućena je postupna izgradnja primjenskog programa. Funkcionalnosti skupa smisleno povezanih blokova objedinjavaju se i koriste u obliku novog bloka na višoj razini usložnjavanja.

3.5.2 Pozadinski sustav znakovlja

Uz osnovne elemente koji čine sustav znakovlja, smanjenju umnog napora korisnika doprinosi korištenje pomoćnih elemenata, odnosno pozadinskog sustava znakovlja (engl. *secondary notation*). Iako ne dodaje novo značenje osnovnom sustavu znakovlja, pozadinski sustav znakovlja doprinosi razumijevanju i jednostavnijem tumačenju zapisanog modela. Primjerice, korištenje posebnih znakova u zapisu telefonskih brojeva predstavlja primjenu pozadinskog sustava znakovlja. Osnovni sustav znakovlja koji čine znamenke dekadskog brojevnog sustava obogaćen je pozadinskim oznakama u obliku zagrada, crtica ili praznih znakova. Na primjer, telefonski broj za područje Grada Čakovca za poziv upućen iz inozemstva zapisanim osnovnim sustavom znakovlja

```
385406129897
```

moguće je zapisati primjenom pozadinskog sustava znakovlja u obliku

```
385-40-612-9897
```

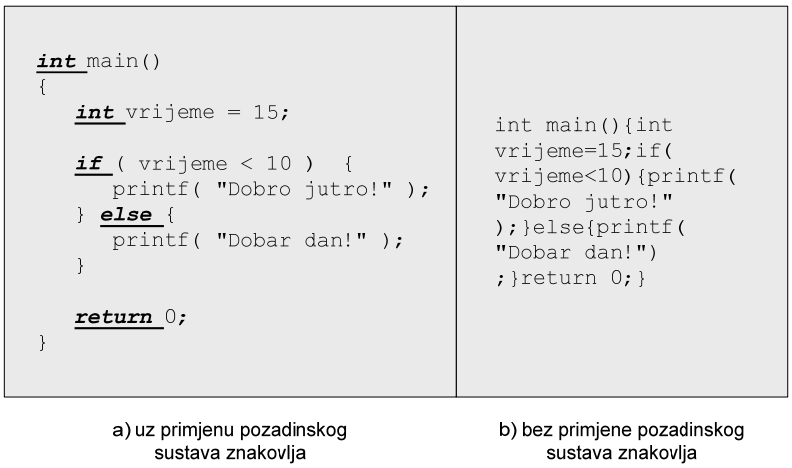
Korištenjem pozadinskih oznaka u obliku crtica, telefonski broj se rastavlja na jasno uočljive cjeline koje redom predstavljaju pozivni broj Republike Hrvatske (385), pozivni broj Grada Čakovca (40), pozivni broj gradske četvrti (612) te broj pozivane osobe (9897). Primjenom smislenog objedinjavanja radi lakšeg pamćenja, zadani telefonski broj moguće je zapisati u obliku

```
REPUBLIKA HRVATSKA-GRAD ČAKOVEC-612-9897
```

Budući da pozadinski sustav znakovlja ne utječe na značenje osnovnog skupa znakovlja, isto značenje modela moguće je postići različitim načinima primjene pozadinskog znakovlja. Druga mogućnost zapisa promatranog telefonskog broja jest oblik

385-406-129-897

gdje je uvjet za grupiranje znamenaka u skupine podjednak broj znamenaka u skupini.



```
int main()
{
   int vrijeme = 15;

   if ( vrijeme < 10 )  {
      printf( "Dobro jutro!" );
   } else {
      printf( "Dobar dan!" );
   }

   return 0;
}
```

a) uz primjenu pozadinskog sustava znakovlja

```
int main(){int
vrijeme=15;if(
vrijeme<10){printf(
"Dobro jutro!"
);}else{printf(
"Dobar dan!")
;}return 0;}
```

b) bez primjene pozadinskog sustava znakovlja

Slika 3.9. Isječak programa napisanog u programskom jeziku C sa i bez korištenja pozadinskog sustava znakovlja

Pozadinski sustav znakovlja nalazi široku primjenu u programiranju. Naročito je zastupljen u programskim jezicima zasnovanim na slovčanom zapisu u obliku uvlačenja redaka (engl. *indentation*) i naglašavanja sintaksnih cjelina (engl. *syntax highlighting*). Uvlačenje redaka nema utjecaja na rad jezičnog procesora, ali značajno doprinosi čitljivosti napisanog programa. Slika 3.9 prikazuje primjer programa napisanog u programskom jeziku C sa i bez primjene pozadinskog sustava znakovlja u obliku uvlačenja redaka i naglašenog isticanja ključnih riječi.

Primjena za oblikovanje potrošaču prilagođenog programiranja

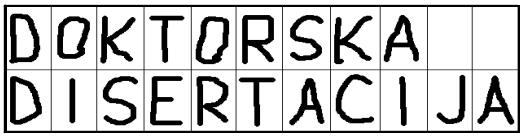
Mjera pozadinskog sustava znakovlja upotrijebljena je tijekom ostvarenja potrošaču prilagođene programske paradigme za izbor elemenata za prikaz programa. Program izgrađen od strane potrošača zapisuje se u dvodimenzionalnu tablicu u kojoj vrijeme istodobno i nezavisno napreduje u okomitom i vodoravnom smjeru. Tablična organizacija programa slična je grafičkim jezicima zasnovanim na čvorovima i usmjerenim granama, ali se povezivanje elemenata programa, umjesto linijskim poveznicama, izvodi njihovim zapisivanjem u susjedne ćelije tablice. Elementi programa zapisani u susjednim ćelijama izvode se slijedno jedan iza drugog, dok su elementi razdvojeni praznim ćelijama vremenski nezavisni. Tablični prikaz s dvosmjernim napredovanjem vremena pojednostavljuje oblikovanje složenih vremenskih uzoraka izvođenja programa jer se modeliranje slijednog i istodobnog izvođenja te sinkronizacije, ostvaruje bez korištenja potrošaču nerazumljivih

sinkronizacijskih mehanizama i programskih elemenata za pokretanje istodobnih procesa. U tabličnom prikazu programa, ćelije tablice organizirane u retke i stupce predstavljaju pozadinski sustav znakovlja. Podjela dvodimenzionalnog prostora na diskretne elemente u obliku ćelija olakšava poravnavanje programskih cjelina kako bi vremenski zavisni i vremenski nezavisni dijelovi programa bili jasno uočljivi i međusobno razlučivi. Nadalje, pozadinski sustav znakovlja koristi se za nadopunjavanje prikaza programa slobodnim zapisom korisničkih komentara. Između ključnih riječi kojima se opisuju elementi programa zapisani u ćelijama tablice dozvoljeno je ubacivati proizvoljne dijelove teksta pisane prirodnim jezikom.

3.5.3 Preuranjeno odlučivanje

Zapisivanje rješenja primjenom sustava znakovlja ponekad od korisnika zahtijeva donošenje odluke o redoslijedu poduzimanja međukoraka prema završnom rješenju prije nego što su sve informacije potrebne za donošenje odluke raspoložive. S druge strane, javljaju se i slučajevi u kojima su u promatranom trenutku sve potrebne informacije raspoložive, ali potrebne korake nije moguće u potpunosti izvršiti zbog ograničenja u sustavu znakovlja. Ovo svojstvo sustava znakovlja opisuje se mjerom umnog napora pod nazivom preuranjeno odlučivanje (engl. *premature commitment*).

a) sustav znakovlja s istaknutim problemom preuranjenog odlučivanja



b) ublažavanje problema preuranjenog odlučivanja

Slika 3.10. Problem preuranjenog odlučivanja tijekom ručnog upisivanja riječi u polje unaprijed zadanih dimenzija na listu papira

Problem preuranjenog odlučivanja javlja se u slučajevima kada sustav znakovlja sadrži zavisnosti među elementima koje ograničavaju redoslijed korištenja elemenata u sustavu znakovlja ili zahtijevaju unaprijedno predviđanje budućih koraka prema završnom rješenju. Slika 3.10 prikazuje jednostavan primjer utjecaja preuranjenog odlučivanja na način primjene sustava znakovlja. Ručno upisivanje riječi u predviđeno polje za unos teksta na papiru zahtijeva unaprijedno predviđanje duljine ispisane riječi te na osnovu toga

unaprijednu procjenu širine pojedinih znakova. Ako se tijekom procjene donese pogrešna odluka, završne znakove ispisane riječi potrebno je stješnjavati kako bi cijela riječ stala u predviđeno polje, kao što je prikazano na slici 3.10.a. Ublažavanje utjecaja preuranjenog odlučivanja prikazano je slikom 3.10.b. Pod pretpostavkom da je unaprijed moguće predvidjeti najveću moguću duljinu riječi koja će biti upisana u polje, problem preuranjenog odlučivanja moguće je ublažiti podjelom polja za unos teksta na pojedinačna polja za unos pojedinih znakova unutar kojih se ti znakovi moraju pojavljivati.

Drugi svakodnevni primjer sustava znakovlja s istaknutim problemom preuranjenog odlučivanja je jednostavno ručno računalo s osnovnim aritmetičkim funkcijama, prikazano slikom 3.11.a. Izračunavanje aritmetičkog izraza

```
3*(4+5)/(6-7)
```

uz poštivanje relacija prednosti i asocijativnosti operatora zahtijeva unos znakova u dva koraka prema redoslijedu

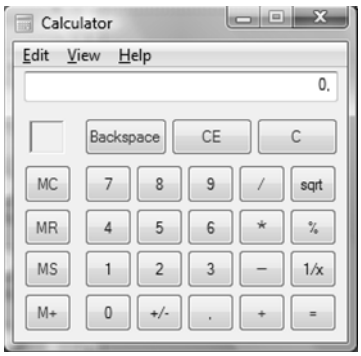
```
6  -  7  =  M+
```

```
4  +  5  =  *  3  =  /  MR  =
```

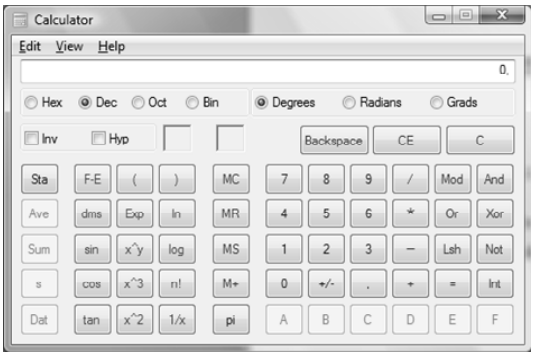
Primjenom jednostavnog ručnog računala, za izračunavanje zadanog izraza potrebno je pribjeći korištenju ugrađene memorije ručnog računala za privremeno očuvanje međurezultata (6-7). Ručnim računalom za napredno izračunavanje prikazano slikom 3.11.b koje pruža mogućnost korištenja zagrada pri upisivanju aritmetičkih izraza, gornji je izraz moguće zadati u jednom koraku prema redoslijedu

```
3  *  (  4  +  5  )  /  (  6  -  7  )  =
```

koji odgovara uobičajenom načinu zapisivanja aritmetičkih izraza.



a) jednostavno ručno računalo s osnovnim aritmetičkim operacijama

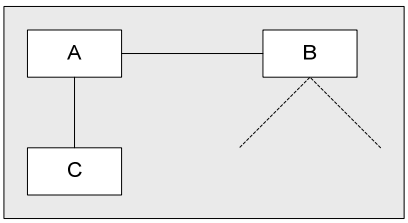


b) ručno računalo za napredno izračunavanje

Slika 3.11. Problem preuranjenog odlučivanja tijekom izračunavanja aritmetičkih funkcija primjenom ručnog računala

Programski jezici česti su primjeri sustava znakovlja s istaknutim problemom preuranjenog odlučivanja. Primjer iz područja grafičkog programiranja primjenom dijagrama prikazan je slikom 3.12. Uređivači za iscrtavanje dijagrama u obliku ćelija i linija (engl. *box-and-line editor*) često zahtijevaju postavljanje obiju ćelija na radnu površinu prije nego što je među njima moguće iscrtati liniju koja ih povezuje. Nerijetko se, međutim, javljaju slučajevi u kojima je za promatranu ćeliju korisniku unaprijed poznat broj ćelija s kojima promatrana ćelija mora biti povezana, ali nije poznata vrsta ćelija na koje se povezivanje odnosi. Iako je broj linija koje je potrebno iscrtati poznat, uređivač sprječava korisnika u iscrtavanju linija ako obje krajnje točke iscrtane linije nisu povezane na neke od iscrtanih ćelija.

Preuranjeno odlučivanje nepovoljno utječe na umno opterećenje korisnika jer unaprijedno donošenje odluka zahtijeva predviđanje budućih međukoraka prema završnom rješenju. Predviđanje budućih međukoraka zauzima kratkoročno pamćenje korisnika. Budući da je za kratkoročno pamćenje svojstven ograničen broj činjenica kojima je moguće istovremeno rukovati, preuranjeno odlučivanje smanjuje količinu kratkoročnog pamćenja koja ostaje raspoloživa za rukovanje činjenicama u sadašnjim koracima.



Slika 3.12. Problem preuranjenog odlučivanja tijekom grafičkog programiranja primjenom dijagrama

Pokusna ispitivanja u području psihologije programiranja pokazala su da postoje različiti načini kojima programeri zaobilaze okolnosti koje zahtijevaju preuranjeno odlučivanje. Neki od postupaka za ublažavanje utjecaja preuranjenog odlučivanja su skiciranje približnog rješenja problema na papiru prije zapisivanja u zadanom sustavu znakovlja, djelomično izvršavanje nepotpunih koraka koji se dovršavaju u kasnijim fazama zapisa rješenja i slično [24].

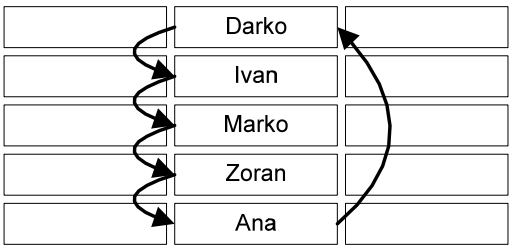
Primjena za oblikovanje potrošaču prilagođenog programiranja

Tijekom oblikovanja modela potrošaču prilagođenog programiranja, mjera preuranjenog odlučivanja iskorištena je za definiranje svojstva *upravljivosti i uočljivosti vremenske relacije* koje je opisano u poglavlju 6.2.8. Tim svojstvom zahtijeva se prikaz programa u obliku koji je pogodan za uređivanje vremenskih odnosa među programskim cjelinama. Svojstvom upravljivosti i uočljivosti vremenske relacije omogućeno je

povezivanje programskih cjelina u složene uzorke tijeka izvođenja programa, kao što je slijedno i istodobno izvođenje te sinkronizacija rada pojedinih cjelina. Izravnim utjecajem potrošača na redoslijed izvođenja elemenata za izgradnju programa, izbor elemenata moguće je obaviti bez potrebe za istodobnim donošenjem odluke o njihovim vremenskim odnosima. Radni tijek nad skupom elemenata u svakom je trenutku moguće nadopuniti uvođenjem novih programskih cjelina i njihovo objedinjavanje s prethodno izgrađenim programom. Takva programska paradigma dopušta izgradnju radnog tijeka primjenom iterativnog postupka tijekom kojeg je moguće višestruko prebacivanje usredotočenosti s izbora programskih elemenata na uspostavu vremenskih odnosa i obrnuto.

3.5.4 Prionljivost

Prionljivost sustava znakovlja (engl. *viscosity*) je mjera otpornosti zapisanog modela na male i lokalizirane promjene. Prepoznate su dvije najčešće vrste prionljivosti. Uvišestručena prionljivost (engl. *repetition viscosity*) je pojava kod koje je istu vrstu izmjene potrebno provesti na više mjesta ili u više istovjetnih koraka u modelu. Vezana prionljivost (engl. *knock-on viscosity*) je pojava kod koje jedna izmjena uzrokuje čitav niz novih izmjena koje je potrebno izvršiti da bi se očuvala dosljednost modela.



Slika 3.13. Primjer sustava znakovlja s istaknutom uvišestručenom prionljivošću

Primjer uvišestručene prionljivosti prikazan je slikom 3.13. Slika prikazuje uređivač tabličnih proračuna (engl. *spreadsheet editor*). Uvišestručena prionljivost javlja se u slučajevima kada je potrebno zamijeniti redoslijed ćelija, primjerice zbog sortiranja sadržaja abecednim redom. U prikazanom primjeru, izmjena koju je potrebno izvršiti je premještanje zadnje ćelije na vrh tablice uz pomicanje svih ostalih ćelija za jedno mjesto prema dolje. Pod pretpostavkom da uređivač koji stoji na raspolaganju nema mogućnost preseljavanja ćelija pomicanjem miša (engl. *drag-and-drop*), potrebne izmjene je potrebno obaviti brisanjem sadržaja ćelija i ponovnim upisivanjem ispravnih vrijednosti. U prikazanom primjeru, obavljanje potrebne izmjene zahtijeva pet operacija brisanja i pet operacija ponovnog upisivanja. Uz postavljena ograničenja, u zadanom sustavu znakovlja postoji visoki stupanj

prionljivosti. Iako stanje modela u svakom međukoraku jasno odražava sve dotad načinjene izmjene, uvišestručena prionljivost nameće znatno umno opterećenje na korisnika jer je potrebno voditi brigu o velikom skupu izmjena koje je potrebno izvršiti.

Uvođenjem mogućnosti premještanja ćelija pomicanjem miša (engl. *drag-and-drop*), uz automatsko pomicanje istisnutih ćelija na slobodno mjesto u tablici, zahtijevanu izmjenu moguće je obaviti u jednom koraku. Napredno svojstvo uređivača u ovom slučaju smanjuje stupanj prionljivosti sustava znakovlja i doprinosi smanjenju umnog napora korisnika.

Primjer vezane prionljivosti jest ubacivanje slika u dokument unutar kojeg su slike naslovljene rastućim brojčanim oznakama. Ubacivanje nove slike na bilo koje mjesto koje u dokumentu prethodi posljednjoj slici u nizu zahtijeva ažuriranje brojčanih oznaka svih slika koje slijede iza mjesta na koje se ubacuje nova slika. Suvremeni uređivači teksta (engl. *text editor, text processor*) dozvoljavaju automatsko brojčano naslovljavanje slika i sličnih objekata, čime se smanjuje problem vezane prionljivosti.

Prionljivost sustava znakovlja naročito je važna za programske jezike jer su istraživanja pokazala da je tijekom cjelokupnog procesa razvoja programskih sustava, od planiranja i izrade specifikacije do oblikovanja i programske izrade, stalno prisutna potreba za izmjenama napisanog modela. Potreba za izmjenama nije nužno posljedica površnog pristupa ili propusta tijekom izrade programskih sustava, već je posljedica potrebe za stalnim unaprjeđivanjem razvijenog proizvoda. S druge strane, potreba za izmjenama javlja se i zbog postupnog razvoja modela. U početnim koracima definira se globalna slika konačnog modela u obliku velikih sastavnih cjelina od kojih se model sastoji, dok se u završnim koracima definiraju pojedinosti unutar pojedine cjeline.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Tijekom oblikovanja modela potrošaču prilagođenog programiranja, mjera prionljivosti iskorištena je za definiranje svojstva *blokovske izgradivosti primjenskog programa* koje je opisano u poglavlju 6.2.1, svojstva *nezavisnosti povezujućih elemenata i značenja blokova* koje je opisano u poglavlju 6.2.6, svojstva *upravljivosti i uočljivosti vremenske relacije* koje je opisano u poglavlju 6.2.8 i svojstva *potpune izgradivosti logike za usložnjavanje blokova od strane potrošača* koje je opisano u poglavlju 6.2.9. Izgradnjom programa od programskih blokova krupne zrnatosti povezanih povezujućim elementima koji su nezavisni od primjenskih svojstava blokova, smanjuje se otpornost programa na izmjene. Izmjene programa moguće je provoditi uklanjanjem bilo kojeg bloka iz radnog tijeka ili uvođenjem novih blokova u radni tijek, uz odgovarajuću prilagodbu povezujućih elemenata.

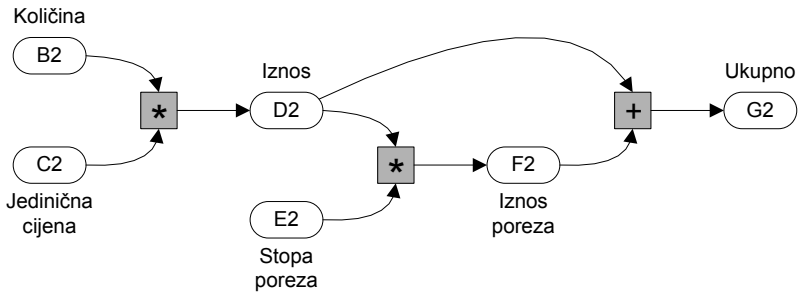
Potpunom izgradivošću logike za usložnjavanje blokova od strane potrošača, potrošač dobiva mogućnost povezivanja programskih blokova u proizvoljni radni tijek koji nije predodređen izborom vrste blokova. Upravljivošću i uočljivošću vremenske relacije, omogućava se ažuriranje vremenskih odnosa u izabranom dijelu programa te dodavanje novih ili uklanjanje postojećih programskih cjelina, bez narušavanja vremenskih odnosa u ostalim, vremenski nezavisnim dijelovima programa.

3.5.5 Skrivene zavisnosti

Skrivena zavisnost je pojava koja se javlja među elementima sustava znakovlja kada jedan element ovisi o drugom na način da njihova zavisnost nije jasno uočljiva. Skrivene zavisnosti česta su pojava u programiranju, naročito u programskim jezicima koji podržavaju uporabu pokazivača (engl. *pointer*). Uporaba pokazivača u računalni program uvodi zavisnost koja je vidljiva u smjeru od pokazivača prema varijabli na koju pokazivač pokazuje, ali nije vidljiva u obrnutom smjeru. Skrivene zavisnosti svojstvene su i HTML stranicama. Stranica koja sadrži poveznicu (engl. *link*) na drugu stranicu uvodi skrivenu zavisnost koju nije moguće uočiti na stranici na koju se poveznica odnosi.

	A	B	C	D	E	F	G
1	Vrsta robe	Količina	Jedinična cijena	Iznos	Stopa poreza	Iznos poreza	Ukupno
2	Olovka	3	13,99	=B2*C2	0,22	=D2*E2	=D2+F2

a) tablični proračun



b) graf zavisnosti

Slika 3.14. Tablični proračun kao primjer sustava znakovlja s izrazitim sklonostima za pojavu skrivenih zavisnosti

U sustavu znakovlja koji dopušta pojavu skrivenih zavisnosti moguća je pojava ulančavanja (engl. *chaining*, *trailing*) i grananja (engl. *branching*) skrivenih zavisnosti. Cjelokupni sustav skrivenih zavisnosti u zadanom modelu naziva se mrežom skrivenih

zavisnosti. Sustav znakovlja za koji je svojstvena pojava ulančanih i razgranatih skrivenih zavisnosti je tablični proračun (engl. *spreadsheet*). Slika 3.14.a prikazuje isječak tabličnog proračuna koji sadrži nekoliko ulančanih i razgranatih skrivenih zavisnosti. Promatranjem prikazanog sustava znakovlja u obliku tabličnog proračuna, zbog ulančanih skrivenih zavisnosti, teško je uočiti veze između ćelija B2 i C2 s ćelijom G2. Cjelokupna mreža skrivenih zavisnosti u prikazanom primjeru prikazana je grafom zavisnosti na slici 3.14.b.

Sustav znakovlja koji dopušta pojavu skrivenih zavisnosti opterećuje umno djelovanje korisnika. U nedostatku jasno istaknutih zavisnosti koje postoje u modelu, pronalaženje tražene informacije zahtijeva pretraživanje mreže zavisnosti, što produljuje vrijeme umnog djelovanja, povećava rizik od pogreške, narušava prirodni tijek misli i smanjuje pouzdanost izrade modela zadanim sustavom znakovlja [24, 34]. Umni napor koji je potrebno uložiti za otkrivanje skrivenih zavisnosti u sustavu znakovlja ovisi o duljini lanca zavisnosti, stupnju razgranatosti i količini napora potrebnog za praćenje pojedinih poveznica u mreži zavisnosti.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Tijekom oblikovanja modela potrošaču prilagođenog programiranja, mjera skrivenih zavisnosti iskorištena je za definiranje svojstva *jednakosti korištenja i povezivanja blokova* koje je opisano u poglavlju 6.2.5, svojstva *upravljivosti i uočljivosti vremenske relacije* koje je opisano u poglavlju 6.2.8 i svojstva *potpune izgradivosti logike za usložnjavanje blokova od strane potrošača* koje je opisano u poglavlju 6.2.9. Budući da je jedno od svojstava programiranja prilagođenog potrošaču mogućnost izgradnje primjenskog programa povezivanjem gotovih programskih blokova, postojanje skrivenih zavisnosti razmatra se na razini povezivosti blokova. Svojstva jednakosti korištenja i povezivanja blokova te potpune izgradivosti logike za usložnjavanje blokova od strane potrošača odnose se na pojavu skrivenih zavisnosti u podatkovnim vezama između programskih blokova. Svojstvo upravljivosti i uočljivosti vremenske relacije odnosi se na pojavu skrivenih zavisnosti u prikazu redoslijeda izvođenja operacija nad blokovima.

Izjednačavanjem programskog povezivanja blokova unutar paradigme i ručnog korištenja blokova izvan paradigme te prepuštanjem izgradnje logike za povezivanje blokova potrošaču, sve podatkovne zavisnosti kojima potrošač rukuje na razini povezivanja blokova svjesno su ugrađene u radni tijek primjenskog programa. U prikazu programa vidljive su sve operacije koje se izvode nad pojedinačnim blokovima, kao i sve podatkovne veze kojima se uspostavlja prijenos informacija između različitih blokova.

Vremenski odnosi nad operacijama za upravljanje izvođenjem blokova oblikuju se i prikazuju tablicom u kojoj vrijeme istodobno i nezavisno napreduje u okomitom i vodoravnom smjeru. Razmještajem operacija za upravljanje izvođenjem blokova po ćelijama dvodimenzionalne tablice moguće je oblikovati složene uzorke izvođenja programa, kao što je slijedno i istodobno izvođenje te razdvajanje i spajanje vremenskih tijekova, isključivo primjenom jednostavne relacije vremenskog prethođenja, odnosno vremenskog slijeđenja. Vremenski slijedne operacije zapisuju se u dvije susjedne vodoravne ili okomite ćelije tablice. Vremenski nezavisne operacije zapisuju se u nesusjedne ćelije tablice razdvojene barem jednom praznom ćelijom. Tabličnim prikazom programa postignuto je uravnoteženje između količine informacija kojima se opisuju vremenske zavisnosti u programu i čitljivosti programskog zapisa. Izravne vremenske zavisnosti jasno su istaknute u zapisu programa. Prostorno susjedne ćelije tablice sadrže vremenski slijedne operacije nad programskim blokovima. Prostorno razdvojene ćelije tablice sadrže vremenski nezavisne operacije nad blokovima. Neizravne vremenske zavisnosti koje proizlaze iz svojstva tranzitivnosti vremenske relacije nisu posebno istaknute kako se prikaz programa ne bi nepotrebno opterećivao informacijama do kojih se dolazi jednostavnim zaključivanjem. Izvođenje zaključaka o postojanju neizravnih vremenskih zavisnosti provodi se na osnovi analize prostornog razmještaja i popunjenosti ćelija. Ako između dviju nepraznih i nesusjednih ćelija tablice postoji prostorni put kroz skup susjednih i nepraznih ćelija u bilo kojem od dva smjera napredovanja vremena, onda su nesusjedne ćelije tablice povezane neizravnom vremenskom relacijom. Pronalaženje tranzitivnih vremenskih zavisnosti u tabličnom prikazu programa je jednostavno jer se zasniva na razlučivosti praznih od nepraznih ćelija, što je na zaslonu računala prilično lako uočljivo. Prednosti tabličnog prikaza programa iscrpno su opisane u poglavlju 11 i [107].

3.5.6 Uočljivost i usporedivost

Uočljivost (engl. *visibility*) je svojstvo sustava znakovlja za koji vrijedi da je tražena informacija u zapisanom modelu dostupna i uočljiva bez potrebe za ulaganjem umnog napora. Primjer sustava znakovlja koji djelomično zadovoljava svojstvo uočljivosti je telefonski imenik u tiskanom obliku. Tiskano izdanje telefonskog imenika prilagođeno je za pretraživanje po imenu i prezimenu i pronalaženje pripadajućeg telefonskog broja. Korištenje imenika na taj način odlikuje se visokim stupnjem uočljivosti. Međutim, tiskano izdanje telefonskog imenika nije pogodno za pronalaženje imena i prezimena vlasnika za zadani telefonski broj jer ne postoji učinkovit način pretraživanja zapisa po zadanom

telefonskom broju. S obzirom na postupak traženja imena i prezimena na osnovi telefonskog broja, promatrani sustav znakovlja ima loše svojstvo uočljivosti.

Svojstvo uočljivosti u programskim jezicima često se definira kao mogućnost postavljanja cjelokupnog programa na jednu stranicu zaslona računala. U počecima programerske struke, računalni programi bili su dovoljno kratki da na taj način definirano svojstvo uočljivosti bude ispunjeno. Kako je veličina računalnih programa porasla za nekoliko redova veličine, svojstvo uočljivosti nastojalo se poboljšati uvođenjem potprograma i programskih knjižnica u programske jezike. Dodatno, u današnjim razvojnim alatima korisniku na raspolaganju stoje pomoćne funkcije koje, osim prikaza teksta napisanog programa, omogućavaju grafički pregled odnosa među potprogramima i knjižnicama uključenim u programski sustav, pregled hijerarhije razreda i komponenata i slično. Uvođenjem pomoćnih funkcija olakšava se pronalaženje tražene informacije u programu i ublažava nedostatak uočljivosti.

Usporedivost (engl. *juxtaposability*) je svojstvo skupa znakovlja koje omogućava jednostavnu usporedbu bilo koja dva elementa zapisanog modela postavljanjem elemenata jedan uz drugi. Primjer sustava znakovlja koji ne zadovoljava svojstvo usporedivosti su knjige u tiskanom obliku s dvostranim otiskom. Usporedba dvaju grafičkih prikaza koji su otisnuti na istom listu papira, ali s različitih strana, nije moguća istovremenim promatranjem obaju prikaza.

Programski alati na različite načine podižu stupanj usporedivosti elemenata zapisanog modela. Preglednici knjiga u elektroničkom obliku omogućavaju istovremeno postavljanje dviju stranica knjige na zaslon računala na način da je svaka stranica prikazana na polovici zaslona. Uz označavanje svih stranica između dviju promatranih stranica nevidljivima, bilo koje dvije stranice moguće je istovremeno postaviti na zaslon. Suvremeni razvojni alati za programiranje također često omogućuju otvaranje istog modela u dva različita prozora uređivača, što omogućuje postavljanje pokazivača položaja (engl. *cursor*) na dva različita mjesta u modelu i istovremenu usporedbu dvaju elemenata modela. S obzirom da u tom slučaju svakom modelu pripada polovica ukupne površine zaslona, razvojni alati na taj način povećavaju mjeru usporedivosti uz žrtvovanje mjere uočljivosti.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Tijekom oblikovanja modela potrošaču prilagođenog programiranja, mjera uočljivosti i usporedivosti iskorištena je za definiranje svojstva *upravljivosti i uočljivosti vremenske relacije* koje je opisano u poglavlju 6.2.8. Tijekom izbora elemenata za ostvarenje potrošaču

prilagođene programske paradigme, mjera uočljivosti i usporedivosti iskorištena je za izbor tehnike prikaza programa koja je opisana u poglavlju 11. Prikaz programa zasniva se na dvodimenzionalnom tabličnom zapisu s dvosmjernim napredovanjem vremena. Istodobno napredovanje vremena u okomitom i vodoravnom smjeru omogućava pojednostavljeno oblikovanje složenih uzoraka izvođenja programa, kao što je slijedno i istodobno izvođenje te razdvajanje i spajanje vremenskih tijekova, isključivo prostornim razmještajem programskih cjelina unutar ćelija tablice. Osim što pojednostavljuje oblikovanje programa, tablična organizacija programa pojednostavljuje i usporedbu vremenskih svojstava različitih primjenskih programa. Sličnost tijeka izvođenja dvaju programa proizlazi iz sličnosti prostornog razmještaja programskih cjelina, što je zbog dvodimenzionalne tablične organizacije programa uočljivo na prvi pogled.

3.5.7 Izravnost relacije preslikavanja

Zapisivanje modela primjenom sustava znakovlja zahtijeva preslikavanje stanja modeliranog prostora u model modelirajućeg prostora. Što je modelirajući prostor sličniji modeliranom prostoru, to je relacija preslikavanja izravnija, a postupak preslikavanja jednostavniji, odnosno manje umno zahtjevan. U idealnom slučaju, svaki element i pojam iz modeliranog prostora trebao bi se preslikavati u apstraktni element istog naziva i istih svojstava u sustavu znakovlja. Svojstvo izravnosti relacije preslikavanja (engl. *closeness of mapping*) je mjera sličnosti zapisanog modela sa stanjem modeliranog svijeta.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Tijekom oblikovanja modela potrošaču prilagođenog programiranja, mjera izravnosti relacije preslikavanja iskorištena je za definiranje svojstva *samodostatnosti blokova* koje je opisano u poglavlju 6.2.3 i svojstva *jednakosti korištenja i povezivanja blokova* koje je opisano u poglavlju 6.2.5. Izborom samodostatnih programskih blokova, program se gradi od elemenata koje su potrošači izvan konteksta programske paradigme navikli koristiti za obavljanje uobičajenih zadaća primjenom računala. Primjena blokova izvan programske paradigme predstavlja modelirani prostor. Iskorištavanjem poznatih blokova u svojstvu elemenata za izgradnju programa, postiže se izravnost preslikavanja poznatog prostora izravne primjene blokova u novonastali prostor izgradnje programa od skupa blokova.

U programskoj paradigmi predloženoj u okviru doktorske disertacije, modelirani prostor sastoji se od skupa udomljenika koje potrošač koristi za obavljanje složenih zadaća putem računala. Pojedinačni udomljenici obavljaju pojedine dijelove složenog posla, dok se

odabirom i primjenom odgovarajućeg skupa udomljenika postiže cjelokupna funkcionalnost prema zamislima potrošača. Međudjelovanjem putem grafičkog korisničkog sučelja prikazanog u pregledniku *World Wide Web* sadržaja, potrošač upravlja izvođenjem pojedinačnih udomljenika, dok preuzimanjem i preslikavanjem sadržaja (engl. *copy/paste*) uspostavlja podatkovne veze između različitih udomljenika. U predloženom ostvarenju programske paradigme, program se gradi od istog skupa udomljenika koji se koriste za obavljanje iste zadaće tijekom izravnog rukovanja. Programska logika za povezivanje skupa udomljenika u radni tijek gradi se definiranjem radnji nad elementima grafičkog korisničkog sučelja udomljenika, na jednak način na koji su ih potrošači navikli koristiti izvan konteksta programske paradigme. Ovakvim izborom programskih blokova i jezika za njihovo povezivanje u radni tijek, programska paradigma oslanja se na znanje i iskustvo potrošača stečeno isključivo redovnim školovanjem i uporabom primjenskih programa za *World Wide Web*.

3.5.8 Dosljednost

Svojstvo dosljednosti (engl. *consistency*) zahtijeva da se isti elementi sustava znakovlja koriste na jednak način, da se slični elementi sustava znakovlja koriste na sličan način te da se iste operacije nad različitim elementima provode na jednak način bez obzira na okolnosti primjene. Sustavi znakovlja sa svojstvom dosljednosti nameću manji umni napor tijekom učenja i primjene jer je mali skup naučenih pravila moguće primijeniti na veliki broj elemenata sustava znakovlja.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Tijekom oblikovanja modela potrošaču prilagođenog programiranja, mjera dosljednosti iskorištena je za definiranje svojstva *jednakosti korištenja i povezivanja blokova* koje je opisano u poglavlju 6.2.5, svojstva *nezavisnosti povezujućih elemenata i značenja blokova* koje je opisano u poglavlju 6.2.6 i svojstva *potpune izgradivosti logike za usložnjavanje blokova od strane potrošača* koje je opisano u poglavlju 6.2.9. Izjednačavanjem korištenja i povezivanja blokova, postiže se dosljedna primjena blokova unutar i izvan konteksta programske paradigme. Nezavisnošću povezujućih elemenata i značenja blokova, omogućuje se oblikovanje programa primjenom konačnog skupa elemenata za povezivanje programskih blokova. Pojedini povezujući elementi uvijek se koriste na jednak način, bez obzira na primjenska svojstva programskih blokova koje povezuju. Potpunom izgradivošću logike za povezivanje blokova od strane potrošača,

programska paradigma omogućuje jednaku razinu izražajnosti kao što se postiže izravnim rukovanjem blokovima izvan konteksta paradigme.

3.5.9 Samoopisivost

Samoopisivost (engl. *role-expresiveness*) sustava znakovlja je mjera koja određuje količinu informacija o značenju elemenata sustava znakovlja koju je moguće iščitati iz prikazanog modela bez prethodnog poznavanja njihova značenja. Naredba pridruživanja u programskom jeziku COBOL u obliku

```
MULTIPLY A BY B GIVING C
```

primjer je samoopisivog elementa sustava znakovlja. Iz prikazanog primjera je čak i bez unaprijednog poznavanja sintaksnih pravila jezika jednostavno zaključiti da se umnožak varijabli *A* i *B* sprema u varijablu *C*. S druge strane, programski jezik Forth sadrži značajan broj elemenata kojima je teško utvrditi značenje bez poznavanja pravila jezika. Primjerice, oznaka točke

```
.
```

u programu napisanom u programskom jeziku Forth predstavlja naredbu za ispis znaka za novi redak (engl. *newline character*).

Dobra svojstva samoopisivosti pokazuju sustavi znakovlja koji se sastoje od elemenata koji uključuju dovoljnu, ali ne prekomjernu, količinu informacija o vlastitom značenju. Pretjerano obogaćivanje elemenata sustava znakovlja informacijama o značenju nepotrebno opterećuje umno djelovanje korisnika. Povećanjem količine informacija koju je moguće iščitati iz prikaza modela, smanjuje se udio informacija koji je moguće zadržati u kratkoročnom pamćenju. S druge strane, u sustavima znakovlja sa šturim obogaćivanjem elemenata informacijama o značenju moguća je pojava pretjerane sličnosti modela koji se po značenju i funkciji znatno razlikuju.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Tijekom oblikovanja modela potrošaču prilagođenog programiranja, mjera samoopisivosti iskorištena je za definiranje svojstva *opažajnog doživljaja značenja blokova* koje je opisano u poglavlju 6.2.2 i svojstva *opažajnog doživljaja prikaza programa* koje je opisano u poglavlju 6.2.10. Svojstvom opažajnog doživljaja značenja blokova zahtijeva se oblikovanje programske paradigme u kojoj su značenje i način primjene pojedinog programskog bloka potrošaču poznati na osnovi vidljive predodžbe bloka. Time se izbjegava

potreba za školovanjem potrošača ili za korištenjem priručnika za uporabu blokova. U programskoj paradigmi predloženoj u okviru doktorske disertacije, programski blokovi pojavljuju se u obliku udomljenika kojima se rukuje putem potrošaču razumljivog grafičkog korisničkog sučelja prikazanog u pregledniku *World Wide Web* sadržaja. Svojstvom opažajnog doživljaja prikaza programa zahtijeva se oblikovanje programske paradigme u kojoj je programska logika za povezivanje blokova u radni tijek potrošaču prikazana na način koji ne zahtijeva prethodno učenje ili iskustvo u primjeni paradigme. U programskoj paradigmi predloženoj u okviru doktorske disertacije, program je prikazan o obliku potrošaču razumljivih radnji nad elementima grafičkog korisničkog sučelja udomljenika, kao što je upisivanje sadržaja u polje za unos teksta, pritisak mišem na tipku ili izbor stavke iz padajućeg izbornika. Za razumijevanje značenja programa dovoljno je znanje o korištenju primjenskih programa putem grafičkog korisničkog sučelja prikazanog u pregledniku *World Wide Web* sadržaja, kojim ovladava većina današnjih korisnika računala.

3.5.10 Podložnost pogreškama

Pogreške tijekom primjene sustava znakovlja moguće je svrstati u dvije skupine. Prvu skupinu čine logičke pogreške koje nastaju kao posljedica pogrešno oblikovanog rješenja problema. Posljedice logičkih pogrešaka su modeli koji su zapisani u skladu s pravilima sustava znakovlja, ali predstavljaju pogrešno stanje modeliranog prostora. Drugu skupinu čine slučajne pogreške koje su posljedica nedovoljnog poznavanja sustava znakovlja ili nepažnje tijekom zapisa modela. Svojstvo podložnosti pogreškama (engl. *error-proneness*) odnosi se na mjeru otpornosti sustava znakovlja prema nastanku slučajnih pogrešaka. Većina današnjih programskih jezika koji koriste slovčani oblik zapisa programa koristi uparene graničnike (engl. *paired-delimiter*) za omeđivanje programskih cjelina. Primjeri uparenih graničnika su vitičaste zagrade u programskom jeziku C i Java, odnosno *begin-end* parovi ključnih riječi u programskom jeziku Pascal za omeđivanje programskih blokova. Na sličan način gotovo svi programski jezici koriste okrugle zagrade za omeđivanje funkcijskih parametara, odnosno aritmetičkih i logičkih izraza. Programski jezici koji koriste uparene graničnike primjeri su sustava znakovlja podložnih pogreškama, jer postoji velika vjerojatnost da korisnik nenamjerno zaboravi zapisati završni graničnik na kraj programske cjeline. Suvremeni uređivači programa poboljšavaju svojstvo podložnosti pogreškama koje su uzrokovane uparenim graničnicima na način da u model automatski dodaju završni graničnik nakon što korisnik upiše početni.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Tijekom oblikovanja modela potrošaču prilagođenog programiranja, mjera podložnosti pogreškama iskorištena je za definiranje svojstva *blokovske izgradivosti primjenskog programa* koje je opisano u poglavlju 6.2.1 i svojstva *izgradivosti primjenskog programa putem grafičkog korisničkog sučelja* koje je opisano u poglavlju 6.2.7. Izgradnjom programa zasnovanom na povezivanju gotovih programskih blokova, smanjuje se mogućnost nastanka i pojednostavljuje postupak otkrivanja logičkih pogrešaka jer se program gradi povezivanjem programskih komponenti krupne zrnatosti čija je funkcionalnost unaprijed izgrađena i provjerena. Izgradnjom programa putem grafičkog korisničkog sučelja koje potrošača vodi kroz postupak izgradnje programa, smanjuje se ili se u potpunosti izbjegava mogućnost nastanka leksičkih i sintaksnih pogrešaka koje su česte kod programskih jezika zasnovanih na slovčanom zapisu.

3.5.11 Složenost rasuđivanja

Povezivanjem osnovnih elemenata sustava znakovlja u veće skupine nastaju složene cjeline čije je značenje određenom načinom povezivanja i međusobnim odnosima osnovnih elemenata. Kasnije tumačenje značenja nastalih cjelina zahtijeva određeni stupanj umnog djelovanja za rasuđivanje i raščlambu značenja složenog oblika. Složenost rasuđivanja (engl. *hard mental operations*) je mjera umnog napora koji je potrebno uložiti za tumačenje značenja složene cjeline zapisanog modela. Prirodnim jezikom često je moguće izreći rečenice relativno jednostavnog značenja, ali na način da je tumačenje tog značenja iznimno teško. Primjerice, rečenica

```
Bez da se ne dogodi slučaj da kosilica nije u garaži
ili ako se dogodi slučaj da nema ulja u spremniku, a
nije slučaj da ključ spremišta za ulje visi na kuki,
ne moraš kositi travu.
```

je zbog složenog uvjeta s nekoliko razina negacija iznimno teška za raščlambu i rasuđivanje značenja. Istu je tvrdnju moguće izreći rečenicom

```
Ako kosilica nije u garaži ili ako nema ulja u
spremniku, a ključ spremišta za ulje ne visi na kuki,
ne moraš kositi travu.
```

koja je znatno jednostavnija za rasuđivanje značenja.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Tijekom oblikovanja modela potrošaču prilagođenog programiranja, mjera složenosti rasuđivanja iskorištena je za definiranje svojstva *blokovske izgradivosti primjenskog programa* koje je opisano u poglavlju 6.2.1 i svojstva *višerazinskog usložnjavanja blokova* koje je opisano u poglavlju 6.2.4. Svojstvom blokovske izgradivosti primjenskog programa, složenost rasuđivanja smanjena je zbog toga što se od potrošača ne zahtijeva usredotočenost na pojedinosti unutarnje građe pojedinih blokova, nego na mogućnost njihova povezivanja u radni tijek s ostalim blokovima. Svojstvom višerazinskog usložnjavanja blokova, složenost rasuđivanja smanjena je mogućnošću postupne izgradnje primjenskog programa.

3.5.12 Postupno vrednovanje

Postupno vrednovanje (engl. *progressive evaluation*) modela zapisanog sustavom znakovlja obrnuta je primjena teorije višeprolaznog iščitavanja koja je opisana u poglavlju 3.4.3. Dok teorija višeprolaznog iščitavanja objašnjava pojavu postupnog tumačenja značenja napisanog programa, postupno vrednovanje odnosi se na pojavu postupne provjere ispravnosti rada tijekom pisanja novog programa. Svojstvo postupnog vrednovanja korisniku omogućava provjeru ispravnosti djelomično napisanog modela.

Primjer sustava znakovlja koji zadovoljava svojstvo postupnog vrednovanja je tablični proračun (engl. *spreadsheet*). S obzirom da se svakom promjenom sadržaja podatkovnih ćelija ili ćelija koje sadrže formule pokreće ponovno izračunavanje svih formula, učinak svake izmjene korisniku je odmah uočljiv. Primjer sustava znakovlja koji ne zadovoljava svojstvo postupnog vrednovanja je grafički programski jezik *LabVIEW* [85, 86] u kojem se programski modeli grade uz pomoć ćelija koje predstavljaju mjesta za obradu podataka i linija koje predstavljaju smjerove toka podataka između ćelija. Programski jezik *LabVIEW* ne dopušta pokretanje napisanog programa prije nego što su na sve spojnice svih ćelija spojene odgovarajuće podatkovne linije i prije nego što je svaka linija s oba kraja spojena na spojnicu ćelije. Sustavi znakovlja koji djelomično zadovoljavaju svojstvo postupnog vrednovanja su programski jezici poput jezika C, Java ili Pascal. Iako u tim jezicima nije dopušteno pokretanje programa u kojemu prethodno nisu definirane sve funkcije koje se u programu pozivaju, omogućeno je pokretanje programa ako su te funkcije tek djelomično ostvarene. Korisnici tih jezika iskorištavaju mogućnost gradnje kostura funkcije, odnosno funkcije koja prema pozivajućim funkcijama izlaže odgovarajuće sučelje, ali njezina unutarnja struktura još nije definirana.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Tijekom oblikovanja modela potrošaču prilagođenog programiranja, mjera postupnog vrednovanja iskorištena je za definiranje svojstva *višerazinskog usložnjavanja blokova* koje je opisano u poglavlju 6.2.4. Svojstvom višerazinskog usložnjavanja blokova omogućena je postupna izgradnja primjenskog programa kroz više koraka usložnjavanja. Na svakoj razini usložnjavanja moguće je provjeriti ispravnost rada složenog programskog bloka koji je dobiven povezivanjem skupa jednostavnijih blokova.

3.6 Utjecaj govornog jezika na doživljaj vremena

Kako bi se oblikovala programska paradigma s potrošaču pogodnim načinom za izražavanje vremenskih odnosa među programskim cjelinama, proučeni su kognitivni modeli koji utječu na doživljaj vremena. S obzirom na to da je vremenske odnose tijekom zapisivanja programskog rješenja potrebno pretvoriti u odgovarajuće prostorne odnose među elementima programskog jezika, u okviru doktorske disertacije proučeni su kognitivni modeli preslikavanja vremenskih odnosa u prostorne odnose. Istraživanja u području psihologije programiranja pokazala su da značajan utjecaj na način izražavanja vremenskih odnosa u prostoru ima govorni jezik [62]. Utjecajem govornog jezika na način razmišljanja, djelovanja i doživljavanja apstraktnih pojmova bavi se grana eksperimentalne psihologije pod nazivom jezična predodređenost (engl. *linguistic determinism*).

U govornim jezicima, za izražavanje vremenskih odnosa nerijetko se primjenjuju prostorne priložne oznake. Primjerice, u rečenici

"*Pisanje doktorske disertacije je **iza** nas, a usmena obrana **ispred** nas.*"

priložne oznake mjesta *iza* i *ispred* koriste se za označavanje trenutka koji slijedi nakon pisanja doktorske disertacije, a prethodi usmenoj obrani.

Analizom različitih govornih jezika, pokazalo se da postoje razlike u načinu na koji izvorni govornici doživljavaju vrijeme. U engleskom govornom području, za izražavanje vremenskih odnosa najčešće se koriste priložne oznake za označavanje vodoravnih prostornih odnosa, kao što su *ispred* i *iza*, odnosno *front*, *back*, *ahead* i *behind*. Slična razmatranja vrijede i za hrvatski jezik. S druge strane, u mandarinskoj inačici kineskog jezika, za izražavanje vremenskih odnosa se, osim priložnih oznaka za označavanje vodoravnih prostornih odnosa *qián* i *hòu*, odnosno *ispred* i *iza*, podjednako često koriste i priložne oznake za označavanje okomitih prostornih odnosa, kao što su *shàng* i *xià*, odnosno *iznad* i *ispod*.

Uporaba govornog jezika određuje način na koji izvorni govornici različitih jezika doživljavaju tijek vremena. Dok u engleskom i hrvatskom govornom području govornici najčešće vrijeme doživljavaju i prikazuju u vodoravnom smjeru, govornici mandarinskog kineskog jezika podjednako često vrijeme doživljavaju i prikazuju i u vodoravnom i u okomitom smjeru. Iako su ove tvrdnje zbirni rezultat statističkih podataka dobivenih pokusnim ispitivanjima, postoje zapažanja i brojni primjeri kada govornici engleskog i sličnih jezika za izražavanje vremenskih odnosa također koriste priložne oznake za označavanje okomitih prostornih odnosa.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Tijekom oblikovanja potrošaču prilagođenog programiranja, teorija jezične predodređenosti je, uz teoriju višeprolaznog iščitavanja koja je opisana u poglavlju 3.4.3, iskorištena za oblikovanje okoline za prikaz programa i modeliranje vremenskih odnosa među programskim cjelinama. Za modeliranje vremenskih odnosa koristi se dvodimenzionalna tablična ploha s dvosmjernim napredovanjem vremena. Unutar tablice, vrijeme istodobno napreduje odozgo prema dolje i slijeva na desno. Događaj koji je zapisan u promatranoj ćeliji tablice izvodi se prije svih događaja koji su zapisani u susjednim ćelijama ispod ili desno od promatrane ćelije. Praznom ćelijom označava se prekid vremenskog tijeka, čime je omogućeno modeliranje vremenski nezavisnih događaja. Dvosmjerno napredovanje vremena omogućuje jednostavno oblikovanje složenih uzoraka tijeka izvođenja programa, kao što je slijedno i istodobno izvođenje te sinkronizacija, bez potrebe za korištenjem posebnih, potrošaču nerazumljivih, sinkronizacijskih mehanizama. Zapis programa primjenom dvodimenzionalne tablične plohe opisan je u poglavlju 11.

4 Odnos potrošača prema tehnologiji

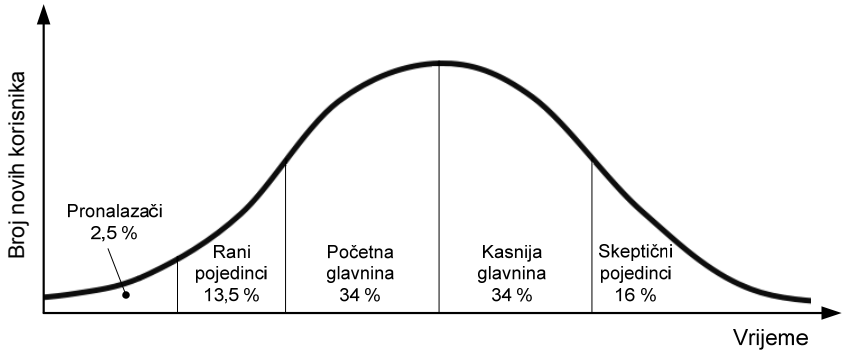
Iskustvo potrošača stečeno primjenom i rukovanjem raznoraznim oblicima tehnologijskih proizvoda i usluga je, uz spoznaje iz područja kognitivnih znanosti, drugi ključni čimbenik na koje je oslonjeno oblikovanje potrošaču prilagođenog programiranja. Potrošačima računala smatraju se ljudi koji su ovladali vještinama korištenja računalnih primjenskih programa, ali ne nužno i programiranjem računala. S gledišta potrošača, programiranje računala moguće je smatrati oblikom tehnologijske inovacije čijom je primjenom potrebno ovladati. Ako se potrošačima omogući programiranje računala primjenom istih elemenata, tehnika i pravila kojima su već ovladali i koje su navikli upotrebljavati tijekom primjene računala, programiranje računala ne zahtijeva dodatni oblik učenja. Osim toga, iskorištavanje poznatih koncepata ima povoljni psihološki učinak jer smanjuje ili u potpunosti uklanja otpor potrošača uzrokovan tromošću tijekom privikavanja na novi oblik tehnologije.

U ovom poglavlju analizirana su znanja i vještine potrošača stečene dugogodišnjom primjenom računalnih primjenskih programa te mogućnosti njihova iskorištavanja za oblikovanje programiranja prilagođenog potrošaču. U poglavlju 4.1 provedena je analiza odnosa potrošača prema tehnologiji te postupak prilagodbe potrošača na pojavu tehnologijskih inovacija. Razmatran je tijek prilagodbe potrošača na nadovezujuće i korjenite inovacije kao dva osnovna oblika tehnologijskih inovacija. U poglavlju 4.2 prikazan je odnos potrošača prema primjeni računala. Prikazano je smanjenje složenosti međudjelovanja čovjeka i računala kroz povijest razvoja računalnih sustava, uz istodobni porast vještina potrošača uzrokovan sve raširenijom primjenom računalnih programa. U poglavlju 4.3

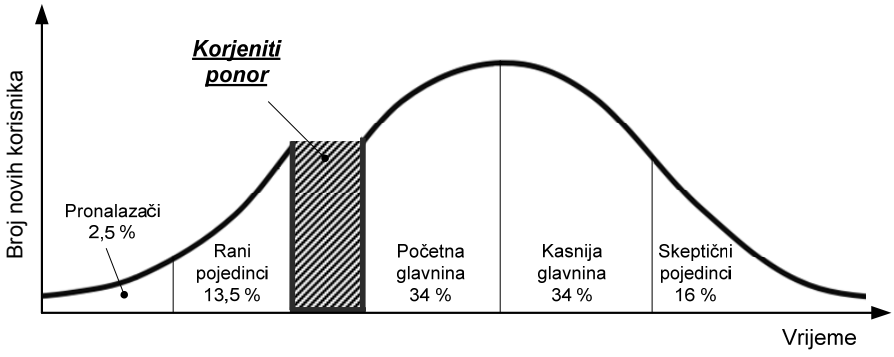
prikazan je odnos potrošača prema programiranju. Uspoređena su znanja i vještine potrošača sa složenošću programiranja računala te su predložene smjernice za iskorištavanje stečenih vještina tijekom oblikovanja programiranja prilagođenog potrošaču.

4.1 Potrošači i tehnologijske inovacije

Iz teorije marketinga i istraživanja tržišta koje se provodi prije plasmana novog proizvoda ili usluge na tržište poznato je da se prihvaćanje tehnologijskih inovacija podvrgava normalnoj razdiobi koja je prikazana grafovima na slici 4.1 [66, 67]. S obzirom na stupanj inovativnosti, tehnologijske inovacije moguće je podijeliti na nadovezujuće (engl. *continuous innovation*) i korjenite inovacije (engl. *disruptive innovation*). Nadovezujuće inovacije svojstvene su za tehnologijska rješenja koja se na tržište plasiraju kao unaprjeđenje već uhodane tehnologije, usluge ili proizvoda. S druge strane, korjenite inovacije svojstvene su uslugama ili proizvodima čija primjena unosi značajnu promjenu u radne i životne navike korisnika.



a) Usvajanje nadovezujuće inovacije



b) Usvajanje korjenite inovacije

Slika 4.1. Odziv potrošača na pojavu tehnologijskih inovacija na tržištu

Slikom 4.1.a prikazan je postupak usvajanja nadovezujuće inovacije, dok je slikom 4.1.b prikazan postupak usvajanja korjenite inovacije. Na vodoravnim osima prikazano je

vrijeme proteklo od trenutka plasmana nove tehnologije na tržište, dok je na okomitim osima prikazan priljev novih korisnika kroz vrijeme. Površina ispod krivulje pokazuje ukupni broj korisnika nove tehnologije do promatranog trenutka u vremenu.

Prema krivulji prikazanoj na slici 4.1.a, postupak prihvaćanja nadovezujuće tehnologijske inovacije započinje primjenom nove tehnologije od strane razmjerno malog broja korisnika, najčešće samih inovatora. Nakon početne faze, novu tehnologiju počinje primjenjivati nešto veći, ali još uvijek razmjerno mali broj korisnika koji su skloni eksperimentiranju s novim stvarima. Tek u trećoj fazi dolazi do masovnije primjene tehnologije. Primjena tehnologije doživljava puni zamah u četvrtoj fazi nakon što su mogućnosti i prednosti njezine primjene već dokazane. Naposljetku, novu tehnologiju počinju primjenjivati i razmjerno male skupine skeptičnih pojedinaca, ponajviše zbog toga što ih okolnosti u kojima žive i rade s vremenom na to primoravaju.

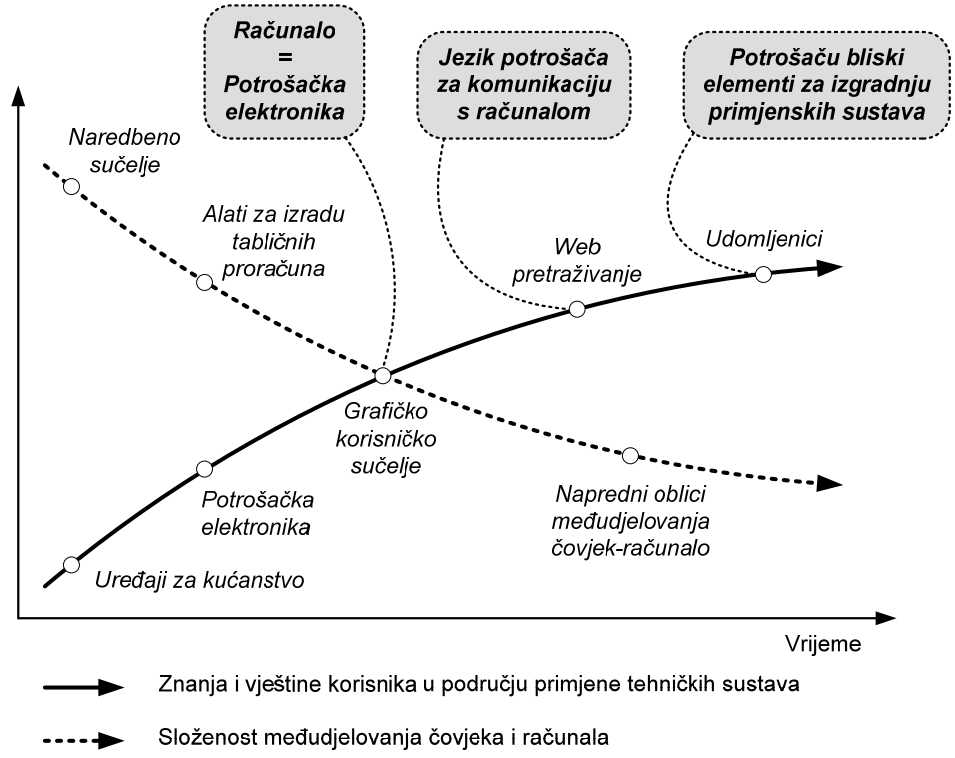
Iz grafičkog prikaza na slici 4.1.a vidljiva je tromost korisnika pri prelasku na primjenu nove tehnologije ili na novi način primjene postojeće tehnologije. Da bi se smanjilo vrijeme potrebno za masovnu primjenu inovacije, nova tehnologijska rješenja se na tržište često plasiraju kao unaprjeđenje neke već uhodane tehnologije. S druge strane, ako inovacija unosi korjenitu promjenu u način primjene tehnologije ili način na koji nova tehnologija utječe na radne i životne navike korisnika, odnosno ako se radi o korjenitoj inovaciji, onda je vrijeme potrebno za usvajanje nove tehnologije među širom skupinom korisnika dodatno produženo. U slučaju korjenite inovacije uočljiva je pojava korjenitog ponora između druge i treće faze primjene tehnologije, kao što je istaknuto na slici 4.1.b. Za korjenite inovacije je svojstveno da je potrebno znatno više vremena i sredstava za promidžbu tehnologije na tržištu kako bi se potaknula njezina masovna primjena.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Istaknute razlike između nadovezujućih i korjenitih inovacija iznimno su važne za definiranje područja programiranja prilagođenog potrošaču. S obzirom da su ključne značajke programiranja prilagođenog potrošaču dostupnost i prikladnost za primjenu od strane najšireg kruga potrošača računalnih sustava, postupak programiranja potrebno je oblikovati uz maksimalno oslanjanje na postojeće vještine potrošača stečene višegodišnjim iskustvom u primjeni računala. Na taj način je programiranje moguće oblikovati kao nadovezujuću inovaciju koja ubrzava postupak usvajanja metodologije među potrošačima.

4.2 Potrošači i primjena računala

Osvrtom na način rukovanja i upravljanja tehničkim sustavima koje su ljudi koristili kroz povijest čovječanstva, vidljivo je da je gotovo svaki poznati tehnički sustav opremljen odgovarajućim sučeljem putem kojeg se ostvaruje međudjelovanje s čovjekom. Sučelja tehničkih sustava prema čovjeku pojavljuju se u raznim oblicima, od najstarijih mehaničkih ostvarenih u sklopovlju do suvremenih elektroničkih ostvarenih uglavnom u obliku računalnih programa. Ljudi su iskustvom stečenim kroz primjenu raznovrsnih mehaničkih i elektroničkih uređaja usvajali način rukovanja tehničkim sustavima. U općenitom slučaju, moguće je reći da su ljudi kroz primjenu tehničkih sustava putem odgovarajućih sučelja učili i usvojili jezik komunikacije s tehničkim sustavima. Slikom 4.2 prikazana je usporedba znanja i vještina potrošača sa složenošću međudjelovanja s računalnim sustavima.



Slika 4.2. Usporedba znanja i vještina potrošača sa složenošću međudjelovanja s računalnim sustavima

Iz slike 4.2 vidljiv je porast usvojene količine vještina i znanja kod potrošača koja su potrebna za rukovanje različitim oblicima tehničkih sustava. Postupno proširivanje vještina i znanja kod potrošača posljedica je tehnologijskog napretka koji je potrošačima omogućio jednostavnu primjenu čak i najnaprednijih oblika tehničkih sustava. Jednostavnost primjene računala postignuta je uvođenjem grafičkog korisničkog sučelja za potrebe međudjelovanja

čovjeka i računala. S druge strane, masovnu uporabu računala i nagli priljev broja potrošača potaknula je pojava primjenskih programa za *World Wide Web*.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Na slici 4.2 istaknute su tri ključne točne kojima su stvoreni uvjeti za oblikovanje programiranja prilagođenog potrošaču. Prva ključna točka predstavlja prekretnicu koja je omogućila široku primjenu računala, odnosno prelazak računala iz područja profesionalne u područje potrošačke elektronike. Upravljanje računalnim sustavima primjenom naredbenog sučelja zahtijevalo je poznavanje posebnih naredbi operacijskih sustava za upravljanje datotekama te za pokretanje, podešavanje, uporabu i zaustavljanje primjenskih programa. Pojavom alata za izradu tabličnih proračuna (engl. *spreadsheet*) [87, 89], učinjen je značajan korak prema razdoblju široke primjene računala među potrošačima, naročito u poslovanju. Alati za izradu tabličnih proračuna su najčešće korištena skupina programskih alata u povijesti računalnih sustava [129]. Prekretnicom koja je omogućila primjenu računala među širokom zajednicom potrošača smatra se početak primjene grafičkog korisničkog sučelja (engl. *graphical user interface – GUI*) [16, 17] za upravljanje operacijskim sustavima i primjenskim programima.

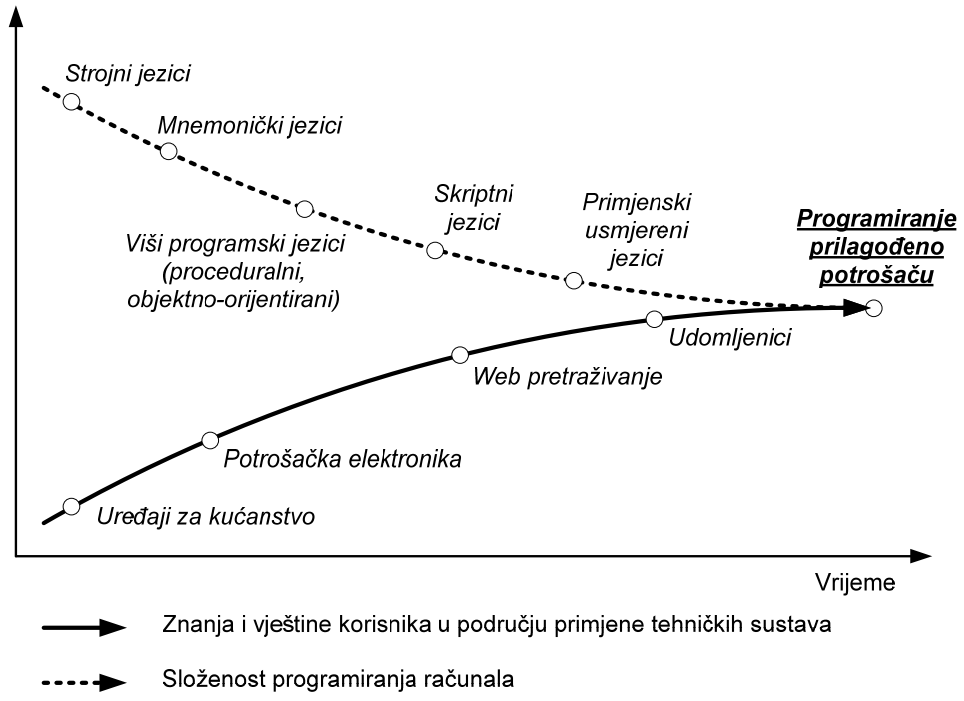
Druga ključna točka predstavlja pojavu skupine primjenskih programa koja je imala najsnažniji utjecaj i najznačajniji doprinos masovnoj primjeni računala u svojstvu potrošačke elektronike. To je skupina primjenskih programa zasnovanih na sustavu *World Wide Web* s kojima potrošači ostvaruju međudjelovanje putem preglednika *World Wide Web* sadržaja. Međudjelovanje potrošača s računalom putem grafičkog korisničkog sučelja primjenskih programa prikazanih u pregledniku *World Wide Web* sadržaja u današnje je vrijeme moguće smatrati jezikom komunikacije potrošača s računalom.

Treća ključna točka predstavlja pojavu udomljenika, odnosno priručnih primjenskih programa za izvođenje u pregledniku *World Wide Web* sadržaja. Svojstvo udomljenika je mogućnost njihova udomljavanja unutar proizvoljne *World Wide Web* stranice koja se naziva udomiteljskom stranicom. Izborom većeg broja nezavisnih udomljenika različitih funkcionalnosti i njihovim postavljanjem unutar zajedničke udomiteljske stranice, moguće je oblikovati okolinu u kojoj je primjenom skupa udomljenika moguće obavljati složene poslove po volji potrošača. S obzirom da se udomljenicima rukuje na isti način kao i *World Wide Web* stranicama, njihovom pojavom nastali su potrošaču bliski elementi koje je moguće iskoristiti za izgradnju proizvoljnih primjenskih programa. U kontekstu programiranja prilagođenog potrošaču, udomljenike je moguće smatrati potrošaču bliskim

komponentama čijim je povezivanjem u cjelinu moguće graditi primjenske programe po volji potrošača.

4.3 Potrošači i programiranje

Iako su primjenom grafičkog korisničkog sučelja prikazanog u pregledniku *World Wide Web* sadržaja kao središnjeg oblika međudjelovanja između čovjeka i računala te pojavom udomljenika kao potrošaču bliskih komponenata za gradnju primjenskih programa stvoreni preduvjeti za pronalazak potrošaču bliske metodologije programiranja računala, programiranje još uvijek nije doseglo točku u kojoj bi postalo dostupno širokom krugu potrošača. Osnovni razlog za takvo stanje je činjenica da su se programski jezici i programske paradigme razvijale u apstraktnom prostoru koji zahtjeva način razmišljanja i djelovanja koji nije vezan uz uobičajene postupke koji se koriste tijekom primjene računala.



Slika 4.3. Usporedba znanja i vještina potrošača sa složenošću programiranja računala

Slikom 4.3 prikazana je usporedba znanja i vještina potrošača sa složenošću programiranja računala. Razvijajući se od strojnih, preko mnemoničkih i jezika opće namjene, do primjenski usmjerenih jezika, programski jezici postajali su sve jednostavniji za primjenu. Pojednostavljenje primjene programskih jezika očitovala se u sve većem odmaku programskih paradigmi od arhitekture i sklopovske građe računala te težnji prema sve većoj usredotočenosti programera prema primjenskim svojstvima problema koji se rješava.

Usprkos tom značajnom pojednostavljenju, programeri i u današnje vrijeme prolaze dugotrajne i skupe oblike obrazovanja kako bi se osposobili za programiranje računala. Znanja i vještine koje se stječu iskustvom kroz primjenu računala nisu primjenjiva za programiranje računala.

Primjena za oblikovanje potrošaču prilagođenog programiranja

Cilj programiranja prilagođenog potrošaču je pronalazak novih oblikovnih postupaka i programskih jezika za daljnje smanjenje složenosti programiranja računala. Kao sljedeći važan korak u razvoju programiranja nazire se programska paradigma koja je maksimalno oslonjena na znanja i vještine stečene kroz primjenu računala, bez potrebe za dodatnim obrazovanjem potrošača. U programiranju koje je prilagođeno potrošaču, udomljenike je moguće iskoristiti u svojstvu osnovnih elemenata za izgradnju složenih primjenskih programa po volji potrošača, a radnje nad elementima grafičkog korisničkog sučelja udomljenika u svojstvu programskog jezika. Udomljenici u svojstvu gradivnih komponenti i radnje nad elementima grafičkog korisničkog sučelja u svojstvu programskog jezika dva su ključna čimbenika na kojima se zasniva ostvarenje potrošaču prilagođenog programiranja predloženo u okviru ove doktorske disertacije.

5 Razvoj primjenskih programa prilagođen krajnjem korisniku

Uključivanje krajnjih korisnika u razvoj računalnih primjenskih programa već je duže vrijeme predmet istraživanja u području računalnih znanosti, naročito programskog inženjerstva. Pojam krajnjih korisnika koristi se za stručnjake u određenom području ljudske djelatnosti koji za obavljanje posla koriste računalne primjenske programe, a nisu školovani za njihov razvoj. Istraživanjem metodologija i postupaka za uključivanje krajnjih korisnika u razvoj primjenskih programa bavi se grana programskog inženjerstva koja se naziva razvojem prilagođenim krajnjem korisniku (engl. *end-user development – EUD*) [19, 20, 21], odnosno programiranjem prilagođenim krajnjem korisniku (engl. *end-user programming – EUP*) [68]. Dok se područje programiranja prilagođenog krajnjem korisniku ograničava na programsko znakovlje i tehnike programiranja pogodne za primjenu od strane krajnjeg korisnika, razvoj prilagođen krajnjem korisniku obuhvaća i ostale elemente životnog vijeka programske potpore, kao što su oblikovanje, ispitivanje i održavanje. U okviru doktorske disertacije, proučene su programske paradigme za razvoj primjenskih programa od strane krajnjeg korisnika. Pojedini elementi tih paradigmi ocijenjeni su s obzirom na prikladnost za oblikovanje programiranja prilagođenog potrošaču.

Metodologije, tehnike i razvojni alati prilagođeni krajnjem korisniku oblikuju se na način da olakšavaju razvoj primjenskih programa iz određenog područja primjene. Usprkos tome što je razvojem prilagođenim krajnjem korisniku obuhvaćen širok opseg metodologija, tehnika i razvojnih alata za različita područja ljudske djelatnosti, razvoj prilagođen krajnjem korisniku prevladavajuću je ulogu zauzeo u dva područja primjene računalnih primjenskih programa. Razvoj primjenskih programa za potporu znanstvenim istraživanjima te razvoj

primjenskih programa za potporu poslovanju dva su područja programskog inženjerstva u kojima krajnji korisnici primjenskih programa najčešće sudjeluju i u svojstvu graditelja programa.

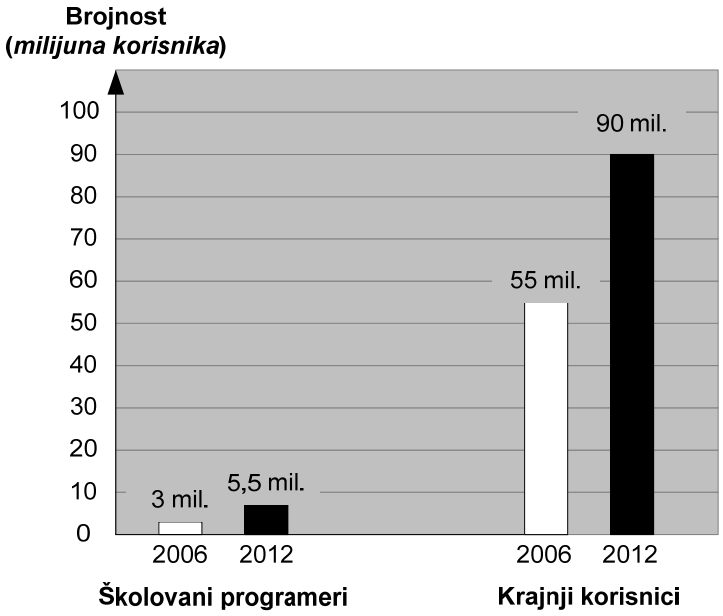
Metodologije, tehnike, postupci i programski alati za razvoj primjenskih programa prilagođen krajnjem korisniku dijele se u pet osnovnih razreda koji obuhvaćaju programiranje primjenom grafičkih oznaka, programiranje primjenom tabličnih proračuna, programiranje primjenom obrazaca, programiranje na pokaznom primjeru i programiranje prirodnim jezikom. U ovom poglavlju opisane su glavne značajke svakog od tih pet osnovnih razreda te njihovi najznačajniji predstavnici. U poglavlju 5.1 prikazani su statistički podaci s usporedbom broja krajnjih korisnika uključenih u razvoj primjenskih programa u odnosu na broj školovanih programera. Iz prikazanih statističkih podataka moguće je iščitati važnost područja razvoja primjenskih programa koji je prilagođen krajnjem korisniku. U poglavlju 5.2 opisane su glavne značajke svakog od pet osnovnih razreda programskih paradigmi za razvoj prilagođen krajnjem korisniku te programski alati koji su tipični predstavnici pojedinog razreda. U poglavlju 5.3 prikazana je ocjena prikladnosti pojedinih elemenata tih razreda za ostvarenje programske paradigme prilagođene potrošaču.

5.1 Krajnji korisnici u ulozi graditelja primjenskih programa

Osnovni razlog i poticaj krajnjim korisnicima za upuštanje u samostalni razvoj primjenskih programa je izrada posebno prilagođenih, odnosno poosobljenih inačica programske potpore koje zbog razmjernog uskog područja primjene nisu dobavljive s tržišta. Razvoj poosobljenih primjenskih programa razlikuje se od uobičajenog razvoja primjenskih programa po broju korisnika i očekivanom životnom vijeku pojedinog izgrađenog primjerka programa [68]. S obzirom na poosobljenu prirodu, takvi primjenski programi najčešće su namijenjeni uporabi od strane pojedinaca ili vrlo uskog kruga korisnika. S druge strane, poticaj za oblikovanje posebno prilagođenih primjenskih programa su zgodom oblikovane i neplanirane situacije koje traju određeno vrijeme i nakon toga prestaju. Prestanak djelovanja zgodom oblikovanih ili neplaniranih situacija nerijetko povlači i prestanak potrebe za primjenom za tu priliku oblikovanog primjenskog programa. Očekivani životni vijek poosobljenih primjenskih programa je, stoga, neusporedivo kraći od životnog vijeka primjenskih programa za opću namjenu. S druge strane, neplanirane situacije koje zahtijevaju postojanje posebno prilagođenih primjenskih programa često iziskuju vrlo kratko vrijeme za isporuku programa. Mali broj korisnika, kratki životni vijek te kratko vrijeme isporuke čimbenici su koji nepovoljno utječu na ekonomsku isplativost razvoja poosobljenih

primjenskih programa. Za potrebe razvoja poosobljenih primjenskih programa se, stoga, primjenjuje samostalni razvoj primjenskih programa od strane krajnjih korisnika.

Unazad nekoliko godina, proveden je niz istraživanja s ciljem utvrđivanja broja krajnjih korisnika koji samostalno sudjeluju u razvoju primjenskih programa. Slika 5.1 prikazuje usporedbu broja krajnjih korisnika koji samostalno sudjeluju u razvoju primjenskih programa u odnosu na broj školovanih programera u Sjedinjenim Američkim Državama u 2006. godini [71]. Na osnovi rezultata istraživanja izrađuju se procjene porasta broja krajnjih korisnika u svojstvu samostalnih razvijatelja primjenskih programa za buduće razdoblje. Osim podataka za 2006. godinu, slikom 5.1 prikazana su i predviđanja broja krajnjih korisnika u svojstvu samostalnih razvijatelja primjenskih programa za 2012. godinu.



Slika 5.1. Usporedba broja krajnjih korisnika koji samostalno sudjeluju u razvoju primjenskih programa u odnosu na broj školovanih programera u Sjedinjenim Američkim Državama

Rezultati istraživanja pokazuju da je u 2006. godini u Sjedinjenim Američkim Državama broj krajnjih korisnika u svojstvu samostalnih razvijatelja primjenskih programa dosegao 55 milijuna u odnosu na 3 milijuna školovanih programera [71]. Najveći udio u ukupnom broju krajnjih korisnika u svojstvu samostalnih razvijatelja primjenskih programa zauzimaju korisnici tabličnih proračuna. Programske alate za izradu tabličnih proračuna u današnje vrijeme uspješno primjenjuje više od 45 milijuna krajnjih korisnika [72]. Prihvaćenost programskih alata za izradu tabličnih proračuna među skupinom krajnjih korisnika posljedica je bliskosti funkcijske paradigme koja se koristi za izradu te vrste primjenskih programa s naučenim doživljajem matematičkih formula. Primjenski programi

izgrađeni u obliku tabličnih proračuna široku primjenu pronalaze u području vođenja poslovanja. Ostali oblici programiranja sa značajnim udjelom u području razvoja primjenskih programa od strane krajnjih korisnika su pisanje upita za pristup bazama podataka, pisanje skriptnih programa za podešavanje radne okoline u ljusci operacijskog sustava te izrada *World Wide Web* stranica primjenom alata zasnovanih na grafičkim predlošcima.

Predviđanja za budućnost ukazuju da će broj krajnjih korisnika do 2012. godine dosegnuti brojku od 90 milijuna [71], dok će broj školovanih programera porasti na 5,5 milijuna. Jedan od najsnažnijih poticaja za porast broja krajnjih korisnika koji samostalno sudjeluju u razvoju programskih sustava jest rasprostranjenost globalne računalne mreže Internet i sveprisutnost primjenskih programa zasnovanih na sustavu *World Wide Web*. Gotovo svaki korisnik koji u današnje vrijeme raspolaže osobnim računalom s pristupom na mrežu Internet, za ispunjavanje svojih svakodnevnih radnih i osobnih obveza koristi neki od oblika elektroničkih usluga dostupnih putem te mreže.

5.2 Razredba postupaka za razvoj primjenskih programa od strane krajnjeg korisnika

Metodologije, tehnike, postupci i programski alati za razvoj primjenskih programa prilagođen krajnjem korisniku dijele se u pet osnovnih razreda koji obuhvaćaju programiranje primjenom grafičkih oznaka, programiranje primjenom tabličnih proračuna, programiranje primjenom obrazaca, programiranje na pokaznom primjeru i programiranje prirodnim jezikom.

5.2.1 Programiranje primjenom grafičkih oznaka

Programiranje primjenom grafičkih oznaka (engl. *visual programming*) u području programskog inženjerstva počelo se primjenjivati početkom osamdesetih godina dvadesetog stoljeća [80]. Osnovna značajka ove vrste programiranja je primjena grafičkih elemenata u postupku izgradnje programske potpore koji zamjenjuju ili upotpunjuju programiranje zasnovano na slovčanom zapisu. Grafički elementi koji se koriste za izgradnju programa pojavljuju se u obliku slika ili geometrijskih likova koji predstavljaju čvorove za obradu informacija. Pojedini grafički elementi povezuju se usmjerenim granama koje predstavljaju tijek izvođenja programa ili tok podataka između elemenata obrade.

Pojava programiranja primjenom grafičkih oznaka u odnosu na programiranje slovčanim programskim jezicima izazvala je u području programskog inženjerstva preokret

sličan onome koji je pojava viših programskih jezika izazvala u odnosu na programiranje strojnim jezicima. Budući da je manje podložno sintaksnim pogreškama, programiranje primjenom grafičkih oznaka u određenim okolnostima poboljšava razumljivost i olakšava održavanje napisanih programa te podiže razinu učinkovitosti programera. Tablicom 5.1 prikazani su programski jezici i razvojni alati koji su tipični predstavnici programiranja primjenom grafičkih oznaka. Unutar tablice, programski jezici i razvojni alati razvrstani su prema području primjene.

Tablica 5.1. Tipični predstavnici programskih jezika i razvojnih alata u području programiranja primjenom grafičkih oznaka

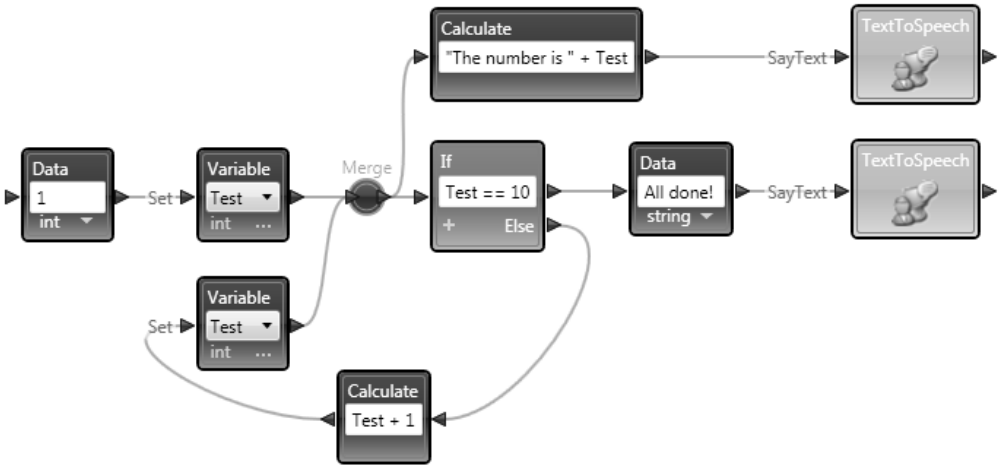
Područje primjene	Predstavnici razreda
Opća namjena	AppWare (pojavljuje se i pod nazivom MicroBrew) Befunge (dvodimenzionalni slovčani jezik) Unified Modeling Language (UML) DRAKON — Prograph Flow — Squeak Lava — Subtext Lily — Tersus Limnor — ThingLab
Računalna grafika i animacija	Cantata — Quest3D MST Workshop — virtools OpenDX — Vsxu Quartz Composer
Sustavi upravljanja	Ladder logic — Sequential function chart Flowcode — VisSim SCADE — EICASLAB Simulink
Robotika	Microsoft Visual Programming Language
Obrada višemedijskih sadržaja	AudioMulch — Scala Multimedia Max — SynthMaker OpenMusic — SynthEdit Reaktor — Vvvv
Raspodijeljeni računalni sustavi	PointDragon
Ugrađeni računalni sustavi	Ptolemy
Paralelno programiranje	CODE
Poslovni informacijski sustavi	Business Process Modeling Notation (BPMN) OutSystems language
Izrada računalnih igara	AgentSheets

Potpora odlučivanju	Analytica
Ispitivanje programske potpore	eXpecco
Projektiranje elektroničke instrumentacije	LabVIEW (G) VEE
Prevođenje prirodnog jezika	Kwikpoint
Obrazovni programski jezici	Baltie ToonTalk

Slikom 5.2 prikazan je primjer izrade računalnog programa primjenom programskog jezika zasnovanog na grafičkim oznakama. Dok slika 5.2.a prikazuje primjer programa izrađenog na uobičajeni način, primjenom programskog jezika zasnovanog na slovčanom zapisu, slika 5.2.b prikazuje istovjetni program izrađen programiranjem primjenom grafičkih oznaka. Prikazani primjer izrađen je u programskom jeziku zasnovanom na grafičkim oznakama pod nazivom *Microsoft Visual Programming Language* koji je sastavni dio programskog sustava *Microsoft Robotics Studio* [84].

```
int Test = 1;
do {
  TextToSpeech.SayText("The number is" + Test);
  Test = Test + 1;
}
while (Test == 10);
TextToSpeech.SayText("All done!");
```

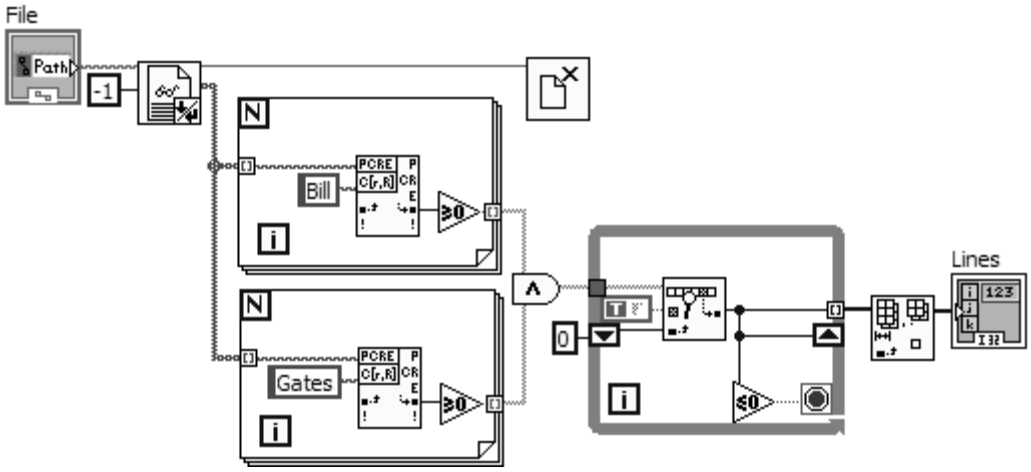
a) Program izrađen primjenom programskog jezika zasnovanog na slovčanom zapisu



b) Program izrađen primjenom programskog jezika zasnovanog na grafičkim oznakama

Slika 5.2. Primjer izrade programa opće namjene programiranjem primjenom grafičkih oznaka

Usporedbom istovjetnog računalnog programa zapisanog slovčanim i grafičkim oznakama, u programu zapisanom grafičkim oznakama moguće je uočiti veću preglednost toka podataka i tijeka izvođenja programa. Primjerice, dok je programska petlja u programu zapisanom grafičkim oznakama jasno uočljiva skupom usmjerenih grana koje čine zatvoreni put između mjesta za obradu podataka, u programu zapisanom slovčanim oznakama petlju je moguće prepoznati isključivo po ključnim riječima *do* i *while*. Ovo i slična svojstva čine programiranje primjenom grafičkih oznaka prikladnije za krajnjeg korisnika jer struktura programa nije određena ključnim riječima, već je vidljiva iz prostornog razmještaja programskih elemenata. S druge strane, upravo uočljivost i preglednost, koja se smatra odlikom programiranja primjenom grafičkih oznaka, za veće programe postaje jedan od glavnih nedostataka ove tehnike jer u napisanom programu najčešće nije moguće potpuno izbjeći križanje usmjerenih grana. Jedan od postupaka kojima se pokušava nadomjestiti ovaj nedostatak je objedinjavanje skupa programskih elemenata u cjeline na višim razinama složenosti koje je u nastavku moguće koristiti kao osnovne programske elemente. Drugi način poboljšanja preglednosti programa je označavanje usmjerenih grana različitim bojama ili različitim nijansama sjenčanja.



Slika 5.3. Primjer izrade programa posebne namjene programiranjem primjenom grafičkih oznaka

Iako je upotrebljivo za izradu programa opće namjene, programiranje primjenom grafičkih oznaka naročito je pogodno za primjenu od strane krajnjeg korisnika ako se u programskom jeziku koriste posebno prilagođene grafičke oznake koje su svojstvene određenom području primjene. Primjer programskog alata zasnovanog na programiranju primjenom grafičkih oznaka koji je posebno prilagođen za izradu i simulaciju električnih shema elektroničke instrumentacije je alat *LabVIEW* [85, 86]. Primjer izrade programa posebne namjene u programskom alatu *LabVIEW* prikazan je slikom 5.3. U programskom

alatu *LabVIEW*, mjesta obrade podataka predstavljaju raznovrsne elektroničke uređaje i komponente, kao što su otpornici, pojačala, logički sklopovi ili osciloskopi, dok usmjerene grane predstavljaju električne vodove.

5.2.2 Programiranje primjenom tabličnih proračuna

Primjena tabličnog proračuna (engl. *spreadsheet*) u računovodstvu je bila poznata davno prije pojave prvih elektroničkih računala [87]. List papira s ucrtanim recima i stupcima u obliku dvodimenzionalne tablice od davnina se koristio za proračun i vođenje evidencije o prihodima, rashodima, porezima i ostalim podacima povezanim s financijskim poslovanjem poduzeća. Umjesto tabličnih proračuna s ucrtanom tablicom na listu papira, u današnje vrijeme koriste se osuvremenjene inačice tabličnih proračuna u obliku programa pogodnih za izvođenje na računalu. Tablicom 5.2 prikazani su programski jezici i razvojni alati koji su tipični predstavnici programiranja primjenom tabličnih proračuna. Unutar tablice, programski jezici i razvojni alati razvrstani su prema području primjene.

Tablica 5.2. Tipični predstavnici programskih jezika i razvojnih alata u području programiranja primjenom tabličnih proračuna

Područje primjene	**Predstavnici razreda**	
Numerički proračuni (alati za lokalno računalo)	VisiCalc Perfect Calc Lotus 1-2-3 Microsoft Excel Apple Numbers OpenOffice.org Calc Gnumeric Lotus Symphony KSpread ZCubes-Calci Advantage	Lotus Improv Javelin Software Lotus Jazz MultiPlan Borland Quattro Pro Silk SuperCalc Wingz Target Planner Calc Spreadsheet 2000 3D-Calc
Numerički proračuni (mrežni alati)	Google Docs & Spreadsheets Simple Spreadsheet wikiCalc BadBlue EditGrid	Expresso spreadsheet ThinkFree Calc ZCubes – Calci Zoho Office Suite Resolver One
Kompozicija usluga	SpreadATOR Husky	

Tvorcem elektroničke inačice tabličnog proračuna smatra se Daniel Bricklin koji je 1978. godine kao student *Harvard Business School* na američkom sveučilištu *Harvard* predstavio idejno rješenje za tu vrstu računalnih programa. Njegove zamisli kasnije su pretvorene u računalni program *VisiCalc* [88] koji se danas smatra prvim tabličnim proračunom za elektronička računala. Iako je nekoliko istraživača, među kojima su najistaknutiji bili Richard Mattessich [89] početkom te Rene Pardo i Remy Landau krajem šezdesetih godina dvadesetog stoljeća, pokušavalo ostvariti elektroničku verziju tabličnog proračuna prije pojave alata *VisiCalc*, *VisiCalc* se smatra prvim tabličnim proračunom i prvim programom koji je naišao na široku primjenu u području osobnih računala [72].

Suvremeno doba računalnih programa zasnovanih na tabličnim proračunima obilježili su alati *Lotus 1-2-3* [90] tvrtke *IBM* i *Microsoft Excel* [91] tvrtke *Microsoft*. U najnovije vrijeme, početkom dvadeset i prvog stoljeća, pod sve jačim utjecajem primjenskih programa zasnovanih na globalnoj mreži Internet, na tržištu se pojavilo i nekoliko mrežnih inačica tabličnih proračuna, među kojima je najpoznatiji *Google Docs & Spreadsheet* [92] tvrtke *Google*.

Programiranje primjenom tabličnih proračuna zasniva se na paradigmi funkcijskog programiranja (engl. *functional programming*) [93] i predstavlja njegov najrašireniji oblik. Tablični proračuni ograničeni su na podskup paradigme funkcijskog programiranja koji se naziva funkcijskim programiranjem prvog reda (engl. *first order functional programming*) [94], što znači da se za izvođenje proračuna koriste isključivo funkcije prvog reda. Funkcijama prvog reda svojstveno je da se kao ulazni argumenti i povratni rezultati koriste isključivo vrijednosti, za razliku od funkcija viših redova u kojima je kao ulazne argumente i povratne rezultate, uz vrijednosti, moguće koristiti i funkcije. Bliskost paradigme funkcijskog programiranja koja se primjenjuje za izradu tabličnih proračuna među skupinom krajnjih korisnika u području vođenja poslovanja zasniva se na sličnosti s matematičkim formulama naučenih tijekom redovnog školovanja.

Slikom 5.4 prikazana je tipična razvojna okolina za izradu tabličnih proračuna. Tablični proračun moguće je promatrati na dvije razine. Razina programskog ostvarenja tabličnog proračuna prikazana je slikom 5.4.a, dok je razina izvođenja tabličnog proračuna prikazana slikom 5.4.b. Razina programskog ostvarenja zasniva se na tabličnoj radnoj plohi s pogledom na konstante i formule koje se koriste u proračunu. Tijekom programskog ostvarenja, korisniku je omogućeno unošenje podataka u ćelije tablice. Vrste podataka koje je dozvoljeno unositi dijele se u dvije skupine. Prvu skupinu čine konstantne vrijednosti, kao što je prikazano u stupcu A na slici 5.4.a. Drugu vrstu podataka čine matematičke ili logičke

formule koje nad unesenim konstantnim vrijednostima obavljaju određene matematičke ili logičke operacije i na osnovu njih izračunavaju nove vrijednosti. Taj je slučaj prikazan u stupcu B na slici 5.4.a. Razina izvođenja zasniva se na tabličnoj radnoj plohi s pogledom na vrijednosti koje su dobivene izračunavanjem formula zadanih na razini programskog ostvarenja.

	A	B	C	D	E
1	10	=A1*2			
2	20	=A2*2			
3	30	=A3*2			
4					
5		=B1+B2+B3			
6		=SUM(B1:B3)			
7					
8					
9					
10					

a) Radna ploha tabličnog proračuna s pogledom na formule

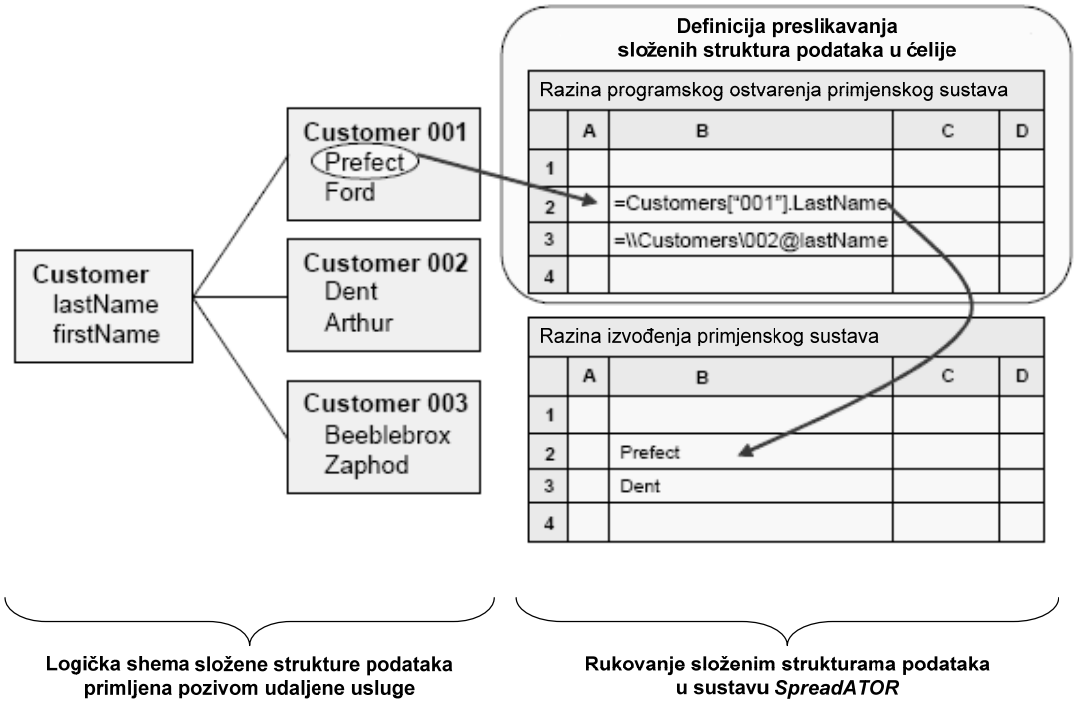
	A	B	C	D	E
1	10	20			
2	20	40			
3	30	60			
4					
5		120			
6		120			
7					
8					
9					
10					

b) Radna ploha tabličnog proračuna s pogledom na izračunate vrijednosti

Slika 5.4. Primjer razvojne okoline za izradu programa primjenom tabličnog proračuna

Osim za oblikovanje matematičkih proračuna, tablično programiranje u posljednje vrijeme nalazi primjenu i za oblikovanje računalnih programa za šire područje primjene, primjerice prikupljanje i obradu podataka dostupnih iz više udaljenih izvorišta te kompoziciju usluga. Ova dva područja primjene tabličnih proračuna potaknuta su uvođenjem uslužno-usmjerene arhitekture (engl. *service-oriented architecture – SOA*) [95, 96] u poslovne informacijske sustave [99]. Primjeri programskih alata koji objedinjuju programiranje primjenom tabličnog proračuna s uslužno-usmjerenom arhitekturom su *StrikeIron OnDemand Web Services for Microsoft Excel* [127] i *SpreadATOR* [72]. Primjenom posebno izgrađenih programskih dodataka, ovi programski alati omogućuju pozivanje udaljenih programskih usluga zasnovanih na porodici tehnologija *Web Services* [102, 103] ili udaljenih izvorišta podataka primjenom tehnologije *RSS* [126].

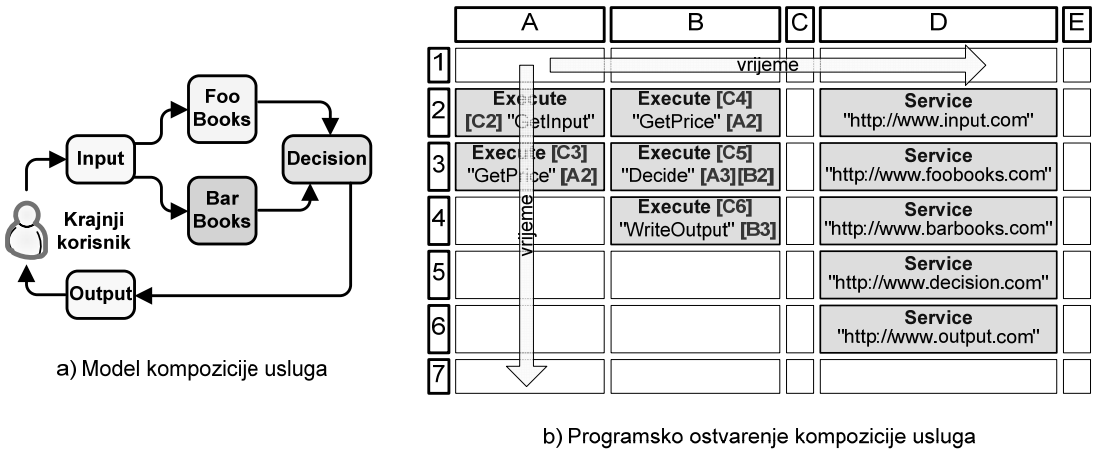
Podaci pribavljeni od udaljenih usluga ili izvorišta podataka smještaju se u ćelije tabličnog proračuna, gdje je nad njima moguće izvršiti daljnju obradu primjenom posebno oblikovanih formula za rukovanje podacima. Budući da su podaci pribavljeni od udaljenih usluga ili izvorišta podataka uglavnom organizirani u složene višerazinske strukture, formule za rukovanje podataka omogućuju izravnavanje strukture podataka kako bi se složeni podaci mogli prikazati u ćelijama tabličnog proračuna kojima je moguće prikazati isključivo jednostavne strukture podataka. Slikom 5.5 prikazan je primjer formula za rukovanje složenim strukturama podataka u sustavu *SpreadATOR*. Slično kao i kod osnovnog oblika tabličnog proračuna prikazanog na slici 5.4, na slici 5.5 istaknute su dvije razine tabličnog proračuna u sustavu *SpreadATOR*. Na razini programskog ostvarenja vidljive su formule za rukovanje složenim strukturama podataka, dok su na razini izvođenja vidljivi izravnani podaci prikazani u ćelijama tabličnog proračuna.



Slika 5.5. Primjer razvojne okoline sustava *SpreadATOR* zasnovane na objedinjavanju tabličnog proračuna i uslužno-usmjerene arhitekture

Primjena tabličnog proračuna za krajnjem korisniku prilagođeno oblikovanje kompozicije usluga prikazana je projektom *Husky* [107] i istoimenim programskim alatom koji je izrađen na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu. Slikom 5.6.a prikazan je primjer modela kompozicije usluga primjenom grafa tijeka izvođenja, dok je slikom 5.6.b prikazan primjer programskog ostvarenja tog modela u programskom sustavu *Husky*.

Programski alat *Husky* od osnovnog oblika tabličnog proračuna nasljeđuje koncept formula i naslovljavanja ćelija u svrhu dohvata podataka. Budući da se koristi isključivo za oblikovanje kompozicije usluga, ali ne i za prikaz rezultata njezina izvođenja, programski alat *Husky* omogućuje spremanje složenih struktura podataka u obliku XML poruka u ćelije tabličnog proračuna. Pozivanje usluga u programskom alatu *Husky* izvodi se posebno oblikovanom formulom *Execute* čijim se izvođenjem poziva udaljena usluga, a rezultat poziva sprema se u ćeliju u kojoj je formula zapisana. Naslovljavanjem ćelija omogućena je primjena rezultata prethodno izvedenih formula u ostalim formulama, čime je omogućena kompozicija usluga, odnosno prosljeđivanje rezultata jedne usluge kao ulaznih podataka pri pozivu druge usluge. Osim formule za poziv udaljene usluge, programski alat *Husky* sadrži i formule za definiciju pristupne točke usluge te skup formula za rukovanje udaljenim mehanizmima za komunikaciju i sinkronizaciju rada usluga.



Slika 5.6. Primjena tabličnog proračuna za oblikovanje kompozicije usluga u programskom sustavu *Husky*

Odstupanje od funkcijske paradigme i novost koji uvodi alat *Husky* u odnosu na osnovni model tabličnog proračuna je postojanje izričitog smjera napredovanja vremena u tablici. Kako bi se olakšalo oblikovanje uzoraka istodobnog izvođenja dijelova programa, vrijeme u tablici istodobno napreduje u dva smjera, odozgo prema dolje i slijeva na desno. Susjedne neprazne ćelije u tablici međusobno su vremenski zavisne, dok se vremenska nezavisnost postiže odvajanjem formula praznim ćelijama.

Široka prihvaćenost programiranja primjenom tabličnih proračuna među krajnjim korisnicima potaknula je istraživače na prilagodbu ove vrste programiranja i na ostala područja primjene. Objedinjavanje osnovnog oblika tabličnog proračuna s jezikom *Forms/3* omogućilo je primjenu proceduralne paradigme programiranja i paradigme toka podataka u tabličnom proračunu [94]. Ograničenja tabličnih proračuna na paradigmu funkcijskog

programiranja prvog reda uklonjena su uvođenjem proširenja koja omogućuju izgradnju dijela programske logike u funkcijskom jeziku *Haskell* [93, 128]. Objedinjavanjem tehnika za obradu teksta i izradu tabličnih proračuna kao dviju najčešće korištenih vrsta primjenskih programa za osobna računala, izgrađen je alat koji prikrivanjem razlika između teksta i podataka pojednostavljuje izradu poslovnih izvješća [129].

5.2.3 Programiranje primjenom obrazaca

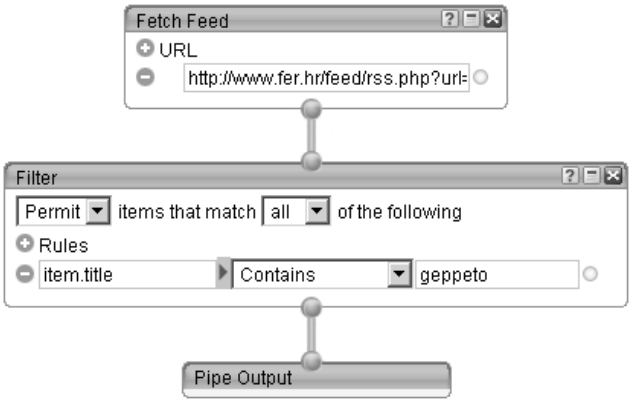
Programiranje primjenom obrazaca (engl. *form-based programming*) [131, 132] je vrsta programiranja koja krajnjem korisniku omogućava izgradnju razmjerno složenih primjenskih programa bez potrebe za poznavanjem programskog jezika i uobičajenih tehnika programiranja. Izvodi se popunjavanje skupa elektroničkih obrazaca prikazanih na zaslonu računala. Tablicom 5.3 prikazani su programski jezici i razvojni alati koji su tipični predstavnici programiranja primjenom obrazaca. Unutar tablice, programski jezici i razvojni alati razvrstani su prema području primjene.

Tablica 5.3. Tipični predstavnici programskih jezika i razvojnih alata u području programiranja primjenom obrazaca

Područje primjene	**Predstavnici razreda**	
Primjenski programi za web	Yahoo!Pipes iGoogle Gadget Maker Web Marquee Wizard	Formgenics CSS Generator Liquid XML Studio
Baze podataka	Microsoft Access QuickWizard Lattice.SPGen	Firebird Data Wizard EazyCode EazySQL
Računalna grafika i animacija	Alice WireFusion	Microsoft PowerPoint
Kompozicija usluga	Programmable Internet Environment (PIE) Axis2 (WSDL2Java + Java2WSDL)	
Automatizacija međudjelovanja čovjek-računalo	Automator HydraMouse	

Primjer izgradnje računalnog programa postupkom programiranja primjenom elektroničkih obrazaca prikazan je slikom 5.7. Na slici je prikazana izrada programa u programskom alatu *Yahoo!Pipes* [177, 178, 184, 185, 186] kojim se iz *RSS* izvorišta podataka na adresi *http://www.fer.hr/* dohvaća skup objavljenih vijesti, a potom se iz skupa vijesti izdvajaju one koje u naslovu sadrže ključnu riječ *geppeto*.

Predodžba elektroničkih obrazaca krajnjem je korisniku bliska na osnovi iskustva stečenog primjenom operacijskih sustava i primjenskih programa izloženih putem grafičkog korisničkog sučelja. Naročit doprinos širokoj prihvaćenosti elektroničkih obrazaca među krajnjim korisnicima dala je sveprisutnost primjenskih programa zasnovanih na sustavu *World Wide Web*, gdje su elektronički obrasci jedan od često primjenjivanih načina međudjelovanja korisnika s računalom. Za primjenu programiranja primjenom obrazaca dovoljno je ovladati vještinama rukovanja računalom putem grafičkog korisničkog sučelja.



Slika 5.7. Primjer izrade programa programiranjem primjenom obrazaca

Ključno svojstvo koje programiranje primjenom obrazaca čini prikladnim za krajnjeg korisnika je sličnost postupka programiranja s prethodno naučenim postupcima koji su naslijeđeni iz fizičkog svijeta. Dok su tipke, klizači, okretala, dojavne lampice ili poruke ispisane na zaslonu uređaja poznate na osnovi rukovanja raznoraznim mehaničkim uređajima, upisivanje sadržaja te odabir i označavanje polja poznato je na osnovi ispunjavanja obrazaca otisnutih na papiru.

Slika 5.8. Program sa slike 5.7, izrađen programiranjem primjenom obrazaca, u izvođenju

Na osnovi prikupljenih vrijednosti koje krajnji korisnik definira putem skupa obrazaca, pozadinska programska logika stvara računalni program pogodan za izvođenje na računalu. Program, čija je izrada prikazana na slici 5.7, za vrijeme izvođenja prikazan je na slici 5.8. Programiranje primjenom obrazaca manje se oslanja na vještine korisnika koje su svojstvene za tradicionalne oblike programiranja, a više na izdvajanje ključnih parametara čijim je objedinjavanjem s pozadinskom programskom logikom moguće dobiti program u izvodivom obliku. Zbog ovog svojstva, programiranje primjenom obrazaca naročito je pogodno za usko specijalizirana područja primjene gdje je prikladnim oblikovanjem obrazaca moguće postići zadovoljavajuću razinu općenitosti i izražajnosti razvojnog alata.

5.2.4 Programiranje na pokaznom primjeru

Programiranje na pokaznom primjeru (engl. *programming by demonstration*) [133] je vrsta programiranja koja za izradu programa ne koristi posebno oblikovane postupke, tehnike i programske jezike, već se program izrađuje na osnovi izravne primjene ciljnog računalnog sustava. Djelovanjem u prostoru korisničkog sučelja ciljnog sustava, odnosno izravnim rukovanjem ciljnim sustavom, korisnik na pokaznom primjeru pokazuje postupak rješavanja problema. Na osnovi korisnikovog djelovanja putem korisničkog sučelja ciljnog sustava, stvara se računalni program čijim je izvođenjem ciljnom sustavu omogućeno rješavanje istog ili sličnog problema bez uključenosti čovjeka. Posebno oblikovana programska potpora prati rad korisnika, snima slijed aktivnosti koje on poduzima tijekom pokaznog primjera te ih sprema u obliku računalnog programa za kasnije izvođenje.

Programiranje na pokaznom primjeru predloženo je u okviru doktorske disertacije Dana Halberta na američkom sveučilištu *University of California at Berkeley* 1984. godine [133]. Od nastanka do danas, najčešće je primjenjivano za izradu programa za automatizaciju uredskog poslovanja te programske potpore za upravljanje radom robota [134]. U posljednje vrijeme pronalazi primjenu u području strojnog upravljanja *World Wide Web* sadržajima (engl. *web automation*) [135]. Tablicom 5.4 prikazani su programski jezici i razvojni alati koji su tipični predstavnici programiranja na pokaznom primjeru. Unutar tablice, programski jezici i razvojni alati razvrstani su prema području primjene.

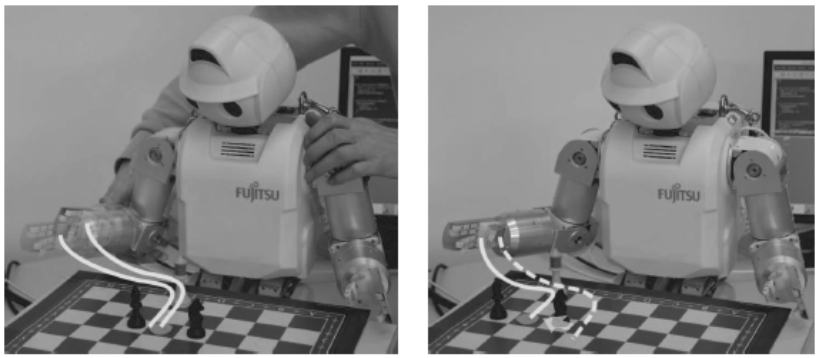
Programiranje na pokaznom primjeru naročito je pogodno za primjenu od strane krajnjeg korisnika jer je to danas jedina poznata tehnika izrade računalnih programa koja omogućava izradu programa bez korisnikove svjesne uključenosti u postupak programiranja. Tijekom programiranja, programiranje na pokaznom primjeru korisniku omogućava zadržavanje umnog režima za primjenu sustava, odnosno ne zahtijeva promjenu umnog

režima iz režima primjene u režim programiranja. Da bi izgradio računalni program za bilo koji sustav programirljiv na pokaznom primjeru, za korisnika je dovoljno poznavanje tehnike izravnog rukovanja tim sustavom.

Tablica 5.4. Tipični predstavnici programskih jezika i razvojnih alata u području programiranja na pokaznom primjeru

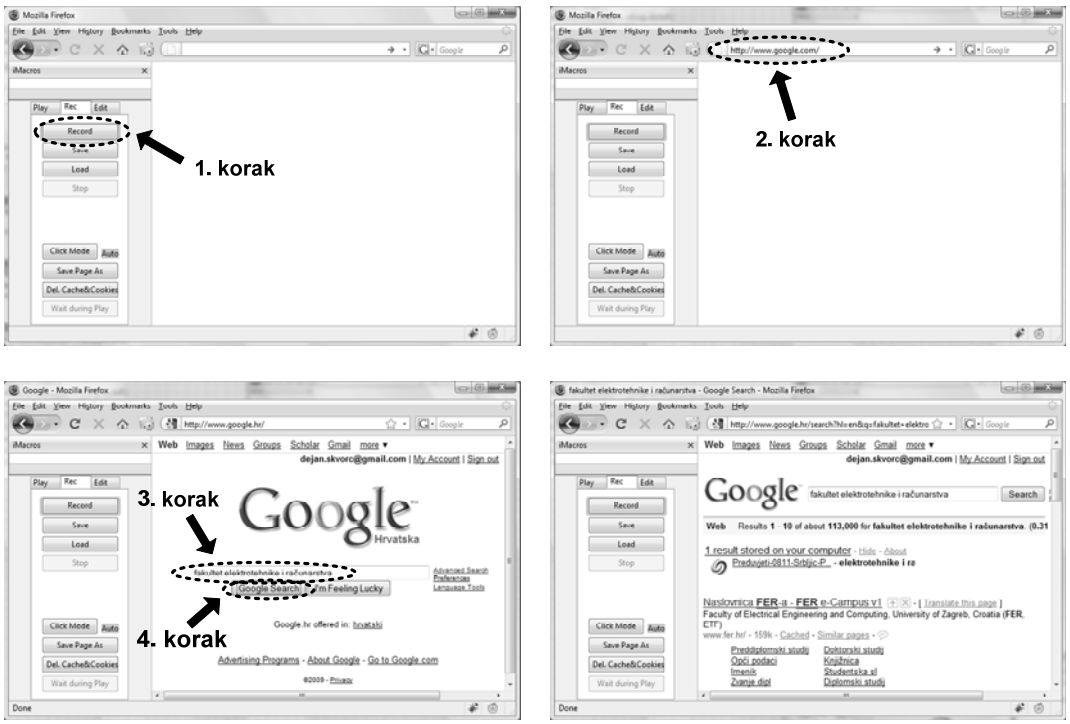
Područje primjene	**Predstavnici razreda**	
Automatizacija međudjelovanja čovjek-računalo za opću namjenu	SmartKey SuperKey KeyWorks Prokey AutoMe PTFB Pro	JitBit Macro Recorder Visual Basic for Applications WinAutomation Workspace Macro Recorder Perfect Macro Recorder Desktop Macros
Automatizacija međudjelovanja čovjek-računalo tijekom rada s web preglednikom	WebVCR iMacros Newbie iRobot Sahi WET	WebAction WebReplay WebRecorder CruiseControl LiquidTest
Obrazovni programski jezici	Stagecast Creator (prethodno Apple's Cocoa)	

Slikom 5.9 prikazan je primjer izrade programa za upravljanje pokretima robotske ruke za pomicanje figura u šahu. U prvom koraku čovjek vlastitim rukama pomiče robotske ruke kako bi pomoću njih obavio pomak šahovske figure. Za to vrijeme, programska potpora ugrađena u robot prati trajektorije koje izvodi čovjek i sprema ih u obliku računalnog programa. U kasnijim koracima, robot je na osnovi zapamćenih trajektorija sposoban samostalno obaviti pokrete robotske ruke kako bi samostalno obavio pomak šahovske figure.



Slika 5.9. Izrada programa za upravljanje pokretima robotske ruke primjenom programiranja na pokaznom primjeru

Slikom 5.10 prikazan je primjer izrade programa za strojno upravljanje *World Wide Web* sadržajima. Slikom je prikazan primjer izgradnje računalnog programa za postavljanje upita u internetsku tražilicu. Izrada programa započinje pokretanjem snimanja korisnikovih radnji, što je omogućeno pritiskom na tipku za snimanje u korisničkom sučelju programskog alata. Nakon što je snimanje pokrenuto, sve radnje koje korisnik izvodi u nastavku koriste se za oblikovanje računalnog programa. U sljedećim koracima korisnik primjenjuje uobičajene radnje kako bi putem preglednika *World Wide Web* sadržaja postavio upit u internetsku tražilicu. U drugom koraku, korisnik upisuje adresu tražilice u adresno polje preglednika. Nakon što je u pregledniku učitano korisničko sučelje tražilice, u polje za postavljanje upita korisnik upisuje ključne riječi za pretraživanje. Naposljetku, u četvrtom koraku pritiskom na tipku za pokretanje pretraživanja pokreće se pretraživanje, a dobiveni rezultati se prikazuju korisniku.

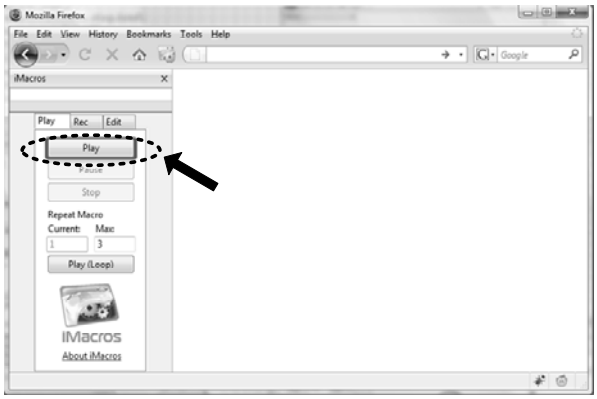


Slika 5.10. Izrada programa za strojno upravljanje *World Wide Web* sadržajima primjenom programiranja na pokaznom primjeru

Na osnovi korisnikovih radnji putem sučelja preglednika *World Wide Web* sadržaja, stvara se računalni program čiji je slovčani zapis prikazan slikom 5.11. Izvođenje programa moguće je pokrenuti pritiskom na tipku za pokretanje snimljenog programa primjenom korisničkog sučelja programskog alata koji omogućava programiranje na pokaznom primjeru. Tipični primjeri programskih alata koji primjenom programiranja na pokaznom

primjeru omogućuju izgradnju programa za strojno upravljanje *World Wide Web* sadržajima su *WebVCR* [136], *iMacros* [137], *Newbie* [138] i *iRobotSoft* [139].

```
URL GOTO=www.google.com

TAG POS=1 TYPE=INPUT:TEXT FORM=NAME:f ATTR=NAME:q
CONTENT=fakultet elektrotehnike i računarstva

TAG POS=1 TYPE=INPUT:SUBMIT FORM=NAME:f
ATTR=NAME:btnG VALUE:Google Search
```



Slika 5.11. Računalni program izgrađen na osnovi programiranja na pokaznom primjeru sa sučeljem za pokretanje izvođenja

Iako je pogodno za izgradnju programa od strane korisnika koji se po prvi puta susreću s tehnikama programiranja, programiranje na pokaznom primjeru pokazuje nedostatke u pogledu izgradnje programskih elemenata za upravljanje tijekom izvođenja programa, kao što je uvjetno grananje i uvjetno ponavljanje. Nadalje, primjenom programiranja na pokaznom primjeru, korisnik se susreće s poteškoćama tijekom izgradnje programa u kojem su pojedini dijelovi podložni izmjenama za vrijeme izvođenja, kao što su korisnički definirani ulazni podaci, vezane varijable u programu i slično. Ovi nedostaci pokušavaju se nadomjestiti tehnikama umjetne inteligencije kojima se od programa izgrađenom na skupu pokaznih primjera, primjenom postupaka strojnog zaključivanja, pokušava izgraditi opći oblik programa. Ovaj je postupak u području programiranja na pokaznom primjeru poznat pod nazivom poopćavanje (engl. *generalization*).

5.2.5 Programiranje prirodnim jezikom

Programiranje prirodnim jezikom (engl. *natural language programming*) [147, 148] u današnje vrijeme predstavlja najsuvremeniji oblik programiranja. Osnovna zamisao ove vrste programiranja je primjena prirodnog jezika za izgradnju računalnih programa. Poticaj za

istraživanje područja programiranja prirodnim jezikom jest činjenica da je za većinu ljudi prirodni jezik najjednostavniji način komunikacije i izražavanja misli. Zbog tog svojstva, programiranje prirodnim jezikom smatra se prikladnom tehnikom za izgradnju računalnih programa od strane korisnika koji ne ovladavaju vještinama potrebnim za primjenu uobičajenih vrsta programskih jezika. Tablicom 5.5 prikazani su programski jezici i razvojni alati koji su tipični predstavnici programiranja prirodnim jezikom. Unutar tablice, programski jezici i razvojni alati razvrstani su prema području primjene.

Tablica 5.5. Tipični predstavnici programskih jezika i razvojnih alata u području programiranja prirodnim jezikom

Područje primjene	Predstavnici razreda
Opća namjena	Metafor
Izrada računalnih igara	Inform 7
Baze podataka	PLANES
Upravljanje sadržajem matrica i tablica	NLC

Programiranje prirodnim jezikom oslanja se na spoznaje iz područja umjetne inteligencije i strojnog zaključivanja, naročito onih koje se odnose na strojnu obradu prirodnog jezika (engl. *natural language processing*). Poteškoće tijekom programiranja prirodnim jezikom predstavlja činjenica da većina prirodnih jezika dopušta izražavanje slobodnim stilom uz mnoštvo izuzetaka od uobičajenih pravila, za razliku od programskih jezika koje je moguće opisati strogim formalnim pravilima u obliku formalnih gramatika. Nadalje, nerijetko se događa da istu rečenicu izrečenu ili zapisanu prirodnim jezikom različiti ljudi tumače na različite načine, od kojih je svaki moguće smatrati ispravnim.

```
"Hello Deductible" by "I.F. Author"

The story headline is "An Interactive Example".

The Living Room is a room. "A comfortably furnished living room."
The Kitchen is north of the Living Room.
The Front Door is south of the Living Room.
The Front Door is a door. The Front Door is closed and locked.
```

Slika 5.12. Primjer računalnog programa zapisanog prirodnim jezikom

Programiranje prirodnim jezikom u današnje je vrijeme još uvijek razmjerno neistraženo područje i predmet je iscrpnih istraživanja. Primjer programskog jezika koji je sličan prirodnom jeziku je programski jezik *Inform 7* [149]. Programski jezik *Inform 7*

nastao je 2006. godine, a namijenjen je za izradu pustolovnih računalnih igara (engl. *adventure games*, *interactive fiction*). Primjer opisa scene za računalnu igru u trodimenzionalnom prostoru primjenom programskog jezika *Inform 7* prikazan je slikom 5.12. Naredbe programskog jezika *Inform 7* poprimaju oblik potpunih rečenica zapisanih prirodnim jezikom. Iz oblika naredbi moguće je raspoznati deklarativni stil programiranja zasnovan na pravilima. Naredbe zapisane prirodnim jezikom prevode se u skup objekata kojima se pridružuju svojstva i veze prema drugim objektima.

5.3 Ocjena prikladnosti za ostvarenje programiranja prilagođenog potrošaču

Osnovni razredi postupaka koji tehnike programiranja približavaju skupini krajnjih korisnika ocijenjeni su s obzirom na prikladnost za ostvarenje programiranja prilagođenog potrošaču. Dok se u području razvoja prilagođenog krajnjem korisniku programske paradigme nastoje prilagoditi stručnjacima u pojedinim granama ljudske djelatnosti, programiranjem prilagođenim potrošaču nastoji se izgraditi programska paradigma za najširi krug potrošača koji su ovladali vještinama uporabe primjenskih programa za *World Wide Web*. Uočene su prepreke koje paradigme i alate namijenjene krajnjim korisnicima još uvijek čine neprikladnima za široki krug potrošača te su izdvojeni povoljni elementi koje je moguće iskoristiti za ostvarenje programske paradigme prilagođene potrošačima.

Programiranje primjenom obrazaca

Od svih prikazanih razreda programiranja prilagođenog krajnjem korisniku, programiranje primjenom obrazaca pokazuje najviše značajki koje su pogodne za primjenu od strane potrošača. Programiranje primjenom obrazaca odlikuje se razumljivošću i jednostavnošću zbog oslonjenosti na iskustva potrošača stečena primjenom grafičkog korisničkog sučelja za upravljanje operacijskim sustavima i primjenskim programima, naročito onih prikazanih u pregledniku *World Wide Web* sadržaja. Tehnika rukovanja obrascima zasnovana na upisivanju sadržaja u polja za unos teksta, izboru stavki iz padajućih izbornika, obilježavanju izbornih polja, pritisku na tipke ili poveznice i sličnim radnjama smatra se usvojenom od strane potrošača. Ako se značenje obrazaca odmakne od računalnih apstrakcija niske razine i prilagodi razini apstrakcije koja je bliska potrošaču, programiranje primjenom obrazaca postaje blisko umnom režimu, znanjima i vještinama potrošača. Nedostatak ove vrste programiranja je zasnovanost na skupu unaprijed definiranih obrazaca koji ograničavaju područje primjene izgrađenih programa.

U predloženom ostvarenju programiranja prilagođenog potrošaču, potrošački primjenski programi grade se povezivanjem udomljenika. Udomljenici su primjenski programi s grafičkim korisničkim sučeljem prikazanim u pregledniku *World Wide Web* sadržaja. Udomljenike je moguće smatrati obrascima za obavljanje potrošaču usmjerenih primjenskih funkcija, kao što je prognoza vremena, pretvorba valuta, pregled zemljovida i slično. Ograničenje izražajnosti uzrokovano unaprijed definiranim skupom obrazaca izbjegnuto je mogućnošću uključivanja proizvoljnih udomljenika, izgrađenih od različitih graditelja i dobavljivih putem mreže Internet, u sustav programske paradigme.

Programiranje na pokaznom primjeru

Programiranje na pokaznom primjeru odlikuje se značajkama razumljivosti i jednostavnosti pod uvjetom da je potrebno izgraditi računalni program za upravljanje radom skupa komponenata koje, osim programskog, dozvoljavaju i ručno upravljanje. Ovo svojstvo iskorišteno je u predloženom ostvarenju programiranja prilagođenog potrošaču. Komponente od kojih potrošači grade primjenske programe su gotovi primjenski programi u obliku udomljenika. Osnovna namjena udomljenika je pružanje grafičkog korisničkog sučelja za ručno upravljanje izvođenjem pozadinskih računalnih procesa. U predloženom ostvarenju programiranja prilagođenog potrošaču, udomljenici su prošireni programskom potporom koja bilježi radnje potrošača tijekom rukovanja skupom udomljenika i stvara računalni program.

Nedostatak tehnike s gledišta potrošača su ograničene mogućnosti poopćavanja programa na osnovi jednog ili više pokaznih primjera. Tehnikom programiranja na pokaznom primjeru moguće je snimiti slijed radnji potrošača nad elementima ciljnog sustava te ih zapamtiti u obliku računalnog programa za kasnije izvođenje bez uključenosti potrošača. Međutim, tehnika nije predviđena da se na osnovi snimljenog slijeda koraka skup elemenata objedini i izloži u obliku jedinstvenog elementa na višoj razini apstrakcije koji je pogodan za daljnju primjenu tehnike. U predloženom ostvarenju programiranja prilagođenog potrošaču, korištena je izmijenjena tehnika koja je zasnovana na plivajućem izborniku za obilazak udomljenika i zadavanje slijeda akcija nad grafičkim korisničkim sučeljima. Osim radnji koje je moguće postići izravnim rukovanjem elementima grafičkog korisničkog sučelja udomljenika, ova tehnika omogućava dodatno obilježavanje elemenata koji služe za unos početnih vrijednosti i prikaz konačnih rezultata izvođenja skupa udomljenika. Obilježeni elementi izdvajaju se u novi udomljenik na višoj razini apstrakcije kojim se nadomješta funkcija skupa udomljenika.

Programiranje primjenom grafičkih oznaka

Osnovna prepreka koja programiranje primjenom grafičkih oznaka čini neprikladnom za potrošače je ovisnost općenitosti i izražajnosti programiranja o odabiru elemenata jezika. Ako grafičke oznake predstavljaju elemente jezika na niskoj razini apstrakcije, onda je općenitost i izražajnost ove vrste programiranja moguće izjednačiti s općenitošću i izražajnošću programskih jezika opće namjene zasnovanih na slovčanom zapisu. Međutim, značenje oznaka u tom slučaju potrošačima nije poznato bez prethodnog učenja jezika ili razvojnog alata. Odabirom elemenata jezika na višim razinama apstrakcije, koje su bliže pojedinim područjima primjene, povećava se razumljivost elemenata, ali se smanjuje općenitost i izražajnost jezika.

Povoljna značajka programiranja primjenom grafičkih oznaka je dvodimenzionalni prostorni prikaz programa. U takvom prikazu programa, na prvi pogled je moguće uočiti odnose među komponentama programa, redoslijed izvođenja i tok podataka. Međutim, ako se povezivanje grafičkih oznaka izvodi primjenom linijskih poveznica, kod složenih programa dolazi do poteškoća s čitljivošću i preglednošću programskog zapisa zbog isprepletenosti grafičkih oznaka i linijskih poveznica.

U predloženom ostvarenju programiranja prilagođenog potrošaču, od programiranja primjenom grafičkih oznaka preuzeto je načelo dvodimenzionalnog prostornog razmještaja elemenata za prikaz programa. Za prikaz programa koristi se dvodimenzionalna tablična ploha. Tablična ploha podijeljena je na ćelije, a pojedina ćelija sadrži zapis po jedne akcije koju potrošač obavlja nad grafičkim korisničkim sučeljem udomljenika. Unutar tablice definirano je dvosmjerno napredovanje vremena. Unutar nepraznih ćelija vrijeme istodobno napreduje odozgo prema dolje i slijeva na desno, dok se praznim ćelijama označava prekid vremenskog tijeka. U takvoj se okolini na prvi pogled jasno raspoznaju vremenski zavisni i vremenski nezavisni dijelovi programa. Prikazom programa u obliku dvodimenzionalne tablice ističu se uzorci tijeka izvođenja programa, kao što je slijedno i istodobno izvođenje te mjesta razdvajanja i spajanja vremenskih tijekova. Uporabom ćelija izbjegava se korištenje linijskih poveznica, što pridonosi čitljivosti i preglednosti prikaza programa.

Programiranje primjenom tabličnih proračuna

Povoljna značajka programiranja primjenom tabličnih proračuna je razumljivost i jednostavnost jer su osnovni elementi za izradu tabličnih proračuna naslijeđeni iz matematike. Međutim, s obzirom da je u pravilu ograničena na izradu matematičkih i

statističkih proračuna te nije pogodna za izradu programa opće namjene, općenitost se ističe kao osnovni nedostatak ove programske paradigme.

U predloženom ostvarenju programiranja prilagođenog potrošaču, od programiranja primjenom tabličnih proračuna preuzeto je načelo dvodimenzionalnog prostornog razmještaja elemenata za prikaz programa unutar ćelija tablice. Iako predloženo ostvarenje nije zasnovano na funkcijskom programiranju, razmještaj programskih elemenata unutar ćelija tablice je načelo koje je poznato velikom broju korisnika tabličnih proračuna. Alati zasnovani na tabličnim proračunima najčešće su primjenjivana vrsta računalnih primjenskih programa [129].

Programiranje prirodnim jezikom

S obzirom na to da je prirodni jezik čovjeku najprirodniji i u većini slučajeva dovoljno izražajan način komunikacije, programiranje prirodnim jezikom pokazuje dobra svojstva u pogledu razumljivosti, jednostavnosti i izražajnosti. Ta svojstva čine postupke koji pripadaju ovom razredu programskih paradigmi pogodnima za primjenu tijekom oblikovanja potrošaču prilagođenog programiranja.

Nedostatak programiranja prirodnim jezikom je snažna oslonjenost na algoritme za obradu prirodnih jezika. Zbog ograničenja postojećih algoritama, punu izražajnost prirodnog jezika najčešće nije moguće u potpunosti iskoristiti. U području razvoja prilagođenog krajnjem korisniku, gdje se paradigma od strane stručnjaka koristi unutar uskog područja primjene, ovaj nedostatak moguće je djelomično ublažiti ograničenjem jezika na ključne riječi i jezične konstrukcije svojstvene tom području primjene. Međutim, primjena prirodno izražajnog načina međudjelovanja uz nametanje umjetnih ograničenja zbog nemogućnosti njegove učinkovite računalne analize nije pogodna jer za potrošača predstavlja teško objašnjivu pojavu.

U predloženom ostvarenju programiranja prilagođenog potrošaču, umjesto prirodnog jezika, kao osnovni oblik komunikacije potrošača i računala tijekom izgradnje programa iskorišteno je grafičko korisničko sučelje primjenskih programa za *World Wide Web*. Prirodni jezik je pritom primijenjen za prikaz i ažuriranje akcija nad elementima grafičkog korisničkog sučelja udomljenika zapisanih na osnovi radnji potrošača. Nadalje, prirodni jezik je upotrijebljen za nadopunjavanje prikaza programa koji se stvara na osnovi radnji potrošača putem grafičkog korisničkog sučelja slobodnim zapisom korisničkih komentara. Između ključnih riječi kojima se opisuju radnje potrošača, dozvoljeno je ubacivati proizvoljne dijelove teksta pisane prirodnim jezikom.

6 Model programiranja prilagođenog potrošaču

Oblikovanju programske paradigme prikladne za primjenu od strane širokog kruga potrošača računalnih primjenskih programa prethodi uspostava modela za ostvarenje pojedinih dijelova te paradigme. Model koji se naziva *modelom programiranja prilagođenog potrošaču* definiran je skupom svojstava koja programsku paradigmu čine prilagođenom umnom režimu, znanjima i iskustvu potrošača. Skup svojstava kojima je definiran model primjenjuje se tijekom izbora elemenata, tehnika i pravila za ostvarenje programske paradigme prilagođene potrošaču. U ovom poglavlju opisan je postupak oblikovanja modela programiranja prilagođenog potrošaču.

Model programiranja prilagođenog potrošaču definiran je na osnovi rezultata istraživanja u području kognitivne znanosti te iskustava potrošača stečenih tijekom primjene informacijsko-komunikacijske tehnologije. Model je oblikovan uzimajući u obzir osobine u kojima se potrošači značajno razlikuju od krajnjih korisnika i školovanih programera, što je opisano u poglavlju 2.1, te vodeći brigu o značajkama primjenskih programa samostalno izgrađenih od strane potrošača, koje su opisane u poglavlju 2.2. Na osnovi modela kognitivne znanosti, psihologije programiranja i mjera umnog napora tijekom primjene sustava znakovlja, koji su opisani u poglavlju 3, definirana su svojstva modela koja programiranje čine *kognitivno bliskim* umnom režimu potrošača. S druge strane, na osnovi analize odnosa potrošača prema tehnologiji, koja je provedena u poglavlju 4, definirana su svojstva modela koja programiranje čine *iskustveno bliskim* umnom režimu potrošača.

U poglavlju 6.1 uspostavljen je radni okvir za vrednovanje značajki programskih elemenata i postupaka programiranja s obzirom na prikladnost za oblikovanje programske

paradigme prilagođene potrošaču. Primjenom definiranog radnog okvira, postavljeni su zahtjevi koje je potrebno ispuniti tijekom oblikovanja programske paradigme kako bi paradigma bila pogodna za primjenu od strane potrošača. Usporedno s definicijom zahtjeva za programsku paradigmu prilagođenu potrošaču, ocjenjene su značajke programskih paradigmi u području razvoja prilagođenog krajnjem korisniku i profesionalnog razvoja programske potpore.

U poglavlju 6.2 uspostavljen je model sa skupom svojstava za ostvarenje programiranja prilagođenog potrošaču. Usporedno s definicijom modela, prikazana je ocjena najzastupljenijih razvojnih alata namijenjenih za izgradnju usloženih primjenskih programa s obzirom na definirana svojstva programiranja prilagođenog potrošaču. Primjenom definiranog modela, u nastavku doktorske disertacije izabrani su elementi, tehnike i pravila kojima je ostvarena programska paradigma prilagođena potrošaču.

6.1 Značajke programiranja prilagođenog potrošaču

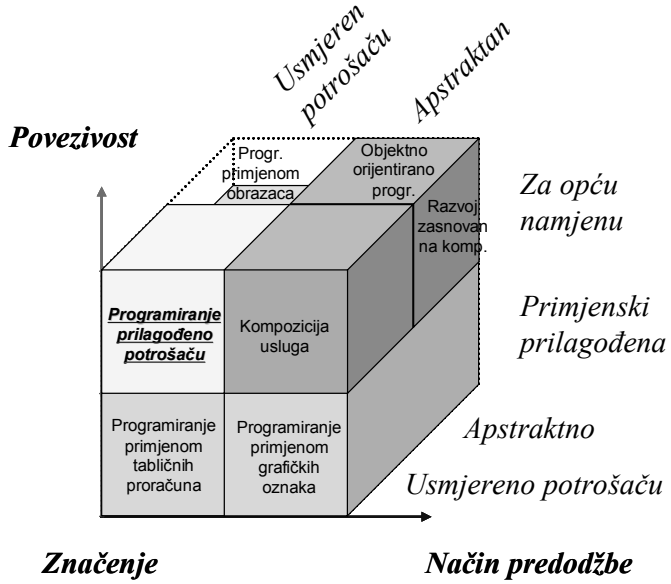
Zahtjevi koje je potrebno ispuniti kako bi programska paradigma zadovoljila elemente kognitivne i iskustvene bliskosti navikama, znanjima i vještinama potrošača opisani su radnim okvirom za vrednovanje značajki programskih elemenata od kojih se grade primjenski programi i postupaka programiranja kojima se osnovni elementi povezuju u primjenski program. Uspostavom radnog okvira, definirane su idealne značajke programiranja prilagođenog potrošaču te su ocjenjene značajke programskih paradigmi u području razvoja prilagođenog krajnjem korisniku i profesionalnog razvoja programske potpore.

6.1.1 Značajke programskih elemenata

Radni okvir za ocjenu značajki programskih elemenata od kojih se povezivanjem u složene cjeline grade primjenski programi prikazan je slikom 6.1. Programski elementi ocijenjeni su s obzirom na tri dimenzije, od kojih je svaka predstavljena dvjema kvalitativnim vrijednostima.

Prva dimenzija za ocjenu programskih elemenata odnosi se na *značenje* koje programski elementi za izgradnju primjenskih programa imaju za potrošača. Značenje elementa odnosi se na funkcionalnost elementa koja se koristi tijekom izgradnje primjenskog programa. S gledišta potrošača, značenje elemenata moguće je kvalitativno ocijeniti kao *apstraktno* ili *usmjereno potrošaču*. Primjerice, programski element čija je funkcionalnost pretvorba valuta moguće je ocijeniti kao element sa značenjem usmjerenim potrošaču jer je

pretvorba valuta funkcionalnost primjenske razine koja je potrošaču razumljiva na osnovi svakodnevnog životnog iskustva. S druge strane, element čija je funkcija pretvorba komunikacijskih protokola moguće je ocijeniti kao apstraktan za potrošača jer većina potrošača tijekom rukovanja računalom ne razmišlja na razini komunikacijskih protokola.



Slika 6.1. Radni okvir za ocjenu bliskosti programskih elemenata umnom režimu potrošača

Druga dimenzija za ocjenu programskih elemenata odnosi se na *način predodžbe* programskih elemenata potrošaču tijekom izgradnje primjenskih programa. Slično kao i značenje, način predodžbe programskih elemenata moguće je kvalitativno ocijeniti kao *apstraktan* ili *usmjeren potrošaču*. Primjerice, objekti kod paradigme *objektno-orijentiranog programiranja* ili komponente kod paradigme *razvoja zasnovanog na komponentama* su ocijenjeni kao apstraktno predočeni programski elementi jer su potrošaču izloženi putem apstraktnih programskih sučelja. Za primjenu objekata ili komponenata, potrošač mora usvojiti koncept javno izložene metode, poziva metode, pozivnih argumenata i povratne vrijednosti. S druge strane, *programiranje primjenom tabličnih proračuna*, gdje su osnovni elementi u obliku matematičkih formula korisniku predočeni na jednak način na koji ih je on navikao zapisivati na papiru, primjer je potrošaču usmjerenog načina predodžbe programskih elemenata.

Treća dimenzija za ocjenu programskih elemenata odnosi se na *povezivost*, odnosno izražajnost povezivanja programskih elemenata u složene primjenske programe. Ako je povezivanje elemenata ograničeno na način da je programske elemente moguće povezivati samo u primjenske programe za određeno područje primjene, onda se govori o *primjenski prilagođenoj povezivosti*. Ako je skup elemenata moguće povezivati u proizvoljne

kombinacije i graditi primjenske programe opće namjene, onda se govori o *povezivosti opće namjene*.

Programiranje prilagođeno potrošaču definirano je kao paradigma programiranja u kojoj su značenje i način predodžbe programskih elemenata usmjereni potrošaču, dok je njihovim povezivanjem moguće graditi primjenske programe opće namjene. Ocjena svojstava nekoliko najzastupljenijih programskih paradigmi opće namjene, kao i onih prilagođenih krajnjim korisnicima u određenim područjima primjene, uključena je u radni okvir prikazan na slici 6.1. O*bjektno-orijentirano programiranje*, *razvoj zasnovan na komponentama* i *kompozicija usluga* primjeri su programskih paradigmi opće namjene s apstraktnim načinom predodžbe programskih blokova. Za razliku od *objektno-orijentiranog programiranja* i *razvoja zasnovanog na komponentama*, značenje elemenata kod *kompozicije usluga* smatra se usmjerenim potrošaču zbog krupne zrnatosti programskih usluga putem kojih su izložene funkcionalnosti potrošačke razine, kao što je prognoza vremena, rezervacija hotelskih soba, elektroničko bankarstvo i slično. S druge strane, većina postupaka *programiranja primjenom tabličnih proračuna*, *grafičkih oznaka* i *obrazaca* ograničena je na usko područje primjene, što ih čini nepogodnim za izgradnju programa opće namjene. Iako p*rogramiranje primjenom obrazaca* pokazuje povoljne značajke u pogledu načina predodžbe programskih blokova, značenje obrazaca tijekom programiranja često je zasnovano na računalno-svojstvenim apstrakcijama, kao što je obrada HTML jezika, konfiguracija SQL upita i slično.

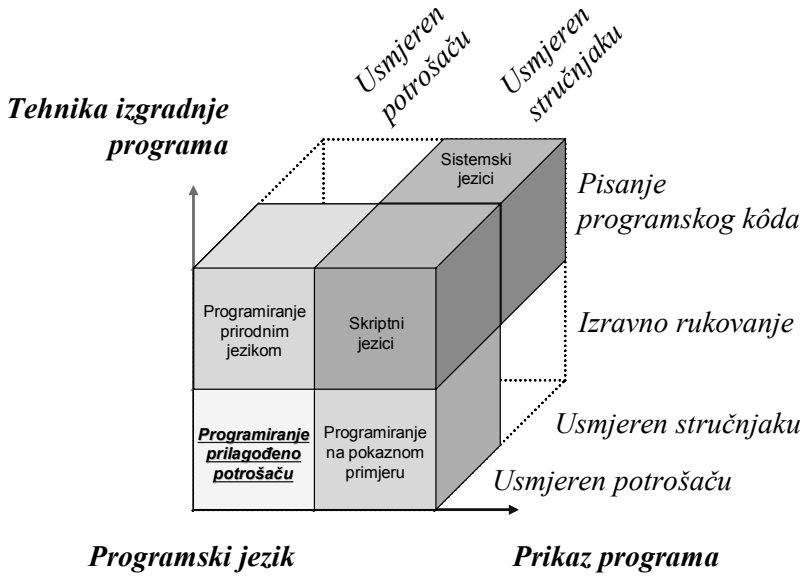
6.1.2 Značajke postupka programiranja

Radni okvir za ocjenu značajki postupka programiranja kojim se skup osnovnih programskih elemenata povezuje u složene cjeline s ciljem izgradnje primjenskih programa prikazan je slikom 6.2. Postupak programiranja ocijenjen je s obzirom na tri dimenzije, od kojih je svaka predstavljena dvjema kvalitativnim vrijednostima.

Prva dimenzija za ocjenu postupka programiranja odnosi se na svojstva *programskog jezika* koji se koristi za definiranje načina međudjelovanja skupa elemenata kako bi se postigla željena funkcionalnost primjenskog programa. Programski jezik moguće je kvalitativno ocijeniti kao *usmjeren potrošaču* ili *usmjeren stručnjaku*. Primjerice, većina *programskih jezika opće namjene* zasnovanih na slovčanom zapisu moguće je ocijeniti stručnjacima usmjerenim jezicima zbog visoke razine apstrakcije tijekom preslikavanja uzoraka međudjelovanja programskih elemenata u slovčane naredbe programskog jezika. S druge strane, *programiranje na pokaznom primjeru* moguće je smatrati postupkom

programiranja s potrošaču usmjerenim programskim jezikom. Naime, kod ove vrste programiranja ne koristi se posebni programski jezik, već se u svojstvu programskog jezika koriste izravne radnje korisnika nad programskim elementima od kojih se gradi program.

Druga dimenzija za ocjenu postupka programiranja odnosi se na *prikaz programa* izgrađenog primjenom programskog jezika. Prikaz programa također je moguće kvalitativno ocijeniti kao *usmjeren potrošaču* ili *usmjeren stručnjaku*. Prikaz programa je usmjeren potrošaču ako su potrošaču za razumijevanje programskog zapisa dovoljna opća znanja i iskustvo, bez potrebe za dodatnom obukom u području programiranja. Ako su za tumačenje programskog zapisa potrebna određena stručna znanja, prikaz programa smatra se usmjerenim stručnjaku.



Slika 6.2. Radni okvir za ocjenu bliskosti postupka programiranja umnom režimu potrošača

Treća dimenzija za ocjenu postupka programiranja odnosi se na *tehniku izgradnje programa*. Tehniku izgradnje programa moguće je kvalitativno ocijeniti kao tehniku zasnovanu na *izravnom rukovanju* ili tehniku zasnovanu na *pisanju programskog kôda*. Tehnikom izravnog rukovanja, računalni program gradi se izravnom primjenom ciljnog sustava od strane potrošača tijekom rješavanja zadanog problema, bez nužne potrebe za potrošačevom svjesnom uključenošću u postupak programiranja. Na osnovi radnji potrošača nad ciljnim sustavom, stvara se računalni program za kasnije automatizirano upravljanje njegovim radom. Tehnikom zasnovanom na pisanju programskog kôda, računalni se program gradi pisanjem naredbi ili oblikovanjem grafičkog modela u izabranom programskom jeziku. S obzirom na izravnost preslikavanja između radnji korisnika i učinaka izvođenja napisanog programa, pogodnijom tehnikom za potrošača smatra se tehnika

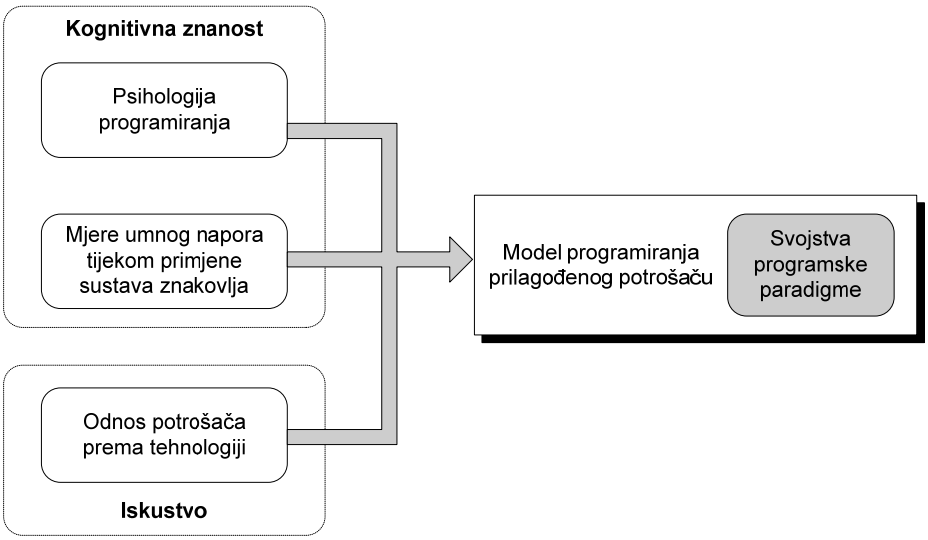
zasnovana na izravnom rukovanju. S druge strane, u većini današnjih programskih jezika opće namjene ili primjenski usmjerenim programskim jezicima koristi se tehnika zasnovana na pisanju programskog kôda.

S gledišta postupka programiranja, *programiranje prilagođeno potrošaču* definirano je kao paradigma programiranja s potrošaču usmjerenim programskim jezikom, potrošaču usmjerenim načinom prikaza izgrađenog programa te tehnikom izgradnje programa zasnovanom na izravnom rukovanju. Ocjena svojstava nekoliko najzastupljenijih skupina programskih jezika i tehnika programiranja u odnosu programiranje prilagođeno potrošaču uključena je u radni okvir prikazan na slici 6.2. Programiranje primjenom *sistemskih jezika* primjenjuje se za razvoj programske logike operacijskih sustava i osnovnih programskih blokova te je zbog toga programski jezik i prikaz programa usmjeren stručnjacima u području programskog inženjerstva, dok je izrada programa zasnovana na tehnici pisanja programskog kôda. Programiranje primjenom *skriptnih jezika* namijenjeno je povezivanju gotovih programskih blokova u složene cjeline pa se koriste programski jezici razumljiviji potrošaču. Međutim, izrada programa još uvijek je zasnovana na tehnici pisanja programskog kôda. *Programiranje na pokaznom primjeru* je pogodno za potrošača jer se za izradu programa koristi tehnika izravnog rukovanja, ali većina poznatih alata za prikaz programa koristi stručnjaku usmjeren zapis. Iako je izrada programa krajnje pojednostavljena, tehnika nije pogodna za naknadnu izmjenu programa od strane potrošača jer se zahtijeva ponavljanje cjelokupnog postupka izravnog rukovanja. *Programiranje prirodnim jezikom* ocijenjeno je pogodnom tehnikom za potrošača, ali zbog potrebe za zapisivanjem teksta koji predstavlja računalni program ne omogućava izgradnju programa kao popratnu pojavu uobičajenih radnji potrošača.

6.2 Model za ostvarenje programiranja prilagođenog potrošaču

Uspostavom radnog okvira za ocjenu bliskosti programskih elemenata i postupaka programiranja navikama, znanjima i vještinama potrošača, definirani su zahtjevi koji se postavljaju tijekom oblikovanja potrošaču prilagođene programske paradigme. Programska paradigma koja zadovoljava postavljene zahtjeve ostvaruje se primjenom modela kojim je definiran način izbora elemenata, tehnika i pravila za oblikovanje i izgradnju primjenskih programa. Shematski prikaz metodologije za oblikovanje modela programiranja prilagođenog potrošaču prikazan je slikom 6.3. Ključni čimbenici koji programsku paradigmu čine prikladnom za potrošača definirani su na osnovi spoznaja iz područja kognitivne znanosti i psihologije programiranja te mjera umnog napora tijekom primjene

sustava znakovlja koje su opisane u poglavlju 3 te na osnovi iskustava potrošača stečenih dugogodišnjom primjenom informacijsko-komunikacijske tehnologije koja su prikazana u poglavlju 4.



Slika 6.3. Oblikovanje modela programiranja prilagođenog potrošaču

Skup svojstava kojima se definira model za ostvarenje programiranja prilagođenog potrošaču čine blokovska izgradivost primjenskog programa, opažajni doživljaj značenja blokova, samodostatnost blokova, višerazinsko usložnjavanje blokova, jednakost korištenja i povezivanja blokova, nezavisnost povezujućih elemenata i značenja blokova, izgradivost primjenskog programa putem grafičkog korisničkog sučelja, upravljivost i uočljivost vremenske relacije, potpuna izgradivost logike za usložnjavanje blokova od strane potrošača te opažajni doživljaj prikaza programa. Uz definiciju modela za ostvarenje potrošaču prilagođene programske paradigme, u ovom je poglavlju prikazana i ocjena najzastupljenijih alata za razvoj usloženih primjenskih programa s obzirom na definirana svojstva.

6.2.1 Blokovska izgradivost primjenskog programa

Svojstvo *blokovske izgradivosti primjenskog programa* zadovoljava programska paradigma koja dopušta izgradnju primjenskog programa iskorištavanjem gotovih programskih blokova. Postupak definiranja svojstva blokovske izgradivosti primjenskog programa prikazan je tablicom 6.1. Svojstvo blokovske izgradivosti primjenskog programa definirano je na osnovi *teorije kratkoročnog pamćenja privremenih činjenica* kao jednog od modela psihologije programiranja, na osnovi *stupnja smislenog objedinjavanja*, *podložnosti pogreškama*, *složenosti rasuđivanja* i *prionljivosti* kao mjera umnog napora tijekom primjene

sustava znakovlja te *komponentnog World Wide Web sustava* kao iskustvenog elementa proizašlog iz odnosa potrošača prema tehnologiji.

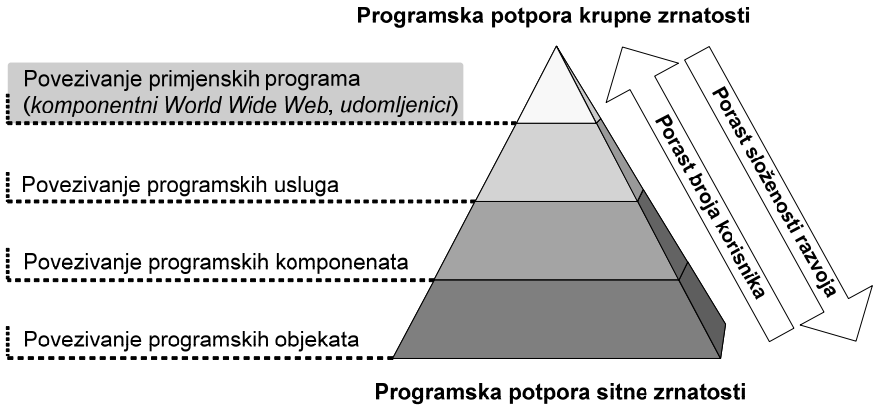
Tablica 6.1. Definicija svojstva blokovske izgradivosti primjenskog programa

Svojstvo prog. prilagođenog potrošaču	Polazište za definiciju svojstva
Blokovska izgradivost primjenskog programa	**Psihologija programiranja** ▪ Teorija kratkoročnog pamćenja privremenih činjenica **Mjere umnog napora** ▪ Stupanj smislenog objedinjavanja ▪ Podložnost pogreškama ▪ Složenost rasuđivanja ▪ Prionljivost **Odnos potrošača prema tehnologiji** ▪ Komponentni *World Wide Web* sustav

Teorijom kratkoročnog pamćenja privremenih činjenica pokazano je da je kratkoročno pamćenje čovjeka tijekom učenja, stvaranja i zaključivanja ograničeno na približno sedam činjenica. Kratkoročno pamćenje potrošača tijekom programiranja zauzeto je planiranjem radnog tijeka primjenskog programa koje uključuje izbor programskih elemenata i njihovo povezivanje u radni tijek prema zakonitostima primjenskog programa. Ako je radni tijek moguće izgraditi povezivanjem programskih blokova koji skrivaju pojedinosti unutarnje građe, a putem izloženog sučelja pružaju potrošaču usmjerene primjenske funkcije, cjelokupni programski blok pojavljuje se kao osnovna činjenica u kratkoročnom pamćenju potrošača.

Povećanjem zrnatosti programskih blokova, postiže se veći *stupanj smislenog objedinjavanja* jer se većim brojem programskih funkcionalnosti rukuje u obliku jedinstvenog bloka. U tom slučaju, isti radni tijek moguće je oblikovati od manjeg broja blokova. Osim što se time rasterećuje kratkoročno pamćenje potrošača, ponovna iskoristivost već izgrađene programske potpore izložene u obliku programskog bloka skraćuje vrijeme razvoja novih programa i smanjuje mogućnost uvođenja pogrešaka u sustav. Programska paradigma na taj način pokazuje niži stupanj *podložnosti pogreškama*. Nadalje, izgradnja primjenskih programa zasnovana na iskorištavanju gotovih programskih blokova potrošaču omogućava djelovanje na razini međusobnih odnosa pojedinih komponenata sustava, umjesto na razini njihove unutarnje građe, što smanjuje *složenost rasuđivanja* tijekom oblikovanja, izgradnje i održavanja primjenskog programa.

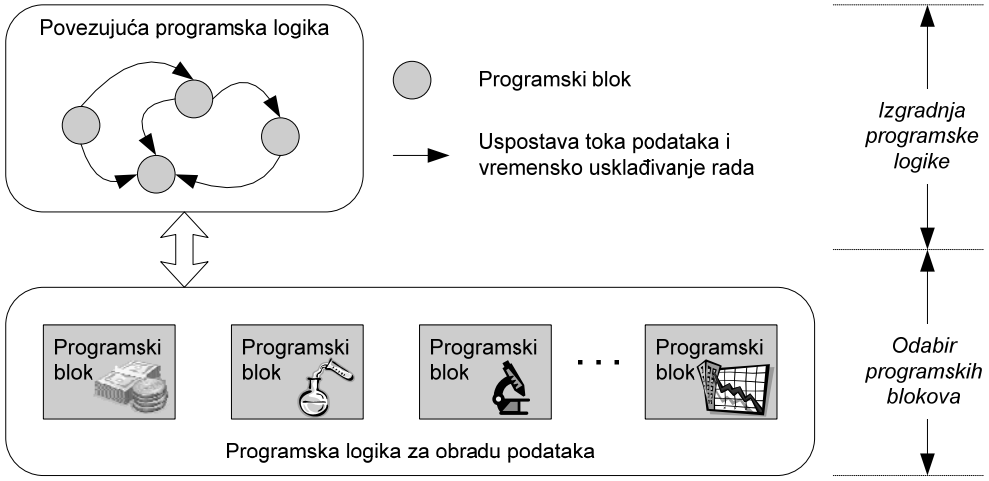
Osim što pokazuje povoljne značajke s gledišta kognitivne analize, povećanje zrnatosti programskih blokova pokazalo je značajne prednosti u odnosu na postupak razvoja sustava iznova i kroz povijest razvoja programskog inženjerstva. Slika 6.4 prikazuje piramidalni složaj programskih paradigmi za oblikovanje programskih sustava koje pokazuju svojstvo blokovske izgradivosti. S obzirom na zrnatost osnovnih programskih elemenata od kojih se gradi programska potpora, programske paradigme dijele se na *oblikovanje povezivanjem programskih objekata*, *oblikovanje povezivanjem programskih komponenti*, *oblikovanje povezivanjem programskih usluga* te *oblikovanje povezivanjem primjenskih programa*. Oblikovanje povezivanjem programskih objekata i oblikovanje povezivanjem programskih komponenti su tradicionalne paradigme razvoja programske potpore na kojima se zasnivaju objektno-orijentirano programiranje [168, 171], odnosno razvoj zasnovan na komponentama [172]. Oblikovanje povezivanjem programskih usluga [104, 117] je programska paradigma koja je nagli zamah doživjela razvojem otvorenih i raznorodnih raspodijeljenih sustava, računalnih spletova [174, 175] i uslužno-usmjerenog računarstva [95, 96].



Slika 6.4. Složaj programskih paradigmi za oblikovanje primjenskih programa sa svojstvom blokovske izgradivosti sustava

Oblikovanje povezivanjem primjenskih programa je programska paradigma u kojoj se novi primjenski program gradi objedinjavanjem i povezivanjem već izgrađenih primjenskih programa koji sami za sebe predstavljaju gotova i potpuna programska rješenja. Gotovi primjenski programi povezuju se u nove, složenije i poosobljene primjenske programe prilagođene pojedinačnim potrebama potrošača. Povezivanjem skupa međusobno nezavisnih primjenskih programa u novi primjenski program nastaje novi razred primjenskih programa koji se naziva usloženim primjenskim programima (engl. *mashups*) [177, 180]. Izgradnja usloženih primjenskih programa povezivanjem programskih blokova krupne zrnatosti u obliku jednostavnijih primjenskih programa osnovna je zamisao *komponentnog World Wide Web sustava* [22] u kojem se primjenski programi grade od udomljenika (engl. *widgets*).

Kretanjem uzduž složaja programskih paradigmi od dna prema vrhu, povećava se zrnatost izgradnje programskih sustava. Povećanjem zrnatosti izgradnje programskih sustava, smanjuje se složenost primjene pojedinih programskih tehnologija jer je složenije programske sustave moguće graditi primjenom manjeg broja programskih elemenata. Smanjivanjem složenosti primjene programske paradigme povećava se broj korisnika koji ovladavaju znanjima i vještinama potrebnim za primjenu promatrane paradigme. Za primjenu paradigmi oblikovanja sitne zrnatosti gdje se računalni sustav oblikuje povezivanjem programskih objekata i komponenti, nužna su znanja i vještine školovanih programera. Razina oblikovanja povezivanjem programskih usluga prikladnija je za širu skupinu korisnika, ali još uvijek zahtijeva poznavanje pojmova poput programskih varijabli, funkcija, funkcijskih argumenata, povratnih vrijednosti, komunikacijskih protokola i slično. Cilj istraživanja provedenog u okviru ove doktorske disertacije jest prijedlog metodologije programiranja bliske potrošačima za čiju su primjenu dovoljna osnovna znanja o uzajamnom djelovanju korisnika i računala putem *World Wide Web* stranica. Za takvu vrstu programiranja prikladno je koristiti paradigmu oblikovanja povezivanjem primjenskih programa izloženih putem grafičkog korisničkog sučelja.

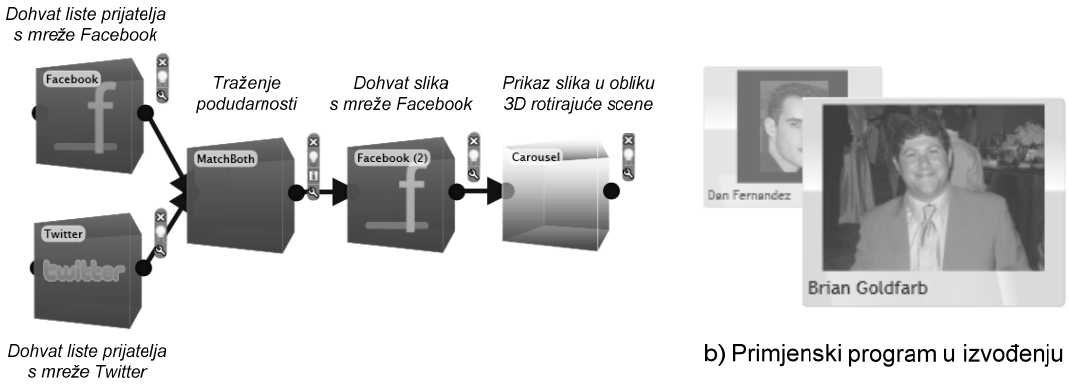


Slika 6.5. Blokovska izgradivost primjenskog programa

Svojstvo blokovske izgradivosti primjenskog programa prikazano je slikom 6.5. U razvojnom postupku sa svojstvom blokovske izgradivosti primjenskog programa razlikuju se programska logika za obradu podataka te povezujuća programska logika. Programska logika za obradu podataka zadužena je za obavljanje jezgrenih funkcija primjenskog programa i sastoji se od konačnog skupa programskih blokova. Pojedini programski blok je nezavisna programska cjelina koju je moguće koristiti kao element programske logike za obradu podataka proizvoljnog primjenskog programa. Povezujućom programskom logikom, skup

blokova povezuje se u radni tijek kojim je definiran tok podataka i redoslijed izvođenja blokova za obradu podataka. Uloga potrošača tijekom oblikovanja primjenskog programa primjenom programske paradigme sa svojstvom blokovske izgradivosti je dvostruka. S jedne strane, od potrošača se očekuje da iz skupa raspoloživih programskih blokova izabere onaj podskup blokova kojima je moguće obaviti pojedine dijelove jezgrene logike primjenskog programa. S druge strane, od potrošača se očekuje izgradnja povezujuće programske logike za povezivanje izabranih programskih blokova u radni tijek prema zakonitostima primjenskog programa.

Izborom programskih blokova koje je moguće proizvoljno povezivati s ostalim blokovima, programska paradigma pokazuje niski stupanj *prionljivosti*. U primjenskom programu izgrađenom povezivanjem blokova niskog stupnja prionljivosti, programski blok moguće je nadomjestiti zamjenskim blokom iste ili slične funkcionalnosti, bez utjecaja na funkcionalnost primjenskog programa. Nadalje, izmjenu funkcionalnosti programa moguće je izvesti izuzimanjem postojećeg ili uvođenjem novog bloka u radni tijek.



Slika 6.6. Primjer izgradnje usloženog primjenskog programa primjenom programskog alata sa svojstvom blokovske izgradivost primjenskog programa

Slikom 6.6 prikazan je primjer izgradnje usloženog primjenskog programa u razvojnom alatu *Microsoft Popfly* [177, 178, 182] koji ima svojstvo blokovske izgradivosti primjenskog programa. Slikom 6.6.a prikazan je postupak izgradnje primjenskog programa, dok je slikom 6.6.b prikazan program u izvođenju. Trodimenzionalnim kockama predstavljeni su osnovni programski blokovi koji se u usloženi primjenski program povezuju definiranjem toka podataka primjenom usmjerenih grana u obliku strelica. Prikazanim primjerom izgrađen je primjenski program za rukovanje podacima s društvenih mreža *Facebook* i *Twitter*. Rezultat izvođenja programa je prikaz slika osoba koje se pojavljuju u svojstvu prijatelja na obje društvene mreže u obliku rotirajuće scene.

Tablica 6.2. Ocjena razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo blokovske izgradivosti primjenskog programa

Naziv alata	Blokovska izgradivost
Microsoft Popfly	+
Yahoo! Pipes	+
Intel Mash Maker	+
Google Mashup Editor	+
iGoogle Gadget Maker	-
IBM QEDwiki	+
OpenKapow	-
Proto	+
Dapper	+
Marmite	+
JackBe Presto	+

Tablicom 6.2 prikazana je ocjena najpoznatijih predstavnika razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo blokovske izgradivosti primjenskog programa. Alati *Microsoft Popfly*, *Proto*, *JackBe Presto*, *Intel Mash Maker*, *IBM QEDwiki*, *Marmite* i *Google Mashup Editor* koriste blokove koji omogućuju pristup mrežnim uslugama primjenske razine, kao što su čitanje vijesti, pregled video sadržaja, naručivanje roba i usluga, pregled zemljovida i društvene mreže, pristup bazama podataka te lokalnu obradu i prikaz podataka korisniku programa. Pritom alati *Microsoft Popfly*, *Proto* i *JackBe Presto* koriste blokove u obliku grafičkih oznaka kao što su trodimenzionalni prostorni elementi ili slikovni elementi u obliku ikona, alati *Intel Mash Maker*, *IBM QEDwiki* i *Marmite* koriste blokove u obliku udomljenika s grafičkim korisničkim sučeljem, dok alat *Google Mashup Editor* koristi komponente izložene putem programskih sučelja. Alat *Yahoo! Pipes* koristi blokove u obliku elektroničkih obrazaca s grafičkim korisničkim sučeljem (engl. *web forms*) za pristup, obradu i prikaz podataka dostupnih iz RSS izvorišta. U alatu *Dapper*, u svojstvu programskih blokova koriste se proizvoljne *World Wide Web* stranice. Programiranje primjenom alata *iGoogle Gadget Maker* i *OpenKapow* zasnovano je na unaprijed definiranim predlošcima primjenskih programa pa ti alati ne zadovoljavaju svojstvo blokovske izgradivosti primjenskog programa.

6.2.2 Opažajni doživljaj značenja blokova

Svojstvom blokovske izgradivosti primjenskog programa i krupne zrnatosti programskih blokova osiguravaju se dobra svojstva programske paradigme što se tiče brzine

razvoja, opterećenja kratkoročnog pamćenja potrošača i podložnosti pogreškama za vrijeme oblikovanja novih primjenskih programa. Funkcionalnosti obuhvaćene pojedinim programskim blokovima moguće je, međutim, na različite načine predstaviti potrošaču. Svojstvom *opažajnog doživljaja značenja blokova* definirano je svojstvo programske paradigme u kojoj se za izgradnju primjenskih programa koriste samoopisivi programski blokovi koji za razumijevanje uloge bloka u postupku izgradnje primjenskog programa ne zahtijevaju dodatni napor, kao što je školovanje ili čitanje priručnika za uporabu.

Postupak definiranja svojstva opažajnog doživljaja značenja blokova prikazan je tablicom 6.3. Svojstvo opažajnog doživljaja značenja blokova definirano je na osnovi *teorije održavanja i preispitivanja znanja* kao jednog od modela psihologije programiranja te na osnovi *samoopisivosti* kao jedne od mjera umnog napora tijekom primjene sustava znakovlja.

Tablica 6.3. Definicija svojstva opažajnog doživljaja značenja blokova

Svojstvo prog. prilagođenog potrošaču	Polazište za definiciju svojstva
Opažajni doživljaj značenja blokova	**Psihologija programiranja** ▪ Teorija održavanja i preispitivanja znanja **Mjere umnog napora** ▪ Samoopisivost

Teorijom održavanja i preispitivanja znanja pokazano je da programiranje nameće dva oblika umnog opterećenja na programera. U prvom koraku potrebno je uložiti umni napor za učenje programskog jezika i paradigme. U drugom koraku potrebno je uložiti umni napor za primjenu naučenog programskog jezika i paradigme za oblikovanje i izgradnju primjenskog programa. Veliki početni napor koji je potrebno uložiti za učenje programskog jezika i paradigme kod većine potrošača smanjuje poticaj za upuštanje u postupak izgradnje programa, naročito ako izgradnja programa nije osnovna djelatnost potrošača. Izborom samoopisivih programskih blokova uklanja se potreba za ulaganjem početnog umnog napora za razumijevanje uloge i načina primjene blokova koje prethodi izgradnji programa.

Svojstvo opažajnog doživljaja značenja blokova izravna je posljedica *samoopisivosti* kao jedne od mjera umnog napora tijekom primjene sustava znakovlja. Mjerom samoopisivosti zahtijeva se izbor programskih blokova za čije je razumijevanje značenja i načina primjene tijekom izgradnje primjenskog programa dovoljna predodžba blokova putem izlaznih naprava računala, bez potrebe za dodatnom obukom za primjenu blokova.

Programske blokove sa svojstvom samoopisivosti, odnosno opažajnog doživljaja, moguće je izabrati na osnovi prethodnog iskustva ciljne skupine potrošača stečenog primjenom računala i računalnih primjenskih programa. Programski blokovi korisnicima su izloženi na tri osnovna načina, ovisno o tome kojoj ciljnoj skupini korisnika su namijenjeni. Prvi način predodžbe programskih blokova je primjena modela zasnovanog na slovčanom zapisu, drugi način je primjena modela zasnovanog na apstraktnim grafičkim oznakama, dok je treći način primjena predodžbenog modela.

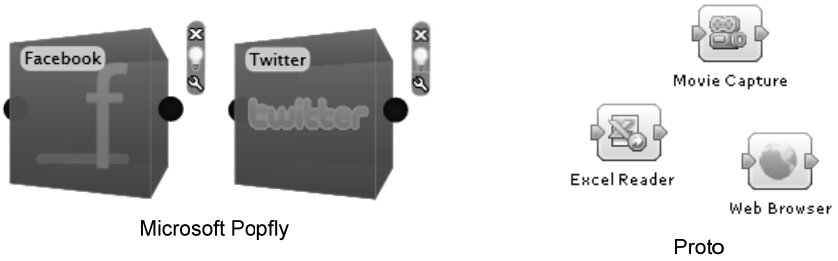
Primjenom modela zasnovanog na slovčanom zapisu, programski blokovi korisniku su predstavljeni u obliku niza slovčanih znakova zapisanih prema pravilima programskog jezika. Većina današnjih programskih jezika opće namjene koristi ovaj način predodžbe programskih blokova. Primjer predodžbe programskih blokova primjenom modela zasnovanog na slovčanom zapisu prikazan je slikom 6.7. Primjerom je prikazan isječak programskog bloka koji je izgrađen primjenom programskog alata *Google Mashup Editor* [177, 178, 188]. Ovaj model predodžbe programskih blokova je najopćenitiji i najizražajniji jer je graditelju omogućeno upravljanje svim pojedinostima vezanim uz programsku logiku i način prikaza sadržaja korisniku. Nedostatak modela je potreba za poznavanjem strogih leksičkih, sintaksnih i semantičkih pravila programskih tehnologija koje se koriste za izgradnju programskih blokova.

```
<gm:page title="Sample - Calendar">

  <gm:data id="myData" data="http:..."/>

  <gm:calendar id="myCalendar" data="${myList}">
    <gm:handleEvent src="myList" event="select"/>
  </gm:calendar>

  <gm:list id="myList" data="${myData}" pagesize="5">
    <gm:handleEvent event="select" src="myCalendar"/>
  </gm:list>

</gm:page>
```

Slika 6.7. Model programskog bloka zasnovan na slovčanom zapisu

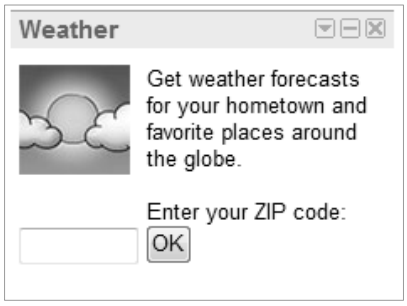
Drugi način predodžbe programskih blokova je primjena apstraktnih grafičkih oznaka. Primjeri predodžbe programskih blokova primjenom apstraktnih grafičkih oznaka prikazani su slikom 6.8. Primjenom apstraktnih grafičkih oznaka, programski blokovi korisniku su predstavljeni u obliku slikovnih elemenata koji se u cjelinu najčešće povezuju strelicama koje određuju smjer toka podataka. Primjerom je prikazan način predodžbe programskih blokova u programskim alatima *Microsoft Popfly* [177, 178, 182, 183] i *Proto* [193].

Iako ne zahtijeva poznavanje programskih tehnologija koje se koriste za izradu programskih blokova, primjena blokova predstavljenih apstraktnim grafičkim oznakama zahtijeva poznavanje ulaznih i izlaznih struktura podataka koje nisu vidljive na osnovi grafičke predodžbe blokova. Razina apstrakcije kod ovog modela još je uvijek na razini načela i pojmova koji potrošaču nisu poznati iz iskustva stečenog svakodnevnim korištenjem primjenskih programa za *World Wide Web*.



Slika 6.8. Model programskog bloka zasnovan na apstraktnim grafičkim oznakama

Razinu apstrakcije i izravnost preslikavanja između modela za oblikovanje i izvodivog oblika primjenskog programa moguće je značajno popraviti primjenom predodžbenog modela programskih blokova. Primjer predodžbe programskog bloka primjenom predodžbenog modela prikazan je slikom 6.9. Predodžbeni model koristi se načelom oblikovanja s izravnim preslikavanjem u izvodivi oblik (engl. *What You See Is What You Get – WYSIWYG*) [16, 205]. Svojstvo načela oblikovanja s izravnim preslikavanjem u izvodivi oblik je jednakost predodžbe programskog bloka za vrijeme oblikovanja i za vrijeme korištenja. Primjenom predodžbenog modela, programski blokovi potrošaču su predstavljeni u obliku grafičkog korisničkog sučelja prikazanog u pregledniku *World Wide Web* sadržaja.



Slika 6.9. Predodžbeni model programskog bloka

S gledišta prikladnosti za primjenu od strane potrošača, predodžbeni model ima višestruke prednosti u odnosu na model zasnovan na slovčanom zapisu i model zasnovan na apstraktnim grafičkim oznakama. Ključna prednost predodžbenog modela je izravni doživljaj izgleda i funkcije programskih blokova za vrijeme oblikovanja. Predodžbeni model

programskih blokova predstavljen grafičkim korisničkim sučeljem prirodni je način doživljaja primjenskih programa za *World Wide Web* od strane potrošača. Iako mreža Internet ne nameće pravila o načinu predodžbe elektroničkih usluga krajnjim potrošačima, ustaljeni i opće prihvaćeni način ponude i primjene usluga upravo je izlaganje putem *World Wide Web* stranica. Međudjelovanje i rukovanje informacijama, uslugama i procesima u globalnoj mreži Internet većina potrošača doživljava kroz korisničko sučelje putem kojeg je dostupna pojedina informacija, usluga ili računalni proces.

Prilog tezi o grafičkom korisničkom sučelju kao prirodno razumljivom i široko prihvaćenom obliku predstavljanja primjenskih sustava za *World Wide Web* daje i sve jača težnja prema preobrazbi primjenskih programa iz oblika za postavljanje na lokalno računalo u oblik dostupan putem *World Wide Web* sustava. Programski alati za koje je postojala dugogodišnja praksa pojavljivanja u obliku za postavljanje na lokalno računalo u današnje se vrijeme počinju nuditi u obliku usluga dostupnih putem *World Wide Web* stranica. Tablicom 6.4 prikazano je nekoliko učestalo primjenjivanih skupina primjenskih programa koji doživljavaju preobrazbu iz oblika za postavljanje na lokalno računalo u oblik dostupan putem *World Wide Web* stranica.

Tablica 6.4. Primjeri preobrazbe primjenskih programa iz oblika za postavljanje na lokalno računalo u oblik dostupan putem *World Wide Web* stranica

	Primjenski programi za postavljanje na lokalno računalo	**Primjenski programi zasnovani na *World Wide Web* sustavu**
Elektronička pošta	Pine, Outlook Express	Squirrel Mail, Hotmail, Gmail
Uredski alati	Microsoft Office, Open Office	Google Docs & Spreadsheets
Izrada *World Wide Web* stranica	Microsoft FrontPage, Adobe Dreamweaver	ezHTML, Google Page Creator
Višemedijski sadržaji	Windows Media Player, Winamp	YouTube, Pandora Radio

Osnovna zamisao paradigme programiranja prilagođenog potrošaču jest povezivanje informacija, usluga i procesa na predodžbenoj razini. Elementi od kojih se grade poosobljeni programi su gotovi primjenski programi izloženi putem grafičkog korisničkog sučelja prikazanog u pregledniku *World Wide Web* sadržaja. Povezivanje informacija, usluga i procesa na predodžbenoj razini obavlja se povezivanjem korisničkih sučelja putem kojih su informacije, usluge ili procesi izloženi potrošaču.

Tablicom 6.5 prikazana je ocjena najpoznatijih predstavnika razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo opažajnog doživljaja

značenja blokova. Svojstvo opažajnog doživljaja značenja blokova zadovoljavaju alati *Intel Mash Maker*, *IBM QEDwiki*, *Dapper* i *Marmite* koji koriste programske blokove u obliku udomljenika ili *World Wide Web* stranica. Pojedinim blokovima ostvaruje se pristup do potrošaču usmjerenih primjenskih funkcionalnosti, kao što je čitanje vijesti, pregled video sadržaja, naručivanje roba i usluga, pretvorba valuta, pregled zemljovida i slično. Značenje blokova predstavljenih u takvom obliku potrošaču je poznato na osnovi iskustva stečenog korištenjem primjenskih programa za *World Wide Web*.

Tablica 6.5. Ocjena razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo opažajnog doživljaja značenja blokova

Naziv alata	Opažajni doživljaj značenja blokova
Microsoft Popfly	-
Yahoo! Pipes	-
Intel Mash Maker	+
Google Mashup Editor	-
iGoogle Gadget Maker	-
IBM QEDwiki	+
OpenKapow	-
Proto	-
Dapper	+
Marmite	+
JackBe Presto	-

U programskim alatima *Microsoft Popfly*, *Proto* i *JackBe Presto* programski blokovi predstavljeni su apstraktnim grafičkim oznakama. U programskom alatu *Yahoo! Pipes* programski blokovi predstavljeni su elektroničkim obrascima koji se sastoje od polja za unos znakova, padajućih izbornika i označnih polja. Iako je primjena tih programskih blokova bliska potrošaču, njihovo je značenje usko povezano sa strukturom podataka dostupnih iz RSS izvorišta. Alat *Google Mashup Editor* koristi se slovčanim zapisom u obliku jezika HTML, XML i Javascript. Alati *iGoogle Gadget Maker* i *OpenKapow* ne koriste programske blokove za izgradnju primjenskih programa.

6.2.3 Samodostatnost blokova

Svojstvo *samodostatnosti programskih blokova* omogućuje postojanje i funkcioniranje programskih blokova izvan konteksta programske paradigme u kojoj se ti blokovi koriste kao elementi za izgradnju primjenskih programa. Ako je elemente koji se koriste kao blokovi

za izgradnju programa moguće istodobno koristiti i kao samostalne primjenske programe, onda programska paradigma pokazuje svojstvo samodostatnosti programskih blokova. Svojstvom samodostatnosti programskih blokova osigurava se da se u programsku paradigmu ne uključuju programski blokovi oblikovani isključivo za potrebe te paradigme, već se kao elementi za izgradnju primjenskih programa iskorištavaju otprije poznati programski elementi.

Postupak definiranja svojstva samodostatnosti blokova prikazan je tablicom 6.6. Svojstvo samodostatnosti blokova definirano je na osnovi *izravnosti relacije preslikavanja* kao jedne od mjera umnog napora tijekom primjene sustava znakovlja te *nadovezujuće inovacije* kao iskustvenog elementa proizašlog iz odnosa potrošača prema tehnologiji.

Tablica 6.6. Definicija svojstva samodostatnosti programskih blokova

Svojstvo prog. prilagođenog potrošaču	Polazište za definiciju svojstva
Samodostatnost blokova	**Mjere umnog napora** ▪ Izravnost relacije preslikavanja **Odnos potrošača prema tehnologiji** ▪ Nadovezujuća inovacija

Idealna potrošaču prilagođena programska paradigma omogućava razvoj primjenskih programa primjenom istih načela, pravila i postupaka koje su potrošači navikli koristiti tijekom uobičajene uporabe primjenskih programa. Na taj način razvoj novih primjenskih programa postaje prirodna nadogradnja na svakodnevnu uporabu postojećih primjenskih programa jer potrošač ne doživljava razliku u primijenjenim postupcima i korištenim tehnikama. S obzirom na to da omogućava zadržavanje istog umnog režima za vrijeme korištenja i za vrijeme izgradnje primjenskih programa, takva programska paradigma za potrošače računalnih primjenskih programa predstavlja oblik *nadovezujuće inovacije*. Tehnologijski proizvodi i usluge zasnovani na nadovezujućim inovacijama znatno brže i lakše nailaze na prihvaćanje od strane potrošača od onih zasnovanih na korjenitim inovacijama koje zahtijevaju značajne promjene u načinu razmišljanja i djelovanja.

Za ostvarenje programske paradigme u obliku nadovezujuće inovacije koriste se programski blokovi koje je moguće ravnopravno koristiti kao samostalne primjenske programe i kao elemente za izgradnju novih primjenskih programa. Ako programski blokovi u okviru programske paradigme zadržavaju isti oblik, funkciju i način primjene u kojem se koriste izvan paradigme, onda je uspostavljena *izravnost relacije preslikavanja* između izravnog rukovanja blokom i izgradnje programa za upravljanje izvođenjem bloka.

Tablica 6.7. Ocjena razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo samodostatnosti blokova

Naziv alata	Samodostatnost blokova
Microsoft Popfly	-
Yahoo! Pipes	-
Intel Mash Maker	+
Google Mashup Editor	-
iGoogle Gadget Maker	-
IBM QEDwiki	+
OpenKapow	-
Proto	-
Dapper	+
Marmite	-
JackBe Presto	-

Tablicom 6.7 prikazana je ocjena najpoznatijih predstavnika razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo samodostatnosti blokova. Alat *Intel Mash Maker* omogućuje iskorištavanje udomljenika zasnovanih na *Google Gadgets* specifikaciji u svojstvu programskih blokova. Tu vrstu udomljenika moguće je koristiti kao gotove primjenske programe izvan konteksta razvojnog alata. Alat *IBM QEDwiki* koristi posebno oblikovane udomljenike za primjenu u poslovnim primjenskim programima. Iako su posebno oblikovani za alat *IBM QEDwiki*, udomljenike je u sličnom obliku moguće koristiti i kao gotove primjenske programe izvan konteksta razvojnog alata. Alat *Dapper* koristi proizvoljne *World Wide Web* stranice u svojstvu programskih blokova. Ostali alati ne zadovoljavaju svojstvo samodostatnosti blokova jer koriste posebno oblikovane blokove koji izvan konteksta alata nemaju definiranu funkciju.

6.2.4 Višerazinsko usložnjavanje blokova

Svojstvo *višerazinskog usložnjavanja blokova* zadovoljava programska paradigma koja dopušta izgradnju primjenskog programa postupnim povezivanjem programskih blokova kroz više razina. Od skupa osnovnih programskih blokova oblikuje se novi programski blok koji se u nastavku izgradnje programa koristi kao osnovni blok.

Postupak definiranja svojstva višerazinskog usložnjavanja blokova prikazan je tablicom 6.8. Svojstvo višerazinskog usložnjavanja blokova definirano je na osnovi *teorije kratkoročnog pamćenja privremenih činjenica* kao jednog od modela psihologije

programiranja te na osnovi *stupnja smislenog objedinjavanja*, *postupnog vrednovanja* i *složenosti rasuđivanja* kao mjera umnog napora tijekom primjene sustava znakovlja.

Tablica 6.8. Definicija svojstva višerazinskog usložnjavanja blokova

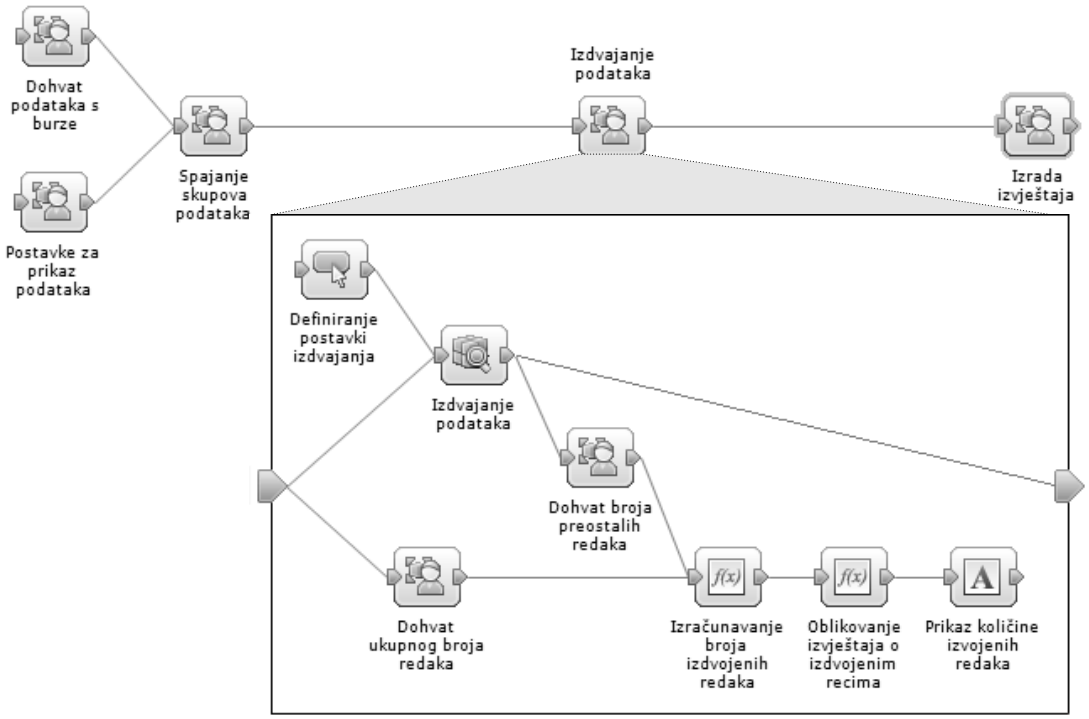
Svojstvo prog. prilagođenog potrošaču	Polazište za definiciju svojstva
Višerazinsko usložnjavanje blokova	**Psihologija programiranja** ▪ Teorija kratkoročnog pamćenja privremenih činjenica **Mjere umnog napora** ▪ Stupanj smislenog objedinjavanja ▪ Postupno vrednovanje ▪ Složenost rasuđivanja

Svojstvo višerazinskog usložnjavanja blokova prikazano je slikom 6.10. U primjeru višerazinskog usložnjavanja programskih blokova prikazanom na slici 6.10, usloženi primjenski program izgrađen je primjenom programskih blokova za *dohvat podataka s burze*, definiranje *postavki za prikaz podataka*, *spajanje skupova podataka*, *izdvajanje podataka* i *izradu izvještaja*. Programski blok za *izradu izvještaja* je složeni programski blok koji se sastoji od sedam programskih blokova na nižoj razini usložnjavanja. Međutim, složeni blok za *izradu izvještaja* je tijekom izgradnje programske logike na gornjoj razini usložnjavanja korišten u svojstvu osnovnog programskog bloka. Prikazani primjer izgrađen je primjenom programskog alata *Proto* [193].

Uloga višerazinskog usložnjavanja programskih blokova u programiranju prilagođenom potrošaču ima višestruke povoljne učinke s gledišta kognitivne znanosti. Prema *teoriji kratkoročnog pamćenja privremenih činjenica*, prividno povećanje veličine kratkoročnog pamćenja čovjeka postiže se nadomještanjem skupa smisleno povezanih činjenica jedinstvenom činjenicom na višoj razini apstrakcije. Pogodovanje toj vrsti prirođene umne aktivnosti, u programskoj paradigmi prilagođenoj potrošaču postiže se izlaganjem skupa povezanih blokova koji predstavljaju zaokruženu cjelinu u obliku novog bloka. Ovaj oblik s*mislenog objedinjavanja* skupa blokova i njihovih međusobnih veza u nastavku izgradnje primjenskog programa omogućuje rukovanje čitavom složenom strukturom blokova kao jedinstvenom činjenicom u kratkoročnom pamćenju potrošača.

Višerazinsko usložnjavanje programskih blokova omogućuje postupno oblikovanje i izgradnju primjenskog programa. Postupnim usložnjavanjem blokova, usredotočenost potrošača zadržava se na pojedinim dijelovima programa, umjesto na cjelokupnom programu odjednom. Osim što se time rasterećuje kratkoročno pamćenje, postupnom izgradnjom

primjenskog programa potrošaču je omogućeno *postupno vrednovanje* vlastitog napredovanja tijekom izrade programa.



Slika 6.10. Višerazinsko usložnjavanje programskih blokova

Svojstvo višerazinskog usložnjavanja programskih blokova omogućava oblikovanje primjenskih programa razmatranjem isključivo dvaju razreda programskih blokova. Tijekom razvoja primjenskog programa, korisnik programske paradigme razlikuje samo osnovne blokove i složeni blok. Osnovni blokovi koriste se za izgradnju složenog bloka. Ako se složeni blok koristi za izgradnju novog složenog bloka na višoj razini usložnjavanja, onda taj blok s gledišta korisnika paradigme poprima svojstva osnovnog bloka. Za korisnika paradigme nije nužno razlikovanje osnovnih od složenih programskih blokova u skupu blokova koje koristi za izgradnju novog bloka. Razlikovanjem isključivo dvaju razreda programskih blokova smanjuje se *složenost rasuđivanja*.

Tablica 6.9. Ocjena razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo višerazinskog usložnjavanja blokova

Naziv alata	Višerazinsko usložnjavanje blokova
Microsoft Popfly	-
Yahoo! Pipes	-
Intel Mash Maker	-
Google Mashup Editor	-

Naziv alata	Višerazinsko usložnjavanje blokova
iGoogle Gadget Maker	-
IBM QEDwiki	-
OpenKapow	-
Proto	+
Dapper	+
Marmite	-
JackBe Presto	-

Tablicom 6.9 prikazana je ocjena najpoznatijih predstavnika razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo višerazinskog usložnjavanja blokova. Višerazinsko usložnjavanje programskih blokova podržavaju alati *Proto* i *Dapper*. Alat *Proto* omogućuje višerazinsko usložnjavanje programskih blokova kao što je prikazano slikom 6.10. Alat *Dapper* omogućuje povezivanje sličnih *World Wide Web* stranica u novu stranicu s objedinjenim funkcionalnostima. Dobivenu stranicu moguće je nastaviti dalje povezivati sa sličnim stranicama. Kod ostalih alata ne postoji mogućnost povezivanja skupa blokova u istovrsni složeni blok. Kod tih je alata funkcionalnost skupa povezanih blokova uglavnom izložena u obliku *World Wide Web* stranica ili *RSS* izvorišta, što nije istovjetno predodžbi programskih blokova unutar alata.

6.2.5 Jednakost korištenja i povezivanja blokova

Svojstvom samodostatnosti programskih blokova, koje je opisano u poglavlju 6.2.3, zahtijeva se oblikovanje programske paradigme u kojoj se programski blokovi pojavljuju u obliku elemenata koje je izvan konteksta programske paradigme moguće koristiti kao samostalne primjenske programe. Svojstvom *jednakosti korištenja i povezivanja blokova* zahtijeva se oblikovanje programske paradigme u kojoj se programsko povezivanje nezavisnih blokova u radni tijek za ostvarenje novog primjenskog programa izvodi istim tehnikama koje se koriste tijekom uporabe tih blokova izvan konteksta programske paradigme.

Postupak definiranja svojstva jednakosti korištenja i povezivanja blokova prikazan je tablicom 6.10. Svojstvo jednakosti korištenja i povezivanja blokova definirano je na osnovi *teorije održavanja i preispitivanja znanja* kao jednog od modela psihologije programiranja, na osnovi *dosljednosti*, *izravnosti relacije preslikavanja* i *skrivenih zavisnosti* kao mjera umnog napora tijekom primjene sustava znakovlja te *nadovezujuće inovacije* kao iskustvenog elementa proizašlog iz odnosa potrošača prema tehnologiji.

U skladu s *teorijom održavanja i preispitivanja znanja*, učenje i primjena programskog jezika za povezivanje programskih blokova u radni tijek olakšani su i ubrzani ako se programski jezik oslanja na prethodno stečena znanja potrošača. Budući da se programska paradigma prilagođena potrošaču zasniva na povezivanju programskih blokova u obliku samostalnih i nezavisnih primjenskih programa, programski jezik za povezivanje blokova sastoji se od elemenata koji predstavljaju izravnu primjenu blokova izvan konteksta paradigme. Iskustvo pokazuje da su u nedostatku primjenskih programa koji nude cjelokupne funkcionalnosti za njihove poosobljene potrebe, potrošači složene zadaće putem računala navikli obavljati usklađenom primjenom skupa nezavisnih primjenskih programa od kojih svaki pruža dio funkcionalnosti. Programski jezik za povezivanje blokova u radni tijek je, stoga, potrebno oblikovati od elemenata za upravljanje izvođenjem pojedinačnih programa i prijenos podataka između pojedinih programa.

Tablica 6.10. Definicija svojstva jednakosti korištenja i povezivanja blokova

Svojstvo prog. prilagođenog potrošaču	Polazište za definiciju svojstva
Jednakost korištenja i povezivanja blokova	**Psihologija programiranja** ▪ Teorija održavanja i preispitivanja znanja **Mjere umnog napora** ▪ Dosljednost ▪ Izravnost relacije preslikavanja ▪ Skrivene zavisnosti **Odnos potrošača prema tehnologiji** ▪ Nadovezujuća inovacija

Izjednačavanjem izravnog rukovanja i programskog povezivanja blokova, programska paradigma zadržava *dosljednost* primjene blokova u svojstvu samostalnih primjenskih programa i u svojstvu elemenata za izgradnju radnog tijeka usloženog primjenskog programa. Dosljednošću primjene skupa blokova unutar i izvan programske paradigme, izgradnja primjenskih programa za potrošača postaje *nadovezujuća inovacija* na njihovu uobičajenu uporabu jer je isti skup znanja, iskustava, tehnika i pravila moguće primijeniti za novi oblik potrošačkog djelovanja. Nadalje, podudarnošću izravnog rukovanja i programskog povezivanja blokova uspostavlja se *izravnost relacije preslikavanja* iz poznatog prostora izravne primjene blokova u novonastali prostor izgradnje programa od skupa blokova.

Osim što se pojednostavljuje izgradnja radnog tijeka novog primjenskog programa, izravnim preslikavanjem elemenata programskog jezika u radnje potrošača tijekom ručnog upravljanja izvođenjem blokova pojednostavljuje se i tumačenje značenja izgrađenog

programa jer se izbjegava pojava *skrivenih zavisnosti* u zapisu programa. Svi elementi povezivosti blokova koje potrošač koristi tijekom izravnog rukovanja blokovima pojavljuju se i u programskom zapisu radnog tijeka.

Tablica 6.11. Ocjena razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo jednakosti korištenja i povezivanja blokova

Naziv alata	Jednakost korištenja i povezivanja blokova
Microsoft Popfly	-
Yahoo! Pipes	-
Intel Mash Maker	+/-
Google Mashup Editor	-
iGoogle Gadget Maker	-
IBM QEDwiki	+/-
OpenKapow	-
Proto	-
Dapper	+
Marmite	-
JackBe Presto	-

Tablicom 6.11 prikazana je ocjena najpoznatijih predstavnika razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo jednakosti korištenja i povezivanja blokova. Ovo svojstvo zadovoljava alat *Dapper* kod kojeg se povezivanje skupa sličnih *World Wide Web* stranica izvodi primjenom istih tehnika kao i tijekom njihova uobičajenog korištenja izvan konteksta razvojnog alata. Svojstvo djelomično zadovoljavaju i alati *Intel Mash Maker* i *IBM QEDwiki*. Kod tih su dvaju alata postupci odabira i prostorne organizacije blokova u obliku udomljenika izjednačeni s postupcima korištenja udomljenika izvan razvojnog alata. Međutim, uspostava toka podataka između udomljenika izvodi se automatizirano na osnovi ugrađenih informacija o podudarnosti značenja i strukture ulaznih i izlaznih podataka, dok se izvan konteksta alata za tu svrhu uobičajeno koriste postupci preuzimanja i preslikavanja sadržaja (engl. *copy/paste*). Ostali alati koriste posebno oblikovane programske blokove koji izvan konteksta razvojnog alata nemaju definiranu funkciju pa postupci korištenja i povezivanja blokova za te alate nisu usporedivi.

6.2.6 Nezavisnost povezujućih elemenata i značenja blokova

Svojstvo *nezavisnosti povezujućih elemenata i značenja blokova* zadovoljava programska paradigma u kojoj se povezivanje blokova obavlja konačnim skupom

povezujućih elemenata koji su nezavisni od primjenske funkcije pojedinih blokova. Oblikovanjem programske paradigme koja zadovoljava ovo svojstvo omogućava se oblikovanje beskonačnog broja različitih uzoraka povezivanja blokova primjenom konačnog skupa povezujućih elemenata.

Postupak definiranja svojstva nezavisnosti povezujućih elemenata i značenja blokova prikazan je tablicom 6.12. Svojstvo nezavisnosti povezujućih elemenata i značenja blokova definirano je na osnovi *teorije kratkoročnog pamćenja privremenih činjenica* kao jednog od modela psihologije programiranja te na osnovi *dosljednosti* i *prionljivosti* kao mjera umnog napora tijekom primjene sustava znakovlja.

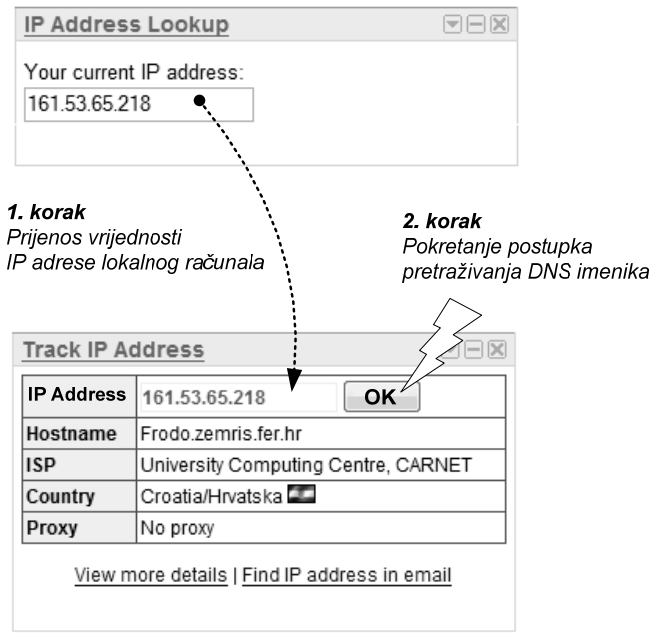
Tablica 6.12. Definicija svojstva nezavisnosti povezujućih elemenata i značenja blokova

Svojstvo prog. prilagođenog potrošaču	Polazište za definiciju svojstva
Nezavisnost povezujućih elemenata i značenja blokova	**Psihologija programiranja** ▪ Teorija kratkoročnog pamćenja privremenih činjenica **Mjere umnog napora** ▪ Dosljednost ▪ Prionljivost

Tijekom izgradnje primjenskog programa, kratkoročno pamćenje potrošača zauzeto je planiranjem uključenosti blokova u radni tijek i analizom primjenjivosti raspoloživih povezujućih elemenata u pojedinim točkama radnog tijeka. *Teorijom kratkoročnog pamćenja privremenih činjenica* pokazano je da je kratkoročno pamćenje, u kojem se odvijaju ključni elementi umnog djelovanja potrošača tijekom izgradnje programa, ograničene veličine. Budući da je planiranje uključenosti blokova u radni tijek svojstveno pojedinom primjenskom programu, ono neizbježno zauzima dio kratkoročnog pamćenja potrošača tijekom čitavog trajanja izgradnje programa. S druge strane, učinkovitost umnog djelovanja to je veća što je veći dio ukupne količine kratkoročnog pamćenja moguće posvetiti planiranju uključenosti blokova u radni tijek. Učinkovitost umnog djelovanja potrošača je, stoga, moguće povećati rasterećenjem onog dijela kratkoročnog pamćenja koje je zauzeto analizom primjenjivosti raspoloživih povezujućih elemenata u pojedinim točkama radnog tijeka. To je moguće postići ako se u programsku paradigmu uključe povezujući elementi nad kojima je analizu primjenjivosti za povezivanje blokova, umjesto tijekom izgradnje svakog pojedinog radnog tijeka, moguće provesti jednokratno tijekom učenja programske paradigme. Na taj način znanje o primjenjivosti povezujućih elemenata postaje ugrađeno u dugotrajno pamćenje potrošača. Kako bi analiza primjenjivosti povezujućih elemenata

postala dijelom procesa učenja programske paradigme, povezujuće elemente potrebno je oblikovati na način da su njihovo značenje i funkcija neovisni o značenju pojedinih programskih blokova i njihovoj ulozi u radnom tijeku primjenskog programa.

Nezavisnost povezujućih elemenata i značenja blokova omogućuje povezivanje proizvoljnih programskih blokova primjenom konačnog broja unaprijed definiranih povezujućih elemenata. Time se postiže *dosljednost* programske paradigme jer se ostvarenje istih ili sličnih uzoraka povezivosti blokova uvijek izvodi primjenom iste vrste povezujućih elemenata, bez obzira na primjenska svojstva radnog tijeka. Nadalje, nezavisnošću povezujućih elemenata i značenja blokova smanjuje se *prionljivost* programske paradigme, odnosno otpornost izgrađenog programa na izmjene, jer je zamjena, dodavanje ili uklanjanje blokova iz radnog tijeka ne sužava niti ne proširuje skup raspoloživih povezujućih elemenata.



Slika 6.11. Primjenski program za pretraživanje DNS imenika

Povoljne značajke programske paradigme sa svojstvom nezavisnosti povezujućih elemenata o značenju programskih blokova pokazane su na dva primjera izgradnje primjenskih programa. Programske blokove za izgradnju primjenskih programa predstavljaju grafička korisnička sučelja prikazana u pregledniku *World Wide Web* sadržaja. Povezivanje programskih blokova ostvaruje se opisivanjem uzajamnog djelovanja potrošača i primjenskog programa putem pripadajućeg korisničkog sučelja. U takvim okolnostima, povezujućim elementima razvojnog modela smatraju se naredbe za opis događaja nad elementima grafičkog korisničkog sučelja.

Slikom 6.11 prikazan je primjer primjenskog programa za pretraživanje DNS imenika. U primjeru se pretpostavlja da potrošač nastoji izgraditi složeni primjenski program za određivanje DNS imena lokalnog računala. Složeni primjenski program izgrađen je od dva programska bloka koji predstavljaju korisnička sučelja prema dva jednostavnija primjenska programa. Prvi program pod nazivom *IP Address Lookup* koristi se za pribavljanje IP adrese potrošačevog računala. Drugi program pod nazivom *Track IP Address* koristi se za pristup DNS imeniku. Na osnovi zadane IP adrese računala, ovaj program omogućava pribavljanje podataka o DNS imenu računala, pružatelju pristupa, državi iz koje se obavlja pristup te poslužitelju zastupniku preko kojeg zadano računalo ostvaruje pristup na mrežu Internet.

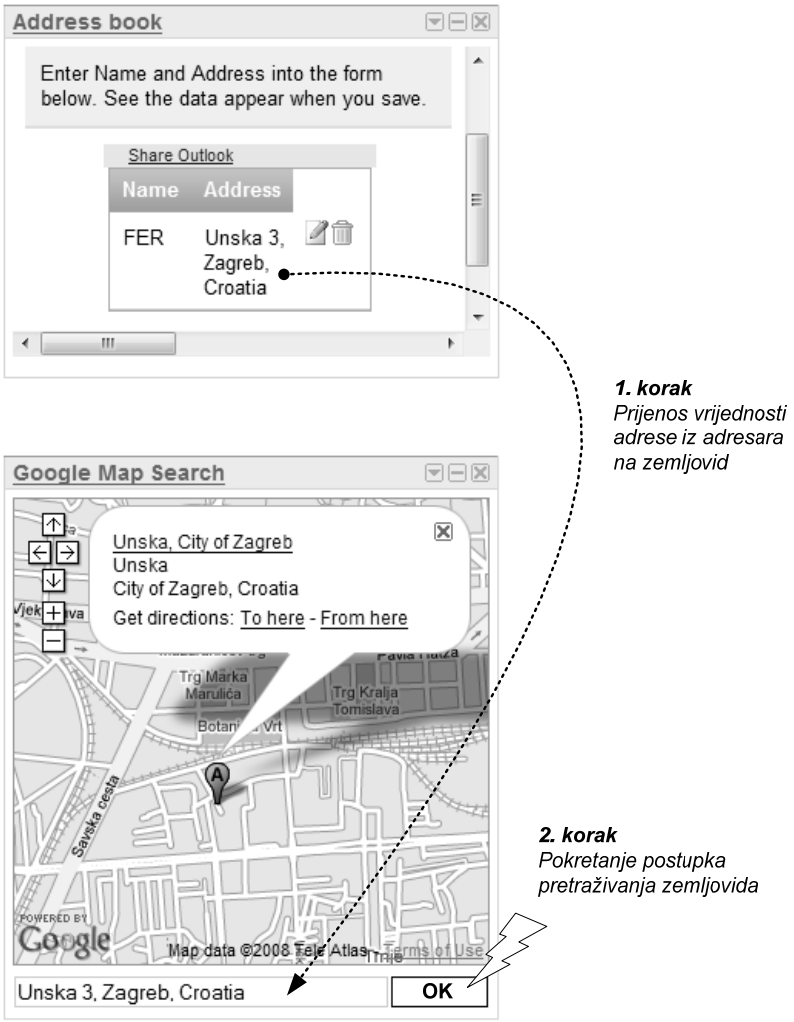
Povezivanje dvaju međusobno nezavisnih primjenskih programa u usloženi primjenski program moguće je obaviti u dva koraka. U prvom koraku potrebno je vrijednost IP adrese lokalnog računala, koja je izlazni podatak programa *IP Address Lookup*, prenijeti u program *Track IP Address* koji kao ulazni podatak prima proizvoljnu IP adresu. U drugom koraku potrebno je u programu *Track IP Address* pokrenuti postupak pretraživanja DNS imenika za zadanu IP adresu.

```
copy Your current IP address at IP Address Lookup
paste into IP Address at Track IP Address
click OK at Track IP Address
```

Slika 6.12. Povezujući elementi za izgradnju primjenskog programa za pretraživanje DNS imenika

S obzirom da su tijekom izgradnje usloženog primjenskog programa osnovni računalni procesi potrošaču predstavljeni o obliku grafičkih korisničkih sučelja, povezivanje dvaju nezavisnih programa moguće je izvesti opisom međudjelovanja korisnika i računala putem korisničkih sučelja dvaju programa. Slikom 6.12 prikazan je računalni program kojim se opisuje povezujuća logika u obliku uzajamnog djelovanje korisnika i računala putem grafičkog korisničkog sučelja programa *IP Address Lookup* i *Track IP Address*. Prikazani program sastoji se od tri naredbe, odnosno tri povezujuća elementa. Prvim povezujućim elementom definirano je da se obavlja operacija preuzimanja vrijednosti zapisane u polju za prikaz teksta pod nazivom *Your current IP address* s korisničkog sučelja programa *IP Adress Lookup*. Drugim povezujućim elementom definirano je da se obavlja operacija upisivanja preuzete vrijednosti u polje za unos teksta pod nazivom *IP address* na korisničkom sučelju programa *Track IP Adress*. Konačno, trećim povezujućim elementom definirano je da se obavlja operacija pritiska na tipku pod nazivom *OK* na korisničkom sučelju programa *Track IP Adress* kojom se pokreće postupak pretraživanja DNS imenika.

Drugi primjer usloženog primjenskog programa prikazan je slikom 6.13. Prikazani program koristi se za pretraživanje zemljovida u elektroničkom obliku. U primjeru se pretpostavlja da potrošač nastoji izgraditi primjenski program za pronalaženje položaja na zemljovidu na kojem se nalazi izabrana adresa iz osobnog adresara. Složeni primjenski program izgrađen je od dva programska bloka koji predstavljaju korisnička sučelja prema dva jednostavnija primjenska programa. Prvi program pod nazivom *Address book* koristi se za pregledavanje zapisa o osobnim kontaktima spremljenim u osobnom elektroničkom adresaru. Drugi program pod nazivom *Google Map Search* koristi se za pretraživanje elektroničkih zemljovida. Na osnovi zadanog naziva ulice u naseljenom mjestu, ovaj program omogućava prikaz položaja tražene točke na zemljovidu.



Slika 6.13. Primjenski program za pretraživanje zemljovida

Slično primjeru usloženog primjenskog programa prikazanog na slici 6.11, i u ovom je slučaju povezivanje dvaju međusobno nezavisnih primjenskih programa u usloženi

primjenski program moguće obaviti u dva koraka. U prvom koraku potrebno je podatak o nazivu ulice i naseljenog mjesta, koja je izlazni podatak programa *Address book*, prenijeti u program *Google Map Search* koji kao ulazni podatak prima proizvoljnu zemljopisnu adresu. U drugom koraku potrebno je u programu *Google Map Search* pokrenuti postupak pretraživanja zemljovida.

```
copy Address at Address book

paste into Address at Google Map Search

click OK at Google Map Search
```

Slika 6.14. Povezujući elementi za izgradnju primjenskog sustava za pretraživanje zemljovida

Slikom 6.14 prikazan je računalni program kojim se opisuje povezujuća logika u obliku uzajamnog djelovanja korisnika i računala putem grafičkog korisničkog sučelja programa *Address book* i *Google Map Search*. Prikazani program sastoji se od tri naredbe kojima se definiraju povezujući elementi za usložnjavanje procesa. Prvim povezujućim elementom definirano je da se obavlja operacija preuzimanja vrijednosti zapisane u polju za prikaz teksta pod nazivom *Address* s korisničkog sučelja programa *Adress book*. Drugim povezujućim elementom definirano je da se obavlja operacija upisivanja preuzete vrijednosti u polje za unos teksta pod nazivom *Address* na korisničkom sučelju programa *Google Map Search*. Konačno, trećim povezujućim elementom definirano je da se obavlja operacija pritiska na tipku pod nazivom *OK* na korisničkom sučelju programa *Google Map Search* kojom se pokreće postupak pretraživanja elektroničkog zemljovida.

Prikazani primjeri pokazuju svojstvo nezavisnosti povezujućih elemenata i značenja programskih blokova za izgradnju primjenskih programa. Dva različita primjenska programa izgrađena su primjenom triju istih povezujućih elemenata. Prikazanim povezujućim elementima redom se opisuju operacija preuzimanja vrijednosti prikazane elementom grafičkog korisničkog sučelja, operacija upisivanja preuzete vrijednosti u element grafičkog korisničkog sučelja te operacija pritiska na element grafičkog korisničkog sučelja. Iako pojedini povezujući elementi imaju dobro definirano značenje za povezivanje programskih blokova, značenje povezujućih elemenata nije vezano uz značenje blokova koji se njima povezuju. Primjerice, povezujućem elementu *click* pridruženo je značenje pritiska na element grafičkog korisničkog sučelja, bez obzira na značenje bloka nad čijim se korisničkim sučeljem djelovanje povezujućeg elementa primjenjuje. Međutim, učinak operacije uzrokovane djelovanjem povezujućeg elementa ima značenje koje je vezano uz značenje bloka. Učinak povezujućeg elementa *click* u prvom slučaju ima značenje pokretanja

pretraživanja DNS imenika, a u drugom pokretanje pretraživanja zemljovida. S obzirom da je zadatak graditelja usloženog primjenskog programa odabir i povezivanje programskih blokova kako bi se ostvarila funkcija složenog procesa, od njega se očekuje poznavanje značenja izabranih programskih blokova i učinaka djelovanja pojedinih povezujućih elemenata nad izabranim blokovima.

Tablica 6.13. Ocjena razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo nezavisnosti povezujućih elemenata i značenja blokova

Naziv alata	Nezavisnost povezujućih elemenata
Microsoft Popfly	+
Yahoo! Pipes	+
Intel Mash Maker	-
Google Mashup Editor	+
iGoogle Gadget Maker	-
IBM QEDwiki	-
OpenKapow	-
Proto	+
Dapper	+
Marmite	-
JackBe Presto	+

Tablicom 6.13 prikazana je ocjena najpoznatijih predstavnika razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo nezavisnosti povezujućih elemenata i značenja blokova. Alati *Microsoft Popfly*, *Yahoo! Pipes*, *Proto* i *JackBe Presto* za povezivanje blokova koriste grafičke elemente u obliku strelica ili linija. Istom vrstom poveznica moguće je povezivati raznovrsne programske blokove. Alat *Google Mashup Editor* koristi se okidačima događaja i funkcijskim pozivima. Iako je značenje pojedinog funkcijskog poziva povezano sa značenjem ciljnog programskog bloka, mehanizam funkcijskog poziva zajednički je za sve blokove. Alat *Dapper* koristi se mehanizmima označavanja i izrezivanja dijelova *World Wide Web* stranica. Ostali alati ne zadovoljavaju svojstvo nezavisnosti povezujućih elemenata i značenja blokova. Alati *Intel Mash Maker*, *IBM QEDwiki* i *Marmite* ne koriste posebne elemente za povezivanje blokova. U tim alatima, blokovi se automatski povezuju na osnovi ugrađenih informacija o podudarnosti značenja i strukture ulaznih i izlaznih podataka. Alati *iGoogle Gadget Maker* i *OpenKapow* ne koriste se programskim blokovima za izgradnju primjenskih programa.

6.2.7 Izgradivost primjenskog programa putem grafičkog korisničkog sučelja

Svojstvo *izgradivosti primjenskog programa putem grafičkog korisničkog sučelja* zadovoljava programska paradigma koja omogućava izgradnju programske logike za povezivanje skupa blokova u radni tijek putem grafičkog korisničkog sučelja. Za takvu je programsku paradigmu moguće izgraditi potpomaganu razvojnu okolinu koja potrošača vodi kroz postupak izgradnje programa.

Postupak definiranja svojstva izgradivosti primjenskog programa putem grafičkog korisničkog sučelja prikazan je tablicom 6.14. Svojstvo izgradivosti primjenskog programa putem grafičkog korisničkog sučelja definirano je na osnovi *teorije kratkoročnog pamćenja privremenih činjenica* kao jednog od modela psihologije programiranja, na osnovi *podložnosti pogreškama* kao jedne od mjera umnog napora tijekom primjene sustava znakovlja te *nadovezujuće inovacije* i *grafičkog korisničkog sučelja* kao iskustvenih elemenata proizašlih iz odnosa potrošača prema tehnologiji.

Tablica 6.14. Definicija svojstva izgradivosti primjenskog programa putem grafičkog korisničkog sučelja

Svojstvo prog. prilagođenog potrošaču	Polazište za definiciju svojstva
Izgradivost primjenskog programa putem grafičkog korisničkog sučelja	**Psihologija programiranja** ▪ Teorija kratkoročnog pamćenja privremenih činjenica **Mjere umnog napora** ▪ Podložnost pogreškama **Odnos potrošača prema tehnologiji** ▪ Nadovezujuća inovacija ▪ Grafičko korisničko sučelje

Izborom samoopisivih programskih blokova i povezujućih elemenata u obliku operacija za izravno rukovanje blokovima, gradivni elementi i programski jezik za oblikovanje radnog tijeka oslanjaju se na znanja i vještine potrošača stečene prethodnom uporabom računalnih primjenskih programa. Na taj način oblikovanje novih primjenskih programa postaje nadovezujuća inovacija na njihovu uobičajenu primjenu, čime se smanjuje ili u potpunosti izbjegava početni umni napor potreban za učenje programske paradigme. Međutim, iako je značenje gradivnih i povezujućih elemenata blisko umnom režimu potrošača, postavlja se pitanje odabira tehnike za njihovo objedinjavanje u radni tijek. Tehnika za izgradnju radnog tijeka također treba biti zasnovana na načelima koja su

potrošaču poznata na osnovi općeg znanja ili iskustva stečenog primjenom računala. Na taj način, uz oblikovanje, i izgradnja radnog tijeka poprima oblik *nadovezujuće inovacije*.

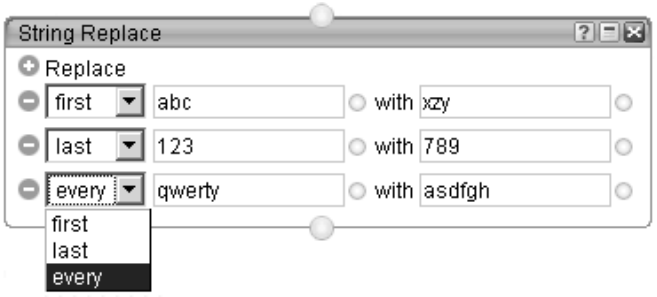
Uobičajena tehnika izgradnje programa u grani profesionalnog razvoja programske potpore je primjena programskih jezika zasnovanih na slovčanom zapisu. Slikom 6.15 prikazan je isječak programa u jeziku *Javascript* s tri pravila za zamjenu uzoraka teksta. Prvim pravilom definirana je zamjena prvog pojavljivanja uzorka *abc* uzorkom *xzy*. Drugim pravilom definirana je zamjena posljednjeg pojavljivanja uzorka *123* uzorkom *789*. Trećim pravilom definirana je zamjena svakog pojavljivanja uzorka *qwerty* uzorkom *asdfgh*.

```
var s = new String();
s = s.replace(/abc/, "xyz");
s = s.replace(/(.*)123/, "$1789");
s = s.replace(/qwerty/g, "asdfg");
```

Slika 6.15. Izgradnja programa primjenom programskog jezika zasnovanog na slovčanom zapisu

Primjena programskih jezika zasnovanih na slovčanom zapisu zahtijeva način razmišljanja i djelovanja koji nije vezan uz uobičajene postupke koje potrošači koriste tijekom primjene računala. Ova tehnika izgradnje programa zahtijeva poznavanje pojmova vezanih uz programsko inženjerstvo, kao što su programske varijable, programski objekti, funkcije i regularni izrazi. Osim neizbježne usredotočenosti na oblikovanje radnog tijeka primjenskog programa, primjena programskih jezika zasnovanih na slovčanom zapisu unosi dodatno umno opterećenje jer je dio *kratkoročnog pamćenja* programera zauzet obradom leksičkih i sintaksnih pravila jezika. Osim toga, tehnika izgradnje programa koja zahtijeva stalnu usredotočenost na leksička i sintaksna pravila jezika pokazuje razmjerno visoki stupanj *podložnosti pogreškama* tijekom stvaranja programskog zapisa.

Prilagodba tehnike izgradnje programa umnom režimu, znanju i iskustvu potrošača je zamjena slovčanog zapisa grafičkim korisničkim sučeljem. Analizom odnosa potrošača prema tehnologiji, *grafičko korisničko sučelje* se pokazalo prevladavajućom tehnikom međudjelovanja čovjeka i računala kojom većina potrošača uspijeva ovladati bez posebnog napora i utroška vremena. Izgradnja programske logike putem grafičkog korisničkog sučelja zasniva se na potpomaganoj razvojnoj okolini koja potrošača vodi kroz postupak izgradnje programa. Na taj način programska paradigma postaje manje podložna pogreškama, a potrošaču se, zbog rasterećenja kratkoročnog pamćenja, ostavlja više prostora za usredotočavanje na primjenska svojstva postavljenog problema. Primjeri potpomaganih razvojnih okolina su korisnička sučelja u obliku elektroničkih obrazaca (engl. *forms*) i računalnih čarobnjaka (engl. *wizards*) [131, 132].



Slika 6.16. Izgradnja programske logike putem grafičkog korisničkog sučelja

Slikom 6.16 prikazan je postupak izgradnje programa istovjetnog onom prikazanom na slici 6.15 primjenom grafičkog korisničkog sučelja u obliku elektroničkog obrasca. Potpomagana razvojna okolina sastoji se od skupa obrazaca s unaprijed definiranom ulogom. Primjenski program gradi se odabirom odgovarajućih obrazaca i popunjavanjem ulaznih polja vrijednostima koje su svojstvene primjenskom programu. U postupku programiranja putem grafičkog korisničkog sučelja koriste se elementi u obliku polja za unos teksta, tipki, označnih polja, izbornih polja i padajućih izbornika koji su potrošaču poznati i razumljivi na osnovi iskustva stečenog svakodnevnom uporabom primjenskih programa za *World Wide Web*. Na osnovi potrošačeva djelovanja nad elementima grafičkog korisničkog sučelja, pozadinska programska potpora stvara zapis programa. Prikazani primjer izgrađen je primjenom programskog alata *Yahoo! Pipes* [177, 178, 184, 185, 186].

Tablica 6.15. Ocjena razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo izgradivosti primjenskog programa putem grafičkog korisničkog sučelja

Naziv alata	Izgradivost putem grafičkog korisničkog sučelja
Microsoft Popfly	+
Yahoo! Pipes	+
Intel Mash Maker	+
Google Mashup Editor	-
iGoogle Gadget Maker	+
IBM QEDwiki	+
OpenKapow	+
Proto	+
Dapper	+
Marmite	+
JackBe Presto	+

Tablicom 6.15 prikazana je ocjena najpoznatijih predstavnika razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo izgradivosti primjenskog programa putem grafičkog korisničkog sučelja. Gotovo svi prikazani alati koriste se grafičkim korisničkim sučeljem za izgradnju logike primjenskog programa. Alati *Microsoft Popfly*, *Proto* i *JackBe Presto* koriste usmjerene grane za povezivanje grafičkih oznaka. Alat *Yahoo! Pipes* koristi grafičke elemente u obliku cijevi koje služe za prijenos podataka između elektroničkih obrazaca, odakle potječe i naziv alata. U alatu *Marmite*, povezivanje blokova izvodi se prostornim razmještajem blokova po radnoj površini alata. Susjedni blokovi međusobno su podatkovno povezani. U alatima *Intel Mash Maker*, *IBM QEDwiki* i *Dapper*, primjenski se program gradi na osnovi rukovanja udomljenicima i *World Wide Web* stranicama putem grafičkog korisničkog sučelja prikazanog u pregledniku *World Wide Web* sadržaja. Od svih prikazanih alata, jedino se alat *Google Mashup Editor* koristi programskim jezikom zasnovanim na slovčanom zapisu pa ne zadovoljava svojstvo izgradnje povezujućih elemenata putem grafičkog korisničkog sučelja.

6.2.8 Upravljivost i uočljivost vremenske relacije

Svojstvo *upravljivosti i uočljivosti vremenske relacije* zadovoljava programska paradigma u kojoj je redoslijed izvođenja programskih blokova povezanih u radni tijek pod nadzorom i utjecajem potrošača. Tim je svojstvom omogućeno definiranje različitih vremenskih relacija nad istim skupom programskih blokova te oblikovanje složenih uzoraka izvođenja programa.

Tablica 6.16. Definicija svojstva upravljivosti i uočljivosti vremenske relacije

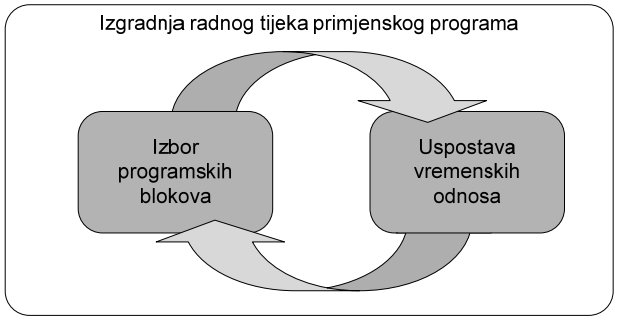
Svojstvo prog. prilagođenog potrošaču	Polazište za definiciju svojstva
Upravljivost i uočljivost vremenske relacije	**Psihologija programiranja** ▪ Teorija višeprolaznog iščitavanja **Mjere umnog napora** ▪ Prionljivost ▪ Preuranjeno odlučivanje ▪ Uočljivost i usporedivost ▪ Skrivene zavisnosti

Postupak definiranja svojstva upravljivosti i uočljivosti vremenske relacije prikazan je tablicom 6.16. Svojstvo upravljivosti i uočljivosti vremenske relacije definirano je na osnovi *teorije višeprolaznog iščitavanja* kao jednog od modela psihologije programiranja te na

osnovi *prionljivosti*, *preuranjenog odlučivanja*, *uočljivosti i usporedivosti* te *skrivenih zavisnosti* kao mjera umnog napora tijekom primjene sustava znakovlja.

Funkcijska svojstva primjenskog programa određena su izborom programskih blokova i redoslijedom njihova izvođenja. Isti skup programskih blokova organiziran u dva različita redoslijeda izvođenja predstavlja dva različita radna tijeka, odnosno dva različita primjenska programa. Kako bi se programskom paradigmom omogućilo povezivanje istog skupa programskih blokova u proizvoljne oblike radnih tijekova, definirano je svojstvo upravljivosti i uočljivosti vremenske relacije.

Upravljivost vremenske relacije koristi se tijekom izgradnje programske logike za povezivanje programskih blokova u radni tijek. Svojstvom upravljivosti vremenske relacije, potrošaču je omogućen utjecaj na redoslijed izvođenja programskih blokova. Uočljivost vremenske relacije koristi se tijekom tumačenja značenja izgrađenog primjenskog programa. Svojstvom uočljivosti vremenske relacije, potrošaču je omogućen uvid i ažuriranje redoslijeda izvođenja programskih blokova.



Slika 6.17. Razdvajanje izbora programskih blokova i uspostave vremenskih odnosa tijekom izgradnje radnog tijeka primjenskog programa

Da bi programska paradigma zadovoljila svojstvo upravljivosti i uočljivosti vremenske relacije, izbor programskih blokova razdvojen je od uspostavljanja vremenskih odnosa koji određuju redoslijed njihova izvođenja. Shematski prikaz postupka izgradnje radnog tijeka primjenskog programa primjenom programske paradigme sa svojstvom upravljivosti i uočljivosti vremenske relacije prikazan je slikom 6.17. Izborom programskih blokova definira se skup funkcijskih cjelina koje obavljaju pojedine dijelove programske logike primjenskog programa. Nakon toga se uspostavljaju vremenski odnosi nad izabranim programskim blokovima koji određuju redoslijed izvođenja pojedinih funkcijskih cjelina. Programska paradigma dopušta izgradnju radnog tijeka primjenom iterativnog postupka tijekom kojeg je moguće višestruko prebacivanje usredotočenosti s izbora programskih blokova na uspostavu vremenskih odnosa i obrnuto.

Razdvajanjem izbora programskih blokova od definiranja redoslijeda njihova izvođenja postižu se višestruke povoljne značajke s gledišta kognitivne znanosti. Programska paradigma pokazuje niži stupanj *prionljivosti* jer je promjenu funkcijskih svojstava primjenskog programa moguće postići zamjenom podskupa programskih blokova zamjenskim blokovima uz zadržavanje prvobitnog redoslijeda izvođenja, promjenom redoslijeda izvođenja uz zadržavanje prvobitnog skupa blokova ili istodobnim izmjenama obaju elemenata.

Primjenom programske paradigme sa svojstvom upravljivosti i uočljivosti vremenske relacije smanjuje se potreba za *preuranjenim odlučivanjem* o konačnom obliku radnog tijeka. U prvom koraku moguće je izgraditi osnovni oblik radnog tijeka, usredotočujući se na obradu informacija unutar pojedinih programskih blokova i tok podataka između blokova. Napredne uzorke tijeka izvođenja programa, kao što je istodobno izvođenje pojedinih blokova te razdvajanje i spajanje tijeka izvođenja, moguće je oblikovati nadogradnjom ili promjenom osnovnog oblika radnog tijeka nakon što su način obrade informacija i tok podataka već definirani.

Svojstvo upravljivosti i uočljivosti vremenske relacije doprinosi razumijevanju značenja napisanog programa i usporedbi sličnosti s drugim programima. Prema *teoriji višeprolaznog iščitavanja*, razumijevanje značenja programa provodi se postupno u više koraka. U pojedinim koracima, usredotočenost potrošača zadržava se na izdvojenim elementima programa, uz potiskivanje preostalih elemenata. Uvidom u vremensku relaciju primjenskog programa, potrošaču je omogućena analiza vremenskih uzoraka povezanosti blokova nezavisno od analize funkcijskih svojstava blokova. Uočljivošću vremenske povezanosti blokova postiže se i *usporedivost* različitih primjenskih programa ako su sastavljeni od istih ili sličnih programskih blokova.

Izravnim uvidom i utjecajem potrošača na redoslijed izvođenja programskih blokova, iz zapisa programa uklanjaju se skrivene zavisnosti jer su vremenski odnosi pojedinih blokova definirani i jasno istaknuti od strane potrošača, umjesto da su posljedica funkcijskih svojstava blokova. Nadalje, izborom prikladne tehnike za definiranje vremenskih odnosa postiže se uravnoteženje između količine informacija kojima se opisuju vremenske zavisnosti u programu i čitljivosti programskog zapisa. S obzirom na svojstvo tranzitivnosti vremenske relacije, tijekom izgradnje radnog tijeka dovoljno je definirati samo izravne relacije vremenskog prethođenja, vremenskog slijeđenja i vremenske nezavisnosti među programskim blokovima. Neizravne vremenske relacije nije potrebno posebno isticati jer postaju očigledne na osnovi niza ulančanih izravnih relacija.

Tablica 6.17. Ocjena razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo upravljivosti i uočljivosti vremenske relacije

Naziv alata	Upravljivost i uočljivost vremenske relacije
Microsoft Popfly	+
Yahoo! Pipes	+
Intel Mash Maker	-
Google Mashup Editor	+
iGoogle Gadget Maker	-
IBM QEDwiki	-
OpenKapow	-
Proto	+
Dapper	-
Marmite	+
JackBe Presto	+

Tablicom 6.17 prikazana je ocjena najpoznatijih predstavnika razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo upravljivosti i uočljivosti vremenske relacije. U alatima *Microsoft Popfly*, *Yahoo! Pipes*, *Proto* i *JackBe Presto*, redoslijed izvođenja pojedinih dijelova primjenskog programa određen je usmjerenim granama koje određuju tok podataka između programskih blokova. U alatu *Marmite*, redoslijed izvođenja programskih blokova određen je njihovim prostornim razmještajem. Programski blokovi izvode se redom odozgo prema dolje. Alati *Intel Mash Maker*, *iGoogle Gadget Maker*, *IBM QEDwiki*, *OpenKapow* i *Dapper* ne omogućuju izravno upravljanje vremenskim odnosima jer su ti odnosi određeni izborom programskih blokova ili predložaka za izradu primjenskih programa.

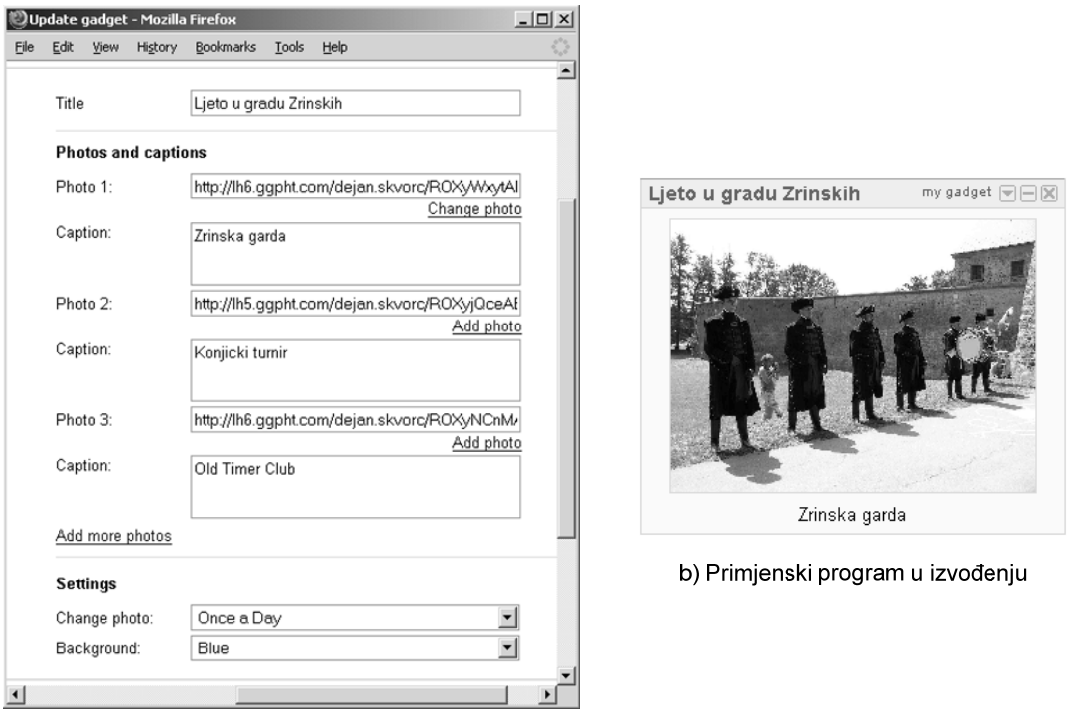
6.2.9 Potpuna izgradivost logike za usložnjavanje blokova od strane potrošača

Svojstvo *potpune izgradivosti logike za usložnjavanje blokova od strane potrošača* mjera je izražajnosti potrošaču prilagođene programske paradigme. Programska paradigma koja zadovoljava ovo svojstvo dopušta programsko povezivanje blokova u bilo koji radni tijek koji je moguće postići izravnim rukovanjem blokovima izvan konteksta programske paradigme. Postupak definiranja svojstva potpune izgradivosti logike za usložnjavanje blokova od strane potrošača prikazan je tablicom 6.18. Svojstvo potpune izgradivosti logike za usložnjavanje blokova od strane potrošača definirano je na osnovi *prionljivosti*, *skrivenih zavisnosti* i *dosljednosti* kao mjera umnog napora tijekom primjene sustava znakovlja.

Tablica 6.18. Definicija svojstva potpune izgradivosti logike za usložnjavanje blokova od strane potrošača

Svojstvo prog. prilagođenog potrošaču	Polazište za definiciju svojstva
Potpuna izgradivost logike za usložnjavanje blokova od strane potrošača	**Mjere umnog napora** ▪ Prionljivost ▪ Skrivene zavisnosti ▪ Dosljednost

U današnje vrijeme na tržištu postoje specijalizirani programski alati koji omogućavaju brzu i jednostavnu izgradnju primjenskih programa unutar ograničenog područja primjene. Takvi programski alati prikladni su za korištenje od strane potrošača jer su uglavnom zasnovani na unaprijed pripremljenim predlošcima za izradu primjenskih programa. Predlošci koje ponuđači nude na tržištu obuhvaćaju ona područja primjene za koje široki krug potrošača pokazuje zanimanje, kao što su predlošci za izradu mrežnih dnevnika, fotografskih albuma, kalendara za uređivanje rasporeda radnih aktivnosti, mrežnog sustava za vođenje osobnih zabilješki i slično. Na slici 6.18 prikazan je primjer razvojne okoline alata *iGoogle Gadget Maker* [189] zasnovane na predlošku za izradu osobnih fotografskih albuma.



a) Izgradnja primjenskog programa

b) Primjenski program u izvođenju

Slika 6.18. Primjer potrošaču prilagođene razvojne okoline zasnovane na predlošcima primjenskih programa

Prednost izgradnje primjenskih programa zasnovane na unaprijed pripremljenim predlošcima je prikladnost za potrošača jer je za izgradnju novog primjenskog programa dovoljno popuniti obrazac s postavkama rada programa. Pozadinska programska potpora koja koristi podatke definirane od strane potrošača unaprijed je definirana i potrošaču nije vidljiva. Kao što prikazuje primjer na slici 6.18.a, za izradu programa za prikaz fotografija, od potrošača se očekuje definiranje odredišta za preuzimanje fotografija koje se prikazuju unutar fotografskog albuma, kratke opise za pojedinu fotografiju te učestalost izmjene fotografija tijekom izvođenja programa. Slikom 6.18.b prikazan je izgrađeni primjenski program za vrijeme izvođenja.

Nedostatak ovakve vrste programskih paradigmi je visoki stupanj *prionljivosti* koji zbog *skrivenih zavisnosti* unutar predložaka ograničava izražajnost razvojnog postupka. Izborom predloška ujedno je određen i radni tijek primjenskog programa. Takav radni tijek nije pod utjecajem potrošača pa ga potrošač ne može proizvoljno definirati niti promijeniti kako bi ga prilagodio vlastitim potrebama. U model programiranja prilagođenog potrošaču je, stoga, uključeno svojstvo potpune izgradivosti logike za usložnjavanje blokova od strane potrošača. Programska paradigma sa svojstvom potpune izgradivosti logike za usložnjavanje blokova od strane potrošača razdvaja izbor programskih blokova od oblikovanja povezujućih elemenata, čime redoslijed izvođenja i podatkovna povezanost blokova ostaje pod utjecajem potrošača.

Osim prionljivosti i skrivenih zavisnosti, treća mjera umnog napora na kojoj se zasniva definicija svojstva potpune izgradivosti logike za usložnjavanje blokova od strane potrošača je *dosljednost*. Svojstvom samodostatnosti blokova, koje je opisano u poglavlju 6.2.3, zahtijeva se oblikovanje programske paradigme u kojoj je blokove za izgradnju programa izvan paradigme moguće istodobno koristiti i kao samostalne primjenske programe. Izvan okvira paradigme, programski se blokovi koriste kao skup međusobno nezavisnih primjenskih programa koji djelovanjem potrošača zajednički obavljaju složenu zadaću koja je kombinacija izvođenja pojedinačnih programa. Djelovanje potrošača tijekom obavljanja složenih zadaća sastoji se od pokretanja izvođenja pojedinačnih programa, određivanje redoslijeda njihova izvođenja i korištenje rezultata izvođenja pojedinih programa u svojstvu ulaznih podataka za druge programe. Kako bi se zadržala dosljednost primjene blokova unutar i izvan programske paradigme, programskim povezivanjem blokova potrebno je omogućiti oblikovanje radnog tijeka za bilo koju funkcionalnost koju je moguće postići uporabom blokova izvan paradigme. Ostavljanjem izbora programskih blokova i načina njihova programskog povezivanja u radni tijek pod utjecajem potrošača, programsko

povezivanje blokova u okviru programske paradigme ostaje dosljedno navikama i iskustvu potrošača stečenim tijekom izravnog rukovanja blokovima izvan paradigme.

Tablicom 6.19 prikazana je ocjena najpoznatijih predstavnika razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo potpune izgradivosti logike za usložnjavanje blokova od strane potrošača. Ovo svojstvo zadovoljavaju alati koji za povezivanje blokova koriste grafičke poveznice u obliku usmjerenih linija jer je skup programskih blokova moguće povezivati po volji potrošača. Tu skupinu čine alati *Microsoft Popfly*, *Yahoo! Pipes*, *Proto* i *JackBe Presto*. Nadalje, svojstvo potpune izgradivosti logike za usložnjavanje blokova od strane potrošača zadovoljava alat *Google Mashup Editor* zbog mogućnosti izgradnje programske logike primjenom jezika XML, HTML i Javascript.

Tablica 6.19. Ocjena razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo potpune izgradivosti logike za usložnjavanje blokova od strane potrošača

Naziv alata	Potpuna izgradivost povezujuće logike
Microsoft Popfly	+
Yahoo! Pipes	+
Intel Mash Maker	-
Google Mashup Editor	+
iGoogle Gadget Maker	-
IBM QEDwiki	-
OpenKapow	-
Proto	+
Dapper	-
Marmite	-
JackBe Presto	+

Svojstvo potpune izgradivosti logike za usložnjavanje blokova od strane potrošača ne zadovoljava alat *Dapper* jer je taj alat namijenjen objedinjavanju *World Wide Web* stranica sa sličnom strukturom i sličnim funkcionalnostima ili različitih pojavnih oblika iste *World Wide Web* stranice. Nadalje, alati *Intel Mash Maker*, *IBM QEDwiki* i *Marmite* ne zadovoljavaju ovo svojstvo zbog automatskog povezivanja programskih blokova na osnovi podudarnosti značenja i strukture ulaznih i izlaznih podataka, što nije pod utjecajem potrošača. Konačno, ovo svojstvo ne zadovoljavaju ni alati *iGoogle Gadget Maker* i *OpenKapow* čija je izražajnost ograničena unaprijed definiranim skupom predložaka primjenskih programa.

6.2.10 Opažajni doživljaj prikaza programa

Svojstvom *opažajnog doživljaja prikaza programa* zahtijeva se prikaz programske logike primjenom znakovlja i nazivlja koji su dio općeg znanja čovjeka ili se koriste tijekom uobičajene primjene računalnih primjenskih programa. Razumijevanje značenja programa izgrađenog primjenom takve programske paradigme ne zahtijeva ulaganje dodatnog napora i vremena za prethodno usvajanje značenja elemenata za prikaz programa.

Postupak definiranja svojstva opažajnog doživljaja prikaza programa prikazan je tablicom 6.20. Svojstvo opažajnog doživljaja prikaza programa definirano je na osnovi *teorije održavanja i preispitivanja znanja* kao jednog od modela psihologije programiranja te na osnovi *samoopisivosti* kao jedne od mjera umnog napora tijekom primjene sustava znakovlja.

Tablica 6.20. Definicija svojstva opažajnog doživljaja prikaza programa

Svojstvo prog. prilagođenog potrošaču	Polazište za definiciju svojstva
Opažajni doživljaj prikaza programa	**Psihologija programiranja** ▪ Teorija održavanja i preispitivanja znanja **Mjere umnog napora** ▪ Samoopisivost

Prema *teoriji održavanja i preispitivanja znanja*, početni napor koji programer ulaže u usvajanje novog programskog jezika moguće je smanjiti izborom *samoopisivog* sustava znakovlja i nazivlja. Primjeri čovjeku razumljivih oblika znakovlja koji su pogodni za prikaz na računalu su prirodni jezik izrečen slovčanim ili glasovnim zapisom, slike, skice, animacije i crteži te grafičko korisničko sučelje u obliku tipki, označnih polja, polja za upis teksta, poveznica i izbornika. Međutim, osim znakovlja, znanju i iskustvu potrošača potrebno je prilagoditi i nazivlje za opis programskih elemenata i njihovih međusobnih veza kako bi prikazana informacija bila razumljiva ciljnoj skupini korisnika. U prikazu programa za široki krug potrošača, potrebno je izbjegavati stručno znakovlje i nazivlje te znakovlje i nazivlje posebno oblikovano za potrebe programske paradigme.

Tablicom 6.21 prikazana je ocjena najpoznatijih predstavnika razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo opažajnog doživljaja prikaza programa. Ovo svojstvo zadovoljavaju alati koji za izgradnju primjenskog programa koriste unaprijed definirane predloške. Prikaz programa u takvim alatima zasniva se na prikazu predložaka ispunjenih vrijednostima definiranim od strane potrošača. Tu skupinu čine alati *iGoogle Gadget Maker* i *OpenKapow*. Svojstvo opažajnog doživljaja prikaza

programa zadovoljavaju i alati *Intel Mash Maker* i *IBM QEDwiki*. Prikaz programa u tim alatima sastoji se od skupa udomljenika kojima je obogaćena *World Wide Web* stranica na kojoj se izvodi primjenski program. Nadalje, svojstvo opažajnog doživljaja prikaza programa zadovoljava alat *Marmite* u kojem se program sastoji od skupa udomljenika s postavkama rada programa poredanih po redoslijedu izvođenja.

Tablica 6.21. Ocjena razvojnih alata za izgradnju usloženih primjenskih programa s obzirom na svojstvo opažajnog doživljaja prikaza programa

Naziv alata	Opažajni doživljaj prikaza programa
Microsoft Popfly	-
Yahoo! Pipes	-
Intel Mash Maker	+
Google Mashup Editor	-
iGoogle Gadget Maker	+
IBM QEDwiki	+
OpenKapow	+
Proto	-
Dapper	-
Marmite	+
JackBe Presto	-

Alati koji za prikaz programa koriste programske jezike zasnovane na slovčanom zapisu, kao što je *Google Mashup Editor*, i alati u kojima je program prikazan apstraktnim grafičkim oznakama, kao što su *Microsoft Popfly*, *Yahoo! Pipes*, *Proto* i *JackBe Presto*, ne zadovoljavaju svojstvo opažajnog doživljaja prikaza programa. U tim je alatima za razumijevanje značenja programa potrebno poznavati značenje naredbi programskog jezika ili posebno oblikovanih grafičkih oznaka. Svojstvo u potpunosti ne zadovoljava ni alat *Dapper* u kojem je program prikazan u obliku skupa objedinjenih *World Wide Web* stranica jer se za parametrizaciju njihova rada koriste programske varijable.

6.2.11 Zbirna ocjena analiziranih programskih alata

Tablicom 6.22 prikazana je zbirna ocjena programskih alata koji su u okviru doktorske disertacije analizirani s obzirom na prikladnost za primjenu od strane širokog kruga potrošača računala. Zbirna ocjena prikazana je u krajnje desnom stupcu tablice. Ocjena je prikazana u obliku broja svojstava programiranja prilagođenog potrošaču koje zadovoljava pojedini analizirani alat. Alati su poredani po padajućim vrijednostima zbirne ocjene.

Tablica 6.22. Zbirna ocjena programskih alata za izgradnju usloženih primjenskih programa

Naziv alata	Blokovska izgradivost primjenskog programa	Opažajni doživljaj značenja blokova	Samodostatnost blokova	Višerazinsko usložnjavanje blokova	Jednakost korištenja i povezivanja blokova	Nezavisnost povezujućih elemenata	Izgradivost prog. putem graf. kor. sučelja	Upravljivost i uočljivost vremenske relacije	Potpuna izgradivost povezujuće logike	Opažajni doživljaj prikaza programa	**ZBIRNA OCJENA**
Dapper	+	+	+	+	+	+	+	-	-	-	**7**
Proto	+	-	-	+	-	+	+	+	+	-	**6**
Intel Mash Maker	+	+	+	-	+/-	-	+	-	-	+	**5,5**
IBM QEDwiki	+	+	+	-	+/-	-	+	-	-	+	**5,5**
Microsoft Popfly	+	-	-	-	-	+	+	+	+	-	**5**
Yahoo! Pipes	+	-	-	-	-	+	+	+	+	-	**5**
JackBe Presto	+	-	-	-	-	+	+	+	+	-	**5**
Marmite	+	+	-	-	-	-	+	+	-	+	**5**
Google Mashup Editor	+	-	-	-	-	+	-	+	+	-	**4**
iGoogle Gadget Maker	-	-	-	-	-	-	+	-	-	+	**2**
OpenKapow	-	-	-	-	-	-	+	-	-	+	**2**

Iz tablice je vidljivo da nijedan od analiziranih alata u potpunosti ne zadovoljava svojstva modela programiranja prilagođenog potrošaču. S gledišta prikladnosti za primjenu od strane širokog kruga potrošača, najnepovoljnije su ocijenjeni alati *iGoogleGadgetMaker* i *OpenKapow*. Zbog zasnovanosti na unaprijed definiranom skupu predložaka primjenskih programa, ti alati potrošaču ne dozvoljavaju oblikovanje proizvoljnog radnog tijeka. S druge strane, u alatima *Dapper* i *Proto*, koji zadovoljavaju najveći broj svojstava programiranja prilagođenog potrošaču, izostaju ključna svojstva potrošaču prilagođene programske paradigme, kao što je mogućnost povezivanja elemenata u proizvoljne oblike radnih tijekova, odnosno opažajni doživljaj gradivnih elemenata, povezujućih elemenata i prikaza programa. U okviru doktorske disertacije je, stoga, predložena nova programska paradigma koja model programiranja prilagođenog potrošaču zadovoljava u potpunosti.

7 Programska paradigma za usložnjavanje udomljenika

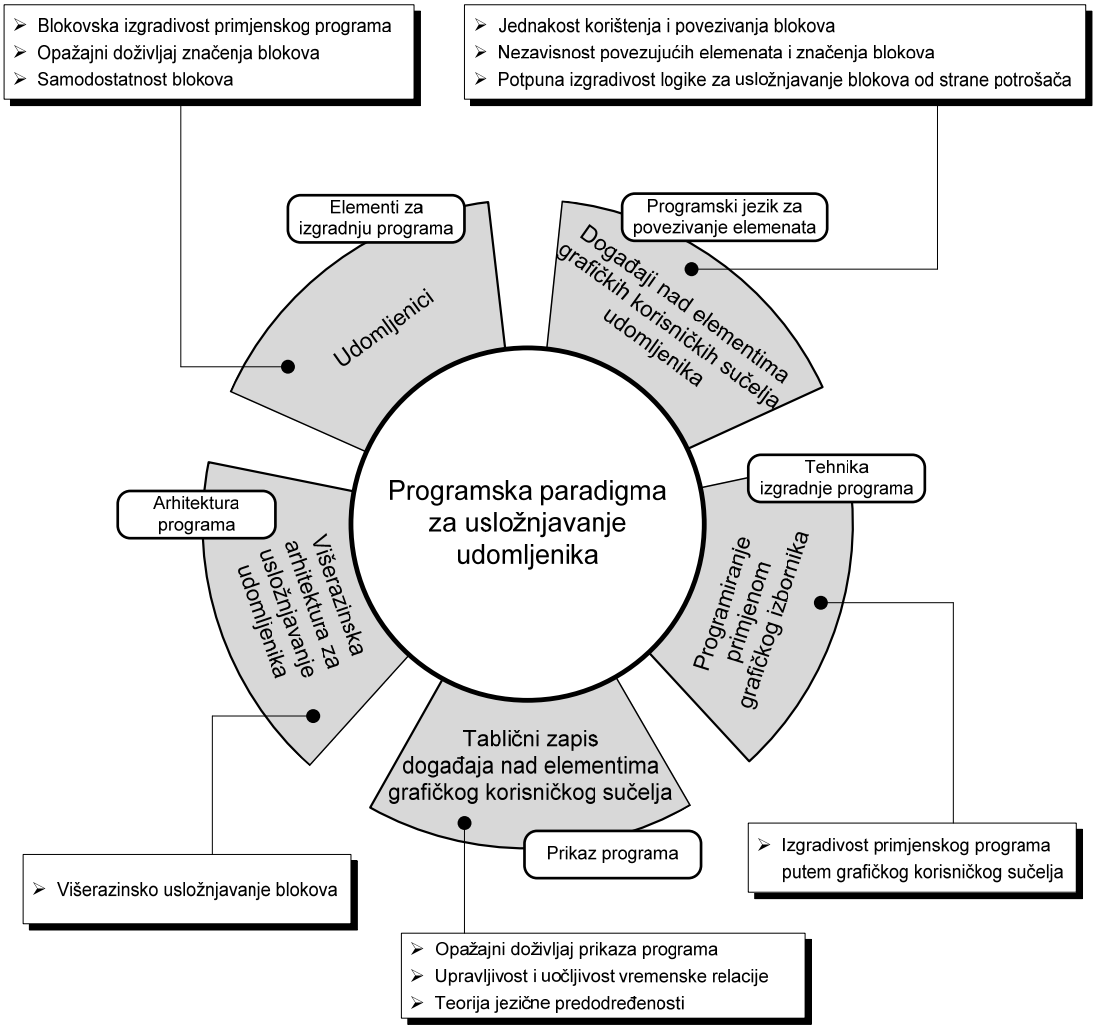
Modelom za ostvarenje programiranja prilagođenog potrošaču koji je opisan u poglavlju 6 definiran je skup svojstava koja programsku paradigmu čine bliskom umnom režimu potrošača. Definirani model moguće je primijeniti za vrednovanje prikladnosti postojećih postupaka i razvojnih alata širokom krugu potrošača, kao što je pokazano u poglavlju 6, ili za oblikovanje nove, potrošaču prilagođene programske paradigme.

U ovom i nekoliko sljedećih poglavlja opisuje se programska paradigma za ostvarenje programiranja prilagođenog potrošaču koja je oblikovana u okviru doktorske disertacije [210]. Pojedini elementi programske paradigme oblikovani su u skladu sa svojstvima definiranim u poglavlju 6. Na osnovi uspostavljenog skupa svojstava, izabrani su elementi za izgradnju primjenskih programa, definiran je jezik za povezivanje elemenata u radni tijek (engl. *workflow*), izabrana je arhitektura za oblikovanje, izgradnju i izvođenje programa, oblikovana je tehnika za izradu programa primjenom izabrane arhitekture i definiranog jezika te je definiran način prikaza programa [210].

Ključno svojstvo predložene programske paradigme je oblikovanje i izgradnja primjenskih programa na predodžbenoj razini, odnosno na razini grafičkih korisničkih sučelja (engl. *graphical user interface – GUI*) primjenskih programa za *World Wide Web*. Za razliku od uobičajenih postupaka programiranja na razini programskih sučelja (engl. *application programming interface – API*) gdje programer rukuje elementima u obliku objekata, funkcija i varijabli, u predloženoj programskoj paradigmi potrošač rukuje elementima u obliku tipki, slika, poveznica, polja za unos teksta i izbornika. Cilj predložene

paradigme je pojednostavljivanje razvoja primjenskih programa do mjere da su za njihovu izgradnju dovoljna znanja i vještine o uzajamnom djelovanju potrošača i računala putem grafičkog korisničkog sučelja prikazanog u pregledniku *World Wide Web* sadržaja.

Predloženo ostvarenje programiranja prilagođenog potrošaču naziva se programskom paradigmom za usložnjavanje udomljenika (engl. *widget composition*) [210]. Elementi paradigme prikazani su slikom 7.1. Za svaki od elemenata istaknuta je uloga u postupku izgradnje primjenskog programa te način njegova oblikovanja na osnovi modela programiranja prilagođenog potrošaču koji je opisan u poglavlju 6.



Slika 7.1. Elementi programske paradigme za usložnjavanje udomljenika

Naziv paradigme dolazi od naziva elemenata koji se koriste za izgradnju primjenskih programa. Novi primjenski programi ne grade se iznova, nego nastaju iskorištavanjem i povezivanjem već izgrađenih primjenskih programa u obliku udomljenika (engl. *widgets*, *gadgets*, *web slices*) [211, 212, 213, 214] kojima se rukuje putem grafičkog korisničkog

sučelja. Odabir programskih elemenata u obliku udomljenika izvršen je na osnovi svojstva *blokovske izgradivosti primjenskog programa*, svojstva *opažajnog doživljaja značenja blokova* te svojstva *samodostatnosti blokova*. Definicija udomljenika, obrazloženje izbora u svojstvu gradivnih elemenata i način primjene tijekom izgradnje primjenskih programa opisani su u poglavlju 7.1.

Povezivanje skupa udomljenika u primjenski program obavlja se posebno oblikovanim programskim jezikom za definiranje radnog tijeka (engl. *workflow*) nad skupom udomljenika. Elemente jezika čine radnje potrošača nad elementima grafičkog korisničkog sučelja. Tim radnjama uspostavlja se tok podataka između udomljenika i pokreće izvođenje pozadinskih procesa. Primjeri elemenata jezika za izgradnju radnog tijeka nad skupom udomljenika su upisivanje sadržaja u polje za unos teksta, izbor stavke iz padajućeg izbornika, označavanje i odznačavanje označnog polja, pritisak na tipku ili poveznicu te preuzimanje i preslikavanje sadržaja (engl. *copy/paste*). Odabir programskog jezika u obliku radnji potrošača nad elementima grafičkog korisničkog sučelja udomljenika izvršen je na osnovi svojstva *jednakosti korištenja i povezivanja blokova*, svojstva *nezavisnosti povezujućih elemenata i značenja blokova* te svojstva *potpune izgradivosti logike za usložnjavanje blokova od strane potrošača*. Obrazloženje izbora programskog jezika i način primjene tijekom izgradnje primjenskog programa opisani su u poglavlju 7.2.

Primjenski program koji nastaje povezivanjem skupa udomljenika u poosobljeni radni tijek izložen je u obliku novog udomljenika. Dobiveni udomljenik sadrži zapis definiranog radnog tijeka i korisničko sučelje za njegovo pokretanje. Moguće ga je koristiti za krajnju primjenu ili za daljnju izgradnju novih primjenskih programa u obliku novih udomljenika. Oblikovanje i izgradnja primjenskih programa zasniva se na višerazinskoj arhitekturi koja omogućava postupno usložnjavanje udomljenika kroz više koraka i udruženo sudjelovanje zajednice potrošača u razvoju primjenskih programa. Odabir višerazinske arhitekture za usložnjavanje udomljenika izvršen je na osnovi svojstva *višerazinskog usložnjavanja blokova*. Obrazloženje izbora višerazinske arhitekture i način primjene tijekom izgradnje primjenskog programa opisani su u poglavlju 7.3.

Programiranje radnog tijeka nad skupom udomljenika i izlaganje objedinjene funkcionalnosti u obliku novog udomljenika izvodi se primjenom grafičkog izbornika. Primjenom grafičkog izbornika, potrošač obilazi skup udomljenika i zadaje slijed radnji koje je potrebno obaviti nad pripadajućim elementima korisničkog sučelja. Odabir tehnike izgradnje programa primjenom grafičkog izbornika izvršen je na osnovi svojstva *izgradivosti primjenskog programa putem grafičkog korisničkog sučelja*. Obrazloženje izbora tehnike

programiranja zasnovane na grafičkom izborniku i način primjene tijekom izgradnje primjenskog programa opisani su u poglavlju 7.4.

Slijed radnji nad elementima grafičkog korisničkog sučelja kojima je definiran radni tijek nad skupom udomljenika zapisuje se i prikazuje u obliku dvodimenzionalne tablice. Organizacija programa u dvodimenzionalnom tabličnom prostoru pojednostavljuje oblikovanje složenih tijekova izvođenja programa, kao što je slijedno i istodobno izvođenje. Odabir tabličnog prikaza programa izvršen je na osnovi svojstva *opažajnog doživljaja prikaza programa*, svojstva *upravljivosti i uočljivosti vremenske relacije* te na osnovi *teorije jezične predodređenosti*. Obrazloženje izbora i način primjene tabličnog prikaza primjenskog programa opisani su u poglavlju 7.5.

7.1 Elementi za izgradnju primjenskih programa

Prvi korak tijekom oblikovanja programske paradigme prilagođene potrošaču je izbor elemenata za izgradnju primjenskih programa. Elementi za izgradnju programa obavljaju pojedine dijelove primjenske logike, a njihovim povezivanjem u radni tijek postiže se cjelokupna funkcionalnost primjenskog programa prema zamislima potrošača.

7.1.1 Oblikovanje elemenata za izgradnju programa

Modelom za ostvarenje programiranja prilagođenog potrošaču koji je opisan u poglavlju 6 definirana su tri svojstva koja se odnose na izbor elemenata za izgradnju programa. Izbor gradivnih elemenata određen je svojstvom *blokovske izgradivosti primjenskog programa*, svojstvom *opažajnog doživljaja značenja blokova* i svojstvom *samodostatnosti blokova*.

Svojstvom *blokovske izgradivosti primjenskog programa* određeno je da se primjenski program gradi povezivanjem gotovih blokova koji obavljaju pojedine dijelove cjelokupne funkcionalnosti programa. Na taj se način pojednostavljuje postupak razvoja programa jer je glavnina programske složenosti skrivena od potrošača unutar blokova. Od potrošača se očekuje odabir odgovarajućih blokova i njihovo povezivanje u radni tijek prema zakonitostima primjenskog programa.

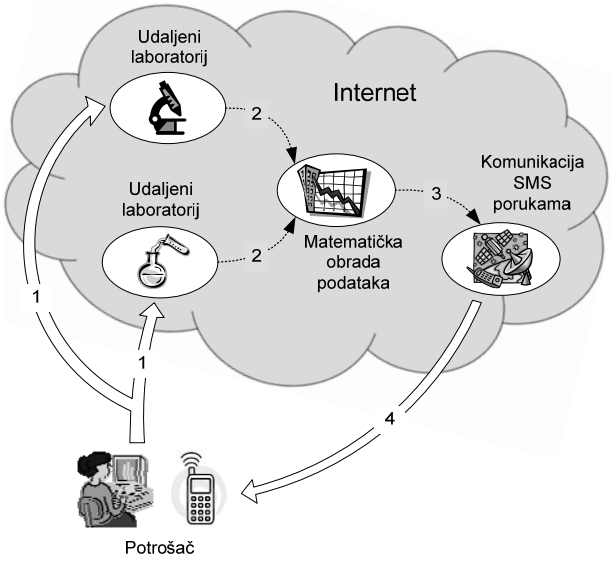
S obzirom na to da se blokovi za ostvarenje programske paradigme pojavljuju u različitim oblicima, odabir vrste blokova izvršen je na osnovi svojstva *opažajnog doživljaja značenja blokova* i svojstva *samodostatnosti blokova*. Svojstvom *opažajnog doživljaja značenja blokova* zahtijeva se izbor samoopisivih programskih blokova koji ne iziskuju

dodatni napor za razumijevanje uloge i načina rukovanja blokom tijekom izgradnje primjenskog programa. Analizom odnosa potrošača prema tehnologiji, pokazano je da je grafičko korisničko sučelje u obliku tipki, slika, poveznica, polja za unos teksta i izbornika prevladavajući oblik predodžbe programskih funkcionalnosti kojim većina potrošača uspijeva ovladati bez posebnog napora i utroška vremena. Izbor programskih blokova za ostvarenje programske paradigme time je sužen na programske blokove kojima se rukuje putem grafičkog korisničkog sučelja.

Svojstvom *samodostatnosti blokova* zahtijeva se izbor programskih blokova koje su potrošači navikli koristiti izvan konteksta programske paradigme. Izborom samodostatnih blokova, razvoj novih primjenskih programa postaje prirodna i neprimjetna nadogradnja uobičajenih navika potrošača. Analizom odnosa potrošača prema tehnologiji, pokazano je da potrošači tijekom svakodnevne primjene računala uspješno primjenjuju programska okružja zasnovana na udomljenicima. U okviru predložene programske paradigme, udomljenici su, stoga, izabrani kao potrošaču prilagođeni programski blokovi za izgradnju primjenskih programa.

7.1.2 Udomljenici

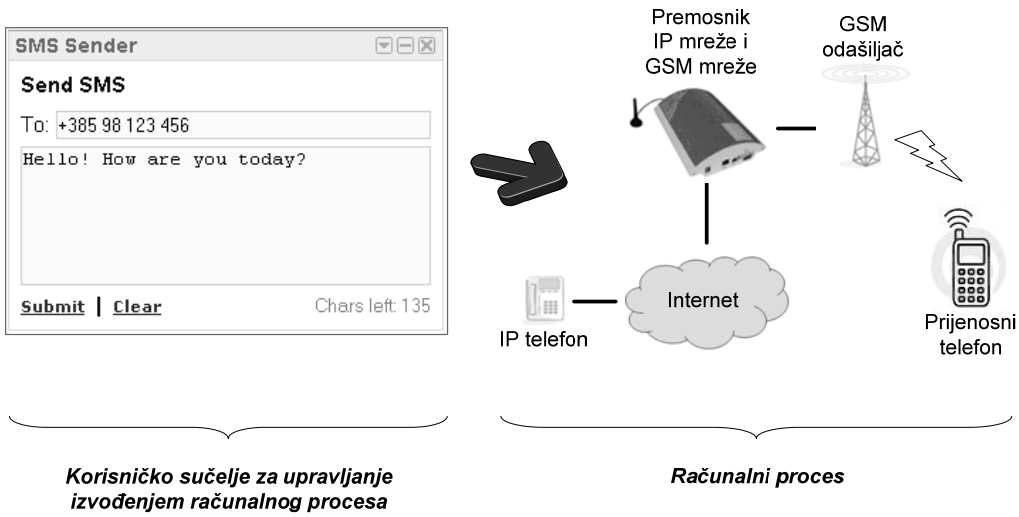
Primjenom programske paradigme za usložnjavanje udomljenika, primjenski programi grade se iskorištavanjem potrošaču usmjerenih programskih funkcionalnosti krupne zrnatosti i njihovim povezivanjem u radni tijek prema zamislima i potrebama potrošača. Programske funkcionalnosti od kojih se grade primjenski programi su računalni procesi na lokalnom računalu ili u mreži Internet.



Slika 7.2. Oblikovanje primjenskog programa povezivanjem skupa nezavisnih računalnih procesa

Slikom 7.2 prikazan je primjer oblikovanja primjenskog programa prema poosobljenim potrebama potrošača. U prikazanom primjeru, zamisao potrošača je oblikovanje programa za udaljeno izvođenje pokusa te obradu i dojavu rezultata. Primjenski program moguće je oblikovati povezivanjem četiriju računalnih procesa koji su dostupni u mreži Internet. Procese za pristup mjernim uređajima u udaljenim laboratorijima, proces za matematičku obradu podataka te proces za komunikaciju SMS porukama moguće je povezati u novi primjenski program prema prikazanoj skici. Izvođenje primjenskog programa pokreće se definiranjem postavki i ulaznih podataka za izvođenje pokusa u udaljenim laboratorijima (1). Rezultati dobiveni pokusnim ispitivanjima prosljeđuju se procesu za matematičku obradu podataka (2), gdje se nad njima obavlja skup matematičkih proračuna. Obrađeni rezultati pokusnih ispitivanja prosljeđuju se procesu za komunikaciju SMS porukama (3) kako bi se dojavili potrošaču putem GSM mreže (4).

Kako bi se potrošaču omogućila maksimalna usredotočenost na funkcijska svojstva računalnih procesa i mogućnosti njihova povezivanja u poosobljeni radni tijek, u programskoj paradigmi za usložnjavanje udomljenika računalni su procesi predstavljeni grafičkim korisničkim sučeljima. Na taj način računalni procesi postaju prikladni za primjenu od strane potrošača jer se prikrivaju pojedinosti organizacijskih raznolikosti i tehnologijske složenosti pojedinih procesa, mjesta gdje se proces izvodi, tehnologije kojom je ostvaren i značajki računalne okoline u kojoj se izvodi.



Slika 7.3. Primjena grafičkog korisničkog sučelja za upravljanje izvođenjem računalnih procesa

Slikom 7.3 prikazano je načelo izlaganja funkcionalnosti računalnih procesa putem grafičkog korisničkog sučelja. Svakom procesu pridruženo je grafičko korisničko sučelje putem kojeg je moguće pratiti stanje i upravljati izvođenjem procesa. Izlaganjem računalnih

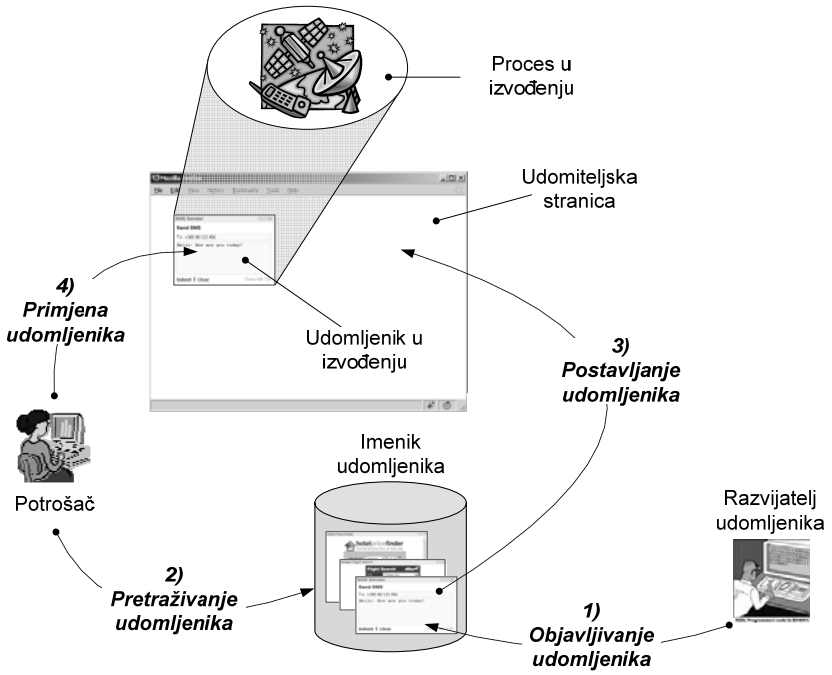
procesa putem grafičkih korisničkih sučelja, potrošaču su izložena samo ona svojstva procesa koje njega kao potrošača zanimaju, dok pojedinosti složene sklopovske i programske arhitekture, programskog ostvarenja i izvođenja procesa ostaju skrivene.

Da bi grafička korisnička sučelja postala primjenjiva za izgradnju primjenskih programa, potrebno je omogućiti jednostavne mehanizme njihova pronalaženja i rukovanja tijekom razvojnog postupka. U programskoj paradigmi za usložnjavanje udomljenika, grafička korisnička sučelja za predodžbu računalnih procesa pojavljuju se u obliku udomljenika (engl. *widgets*, *gadgets*, *web slices*) [211, 212, 213, 214]. Udomljenik je programska komponenta s grafičkim korisničkim sučeljem u obliku *World Wide Web* stranice kojim se upravlja izvođenjem pridruženog pozadinskog procesa. Svojstvo udomljenika je prenosivost te mogućnost udomljavanja i izvođenja unutar druge *World Wide Web* stranice koja se naziva udomiteljskom stranicom (engl. *container page*).

Unutar jedne udomiteljske stranice moguće je udomiti proizvoljni broj udomljenika, čime je omogućeno upravljanje izvođenjem skupa nezavisnih računalnih procesa s jednog mjesta. Svojstva prenosivosti i mogućnosti udomljavanja unutar udomiteljske stranice čine udomljenike pogodnim elementima za primjenu u svojstvu programskih blokova za izgradnju primjenskih programa, dok udomiteljska stranica postaje potrošaču prilagođena razvojna okolina.

Slika 7.4. Primjer udomiteljske stranice sa skupom udomljenika

Slika 7.4 prikazuje primjer udomiteljske stranice koja udomljuje nekoliko nezavisnih udomljenika. Svaki od udomljenika predstavlja korisničko sučelje prema nezavisnom računalnom procesu. Korisnik udomiteljske stranice poosobljava sadržaj stranice postavljanjem proizvoljnog broja i vrste udomljenika na stranicu. Primjer na slici 7.4 prikazuje poosobljenu udomiteljsku stranicu na koju su redom postavljeni udomljenik za praćenje granice dana i noći, udomljenik za prikaz elektroničkog zemljovida, udomljenik s prikazom dnevnog horoskopa, udomljenik za usporedno praćenje vremenskih uvjeta u odabranim gradovima, udomljenik za pregledavanje osobnih fotografskih albuma, udomljenik za pristup pretincu elektroničke pošte te udomljenik s podsjetnikom na dnevne obveze.



Slika 7.5. Arhitektura sustava za upravljanje izvođenjem računalnih procesa zasnovana na udomljenicima

Slikom 7.5 prikazana je arhitektura sustava za upravljanje izvođenjem računalnih procesa zasnovana na udomljenicima. Osnovni elementi prikazane arhitekture su razvijatelji i potrošači udomljenika, skup udomljenika, imenik udomljenika i udomiteljska stranica. Upravljanje sustavom zasniva se na objavljivanju, pretraživanju, postavljanju i primjeni udomljenika. Razvijatelji udomljenika razvijaju udomljenike i objavljuju njihovu dostupnost u imeniku udomljenika (1). Imenik udomljenika je dio arhitekture koji potrošačima omogućuje pretraživanje i pronalaženje odgovarajućih udomljenika. Potrošači udomljenika pretražuju imenik s ciljem pronalaska udomljenika koji zadovoljavaju njihove osobne potrebe (2). Pronalaskom odgovarajućeg udomljenika u imeniku, potrošač izdaje naredbu za

postavljanje udomljenika na udomiteljsku stranicu (3). Udomiteljska stranica je dio arhitekture koja na jednom mjestu omogućuje postavljanje i izvođenje proizvoljnog broja udomljenika. Postavljanjem udomljenika na udomiteljsku stranicu, on postaje udomljenik u izvođenju i spreman je za uporabu. Udomljenici u izvođenju koriste se od strane potrošača za upravljanje izvođenjem pridruženih računalnih procesa putem pripadajućeg grafičkog korisničkog sučelja (4).

Izlaganje računalnih procesa u obliku udomljenika prve su primijenile tvrtke za proizvodnju programskih sustava velikih razmjera *Google*, *Yahoo* i *Microsoft* putem inicijative komponentnog *World Wide Web* sustava (engl. *component web*) [22]. Osnovna zamisao komponentnog *World Wide Web* sustava jest olakšano i prilagodljivo upravljanje sadržajima dostupnima putem *World Wide Web* sustava. Putem inicijative komponentnog *World Wide Web* sustava, razvijateljima primjenskih programa ponuđena je platforma za izlaganje računalnih procesa u mreži Internet putem udomljenika te razvoj i izvođenje udomljenika. Pružateljima usluga omogućeno je pružanje usluga udomiteljskih stranica. Potrošačima je ponuđena mogućnost oblikovanja poosobljenih *World Wide Web* stranica na način da udomiteljsku stranicu popune skupom odabranih udomljenika.

7.2 Programski jezik za povezivanje elemenata

Izbor udomljenika u svojstvu programskih elemenata i udomiteljske stranice u svojstvu razvojne okoline prvi je korak prema ostvarenju programske paradigme za izgradnju primjenskih programa od strane potrošača. Drugi korak je izbor programskog jezika za definiranje radnog tijeka nad skupom udomljenika.

7.2.1 Oblikovanje programskog jezika za povezivanje elemenata

Modelom za ostvarenje programiranja prilagođenog potrošaču koji je opisan u poglavlju 6 definirana su tri svojstva koja se odnose na izbor programskog jezika. Izbor programskog jezika određen je svojstvom *jednakosti korištenja i povezivanja blokova*, svojstvom *nezavisnosti povezujućih elemenata i značenja blokova* i svojstvom *potpune izgradivosti logike za usložnjavanje blokova od strane potrošača*.

Svojstvom *jednakosti korištenja i povezivanja blokova* zahtijeva se da se programski jezik za povezivanje blokova u radni tijek odabere na način da je postupak programskog povezivanja blokova izjednačen s postupkom izravnog rukovanja blokovima koji su potrošači usvojili izvan konteksta programske paradigme. Izborom programskih blokova u obliku udomljenika na taj je način određen i izbor programskog jezika. S obzirom na to da se

udomljenicima rukuje izvođenjem radnji nad elementima pripadajućih grafičkih korisničkih sučelja, radnje nad elementima grafičkog korisničkog sučelja izabrane su kao programski jezik za programsko povezivanje udomljenika u radni tijek.

Izborom programskog jezika u obliku radnji nad elementima grafičkog korisničkog sučelja zadovoljavaju se i preostala dva svojstva modela za ostvarenje programiranja prilagođenog potrošaču. Svojstvo *nezavisnosti povezujućih elemenata i značenja blokova* proizlazi iz konačnog broja različitih vrsta elemenata od kojih se grade grafička korisnička sučelja i konačnog broja različitih radnji koje je moguće izvesti nad pojedinom vrstom elementa. Primjerice, nad elementom korisničkog sučelja u obliku tipke moguće je izvesti radnju jednostrukog i dvostrukog pritiska, dok je nad elementom u obliku izbornika moguće izvesti radnju izbora jedne od ponuđenih stavki. Bez obzira na primjensku funkcionalnost koja se u promatranom udomljeniku postiže izvođenjem određene radnje nad izabranim elementom korisničkog sučelja, mehanizam izvođenja radnje ovisi isključivo o vrsti radnje i vrsti elementa i neovisan je o ulozi elementa u promatranom udomljeniku. Nezavisnošću radnji nad elementima korisničkog sučelja od primjenske funkcije udomljenika, omogućena je primjena konačnog skupa radnji za oblikovanje proizvoljnih primjenskih programa.

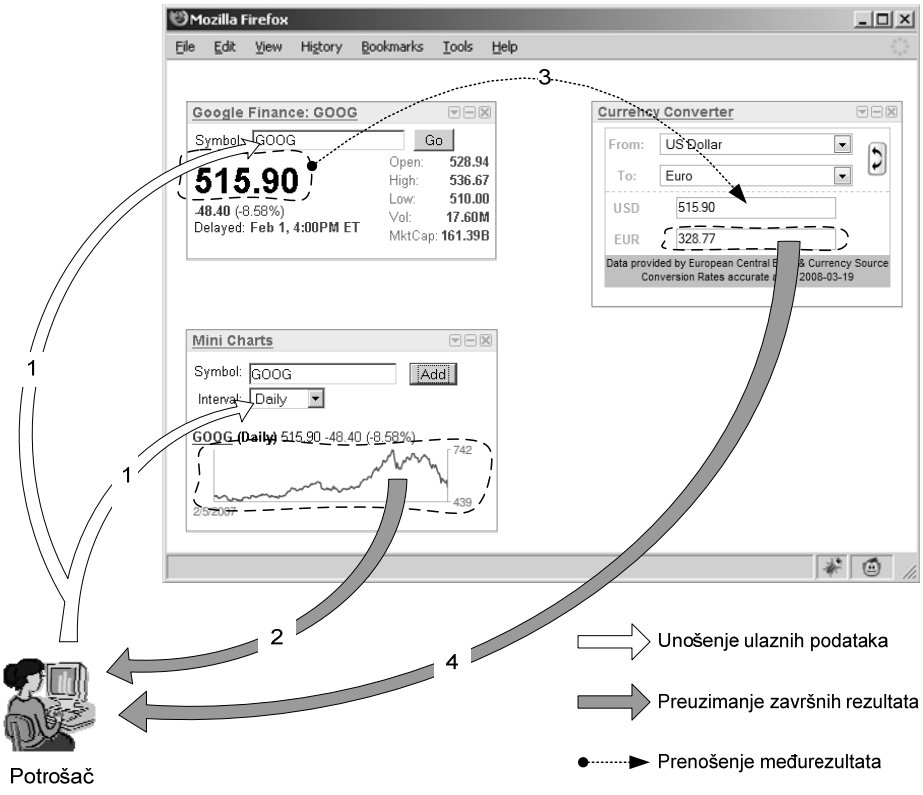
Povezivanje blokova na razini događaja nad grafičkim korisničkim sučeljem udomljenika opravdano je rezultatima analize odnosa potrošača prema tehnologiji. Tom je analizom pokazano da je izvođenje radnji nad elementima grafičkog korisničkog sučelja u obliku tipki, slika, poveznica, polja za unos teksta i izbornika prevladavajući oblik međudjelovanja čovjeka i računala kojim većina potrošača uspijeva ovladati bez posebnog napora i utroška vremena.

Osim jednostavnosti i prikladnosti za primjenu od strane širokog kruga potrošača, grafičko korisničko sučelje se kroz dugogodišnju primjenu pokazalo dovoljno izražajnim oblikom međudjelovanja čovjeka i računala kojim je moguće odgovoriti na gotovo sve zahtjeve potrošača. Budući da je oblikovanjem programa koji oponaša radnje potrošača nad elementima korisničkog sučelja moguće ostvariti bilo koju funkcionalnost koju potrošači postižu izravnim rukovanjem skupom udomljenika, programski jezik zadovoljava svojstvo *potpune izgradivosti logike za usložnjavanje blokova od strane potrošača*.

7.2.2 Radnje nad elementima grafičkog korisničkog sučelja

Radni tijek nad skupom udomljenika oblikuje se radnjama nad elementima grafičkog korisničkog sučelja. Izvođenjem tih radnji, pokreće se izvođenje pojedinačnih udomljenika te se uspostavlja tok podataka između različitih udomljenika.

Slikom 7.6 prikazan je postupak oblikovanja radnog tijeka na razini grafičkih korisničkih sučelja udomljenika. U prikazanom primjeru, potrošač oblikuje složeni proces koji nastaje povezivanjem triju osnovnih procesa. Svaki proces izložen je u obliku udomljenika koji se izvodi unutar udomiteljske stranice. U prvom koraku, potrošač unosi ulazne podatke za izvođenje složenog procesa putem korisničkog sučelja dvaju udomljenika čiji su nazivi *Google Finance* i *Mini Charts* (1). Pritiskom na tipke *Go* i *Add*, pokreće se izvođenje pozadinskih procesa, a na grafičkom korisničkom sučelju udomljenika pojavljuju se izlazni rezultati. Rezultat izvođenja udomljenika *Mini Charts* koristi se kao krajnji rezultat izvođenja složenog procesa (2). S druge strane, rezultat izvođenja udomljenika *Google Finance* predstavlja međurezultat koji se prenosi kao ulazni podatak u udomljenik *Currency Converter* (3). Pokretanjem izvođenja udomljenika *Currency Converter*, na korisničkom sučelju pojavljuje se izlazni rezultat koji se koristi kao drugi krajnji rezultat izvođenja složenog procesa (4).



Slika 7.6. Oblikovanje radnog tijeka na razini grafičkih korisničkih sučelja udomljenika

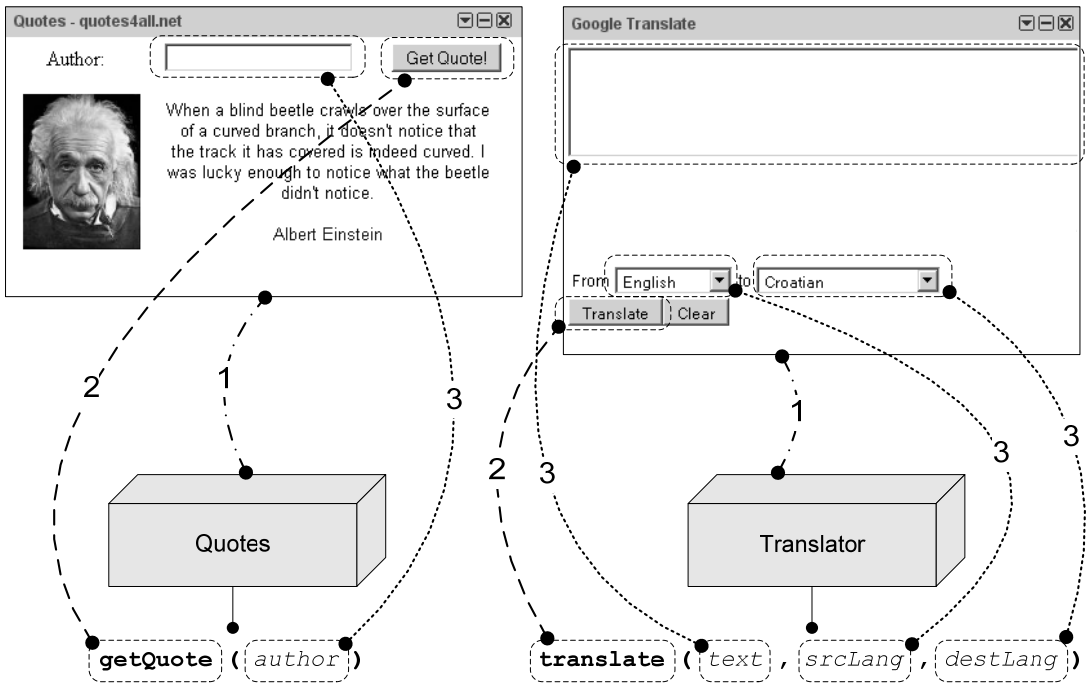
Dok je unošenje ulaznih podataka i tumačenje krajnjih rezultata izvođenja složenog procesa potrebno prepustiti potrošaču, usklađeno pokretanje izvođenja te prenošenje međurezultata između udomljenika moguće je automatizirati izgradnjom odgovarajuće programske logike. Budući da je programsko povezivanje nezavisnih usluga i procesa u

složene radne tijekove namijenjeno za najširi krug potrošača, postupak povezivanja procesa na razini udomljenika zahtijeva metodologiju vrlo blisku svakodnevnom korištenju tih procesa uzajamnim djelovanjem putem grafičkog korisničkog sučelja.

Programskom paradigmom za usložnjavanje udomljenika definiran je programski jezik za povezivanje procesa oponašanjem uzajamnog djelovanja potrošača i računala putem grafičkih korisničkih sučelja udomljenika. Standardne radnje koje korisnik primjenom ulazno-izlaznih naprava računala obavlja putem grafičkog korisničkog sučelja udomljenika preslikavaju se u elemente programskog jezika.

7.2.3 Programirljivost grafičkih korisničkih sučelja

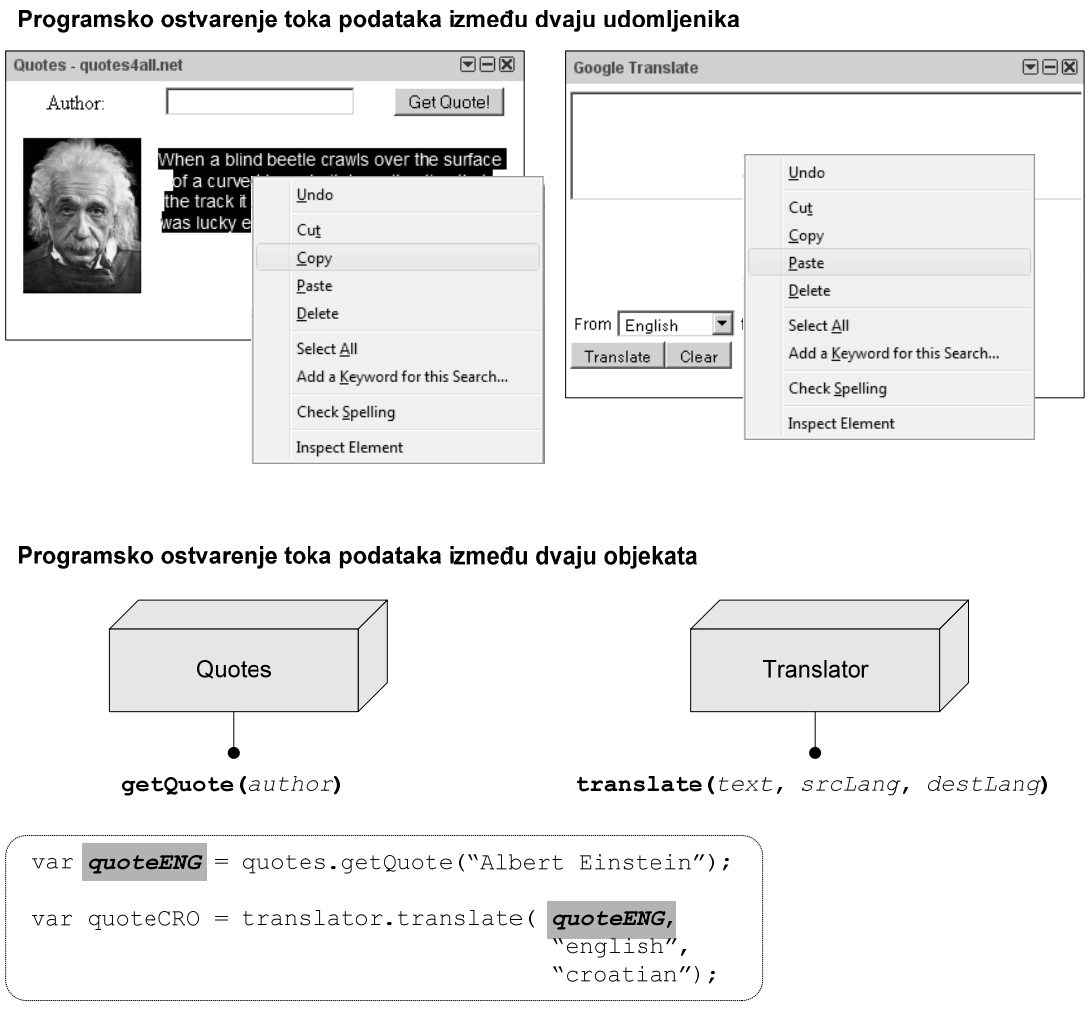
Mogućnosti primjene udomljenika u svojstvu elemenata za izgradnju primjenskih programa i radnji nad elementima grafičkog korisničkog sučelja u svojstvu programskog jezika pokazane su usporedbom svojstava programirljivosti grafičkih korisničkih sučelja sa svojstvima programirljivosti programskih sučelja. Za usporedbu svojstava programirljivosti iskorištena je objektno-orijentirana paradigma programiranja kao jedna od najzastupljenijih programskih paradigmi u grani profesionalnog razvoja programske potpore. Slikom 7.7 prikazano je preslikavanje pojedinih elemenata programirljivosti programskih sučelja na istovjetne elemente programirljivosti grafičkih korisničkih sučelja udomljenika.



Slika 7.7. Preslikavanje elemenata programirljivosti programskih sučelja na istovjetne elemente programirljivosti grafičkih korisničkih sučelja

Ako se primjenski program gradi primjenom paradigme objektno-orijentiranog programiranja, programske funkcionalnosti programerima su izložene u obliku skupa objekata putem odgovarajućih programskih sučelja. S druge strane, u programskoj paradigmi za usložnjavanje udomljenika programske su funkcionalnosti potrošačima izložene u obliku skupa udomljenika putem grafičkih korisničkih sučelja (1).

Funkcionalnosti koje su programerima izložene u obliku članskih metoda nad objektima, potrošačima su izložene u obliku aktivacijskih elemenata grafičkog korisničkog sučelja, kao što su tipke ili poveznice (2). Za pokretanje određene aktivnosti nad objektom, potrebno je obaviti poziv odgovarajuće članske metode, za što je potrebno napisati naredbu u skladu sa sintaksnim pravilima programskog jezika. S druge strane, pokretanje aktivnosti nad udomljenikom postiže se radnjom pritiska aktivacijskog elementa grafičkog korisničkog sučelja.



Slika 7.8. Usporedba svojstava programirljivosti programskih i grafičkih korisničkih sučelja povezanih tokom podataka

Parametri metoda nad objektima kod udomljenika su predstavljeni ulaznim elementima u obliku polja za unos teksta, označnih polja ili padajućih izbornika (3). Prosljeđivanje vrijednosti parametara pri pozivu metode nad objektom kod grafičkih korisničkih sučelja odgovara upisivanju sadržaja u odgovarajući ulazni element udomljenika.

Usporedba svojstava programirljivosti programskih i grafičkih korisničkih sučelja povezanih tokom podataka prikazana je slikom 7.8. Ako je između dvaju objekata potrebno uspostaviti tok podataka, to se postiže primjenom vezane programske varijable. Najprije se pozove metoda nad jednim objektom, povratna vrijednost se spremi u lokalnu varijablu, a nakon toga se vrijednost te varijable koristi kod poziva metode nad drugim objektom. Primjena vezane varijable istaknuta je u odsječku programa za uspostavu toka podataka između dvaju objekata na slici 7.8. Tok podataka između dvaju udomljenika postiže se označavanjem, preuzimanjem i preslikavanjem sadržaja (engl. *copy/paste*) između dvaju elemenata grafičkog korisničkog sučelja.

Tablica 7.1. Usporedba svojstava programirljivosti grafičkih korisničkih sučelja i programskih sučelja

	Programska sučelja	**Grafička korisnička sučelja**
Gradivni elementi	Objekti	Udomljenici
Predodžba funkcionalnosti elementa	Članske metode	Aktivacijski elementi
Pokretanje funkcionalnosti elementa	Poziv članske metode	Pritisak aktivacijskog elementa
Parametrizacija funkcionalnosti elementa	Ulazno-izlazni parametri	Ulazno-izlazni elementi
Pokretanje parametrizirane funkcionalnosti	Prosljeđivanje vrijednosti parametara	Upisivanje sadržaja u ulazne elemente
Tok podataka	Vezane varijable	Označavanje, preuzimanje i preslikavanje sadržaja

Usporedba svojstava programirljivosti grafičkih korisničkih sučelja i programskih sučelja pregledno je prikazana tablicom 7.1. Pronalaskom odgovarajućih elemenata programirljivosti grafičkih korisničkih sučelja za poznate elemente programirljivosti programskih sučelja, pokazano je da je udomljenike, osim što su potrošaču razumljivi po funkcijskim svojstvima i načinu primjene, moguće iskoristiti i kao programske elemente za izgradnju primjenskih programa. Osim toga, pokazano je da je radnje nad elementima grafičkog korisničkog sučelja moguće iskoristiti kao programski jezik za povezivanje udomljenika u radni tijek prema potrebama potrošača.

Izborom udomljenika u svojstvu gradivnih elemenata i radnji nad elementima grafičkog korisničkog sučelja u svojstvu programskog jezika, programiranje postaje prirodna nadogradnja na uobičajene navike potrošača jer postaje istovjetno uobičajenom doživljaju i rukovanju primjenskim programima. Oblikovanje i izgradnja primjenskih programa na razini grafičkih korisničkih sučelja udomljenika ključni je element koji programsku paradigmu za usložnjavanje udomljenika čini bliskom širokom krugu potrošača.

7.3 Arhitektura primjenskih programa

Osim gradivnih elemenata i programskog jezika, znanjima i vještinama potrošača potrebno je prilagoditi i programsku arhitekturu na kojoj se zasniva oblikovanje i izgradnja primjenskih programa. Programskom arhitekturom određen je način usložnjavanja skupa udomljenika u primjenski program te način izlaganja izgrađenog programa potrošačima.

7.3.1 Oblikovanje arhitekture primjenskih programa

Modelom za ostvarenje programiranja prilagođenog potrošaču koji je opisan u poglavlju 6 definirano je svojstvo *višerazinskog usložnjavanja blokova* kojim je određen oblik arhitekture primjenskih programa. Svojstvom višerazinskog usložnjavanja blokova zahtijeva se mogućnost izgradnje primjenskog programa kroz više razina usložnjavanja blokova. Na taj se način olakšava razvoj, smanjuje mogućnost pogreške i omogućava učinkovitije održavanje programa jer se postupnim usložnjavanjem blokova ne zahtijeva usredotočenost potrošača na cjelokupnu strukturu programa, nego na izdvojene programske cjeline.

S obzirom na to da se u predloženoj programskoj paradigmi programski blokovi pojavljuju u obliku udomljenika, za izgradnju primjenskih programa izabrana je višerazinska arhitektura za usložnjavanje udomljenika. Skup udomljenika povezuje se u radni tijek prema potrebama potrošača, a objedinjena funkcionalnost dostupna je u obliku novog udomljenika. Dobiveni udomljenik moguće je koristiti kao primjenski program za krajnju uporabu ili kao programski blok za daljnje usložnjavanje. Takvim izborom programske arhitekture se, uz mogućnost postupne izgradnje, postiže i mogućnost udruženog sudjelovanja zajednice potrošača u razvoju primjenskih programa. Ako se vršni udomljenik koji nastaje kao krajnji rezultat usložnjavanja podijeli sa zajednicom korisnika, bilo kojem članu zajednice omogućena je nadogradnja dobivenog primjenskog programa drugim udomljenicima i nastanak novih primjenskih programa.

7.3.2 Višerazinska arhitektura za usložnjavanje udomljenika

Višerazinskom arhitekturom za usložnjavanje udomljenika omogućeno je usložnjavanje skupa udomljenika kroz više razina. Primjenski program koji nastaje povezivanjem skupa udomljenika pojavljuje se u obliku novog udomljenika koji se naziva složenim udomljenikom. Primjenom višerazinske arhitekture za usložnjavanje udomljenika, primjenski programi izgrađeni od strane potrošača postaju istovjetni elementima od kojih su izgrađeni, što omogućava njihovu primjenu za krajnju uporabu, ali i za daljnju izgradnju novih primjenskih programa od strane istog ili drugih potrošača. Elementi višerazinske arhitekture za usložnjavanje udomljenika su osnovni udomljenici, složeni udomljenici, relacija usložnjavanja udomljenika i relacija upravljanja radom udomljenika.

Osnovni udomljenici

Osnovnim udomljenicima nazivaju se udomljenici na najnižoj razini usložnjavanja. U programskoj paradigmi za usložnjavanje udomljenika, osnovni udomljenici smatraju se nedjeljivim programskim elementima. Putem osnovnih udomljenika, potrošačima je omogućeno upravljanje izvođenjem pozadinskih računalnih procesa koji se izvode na lokalnom računalu ili u mreži Internet. Dostupni su putem *World Wide Web* sustava postavljanjem na javno dostupne poslužitelje i objavljivanjem u javnim imenicima. Graditelji osnovnih udomljenika najčešće su profesionalni programeri. Vlasnici osnovnih udomljenika su pojedinci ili organizacije koji računalne procese u obliku grafičkih korisničkih sučelja nastoje izložiti potrošačima.

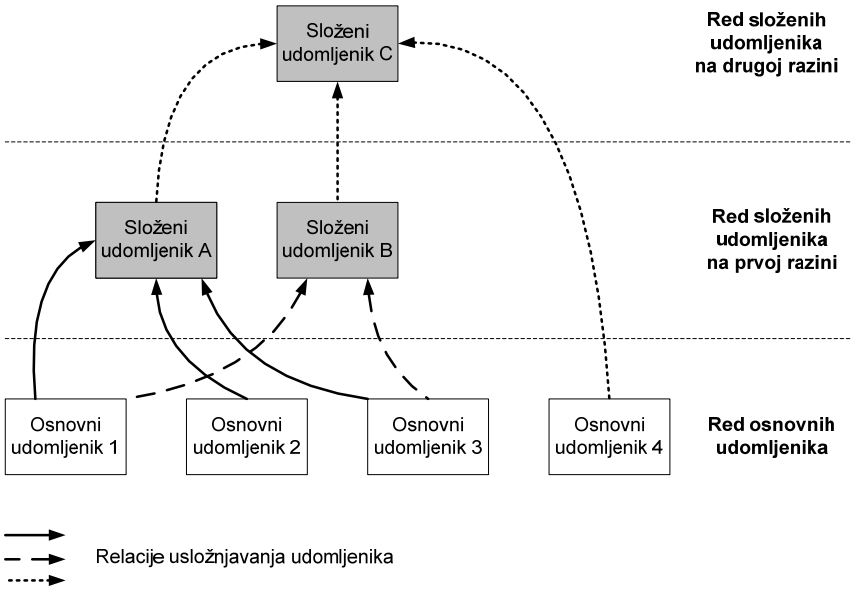
Složeni udomljenik

Složeni udomljenik je udomljenik koji nastaje usložnjavanjem skupa osnovnih udomljenika i putem kojeg je funkcionalnost primjenskog programa izgrađenog primjenom programske paradigme za usložnjavanje udomljenika izložena potrošačima. U programskoj paradigmi za usložnjavanje udomljenika, složeni udomljenik ima dvostruku ulogu. S jedne strane, složeni udomljenik sadrži zapis radnog tijeka definiranog nad skupom osnovnih udomljenika i programsku infrastrukturu za njegovo izvođenje. S druge strane, složeni udomljenik sadrži grafičko korisničko sučelje za pokretanje definiranog radnog tijeka i praćenje rezultata izvođenja povezanog skupa osnovnih udomljenika. Graditelji i vlasnici složenih udomljenika su potrošači, odnosno korisnici programske paradigme za usložnjavanje udomljenika.

Postupak usložnjavanja skupa osnovnih udomljenika u složeni udomljenik sastoji se od definiranja radnog tijeka nad skupom osnovnih udomljenika te od definiranja korisničkog sučelja složenog udomljenika. Radni tijek nad skupom osnovnih udomljenika definira se primjenom programskog jezika koji se sastoji od radnji nad elementima grafičkog korisničkog sučelja, a opisan je u poglavlju 7.2. Korisničko sučelje složenog udomljenika oblikuje se preuzimanjem odgovarajućih elemenata korisničkog sučelja sa osnovnih udomljenika.

Relacija usložnjavanja udomljenika

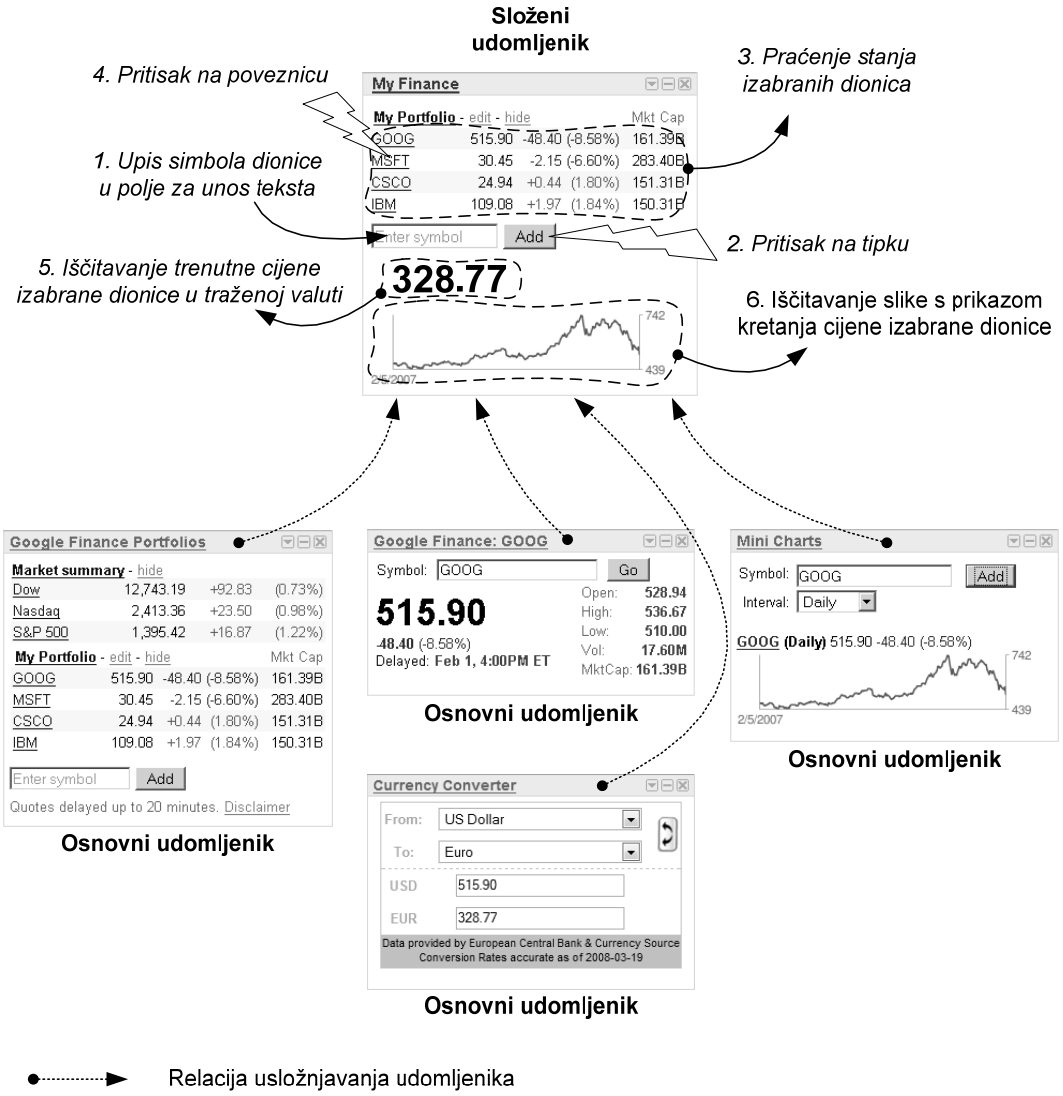
Višerazinskom arhitekturom za usložnjavanje udomljenika omogućeno je povezivanje skupa osnovnih udomljenika u složeni udomljenik, ali i povezivanje skupa složenih udomljenika na nižoj razini usloženosti u složeni udomljenik na višoj razini usloženosti. Arhitektura primjenskog programa zasnovanog na usložnjavanju udomljenika definirana je relacijom usložnjavanja udomljenika, kao što je prikazano na slici 7.9.



Slika 7.9. Relacija usložnjavanja udomljenika

U prikazanom primjeru, relacijom usložnjavanja udomljenika definirano je da su funkcionalnosti osnovnih udomljenika 1, 2 i 3 iskorištene za izgradnju složenog udomljenika A. Osim za izgradnju složenog udomljenika A, funkcionalnosti osnovnih udomljenika 1 i 3 istovremeno su iskorištene i za izgradnju složenog udomljenika B. Funkcionalnosti složenih udomljenika A i B nadalje se koriste za izgradnju složenog udomljenika C u kojem se, uz dva složena udomljenika, koriste i funkcionalnosti osnovnog udomljenika 4. Budući da su izgrađeni isključivo od skupa osnovnih udomljenika, složeni udomljenici A i B pripadaju

redu složenih udomljenika na prvoj razini usloženosti. Složeni udomljenik C pripada redu složenih udomljenika na drugoj razini usloženosti jer, osim jednog osnovnog udomljenika, koristi funkcionalnosti dvaju udomljenika koji pripadaju redu složenih udomljenika na prvoj razini usloženosti.



Slika 7.10. Primjer relacije usložnjavanja udomljenika

Višerazinskom arhitekturom za usložnjavanje udomljenika omogućeno je usložnjavanje udomljenika kroz neograničeni broj razina. Relacija usložnjavanja udomljenika ima svojstvo preslikavanja N:1, što znači da se više osnovnih udomljenika ili složenih udomljenika na nižoj razini usloženosti koristi za izgradnju jednog složenog udomljenika na višoj razini usloženosti. Nad istim skupom udomljenika moguće je istovremeno definirati više različitih relacija usložnjavanja udomljenika. Time se omogućava da bilo koji osnovni udomljenik ili složeni udomljenik na nižoj razini usloženosti bude

iskorišten za izgradnju proizvoljnog broja složenih udomljenika na višim razinama usloženosti. Primjerice, osnovni udomljenici 1 i 3 na slici 7.9 istovremeno se koriste za izgradnju dvaju složenih udomljenika.

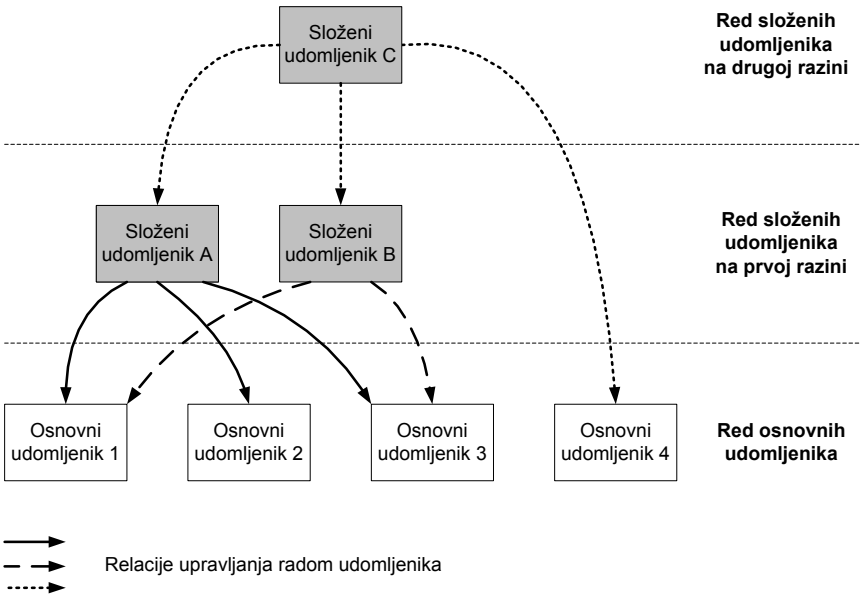
Slikom 7.10 prikazan je primjer usložnjavanja skupa osnovnih udomljenika s ciljem izgradnje složenog udomljenika za upravljanje osobnim financijskim portfeljom. Izgradnjom složenog udomljenika pod nazivom *My Finance*, složeni posao upravljanja osobnim financijskim portfeljom za koji je prije trebalo koristiti četiri različita udomljenika, sada je moguće obaviti putem samo jednog udomljenika. Složeni udomljenik pruža mogućnost dodavanja novih dionica u skup izabranih dionica upisom simbola izabrane dionice u polje za unos teksta *Symbol* (1) i pritiskom na tipku *Add* (2). Stanje skupa izabranih dionica moguće je pratiti u rubrici *My Portfolio* (3). Pritiskom na odgovarajuću poveznicu u rubrici *My Potfolio* (4), obavlja se izbor jedne od prikazanih dionica čiju je trenutnu cijenu u eurima moguće pratiti u polju za prikaz teksta *Price* (5), a kretanje cijene dionice tijekom radnog dana putem slikovnog elementa *Chart* (6). Funkcionalnost složenog udomljenika izgrađena je povezivanjem funkcionalnosti četiriju osnovnih udomljenika. Pojedini dijelovi korisničkog sučelja složenog udomljenika preuzeti su s odgovarajućih osnovnih udomljenika.

Relacija upravljanja radom udomljenika

Tijekom oblikovanja složenog udomljenika, od skupa osnovnih udomljenika ili složenih udomljenika na nižoj razini usloženosti preuzima se dio elemenata grafičkog korisničkog sučelja za pokretanje i praćenje izvođenja definiranog radnog tijeka. Međutim, programska logika osnovnih udomljenika ne preuzima se u složeni udomljenik, već ostaje dio osnovnih udomljenika. Uloga složenog udomljenika tijekom izvođenja radnog tijeka je vremenski usklađeno pozivanje primjenskih funkcionalnosti osnovnih udomljenika i prikazivanje rezultata njihova izvođenja. S obzirom da tijekom izvođenja radnog tijeka složeni udomljenik upravlja radom osnovnih udomljenika, definira se relacija upravljanja radom udomljenika.

Relacija upravljanja radom udomljenika prikazana je slikom 7.11. Relacija upravljanja radom udomljenika ima svojstvo preslikavanja 1:N, što znači da jedan složeni udomljenik na višoj razini usloženosti upravlja radom skupa osnovnih udomljenika ili složenih udomljenika na nižim razinama usloženosti. Primjer na slici 7.11 prikazuje sustav udomljenika nad kojim su definirane tri relacije upravljanja radom udomljenika. Prvom relacijom definirano je da složeni udomljenik A upravlja radom osnovnih udomljenika 1, 2 i 3. Drugom relacijom

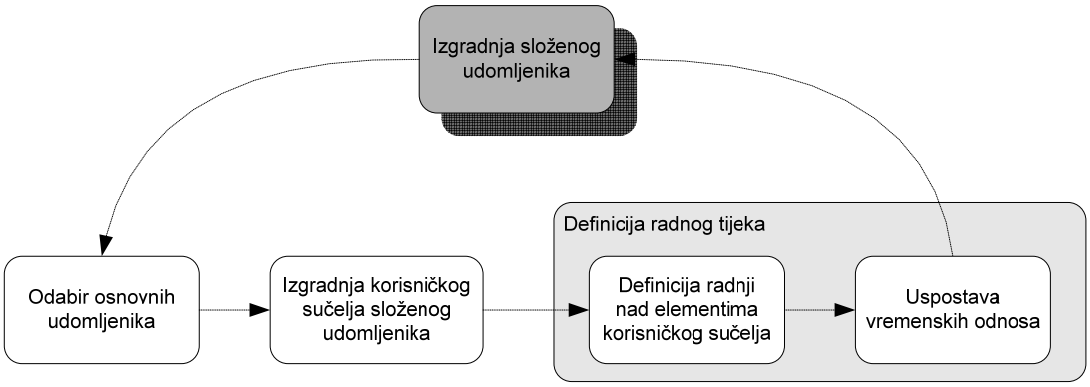
definirano je da složeni udomljenik B upravlja radom osnovnih udomljenika 1 i 3. Trećom relacijom definirano je da složeni udomljenik C upravlja radom složenih udomljenika A i B te osnovnog udomljenika 4.



Slika 7.11. Relacija upravljanja radom udomljenika

Razvojni ciklus za usložnjavanje udomljenika

Postupak izgradnje složenog udomljenika sastoji se od triju koraka, kao što je prikazano shemom na slici 7.12. U prvom koraku odabire se skup osnovnih udomljenika čije se funkcionalnosti koriste za izgradnju složenog udomljenika. U drugom koraku odabiru se elementi korisničkog sučelja osnovnih udomljenika od kojih se gradi korisničko sučelje složenog udomljenika. U trećem koraku definira se radni tijek nad skupom osnovnih udomljenika.

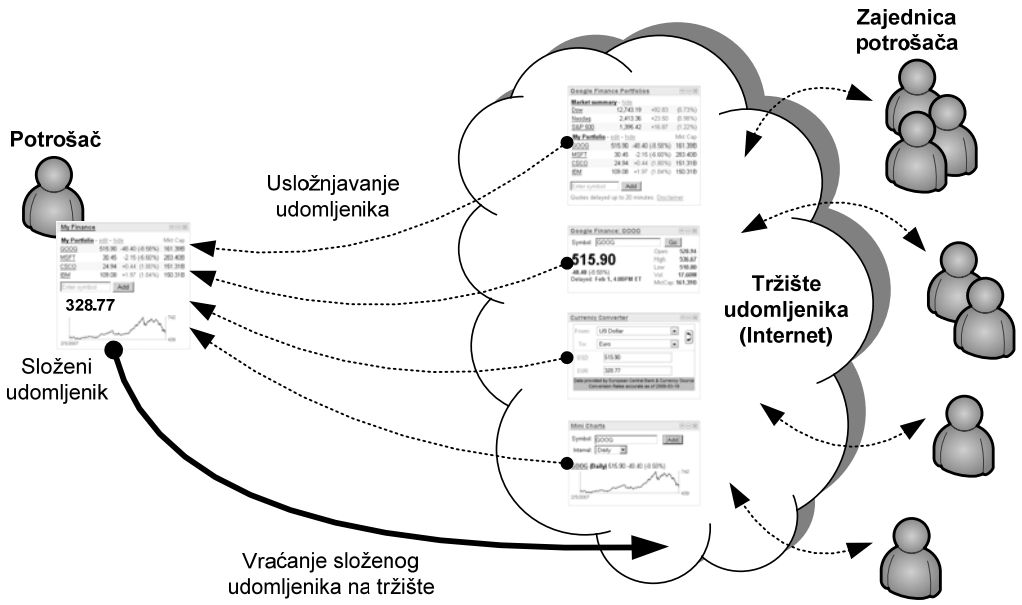


Slika 7.12. Razvojni ciklus za usložnjavanje udomljenika

Izgradnja radnog tijeka sastoji se od dvaju koraka. U prvom koraku, definiraju se radnje koje je potrebno izvršiti nad elementima grafičkog korisničkog sučelja osnovnih udomljenika. U drugom koraku, nad skupom definiranih radnji uspostavljaju se vremenski odnosi kako bi se njihovim pravovremenim izvođenjem osigurao vremenski usklađeni rad osnovnih udomljenika.

Poticanje inovacije i udruženog stvaranja primjenskih programa

Programskom paradigmom za usložnjavanje udomljenika potiče se stvaranje programskih inovacija, dok se izborom višerazinske arhitekture za usložnjavanje udomljenika omogućuje udruženo stvaranje primjenskih programa. Doprinos programske paradigme zasnovane na višerazinskoj arhitekturi za usložnjavanje udomljenika stvaranju inovacija i udruženom stvaranju primjenskih programa prikazan je slikom 7.13.



Slika 7.13. Poticanje inovacije i udruženog stvaranja primjenskih programa primjenom višerazinske arhitekture za usložnjavanje udomljenika

Usložnjavanjem udomljenika dostupnih na globalnom informacijskom tržištu u novi poosobljeni složeni udomljenik nastaje novi primjenski program. Na taj način programska paradigma koja je prikladna za široki krug potrošača svakom sudioniku globalnog informacijskog društva omogućava stvaranje inovativnog doprinosa u obliku računalnih programa. S obzirom na to da su primjenski programi koji nastaju usložnjavanjem udomljenika dostupnih na tržištu opet izloženi u obliku udomljenika, novostvoreni primjenski program pogodan je za vraćanje na tržište u svrhu razmjene s drugim

potrošačima. Višerazinskim usložnjavanjem udomljenika, bilo kojem članu zajednice potrošača omogućeno je daljnje nadograđivanje složenih udomljenika drugim udomljenicima i izgradnja novih primjenskih programa na osnovi prethodnog rada ostalih potrošača.

7.4 Tehnika izgradnje primjenskog programa

Tehnikom izgradnje primjenskog programa definiran je način povezivanja skupa nezavisnih udomljenika u radni tijek prema zamislima potrošača. Izborom tehnike za izgradnju programa potrebno je omogućiti potrošaču razumljiv način definiranja radnji nad elementima grafičkog korisničkog sučelja udomljenika te njihovo spremanje u obliku računalnog programa za kasnije izvođenje.

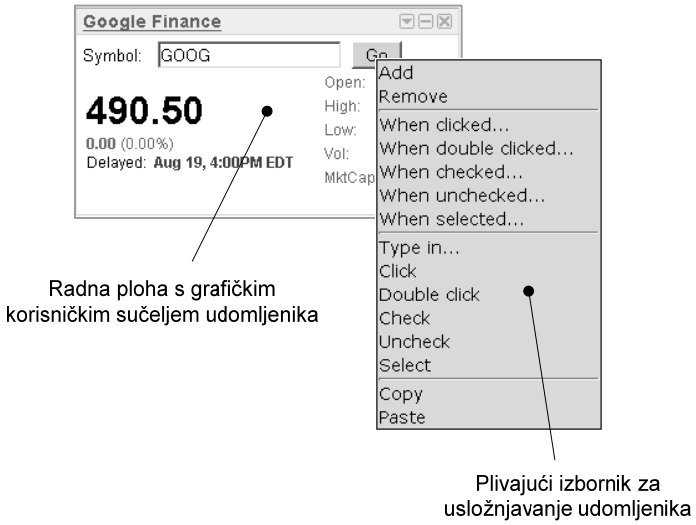
7.4.1 Oblikovanje tehnike izgradnje primjenskog programa

Modelom za ostvarenje programiranja prilagođenog potrošaču koji je opisan u poglavlju 6 definirano je svojstvo *izgradivosti primjenskog programa putem grafičkog korisničkog sučelja*. Ovo svojstvo odnosi se na izbor tehnike za povezivanje programskih blokova u radni tijek prema zakonitostima primjenskog programa. Svojstvom izgradivosti povezujućih elemenata putem grafičkog korisničkog sučelja zahtijeva se da je povezivanje programskih blokova u potpunosti omogućeno putem grafičkog korisničkog sučelja.

S obzirom na to da se u predloženoj programskoj paradigmi programski blokovi pojavljuju u obliku udomljenika, a povezujući elementi u obliku radnji potrošača nad elementima grafičkog korisničkog sučelja, tehnika izgradnje programa zasnovana je na plivajućem grafičkom izborniku (engl. *context menu*) za obilazak udomljenika. Obilaženjem udomljenika, potrošač putem plivajućeg izbornika zadaje slijed radnji koje je potrebno obaviti nad elementima grafičkog korisničkog sučelja. Slijed definiranih radnji sprema se u obliku računalnog programa za kasnije izvođenje na zahtjev potrošača.

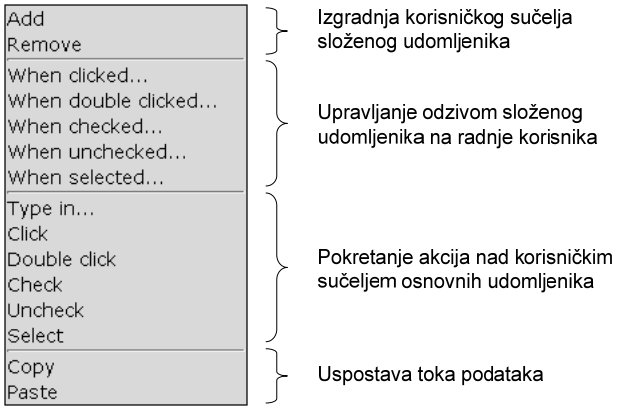
7.4.2 Programiranje primjenom grafičkog izbornika

Tehnika programiranja primjenom plivajućeg grafičkog izbornika koristi se za definiciju radnog tijeka nad skupom osnovnih udomljenika te za izgradnju korisničkog sučelja složenog udomljenika putem kojeg se pokreće definirani radni tijek. Mogućnošću izbora stavki ponuđenih unutar plivajućeg izbornika, potrošač je oslobođen pamćenja elemenata programskog jezika koje je moguće primijeniti u pojedinim koracima izgradnje programa, kao i leksičkih i sintaksnih pravila za zapis programa.



Slika 7.14. Plivajući izbornik za izgradnju složenog udomljenika

Plivajući izbornik za izgradnju složenog udomljenika prikazan je slikom 7.14. Izbornik je objedinjen s udomiteljskom stranicom, a odaziva se na pritisak desne tipke miša unutar radne plohe s korisničkim sučeljem udomljenika. Pojavljuje se na mjestu obavljanja pritiska desnom tipkom miša, čime je omogućen jednostavan obilazak udomljenika. Nakon pojavljivanja plivajućeg izbornika, potrošaču je omogućen izbor jedne od ponuđenih stavki na osnovi koje pozadinska programska logika stvara zapis programa.



Slika 7.15. Funkcijska podjela stavki plivajućeg izbornika za izgradnju složenog udomljenika

S obzirom na funkciju u postupku izgradnje složenog udomljenika, stavke izbornika podijeljene su u četiri skupine, kao što je prikazano slikom 7.15. Prvu skupinu čine dvije stavke izbornika koje se koriste za izgradnju korisničkog sučelja složenog udomljenika. Primjenom ovih dviju stavki, omogućeno je preuzimanje elemenata grafičkog korisničkog sučelja s osnovnih udomljenika i prenošenje na radnu plohu složenog udomljenika, odnosno uklanjanje preuzetih elemenata grafičkog korisničkog sučelja s radne plohe složenog

udomljenika. Drugu skupinu čine stavke izbornika koje služe za upravljanje odzivom složenog udomljenika na radnje potrošača. Primjenom ove skupine stavki, određuje se na koji način se putem grafičkog korisničkog sučelja složenog udomljenika pokreće radni tijek definiran nad skupom osnovnih udomljenika. Treću skupinu čine stavke izbornika za definiciju radnog tijeka nad skupom osnovnih udomljenika. Ovom skupinom stavki definiraju se radnje koje nad elementima grafičkog korisničkog sučelja osnovnih udomljenika izvodi složeni udomljenik kako bi se uspostavio predviđeni radni tijek. Konačno, četvrtu skupinu čine dvije stavke izbornika koje služe za uspostavu toka podataka u skupu udomljenika primjenom mehanizma preuzimanja i preslikavanja sadržaja (engl. *copy/paste*). Primjena pojedinih stavki izbornika podrobnije je opisana u poglavljima 8, 9, 10 i 11.

7.5 Prikaz primjenskog programa

Tehnikom programiranja zasnovanom na grafičkom izborniku stvoren je potrošaču prilagođen postupak izgradnje programa. Osim tehnike izgradnje programa, u okviru potrošaču prilagođene programske paradigme potrebno je oblikovati i potrošaču prilagođen oblik prikaza izgrađenog programa. To je bitno zbog mogućnosti naknadne nadogradnje ili ažuriranja programa, ali i razmjene programa sa zajednicom potrošača.

7.5.1 Oblikovanje tehnike prikaza primjenskog programa

Modelom za ostvarenje programiranja prilagođenog potrošaču koji je opisan u poglavlju 6 definirana su svojstva *opažajnog doživljaja prikaza programa* te *upravljivosti i uočljivosti vremenske relacije* koja se odnose na izbor tehnike za prikaz programa. Prikazom programa potrebno je omogućiti potrošaču razumljiv način predodžbe gradivnih i povezujućih elemenata te uređivanja vremenskih odnosa među programskim cjelinama.

Tehnika prikaza programa oblikovana je iskorištavanjem povoljnih značajki programskih paradigmi iz područja *razvoja prilagođenog krajnjem korisniku* te *teorije jezične predodređenosti* iz područja psihologije programiranja. Skup radnji nad elementima grafičkog korisničkog sučelja udomljenika koje su definirane primjenom grafičkog izbornika prikazuje se slovčanim zapisom koji nalikuje prirodnom jeziku. Ovaj način prikaza odabran je na osnovi povoljnih značajki paradigme *programiranja prirodnim jezikom*.

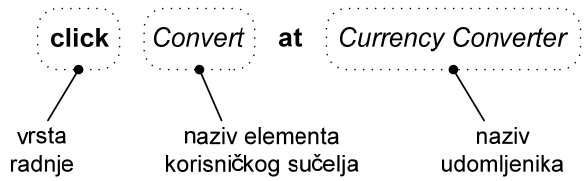
Prikaz i uređivanje vremenskih odnosa među radnjama nad elementima grafičkog korisničkog sučelja udomljenika zasniva se na tabličnom zapisu programa. Tablični zapis programa odabran je na osnovi povoljnih značajki paradigme *programiranja primjenom*

grafičkih oznaka i *programiranja primjenom tabličnih proračuna*. Analizom tih paradigmi, pokazane su prednosti dvodimenzionalne organizacije programa za prikaz radnog tijeka nad skupom elemenata. Dvodimenzionalna organizacija programa omogućava oblikovanje složenih vremenskih uzoraka izvođenja programa, kao što je slijedno i istodobno izvođenje te razdvajanje i spajanje vremenskih tijekova, isključivo primjenom prostornog razmještaja programskih elemenata. Za oblikovanje složenih vremenskih uzoraka, od potrošača se očekuje rukovanje jednostavnim relacijama vremenskog prethođenja, vremenskog slijeđenja i vremenske nezavisnosti bez potrebe za primjenom nerazumljivih sinkronizacijskih mehanizama i elemenata za pokretanje istodobnog izvođenja dijelova programa, kao što su procesi i dretve.

Unutar tablice definiran je vremenski uređaj s dvosmjernim napredovanjem vremena, gdje vrijeme istodobno i nezavisno napreduje u dva smjera: odozgo prema dolje i slijeva na desno. Dvosmjerno napredovanje vremena izabrano je na osnovi *teorije jezične predodređenosti*. Tom je teorijom pokazano da je, kao jedna od posljedica govornog jezika, čovjeku prirodni način predodžbe vremenskih odnosa korištenje okomitih i vodoravnih prostornih odnosa.

7.5.2 Tablični zapis radnji nad elementima grafičkog korisničkog sučelja

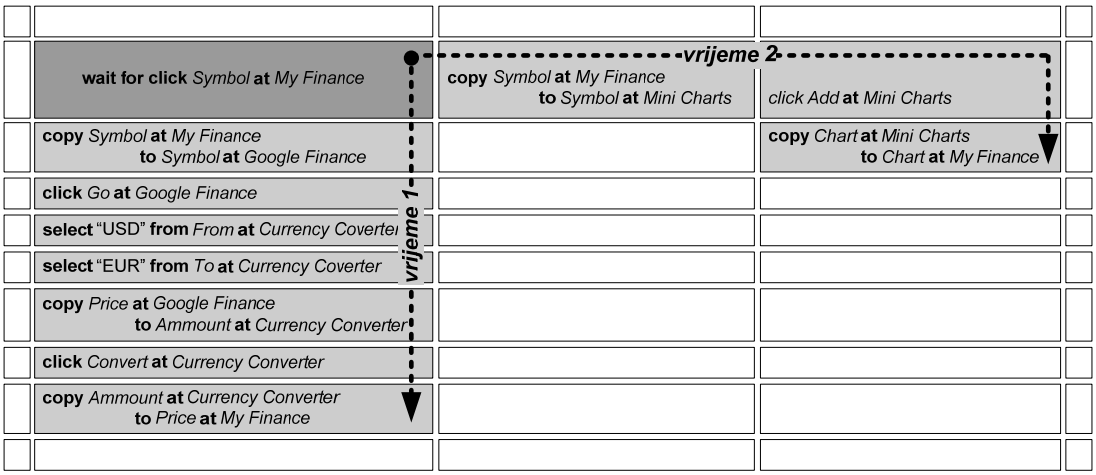
Radni tijek primjenskog programa određen je redoslijedom izvođenja radnji koje nad elementima grafičkog korisničkog sučelja osnovnih udomljenika izvodi složeni udomljenik. Radnje nad elementima grafičkog korisničkog sučelja osnovnih udomljenika definiraju se primjenom grafičkog izbornika, a zapisuju slovčanim zapisom u obliku prirodnog jezika. Primjer zapisa radnji nad elementima grafičkog korisničkog sučelja udomljenika prikazan je slikom 7.16. Zapis pojedine radnje sastoji se od vrste radnje, naziva elementa korisničkog sučelja nad kojim je potrebno obaviti definiranu radnju te naziva udomljenika koji sadrži ciljni element korisničkog sučelja.



Slika 7.16. Prikaz radnji nad elementima grafičkog korisničkog sučelja udomljenika

Vremenski odnosi nad skupom definiranih radnji prikazuju se i uređuju u prostoru dvodimenzionalne tablice, kao što je prikazano slikom 7.17. Tablica se sastoji od ćelija.

Neprazne ćelije tablice sadrže slovčane zapise radnji nad elementima grafičkog korisničkog sučelja udomljenika.



Slika 7.17. Tablični prikaz programske logike složenog udomljenika

Nad nepraznim ćelijama tablice definirano je dvosmjerno napredovanje vremena. Unutar nepraznih ćelija, vrijeme istodobno napreduje odozgo prema dolje i slijeva na desno. Vremenski tijek prekida se nailaskom na praznu ćeliju. Izvođenje radnje zapisane u promatranoj ćeliji može započeti tek nakon što je izvedena radnja zapisana u neposrednoj gornjoj i neposrednoj lijevoj ćeliji. Razdvajanjem radnji praznim ćelijama prekida se vremenski tijek i radnje postaju vremenski nezavisne pa ih je moguće izvoditi proizvoljnim redoslijedom, uključujući i istodobno. Izvođenje programa započinje radnjama koje nemaju niti jednog vremenskog prethodnika. Takve radnje zapisane su u ćelijama koje su odozgo i slijeva omeđene isključivo praznim ćelijama ili se nalaze u gornjem lijevom kutu tablice.

Organizacija programa u tabličnom prostoru s dvosmjernim napredovanjem vremena omogućuje jednostavno oblikovanje i visoku razinu uočljivosti složenih uzoraka tijeka izvođenja programa, kao što je slijedno i paralelno izvođenje. Slikom 7.17 prikazan je primjer programa u kojem se nakon izvođenja radnje zapisane u gornjoj lijevoj ćeliji vremenski tijek razdvaja na dvije nezavisne grane označene oznakama *vrijeme 1* i *vrijeme 2*. Dok je radnje unutar pojedine grane potrebno izvoditi isključivo slijedno, radnje koje pripadaju različitim granama moguće je izvoditi proizvoljnim redoslijedom, uključujući i istodobno.

Tabličnim prikazom programa postignuto je uravnoteženje između količine informacija kojima se opisuju vremenske zavisnosti u programu i čitljivosti programskog zapisa. Izravne vremenske zavisnosti jasno su istaknute u zapisu programa. Prostorno

susjedne ćelije tablice sadrže vremenski slijedne operacije nad programskim blokovima. Prostorno razdvojene ćelije tablice sadrže vremenski nezavisne operacije nad blokovima. Neizravne vremenske zavisnosti koje proizlaze iz svojstva tranzitivnosti vremenske relacije nisu posebno istaknute kako se prikaz programa ne bi nepotrebno opterećivao informacijama do kojih se dolazi jednostavnim zaključivanjem. Izvođenje zaključaka o postojanju neizravnih vremenskih zavisnosti provodi se na osnovi analize prostornog razmještaja i popunjenosti ćelija. Ako između dviju nepraznih i nesusjednih ćelija tablice postoji prostorni put kroz skup susjednih i nepraznih ćelija u bilo kojem od dva smjera napredovanja vremena, onda su nesusjedne ćelije tablice povezane neizravnom vremenskom relacijom. Pronalaženje tranzitivnih vremenskih zavisnosti u tabličnom prikazu programa je jednostavno jer se zasniva na razlučivosti praznih od nepraznih ćelija, što je na zaslonu računala prilično lako uočljivo. Podrobniji opis oblikovanja različitih uzoraka tijeka izvođenja primjenom tablične organizacije programa prikazan je u poglavlju 11.

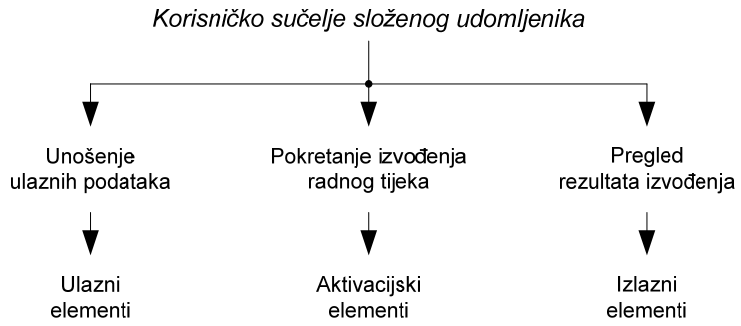
8 Oblikovanje korisničkog sučelja složenog udomljenika

Složeni udomljenik koji nastaje primjenom programske paradigme za usložnjavanje udomljenika sadrži definiciju radnog tijeka za povezivanje skupa osnovnih udomljenika u poosobljeni primjenski program te grafičko korisničko sučelje za pokretanje i praćenje rezultata izvođenja tog radnog tijeka. U ovom poglavlju opisan je postupak oblikovanja i izgradnje korisničkog sučelja složenog udomljenika. Elementi za izgradnju korisničkog sučelja složenog udomljenika preuzimaju se s korisničkog sučelja osnovnih udomljenika. U poglavlju 8.1 prikazana je razredba elemenata grafičkog korisničkog sučelja osnovnih udomljenika s obzirom na ulogu tijekom izvođenja radnog tijeka. Na osnovi prikazane razredbe, oblikuje se grafičko korisničko sučelje složenog udomljenika. U poglavlju 8.2 opisan je postupak izgradnje korisničkog sučelja složenog udomljenika preuzimanjem dijelova korisničkih sučelja osnovnih udomljenika.

8.1 Razredba elemenata korisničkog sučelja osnovnih udomljenika

Izgradnjom korisničkog sučelja složenog udomljenika, potrošaču je omogućeno unošenje početnih vrijednosti za izvođenje radnog tijeka nad skupom osnovnih udomljenika, pokretanje izvođenja radnog tijeka i pregled rezultata izvođenja radnog tijeka. Slikom 8.1 prikazana je uloga korisničkog sučelja složenog udomljenika tijekom izvođenja radnog tijeka i vrste elemenata korisničkog sučelja koje se koriste u pojedinim koracima. Za unošenje početnih vrijednosti koriste se ulazni elementi korisničkog sučelja. Primjeri ulaznih elemenata su polja za unos teksta, označna polja, izborna polja i padajući izbornici. Za

pokretanje izvođenja radnog tijeka koriste se aktivacijski elementi korisničkog sučelja. Primjeri aktivacijskih elemenata su tipke i poveznice. Za pregled rezultata izvođenja radnog tijeka koriste se izlazni elementi korisničkog sučelja. Primjeri izlaznih elemenata su slike, tablice i odlomci za prikaz teksta.



Slika 8.1. Uloga korisničkog sučelja složenog udomljenika

Korisničko sučelje složenog udomljenika oblikuje se preuzimanjem elemenata korisničkog sučelja s osnovnih udomljenika i njihovim postavljanjem na praznu radnu plohu složenog udomljenika. Odabir elemenata za prenošenje s osnovnih u složeni udomljenik obavlja potrošač na osnovi uloge pojedinog elementa u radnom tijeku.

U skupu osnovnih udomljenika povezanih radnim tijekom moguće je uočiti tri različite uloge koje poprimaju pojedini elementi korisničkog sučelja. Dio elemenata korisničkog sučelja potrošač uopće ne koristi. U primjeru na slici 2.2, skup elemenata sučelja objedinjen u rubrici *Market Summary* udomljenika *Google Finance Portfolios* te elementi sučelja *Open*, *High*, *Low*, *Vol* i *MktCap* udomljenika *Google Finance* primjeri su elemenata koje potrošač ne koristi tijekom izvođenja radnog tijeka. Ostali elementi korisničkog sučelja koje potrošač koristi u barem jednom scenariju tijekom izvođenja radnog tijeka dijele se u dvije skupine. Prvu skupinu čine elementi koje potrošač koristi za unošenje početnih vrijednosti i prikaz rezultata izvođenja radnog tijeka. Elementi sučelja objedinjeni u rubrici *My Portfolio*, polje *Enter symbol* i tipka *Add* udomljenika *Google Finance Portfolios*, polje za prikaz cijene dionice u eurima *EUR* udomljenika *Currency Converter* i slikovni element s grafičkim prikazom kretanja cijene dionice udomljenika *Mini Charts* primjeri su elemenata koje potrošač koristi za unošenje početnih vrijednosti i prikaz rezultata izvođenja radnog tijeka. Drugu skupinu čine elementi koje potrošač koristi za izračunavanje i prenošenje međurezultata između pojedinih udomljenika. Primjeri takvih elemenata su polje za unos teksta *Symbol* i tipka *Go* udomljenika *Google Finance*, polje za unos teksta *Symbol*, padajući izbornik *Interval* i tipka *Add* udomljenika *Mini Charts* te polja *From*, *To* i *USD* udomljenika *Currency Converter*.

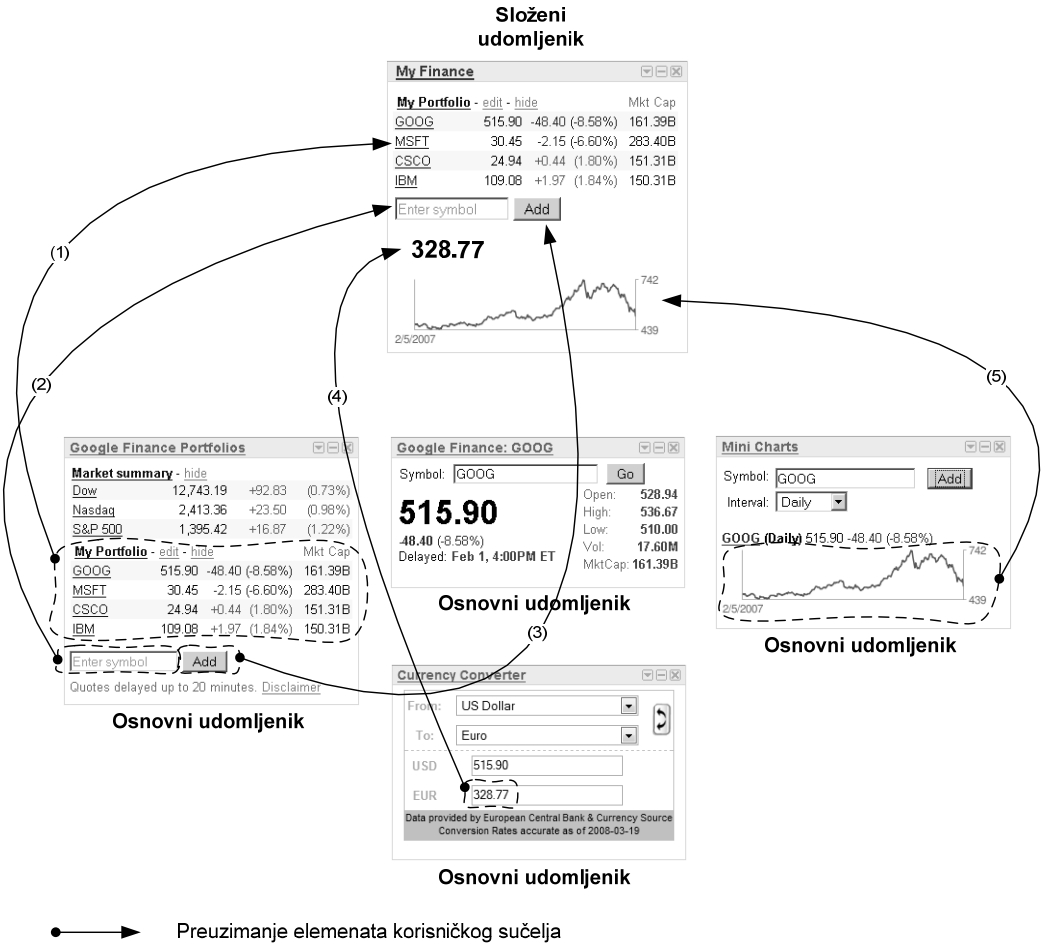
Korisničko sučelje složenog udomljenika oblikuje se preuzimanjem elemenata korisničkog sučelja osnovnih udomljenika koji se koriste za unos početnih vrijednosti, pokretanje izvođenja radnog tijeka i prikaz potrošaču zanimljivih rezultata izvođenja. Na taj način izgrađeno korisničko sučelje složenog udomljenika potrošaču omogućuje upravljanje izvođenjem radnog tijeka bez potrebe za daljnjom primjenom osnovnih udomljenika.

8.2 Izgradnja korisničkog sučelja složenog udomljenika

Slikom 8.2 prikazan je postupak oblikovanja grafičkog korisničkog sučelja složenog udomljenika. Prikazan je primjer oblikovanja sučelja za upravljanje izvođenjem radnog tijeka prikazanog na slici 2.2 koji nastaje usložnjavanjem četiriju osnovnih udomljenika.

Prostor oblikovanja

Prema zamislima potrošača, složeni udomljenik sastoji se od pet elemenata korisničkog sučelja. Elementi sučelja preuzimaju se s tri različita osnovna udomljenika.

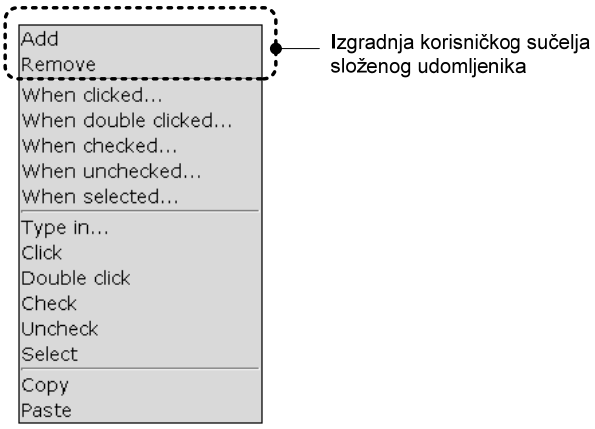


Slika 8.2. Oblikovanje grafičkog korisničkog sučelja složenog udomljenika

Od udomljenika *Google Finance Portfolios* koji se koristi za podešavanje osobnog financijskog portfelja, preuzima se rubrika *My Portfolio* za praćenje stanja skupa izabranih dionica (1), polje *Enter symbol* za unos simbola dionice koji je potrebno dodati u skup izabranih dionica (2) te tipka *Add* za dodavanje zadane dionice u skup izabranih dionica (3). Od udomljenika *Currency Converter* koji se koristi za pretvorbu valuta, preuzima se polje *EUR* za prikaz cijene dionice u eurima (4). Od udomljenika *Mini Charts* koji se koristi za pregled izvještaja o kretanju cijene dionice, preuzima se slikovni element s grafičkim prikazom kretanja cijene dionice tijekom dana. Iako se tijekom oblikovanja radnog tijeka koristi i udomljenik *Google Finance* za pregled izvještaja o trgovanju izabranom dionicom, nijedan od elemenata korisničkog sučelja tog udomljenika nije izabran za oblikovanje sučelja složenog udomljenika. Svi elementi udomljenika *Google Finance* koji se koriste u radnom tijeku sadrže međurezultate izvođenja primjenskog programa.

Prostor izgradnje

Izgradnja korisničkog sučelja složenog udomljenika provodi se primjenom plivajućeg grafičkog izbornika. Plivajućim izbornikom obilazi se skup osnovnih udomljenika te se označavaju elementi korisničkog sučelja koji se prenose u složeni udomljenik. Dijelovi grafičkog izbornika koji se koriste tijekom izgradnje korisničkog sučelja složenog udomljenika istaknuti su slikom 8.3.



Slika 8.3. Primjena grafičkog izbornika za izgradnju korisničkog sučelja složenog udomljenika

Plivajući izbornik sadrži dvije stavke za izgradnju korisničkog sučelja složenog udomljenika. Stavka *Add* koristi se za preuzimanje elemenata korisničkog sučelja osnovnih udomljenika i njihovo postavljanje u složeni udomljenik. Stavka *Remove* koristi se za uklanjanje prethodno preuzetih elemenata korisničkog sučelja s radne plohe složenog udomljenika.

Prostor prikaza

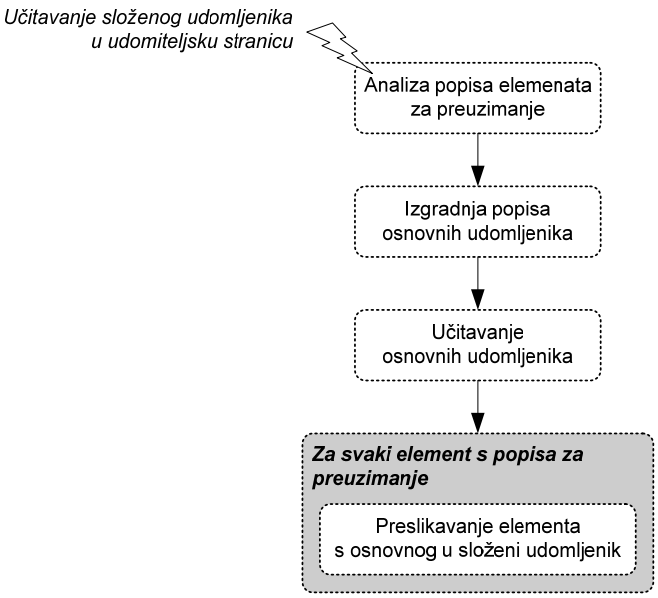
Na osnovi obilaženja osnovnih udomljenika i obilježavanja elemenata korisničkog sučelja primjenom plivajućeg izbornika, stvara se popis elemenata za preuzimanje u složeni udomljenik. Popis elemenata sprema se unutar složenog udomljenika. Pojedinim elementom popisa određena je operacija preuzimanja po jednog elementa korisničkog sučelja s osnovnog udomljenika u složeni udomljenik. Popis elemenata sučelja za preuzimanje s osnovnih u složeni udomljenik prikazana je slovčanim zapisom u obliku

add My Portfolio at Google Finance Portfolios
add Symbol at Google Finance Portfolios
add Add at Google Finance Portfolios
add Price at Google Finance
add Chart at Mini Charts

Prikazanim slovčanim zapisom definiran je popis za preuzimanje pet elemenata korisničkog sučelja s osnovnih udomljenika u složeni udomljenik. Korisničko sučelje složenog udomljenika sastoji se od pet elementa koji se preuzimaju od tri osnovna udomljenika čiji su nazivi redom *Google Finance Portfolios*, *Google Finance* i *Mini Charts*. Od osnovnog udomljenika *Google Finance Portfolios* preuzimaju se elementi *My Portfolio*, *Symbol* i *Add*, od osnovnog udomljenika *Google Finance* preuzima se element *Price*, dok se od osnovnog udomljenika *Mini Charts* preuzima element *Chart*.

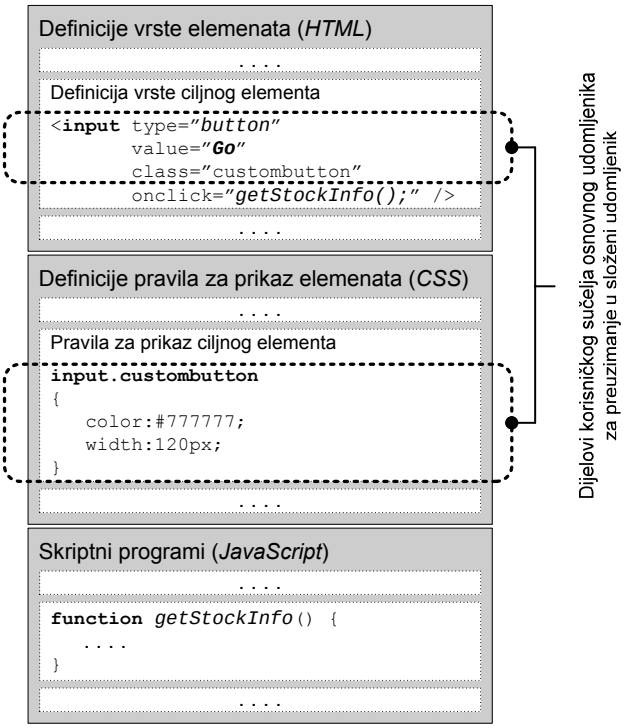
Prostor izvođenja

Preuzimanje elemenata korisničkog sučelja s osnovnih udomljenika u složeni udomljenik prikazano je algoritmom na slici 8.4. Algoritam za preuzimanje elemenata pokreće se tijekom svakog učitavanja složenog udomljenika u udomiteljsku stranicu. Algoritam započinje analizom popisa elemenata korisničkog sučelja za preuzimanje s osnovnih udomljenika u složeni udomljenik. Analizom popisa elemenata utvrđuje se skup osnovnih udomljenika od kojih se preuzimaju elementi. Budući da je za preuzimanje elemenata u složeni udomljenik skup osnovnih udomljenika potrebno dovesti u radno stanje, treći korak algoritma je učitavanje skupa osnovnih udomljenika u udomiteljsku stranicu. Nakon što su svi osnovni udomljenici učitani u udomiteljsku stranicu, započinje postupak preuzimanja elemenata. Za svaki element s popisa za preuzimanje, preslika elementa korisničkog sučelja osnovnog udomljenika preuzima se i postavlja unutar korisničkog sučelja složenog udomljenika.



Slika 8.4. Algoritam za izgradnju korisničkog sučelja složenog udomljenika

Grafičko korisničko sučelje udomljenika izgrađeno je primjenom programskih tehnologija za izgradnju predodžbenih funkcija primjenskih programa za *World Wide Web*. U strukturi grafičkog korisničkog sučelja udomljenika moguće je uočiti tri cjeline, kao što je prikazano slikom 8.5.



Slika 8.5. Dijelovi korisničkog sučelja osnovnog udomljenika koji se preuzimaju u složeni udomljenik

Prvom cjelinom definirane su vrste elemenata od kojih se sastoji korisničko sučelje udomljenika. Definicija vrste elemenata za izgradnju korisničkih sučelja ostvaruje se primjenom označnog jezika *HTML* [217]. Drugom cjelinom definirana su pravila za prikaz elemenata korisničkog sučelja na zaslonu korisničkog računala. Definicije pravila za prikaz elemenata korisničkog sučelja zapisuju se primjenom prikaznog jezika *CSS* [218]. Trećom cjelinom definirani su skriptni programi za upravljanje odzivom pojedinih elemenata korisničkog sučelja na radnje korisnika. Najčešći način ostvarenja skriptnih programa je primjena skriptnog jezika *JavaScript* [219, 220, 221].

Tijekom preslikavanja elemenata korisničkog sučelja, s osnovnog udomljenika u složeni udomljenik prenosi se definicija vrste elementa i pravila za prikaz elementa. Dijelovi definicije elementa korisničkog sučelja koji se odnose na odziv elementa na radnje korisnika ne prenose se u složeni udomljenik. Odziv prenesene preslike elementa korisničkog sučelja na radnje korisnika u složenom se udomljeniku definira programskim jezikom za usložnjavanje udomljenika. Skup elemenata programskog jezika za usložnjavanje udomljenika kojim se upravlja odzivom pojedinih elemenata korisničkog sučelja na radnje potrošača opisan je u poglavlju 11.

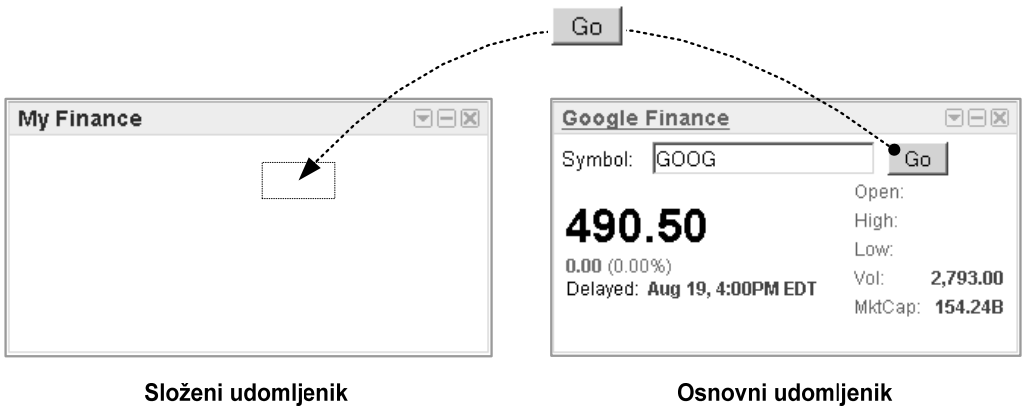
8.2.1 Događaj za preuzimanje elemenata korisničkog sučelja

Popis elemenata korisničkog sučelja osnovnih udomljenika za preuzimanje u složeni udomljenik sastoji se od konačnog skupa *događaja za preuzimanje elemenata korisničkog sučelja*. Za svaki element korisničkog sučelja složenog udomljenika, potrošač definira po jedan događaj za preuzimanje elemenata korisničkog sučelja. Izvođenjem događaja, preslika elementa korisničkog sučelja osnovnog udomljenika postavlja se unutar radne plohe s korisničkim sučeljem složenog udomljenika.

Prostor oblikovanja

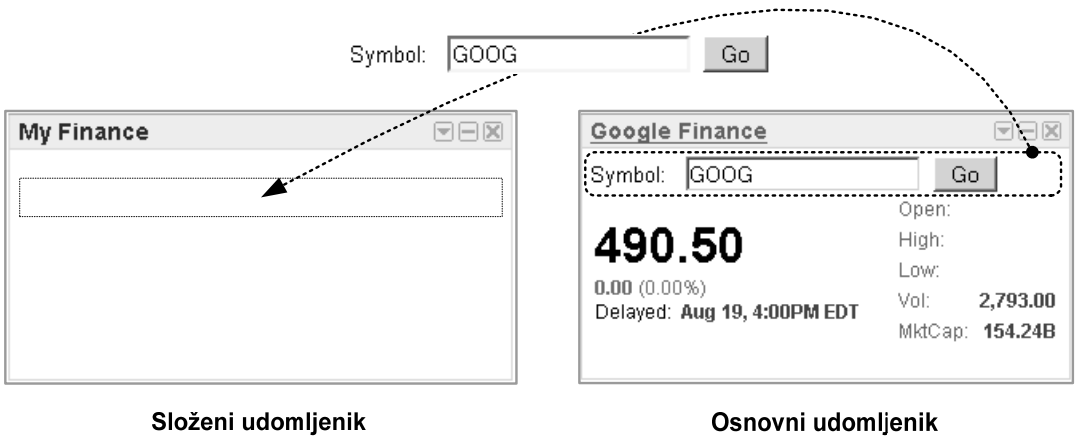
Prostor oblikovanja događaja za preuzimanje elemenata korisničkog sučelja sastoji se od osnovnog i složenog udomljenika. Osnovni udomljenik sadrži korisničko sučelje za upravljanje izvođenjem pridruženog računalnog procesa na lokalnom računalu ili u mreži Internet. Korisničko sučelje složenog udomljenika na početku je prazno. Događajima za preuzimanje elemenata korisničkog sučelja preuzima se jedan po jedan element korisničkog sučelja s osnovnog udomljenika u složeni udomljenik. Izvođenjem događaja za preuzimanje elemenata korisničkog sučelja, korisničko sučelje složenog udomljenika postupno se proširuje novim elementima. Broj događaja za preuzimanje elemenata korisničkog sučelja

jednak je broju elemenata korisničkog sučelja složenog udomljenika koje je potrebno preuzeti od osnovnih udomljenika.



Slika 8.6. Oblikovanje događaja za preuzimanje elementa korisničkog sučelja

Oblikovanje događaja za preuzimanje elemenata korisničkog sučelja prikazano je slikom 8.6. Osnovni udomljenik sadrži element korisničkog sučelja čiju je presliku potrebno preuzeti i postaviti unutar radne plohe korisničkog sučelja složenog udomljenika. Događaj za preuzimanje elemenata korisničkog sučelja oblikuje se tako da se na korisničkom sučelju osnovnog udomljenika odabire element koji je potrebno preuzeti u skup elemenata korisničkog sučelja složenog udomljenika. Preslika izabranog elementa prenosi se s osnovnog udomljenika u složeni udomljenik i postavlja unutar radne plohe njegovog korisničkog sučelja. Primjerom na slici 8.6 prikazano je oblikovanje događaja za preuzimanje elementa korisničkog sučelja u obliku tipke *Go* s osnovnog udomljenika *Google Finance* u složeni udomljenik *My Finance*.

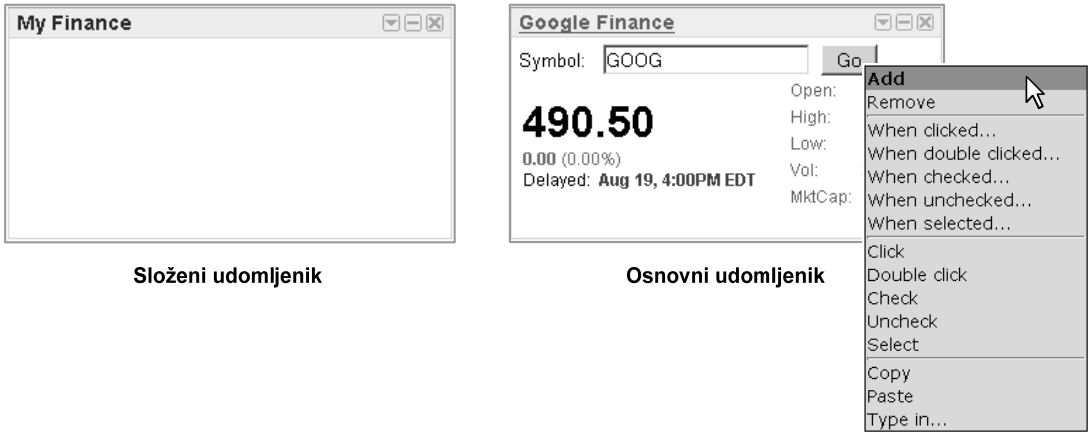


Slika 8.7. Oblikovanje događaja za preuzimanje zbirnog elementa korisničkog sučelja

Osim preuzimanja jednostavnih elemenata, korisničko sučelje složenog udomljenika moguće je oblikovati i preuzimanjem zbirnih elemenata, kao što su obrasci (engl. *form*), rubrike i tablice. Jedan zbirni element sastoji se od skupa jednostavnih elemenata. Oblikovanje događaja za preuzimanje zbirnog elementa korisničkog sučelja u obliku obrasca prikazano je slikom 8.7. Osnovni udomljenik *Google Finance* sadrži element korisničkog sučelja u obliku obrasca čiju je presliku potrebno preuzeti i postaviti unutar radne plohe korisničkog sučelja složenog udomljenika *My Finance*. Prikazani obrazac je zbirni element koji se sastoji od slovčane oznake *Symbol*, polja za unos teksta *Symbol* i tipke *Go*. Događaj za preuzimanje zbirnog elementa korisničkog sučelja oblikuje se tako da se na korisničkom sučelju osnovnog udomljenika odabire zbirni element koji objedinjava skup jednostavnih elemenata. Jednim događajem za preuzimanje elemenata korisničkog sučelja, preslike svih jednostavnih elemenata unutar izabranog zbirnog elementa prenose se s osnovnog udomljenika u složeni udomljenik.

Prostor izgradnje

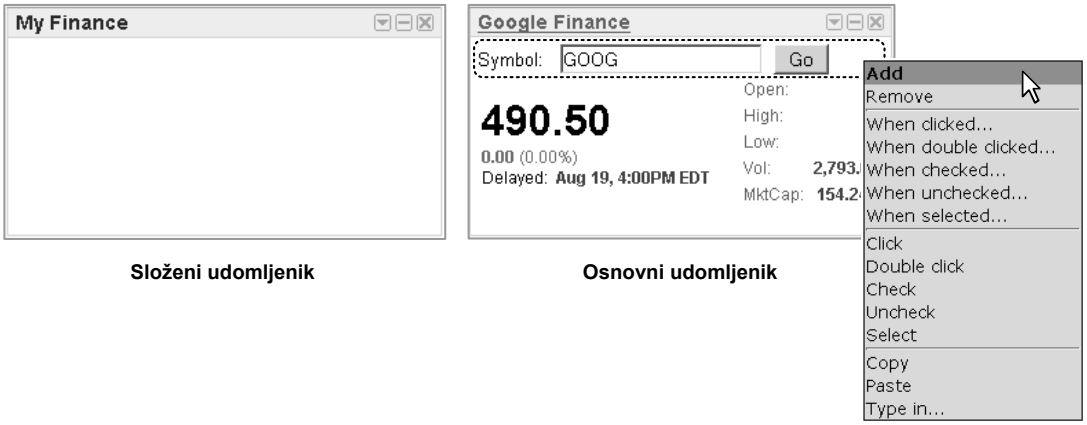
Izgradnja događaja za preuzimanje elemenata korisničkog sučelja prikazana je slikom 8.8. Pritiskom desne tipke miša na izabrani element korisničkog sučelja osnovnog udomljenika, pojavljuje se plivajući izbornik za usložnjavanje udomljenika. Događaj za preuzimanje elementa korisničkog sučelja izgrađuje se odabirom stavke *Add* u plivajućem izborniku. Primjerom na slici 8.8 prikazan je postupak izgradnje događaja za preuzimanje tipke *Go* s osnovnog udomljenika *Google Finance* u složeni udomljenik *My Finance*.



Slika 8.8. Izgradnja događaja za preuzimanje elementa korisničkog sučelja u programskom jeziku za usložnjavanje udomljenika

Primjenom sličnog postupka, moguće je izgraditi događaj za preuzimanje zbirnog elementa korisničkog sučelja. Primjerom na slici 8.9 prikazan je postupak izgradnje događaja

za preuzimanje zbirnog elementa u obliku obrasca s osnovnog udomljenika *Google Finance* u složeni udomljenik *My Finance*. Element u obliku obrasca sastoji se od slovčane oznake *Symbol*, polja za unos teksta *Symbol* i tipke *Go*



Slika 8.9. Izgradnja događaja za preuzimanje zbirnog elementa korisničkog sučelja u programskom jeziku za usložnjavanje udomljenika

U prikazanom primjeru, zbirni element u obliku obrasca prostire se čitavom širinom gornjeg dijela korisničkog sučelja udomljenika *Google Finance*. Za izgradnju događaja za preuzimanje zbirnog elementa, plivajući izbornik potrebno je pokrenuti u točki koja se nalazi unutar područja prostiranja zbirnog elementa, ali istodobno ne pripada niti jednom jednostavnom elementu unutar zbirnog elementa.

Prostor prikaza

Događaj za preuzimanje elementa korisničkog sučelja s osnovnog u složeni udomljenik zapisuje se unutar složenog udomljenika slovčanim zapisom u obliku

add Go at Google Finance

Prikazanim slovčanim zapisom prikazan je događaj za preuzimanje elementa korisničkog sučelja *Go* koji pripada korisničkom sučelju osnovnog udomljenika *Google Finance* u skup elemenata korisničkog sučelja složenog udomljenika. Slovčani prikaz događaja za preuzimanje elemenata korisničkog sučelja sastoji se od ključne riječi *add* kojom je jednoznačno određen događaj za preuzimanje elemenata korisničkog sučelja, naziva elementa korisničkog sučelja koji je potrebno preuzeti s osnovnog u složeni udomljenik, naziva osnovnog udomljenika od kojeg je potrebno obaviti preuzimanje elementa te ključne riječi *at* koja služi za razdvajanje naziva elementa i naziva udomljenika. Isti oblik prikaza

događaja koristi se za preuzimanje jednostavnih i zbirnih elemenata korisničkog sučelja. Ako je u prostoru izgradnje definiran događaj za preuzimanje zbirnog elementa, u slovčanom zapisu događaja umjesto naziva jednostavnog elementa pojavljuje se naziv zbirnog elementa.

elementZaPreuzimanje = **add** *element* **at** *udomljenik*
element = ([**A**..**Z**] + [**a**..**z**] + [**0**..**9**]) $^{+}$
udomljenik = ([**A**..**Z**] + [**a**..**z**] + [**0**..**9**]) $^{+}$

Slika 8.10. Opći oblik prikaza događaja za preuzimanje elementa korisničkog sučelja s osnovnog u složeni udomljenik

Opći oblik prikaza događaja za preuzimanje elemenata korisničkog sučelja s osnovnog u složeni udomljenik prikazan je regularnim definicijama na slici 8.10. Regularnom definicijom *elementZaPreuzimanje* opisano je pravilo za oblikovanje nizova znakova koji predstavljaju valjano zapisane događaje za preuzimanje elemenata korisničkog sučelja. Regularna definicija *elementZaPreuzimanje* definirana je ključnim riječima *add* i *at* te regularnim definicijama *element* i *udomljenik*. Ključna riječ *add* koristi se za jednoznačno prepoznavanje događaja za preuzimanje elemenata korisničkog sučelja. Regularnom definicijom *element* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje elemenata korisničkog sučelja osnovnih udomljenika koji se preuzimaju u složeni udomljenik. Regularnom definicijom *udomljenik* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje osnovnih udomljenika od kojih se elementi korisničkog sučelja preuzimaju u složeni udomljenik. Naslovljavanje elemenata korisničkog sučelja i osnovnih udomljenika obavlja se nizovima znakova koji se sastoje od brojki te velikih i malih slova abecede. Ključna riječ *at* koristi se u svojstvu graničnika između naziva elementa korisničkog sučelja i naziva udomljenika.

popisElemenataZaPreuzimanje = (*elementZaPreuzimanje*) *

Slika 8.11. Opći oblik prikaza popisa događaja za preuzimanje elemenata korisničkog sučelja s osnovnih udomljenika u složeni udomljenik

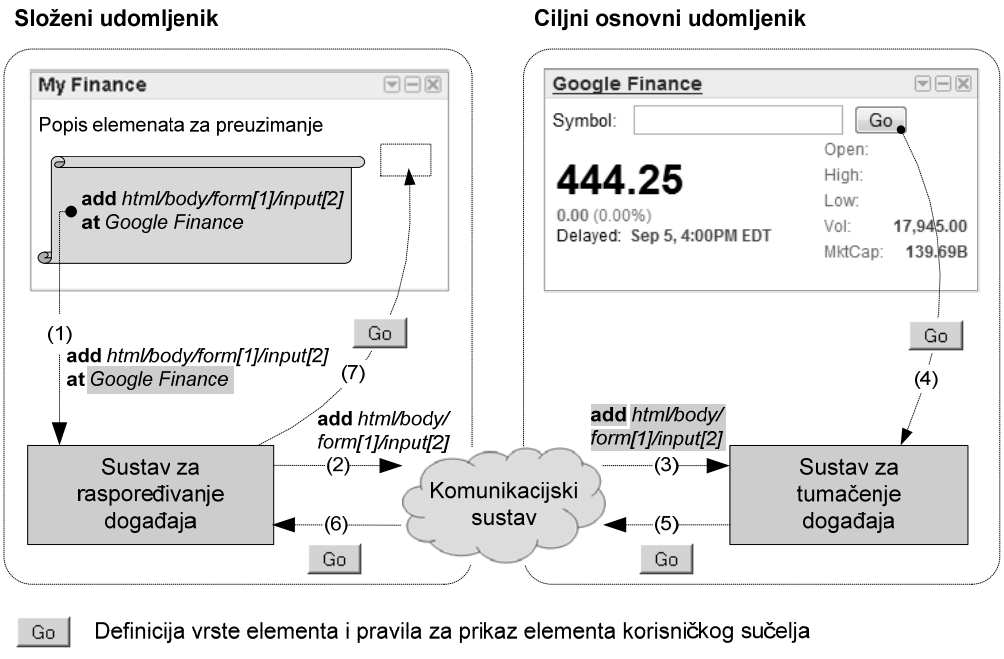
Na osnovi događaja za preuzimanje pojedinačnih elemenata korisničkog sučelja oblikuje se popis događaja kojim je opisano cjelokupno grafičko korisničko sučelje složenog udomljenika. Opći oblik popisa događaja za preuzimanje elemenata korisničkog sučelja prikazan je regularnim definicijama na slici 8.11. Regularnom definicijom *popisElemenataZaPreuzimanje* opisano je pravilo za oblikovanje nizova znakova koji predstavljaju skup valjano zapisanih događaja za preuzimanje elemenata korisničkog sučelja

kojima je određen način izgradnje korisničkog sučelja složenog udomljenika. Regularna definicija *popisElemenataZaPreuzimanje* sastoji se od skupa nadovezanih nizova znakova opisanih regularnom definicijom *elementZaPreuzimanje* kojom je opisano pravilo za zapisivanje pojedinačnih događaja za preuzimanje elemenata korisničkog sučelja.

Prostor izvođenja

Popis događaja za preuzimanje elemenata korisničkog sučelja s osnovnih udomljenika u složeni udomljenik ugrađuje se u programsku strukturu složenog udomljenika, a koristi se na početku izvođenja složenog udomljenika, prema algoritmu prikazanom na slici 8.4. Izvođenjem pojedinih događaja s popisa, odgovarajući elementi korisničkog sučelja preuzimaju se jedan po jedan iz odgovarajućih osnovnih udomljenika i ugrađuju u korisničko sučelje složenog udomljenika.

Izvođenje pojedinačnih događaja za preuzimanje elemenata korisničkog sučelja prikazano je slikom 8.12. Složeni udomljenik sadrži popis događaja za preuzimanje elemenata korisničkog sučelja. Popis događaja izgrađen je u prostoru izgradnje događaja, a unutar složenog udomljenika zapisan je slično kao što je prikazan u prostoru prikaza događaja. U zapisu događaja, simboličko ime elementa korisničkog sučelja zamijenjeno je *XPath* izrazom kojim je određen položaj elementa unutar programske strukture *HTML* dokumenta. Osnovni udomljenik sadrži grafičko korisničko sučelje s ciljnim elementom čiju je presliku potrebno preuzeti i ugraditi u grafičko korisničko sučelje složenog udomljenika.



Slika 8.12. Prostor izvođenja događaja za preuzimanje elementa korisničkog sučelja

Osim grafičkog korisničkog sučelja, osnovni i složeni udomljenik sadrže i programsku infrastrukturu za izvođenje događaja za preuzimanje elemenata korisničkog sučelja. Složeni udomljenik sadrži sustav za raspoređivanje događaja. Sustav za raspoređivanje događaja zadužen je za pronalaženje ciljnog udomljenika od kojeg je potrebno obaviti preuzimanje elementa korisničkog sučelja. Osnovni udomljenik sadrži sustav za tumačenje događaja. Sustav za tumačenje događaja zadužen je za pronalaženje i stvaranje preslike ciljnog elementa korisničkog sučelja osnovnog udomljenika. Osnovni i složeni udomljenik povezani su komunikacijskim sustavom.

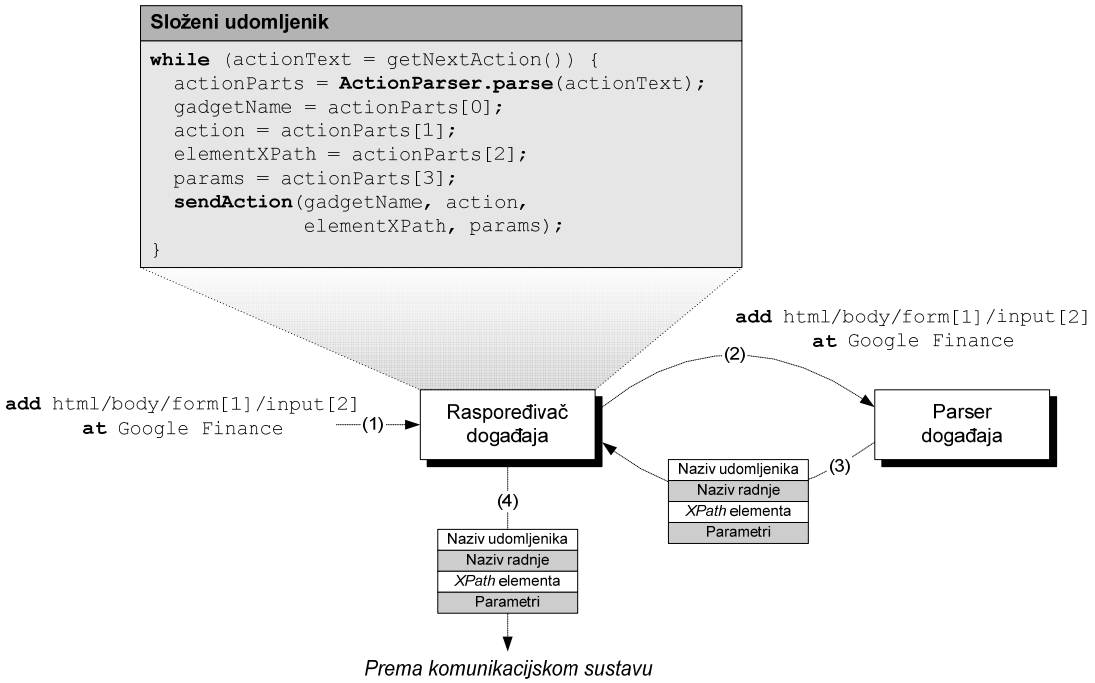
Izvođenje događaja za preuzimanje elemenata korisničkog sučelja odvija se u sedam koraka. Sustav za raspoređivanje događaja dohvaća slovčani zapis događaja za preuzimanje elementa korisničkog sučelja (1). Na osnovi naziva ciljnog udomljenika, zapis događaja se komunikacijskim sustavom prosljeđuje do odredišnog udomljenika (2, 3), gdje ga prihvaća sustav za tumačenje događaja. Na osnovi ključne riječi *add* kojom je jednoznačno određen događaj za preuzimanje elementa korisničkog sučelja, sustav za tumačenje događaja donosi zaključak o vrsti događaja. Nakon utvrđivanja vrste događaja, sustav za tumačenje događaja parsira *XPath* izraz kojim je određen položaj elementa unutar strukture *HTML* dokumenta osnovnog udomljenika. Pretraživanjem strukture korisničkog sučelja osnovnog udomljenika, sustav za tumačenje događaja pronalazi ciljni element korisničkog sučelja i stvara presliku za preuzimanje u složeni udomljenik (4). Preslika elementa korisničkog sučelja prosljeđuje se putem komunikacijskog sustava iz osnovnog u složeni udomljenik (5, 6), gdje se ugrađuje u strukturu korisničkog sučelja složenog udomljenika (7). Opisani postupak koji se odvija u sedam koraka ponavlja se za svaki događaj s popisa događaja za preuzimanje elemenata korisničkog sučelja.

Programsko ostvarenje sustava za izvođenje događaja

Prototip sustava za potrošaču prilagođeno programiranje primjenom programske paradigme za usložnjavanje udomljenika programski je ostvaren proširenjem sustava *Apache Shindig* [222]. *Apache Shindig* je programski sustav otvorenog kôda koji pruža funkcionalnosti posluživanja udomljenika te udomiteljske stranice za njihovo udomljavanje i izvođenje. Izvorni oblik sustava *Apache Shindig* proširen je programskim cjelinama za izvođenje potrošačkih programa definiranih nad skupom udomljenika. Kao što je prikazano na slici 8.12, složeni udomljenik proširen je *sustavom za raspoređivanje događaja*, osnovni udomljenici prošireni su *sustavom za tumačenje događaja*, dok su obje vrste udomljenika te udomiteljska stranica prošireni *komunikacijskim sustavom*.

Sustav za raspoređivanje događaja

Sustav za raspoređivanje događaja je proširenje izvornog oblika sustava *Apache Shindig* koje je ugrađeno u složeni udomljenik. Taj dio sustava zadužen je za parsiranje slovčanog zapisa događaja te određivanje ciljnog udomljenika nad kojim je potrebno izvesti definirani događaj. Slikom 8.13 prikazano je načelo rada i programsko ostvarenje sustava za raspoređivanje događaja. Iako je načelo rada sustava prikazano na primjeru događaja za preuzimanje elementa korisničkog sučelja, isti sustav koristi se i tijekom izvođenja događaja za upravljanje radom udomljenika i događaja za uspostavu toka podataka koji su opisani u poglavljima 9 i 10.



Slika 8.13. Programsko ostvarenje sustava za raspoređivanje događaja

Sustav za raspoređivanje događaja sastoji se od *raspoređivača događaja* i *parsera događaja*. Raspoređivač događaja u ponavljajućoj petlji s popisa događaja dohvaća slovčani zapis događaja (1). Osnovni oblik slovčanog zapisa događaja sastoji se od ključne riječi za određivanje vrste događaja, *XPath* izraza kojim se opisuje položaj elementa u strukturi *HTML* dokumenta ciljnog udomljenika te naziva ciljnog udomljenika. Događaj za upisivanje sadržaja u polje za unos teksta, događaj za izbor stavke iz padajućeg izbornika te događaj za uspostavu toka podataka koji su opisani u poglavljima 9 i 10 koriste prošireni oblik slovčanog zapisa koji sadrži još i dodatne parametre za označavanje slovčanih konstanti, stavki izbornika i podataka za prenošenje između dvaju udomljenika.

Pozivom funkcije *ActionParser.parse*, slovčani zapis događaja prosljeđuje se parseru događaja (2) koji ga rastavlja na niz osnovnih cjelina i ugrađuje u strukturu podataka pogodnu za programsku obradu. Struktura podataka koja sadrži informacije o događaju sastoji se od naziva ciljnog udomljenika, ključne riječi za određivanje vrste događaja, *XPath* izraza za opis putanje do ciljnog elementa i dodatnih parametara za one događaje koji koriste prošireni oblik slovčanog zapisa. Struktura podataka s informacijama o događaju izgrađena tijekom parsiranja vraća se raspoređivaču događaja (3), koji je pozivom funkcije *sendAction* u završnom koraku prosljeđuje komunikacijskom sustavu (4).

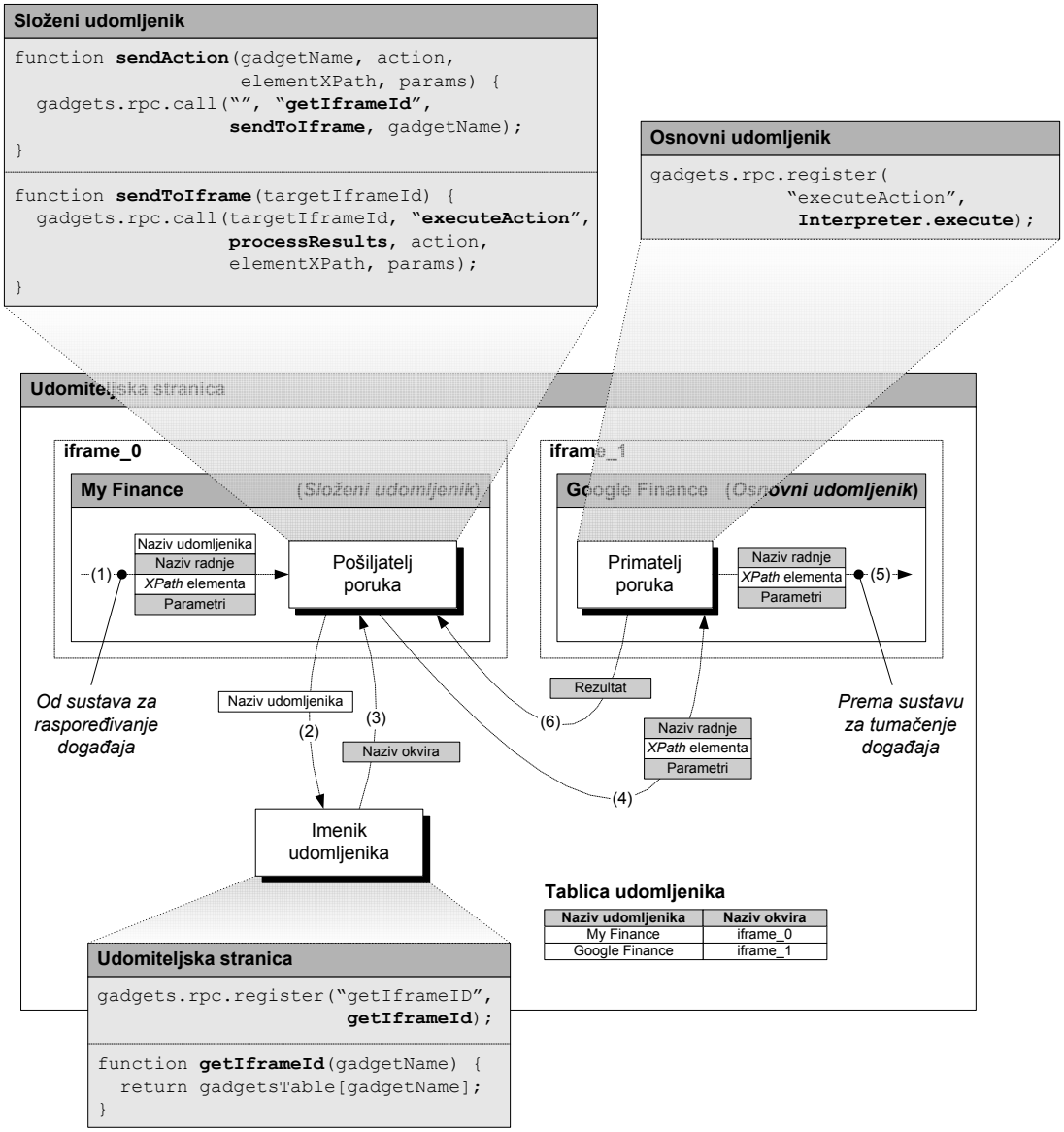
Komunikacijski sustav

Komunikacijski sustav je proširenje izvornog oblika sustava *Apache Shindig* kojim se ostvaruje prosljeđivanje informacija o događajima od složenog udomljenika prema osnovnim udomljenicima kako bi se događaji mogli izvesti nad korisničkim sučeljem osnovnih udomljenika. Dijelovi komunikacijskog sustava ugrađeni su u složeni udomljenik, osnovne udomljenike i udomiteljsku stranicu. Slikom 8.14 prikazano je načelo rada i programsko ostvarenje komunikacijskog sustava.

U okviru sustava *Apache Shindig*, za prikaz udomljenika unutar udomiteljske stranice koriste se stranični okviri. U jeziku *HTML* koji se koristi za ostvarenje udomljenika i udomiteljske stranice, stranični okviri ostvaruju se *iframe* objektima [217]. Svaki udomljenik izvodi se u zasebnom straničnom okviru jedinstvenog naziva unutar udomiteljske stranice. Na slici 8.14 prikazano je izvođenje složenog udomljenika *My Finance* unutar straničnog okvira *iframe_0* i osnovnog udomljenika *Google Finance* unutar straničnog okvira *iframe_1*. Izvođenjem udomljenika u zasebnim straničnim okvirima, postiže se odvojeno izvođenje funkcionalnosti i sadržaja koji pripadaju različitim vlasnicima. U preglednike *World Wide Web* sadržaja ugrađen je mehanizam za sprječavanje pristupa sadržajima i funkcionalnostima unutar straničnog okvira iz ostalih straničnih okvira koji pripadaju različitim vlasnicima (engl. *same origin policy*) [223, 224]. Komunikacija između straničnih okvira moguća je isključivo posredstvom mehanizma za nadzirani pristup sadržaju i funkcionalnostima straničnog okvira.

U sustavu *Apache Shindig*, za nadziranu komunikaciju između straničnih okvira koristi se programska knjižnica *gadgets.rpc* [222]. Komunikacija između udomljenika zasnovana je na nadziranom pozivanju procedura između straničnih okvira. Nadzor komunikacije zasniva se na ograničenom skupu funkcija koje pojedini udomljenici izlažu na korištenje ostalim udomljenicima. Programska knjižnica *gadgets.rpc* iskorištena je za programsko ostvarenje

komunikacijskog sustava za izvođenje potrošačkih programa. Izlaganje funkcionalnosti pozivanog udomljenika pozivajućim udomljenicima izvodi se primjenom funkcije *gadgets.rpc.register*. Pozivanje izložene funkcionalnosti izvodi se primjenom funkcije *gadgets.rpc.call* u pozivajućem udomljeniku.



Slika 8.14. Programsko ostvarenje komunikacijskog sustava

Osnovni udomljenik udomljen u pozivanom okviru koristi funkcijski poziv *gadgets.rpc.register* kako bi složenom udomljeniku u pozivajućem okviru izložio funkciju *Interpreter.execute* kojom se pokreće sustav za tumačenje događaja. Za pozivanje izložene funkcije primjenom funkcijskog poziva *gadgets.rpc.call*, složenom udomljeniku udomljenom u pozivajućem okviru potreban je naziv pozivanog okvira u kojem se izvodi osnovni udomljenik.

Budući da se u slovčanom zapisu događaja pojavljuje informacija o nazivu osnovnog udomljenika, ali ne i informacija o nazivu straničnog okvira unutar kojeg se taj udomljenik izvodi, komunikacija između straničnih okvira odvija se posredstvom udomiteljske stranice. Udomiteljska stranica sadrži tablicu udomljenika u kojoj za svaki učitani udomljenik postoji zapis o nazivu udomljenika i pripadajućem nazivu straničnog okvira u koji je udomljenik učitan. Kako je izvođenje funkcionalnosti straničnih okvira i udomiteljske stranice također međusobno odvojeno, komunikacija između straničnih okvira i udomiteljske stranice također se odvija primjenom mehanizma nadziranog pozivanja javno izloženih procedura.

Komunikacijski sustav sastoji se od tri cjeline. U složeni udomljenik ugrađen je *pošiljatelj poruka* koji putem mehanizma pozivanja javno izloženih procedura straničnih okvira upućuje pozive osnovnom udomljeniku. U udomiteljsku stranicu ugrađen je *imenik udomljenika* koji rukuje tablicom udomljenika i za zadani naziv udomljenika pronalazi naziv straničnog okvira. U osnovni udomljenik ugrađen je *primatelj poruka* koji putem mehanizma javno izloženih procedura straničnih okvira izlaže funkciju za tumačenje događaja.

Na početku rada, osnovni udomljenik i udomiteljska stranica provode postupak izlaganja javno dostupnih funkcija složenom udomljeniku. Osnovni udomljenik koristi funkcijski poziv *gadgets.rpc.register* kako bi složenom udomljeniku izložio funkciju *Interprete.execute* kojom se poziva sustav za tumačenje događaja. S druge strane, udomiteljska stranica koristi funkcijski poziv *gadgets.rpc.register* kako bi složenom udomljeniku izložila funkciju *getIframeId* kojom se iz tablice udomljenika za zadani naziv udomljenika dohvaća naziv straničnog okvira.

Komunikacija složenog i osnovnog udomljenika započinje prihvaćanjem strukture podataka s informacijama o događaju koju pošiljatelju poruka dostavlja sustav za raspoređivanje događaja pozivom funkcije *sendAction* (1). Nakon toga pošiljatelj poruka primjenom funkcijskog poziva *gadgets.rpc.call* na udomiteljskoj stranici poziva funkciju *getIframeId* za dohvat naziva straničnog okvira osnovnog udomljenika. Parametar poziva funkcije je naziv osnovnog udomljenika (2), dok je povratna vrijednost naziv straničnog okvira u koji je udomljenik učitan (3). Osim naziva funkcije *getIframeId* i naziva osnovnog udomljenika, funkciji *gadgets.rpc.call* predaje se i pokazivač na funkciju *sendToIframe* za obradu povratne vrijednosti funkcije *getIframeId*. Pokazivač na funkciju za obradu povratnih rezultata potreban je zbog toga što programska knjižnica *gadgets.rpc* za funkcije koje u pozivajući program vraćaju rezultate izvođenja koristi mehanizam povratnog poziva (engl. *callback*) [221, 222]. Parametar poziva funkcije *sendToIframe* za obradu povratnih vrijednosti je naziv straničnog okvira osnovnog udomljenika koji predstavlja povratnu

vrijednost iz funkcije *getIframeId*. Funkcija *sendToIframe* samostalno se poziva iz programske knjižnice *gadgets.rpc* nakon što završi izvođenje funkcije *getIframeId*.

Nastavak komunikacije složenog i osnovnog udomljenika odvija se u funkciji *sendToIframe* unutar pošiljatelja poruka. Tom funkcijom pošiljatelj poruka poziva sustav za tumačenje događaja osnovnog udomljenika. Osnovnom udomljeniku prosljeđuje se ključna riječ za određivanje vrste događaja, *XPath* izraz za opis putanje do ciljnog elementa i dodatni parametri za one događaje koji koriste prošireni oblik slovčanog zapisa (4). Primatelj poruke prosljeđuje te informacije sustavu za tumačenje događaja (5). Rezultati izvođenja događaja, koji ovise o vrsti događaja, u završnom se koraku vraćaju pošiljatelju poruka u složeni udomljenik (6). Slanje podataka iz složenog u osnovni udomljenik obavlja se funkcijskim pozivom *gadgets.rpc.call* unutar funkcije *sendToIframe* složenog udomljenika. Prihvaćanje rezultata izvođenja događaja obavlja se mehanizmom povratnog poziva funkcije *processResults* čiji se pokazivač koristi u funkcijskom pozivu *gadgets.rpc.call*. Pojedinosti programskog ostvarenja funkcija *Interpreter.execute* i *processResults* podrobnije su opisane u okviru opisa sustava za tumačenje događaja.

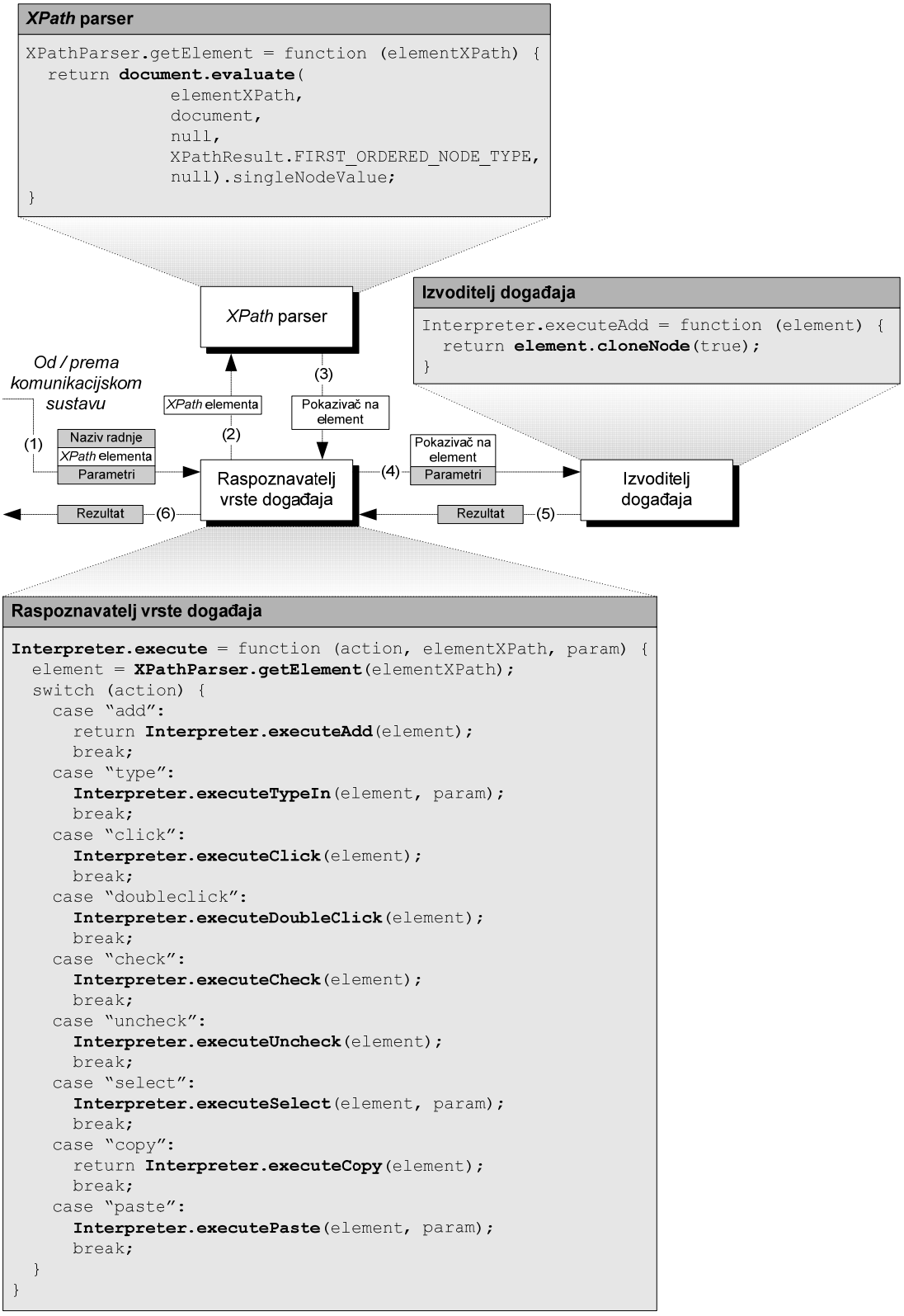
Sustav za tumačenje događaja

Sustav za tumačenje događaja je proširenje izvornog oblika sustava *Apache Shindig* koje je ugrađeno u osnovni udomljenik. Taj dio sustava zadužen je za tumačenje događaja zaprimljenog od složenog udomljenika i njegovo izvođenje nad grafičkim korisničkim sučeljem osnovnog udomljenika. Slikom 8.15 prikazano je načelo rada i programsko ostvarenje sustava za tumačenje događaja.

Sustav za tumačenje događaja sastoji se od *raspoznavatelja vrste događaja*, *XPath parsera* i *izvoditelja događaja*. Raspoznavatelj vrste događaja na osnovi primljene ključne riječi određuje vrstu događaja koju je potrebno izvesti nad elementom grafičkog korisničkog sučelja osnovnog udomljenika. *XPath* parser na osnovi primljenog *XPath* izraza s opisom položaja elementa unutar strukture *HTML* dokumenta osnovnog udomljenika pronalazi naslovljeni element. Izvoditelj događaja izvodi definirani događaj nad pronađenim elementom korisničkog sučelja.

Izvođenje događaja započinje prihvaćanjem strukture podataka koja sadrži ključnu riječ za određivanje vrste događaja, *XPath* izraz za opis putanje do ciljnog elementa i dodatne parametre ako se radi o događaju koji koristi prošireni oblik slovčanog zapisa (1). Ovu strukturu podataka sustavu za izvođenje događaja osnovnog udomljenika putem

komunikacijskog sustava dostavlja sustav za raspoređivanje događaja složenog udomljenika pozivanjem funkcije *Interpreter.execute*.



Slika 8.15. Programsko ostvarenje sustava za tumačenje događaja

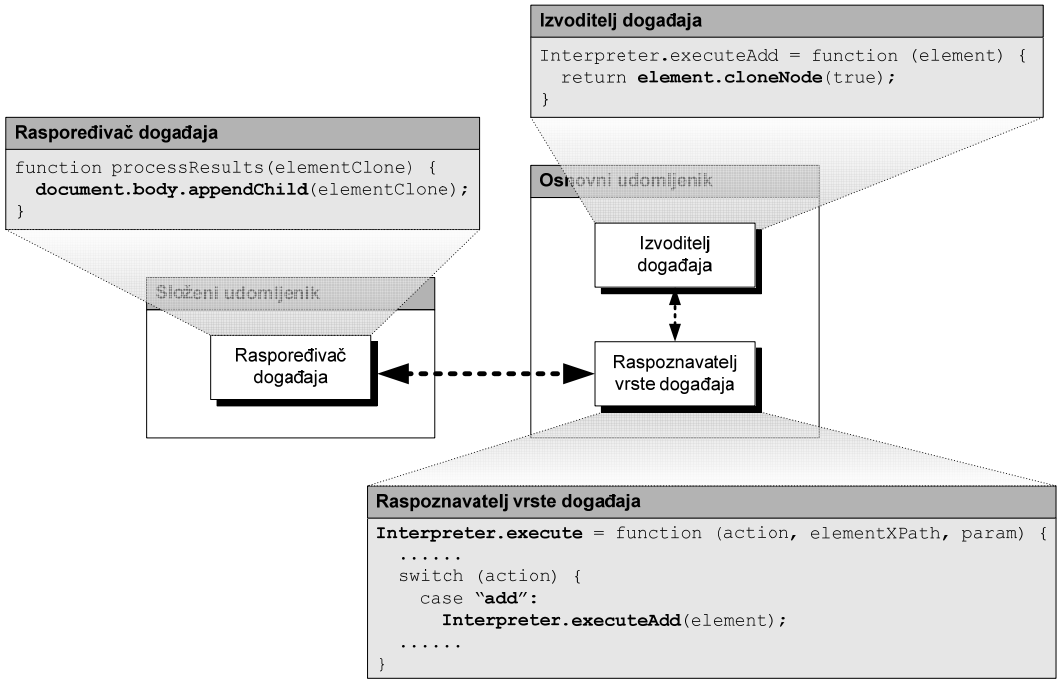
Strukturu podataka prihvaća raspoznavatelj vrste događaja. *XPath* izraz s opisom položaja elementa unutar strukture *HTML* dokumenta osnovnog udomljenika prosljeđuje se *XPath parseru* (2) pozivanjem funkcije *XpathParser.getElement*. Unutar parsera se primjenom funkcije *document.evaluate* za zadani *Xpath* izraz dohvaća pokazivač na traženi element grafičkog korisničkog sučelja osnovnog udomljenika. Funkcija *document.evaluate* dio je standardne programske knjižnice ugrađene u preglednik *World Wide Web* sadržaja kojom se u jeziku *Javascript* programski rukuje *HTML* dokumentima. Pokazivač na pronađeni element korisničkog sučelja se u obliku povratne vrijednosti funkcije *XpathParser.getElement* vraća raspoznavatelju vrste događaja (3).

Nakon pribavljanja pokazivača na element korisničkog sučelja nad kojim je potrebno izvesti primljeni događaj, raspoznavatelj vrste događaja započinje analizu ključne riječi i određuje vrstu događaja. Za obradu pojedinih vrsti događaja od kojih su izgrađeni potrošački programi, izvoditelj događaja sadrži po jednu funkciju. Na osnovi ključne riječi za raspoznavanje vrste događaja, raspoznavatelj vrste događaja poziva odgovarajuću funkciju izvoditelja događaja (4). Tijekom poziva funkcije za izvođenje događaja, izvoditelju događaja prosljeđuje se pokazivač na element nad kojim je potrebno izvesti definirani događaja te dodatne parametre ako su sadržani u zapisu događaja. U završnom koraku postupka, izvoditelj događaja programski izvodi definirani događaj nad naslovljenim elementom grafičkog korisničkog sučelja udomljenika. Rezultati izvođenja događaja vraćaju se raspoznavatelju vrste događaja (5) koji ih putem komunikacijskog sustava prosljeđuje složenom udomljeniku (6).

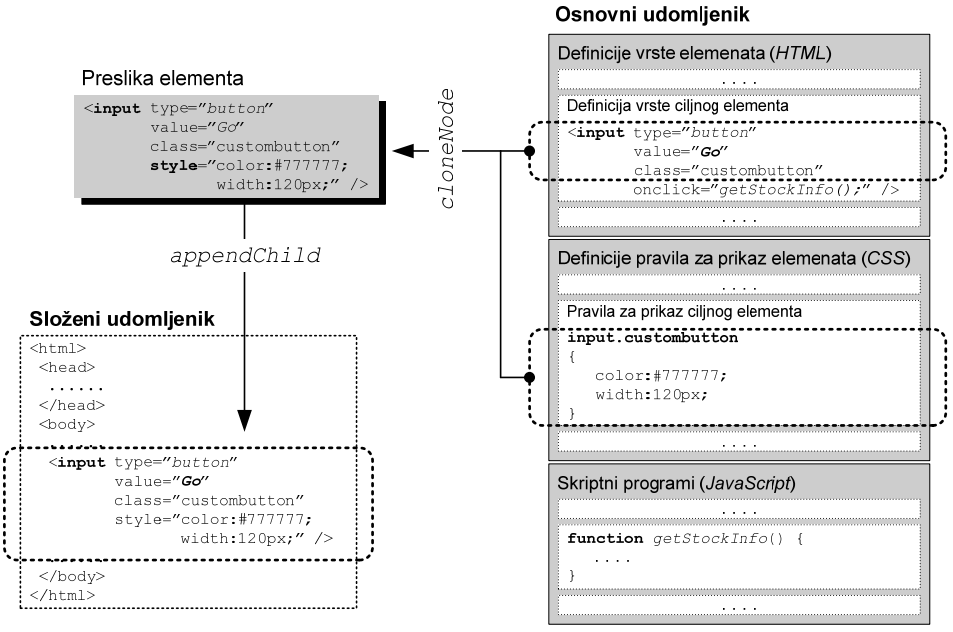
Kao što je prikazano slikom 8.15, osim događaja za preuzimanje elementa korisničkog sučelja s osnovnog u složeni udomljenik koji je označen ključnom riječju *add*, za izgradnju primjenskih programa izgrađenih od strane potrošača koriste se i ostale vrste događaja označene ključnim riječima *type*, *click*, *doubleclick*, *check*, *uncheck*, *select*, *copy* i *paste*. Za pojedinu vrstu događaja, izvoditelj događaja sadrži po jednu funkciju koja ga izvodi. Primjerom na slici 8.15 prikazano je programsko ostvarenje funkcije *Interpreter.executeAdd* za izvođenje događaja za preuzimanje elementa grafičkog korisničkog sučelja s osnovnog u složeni udomljenik. Programska ostvarenja funkcija za izvođenje ostalih vrsta događaja opisana su u poglavljima 9 i 10, usporedno s opisima pojedinih vrsta događaja.

Izvođenje događaja za preuzimanje elementa grafičkog korisničkog sučelja s osnovnog u složeni udomljenik prikazano je slikom 8.16. Slikom 8.16.a prikazan je isječak programskog ostvarenja raspoznavatelja vrste događaja koji je zadužen za prepoznavanje događaja za preuzimanje elementa grafičkog korisničkog sučelja te programsko ostvarenje

izvoditelja događaja i funkcije za obradu rezultata izvođenja. Slikom 8.16.b prikazano je djelovanje korištenih funkcija ugrađenih u preglednik *World Wide Web* sadržaja.



a) Programsko ostvarenje



b) Djelovanje funkcija *cloneNode* i *appendChild*

Slika 8.16. Izvođenje događaja za preuzimanje elementa grafičkog korisničkog sučelja

Događaj za preuzimanje elementa grafičkog korisničkog sučelja s osnovnog u složeni udomljenik raspoznaje se po ključnoj riječi *add*. Nakon prepoznavanja vrste događaja, raspoznavatelj vrste događaja pokreće izvoditelj događaja koji je ostvaren funkcijom

Interpreter.executeAdd. Preuzimanje elementa grafičkog korisničkog sučelja osnovnog udomljenika ostvareno je funkcijskim pozivom *cloneNode* nad programskim objektom kojim je u strukturi *HTML* dokumenta predstavljen ciljni element grafičkog korisničkog sučelja. Funkcijom *cloneNode* stvara se preslika elementa grafičkog korisničkog sučelja koja se u obliku povratne vrijednosti prosljeđuje do složenog udomljenika. Uz definiciju vrste elementa ostvarenu označnim jezikom *HTML*, u presliku elementa ugrađena su i pravila za prikaz elementa na zaslonu računala definirana prikaznim jezikom *CSS*.

Ugradnja preslike preuzetog elementa u strukturu *HTML* dokumenta složenog udomljenika ostvarena je funkcijom *processResults*. Parametar poziva funkcije je preslika elementa dobivena izvođenjem funkcije *cloneNode* nad izvornim elementom korisničkog sučelja osnovnog udomljenika. Unutar funkcije *processResults* koristi se funkcijski poziv *appendChild* kojim se preslika elementa ugrađuje u tijelo *HTML* dokumenta složenog udomljenika. Funkcije *cloneNode* i *appendChild* dio su standardne programske knjižnice ugrađene u preglednik *World Wide Web* sadržaja kojom se u jeziku *Javascript* programski rukuje *HTML* dokumentima.

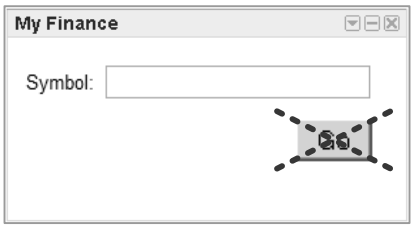
8.2.2 Događaj za uklanjanje preuzetih elemenata korisničkog sučelja

Elemente korisničkog sučelja koji su s osnovnih udomljenika preuzeti u složeni udomljenik ponekad je zbog pogreške u oblikovanju ili zbog ažuriranja složenog udomljenika potrebno ukloniti. Uklanjanje elementa grafičkog korisničkog sučelja osnovnih udomljenika koji su izgradnjom događaja za preuzimanje elemenata korisničkog sučelja obilježeni za preuzimanje u skup elemenata korisničkog sučelja složenog udomljenika obavlja se izgradnjom događaja za uklanjanje preuzetih elemenata korisničkog sučelja.

Prostor oblikovanja

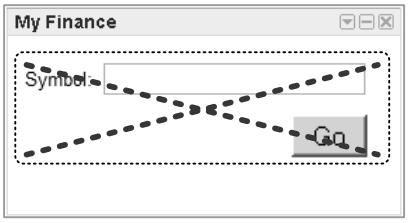
Prostor oblikovanja događaja za uklanjanje preuzetog elementa korisničkog sučelja sastoji se od složenog udomljenika s potpuno ili djelomično izgrađenim korisničkim sučeljem. Događaj za uklanjanje preuzetog elementa korisničkog sučelja oblikuje se tako da se na korisničkom sučelju složenog udomljenika odabere element koji je potrebno ukloniti iz skupa elemenata njegovog korisničkog sučelja. Izabrani element korisničkog sučelja označava se kao suvišan i uklanja se s radne plohe korisničkog sučelja složenog udomljenika. Na mjestu uklonjenog elementa oslobađa se prostor za postavljanje novog elementa korisničkog sučelja.

Slikom 8.17 prikazan je primjer oblikovanja događaja za uklanjanje elementa korisničkog sučelja u obliku tipke *Go* sa složenog udomljenika *My Finance*. Element korisničkog sučelja u obliku tipke *Go* prethodno je preuzet s nekog od osnovnih udomljenika izvođenjem događaja za preuzimanje elementa korisničkog sučelja.



Slika 8.17. Oblikovanje događaja za uklanjanje preuzetog elementa korisničkog sučelja

S obzirom na to da je programskim jezikom za usložnjavanje udomljenika moguća izgradnja događaja za preuzimanje jednostavnih i zbirnih elemenata korisničkog sučelja, osim uklanjanja jednostavnih elemenata korisničkog sučelja, tim programskim jezikom moguće je izgraditi i događaj za uklanjanje zbirnog elementa korisničkog sučelja. Oblikovanje događaja za uklanjanje zbirnog elementa korisničkog sučelja prikazano je slikom 8.18.



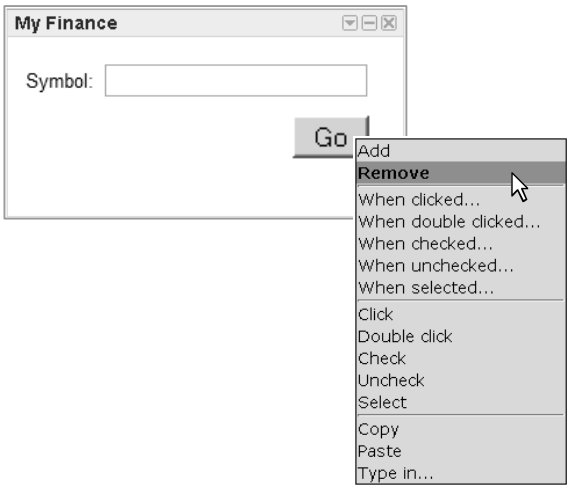
Slika 8.18. Oblikovanje događaja za uklanjanje zbirnog elementa korisničkog sučelja

Događaj za uklanjanje zbirnog elementa korisničkog sučelja oblikuje se tako da se odabire zbirni element koji je zajedno s pripadajućim skupom jednostavnih elemenata potrebno ukloniti s korisničkog sučelja složenog udomljenika. Slikom 8.18 prikazan je primjer oblikovanja događaja za uklanjanje zbirnog elementa korisničkog sučelja u obliku obrasca sa složenog udomljenika pod nazivom *My Finance*. Zbirni element u obliku obrasca sadrži tri jednostavna elementa korisničkog sučelja, kojeg redom čine slovčana oznaka *Symbol*, polje za unos teksta *Symbol* te tipka *Go*.

Prostor izgradnje

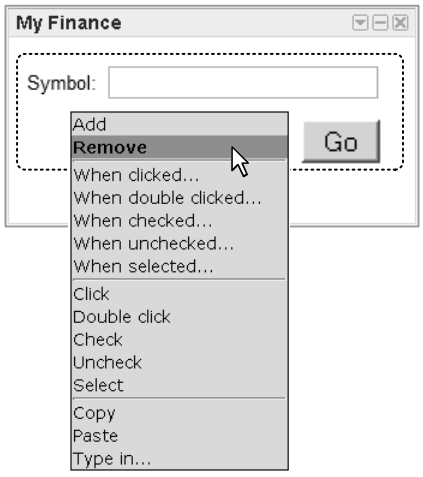
Izgradnja događaja za uklanjanje preuzetog elementa korisničkog sučelja prikazana je slikom 8.19. Pritiskom desne tipke miša na izabrani element korisničkog sučelja složenog

udomljenika, pojavljuje se plivajući izbornik za usložnjavanje udomljenika. Događaj za uklanjanje preuzetog elementa korisničkog sučelja izgrađuje se odabirom stavke *Remove* u plivajućem izborniku. Slikom 8.19 prikazan je postupak izgradnje događaja za uklanjanje tipke *Go* sa složenog udomljenika *My Finance*.



Slika 8.19. Izgradnja događaja za uklanjanje preuzetog elementa korisničkog sučelja

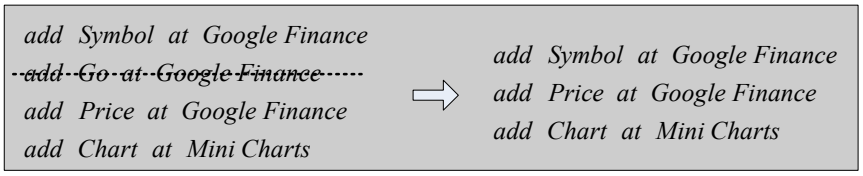
Primjenom sličnog postupka moguće je izgraditi događaj za uklanjanje preuzetog zbirnog elementa korisničkog sučelja. Slikom 8.20 prikazan je postupak izgradnje događaja za uklanjanje zbirnog elementa u obliku obrasca sa složenog udomljenika pod nazivom *My Finance*. Zbirni element u obliku obrasca sastoji se od slovčane oznake *Symbol*, polja za unos teksta *Symbol* te tipke *Go*. Za uklanjanje zbirnog elementa, plivajući izbornik potrebno je pokrenuti u točki koja se nalazi unutar područja prostiranja zbirnog elementa, ali istodobno ne pripada niti jednom jednostavnom elementu unutar zbirnog elementa.



Slika 8.20. Izgradnja događaja za uklanjanje preuzetog zbirnog elementa korisničkog sučelja u programskom jeziku za usložnjavanje udomljenika

Prostor prikaza

Događaj za uklanjanje preuzetog elementa korisničkog sučelja nema posebno definiran način prikaza. Razlog nepostojanja posebnog oblika prikaza ove vrste događaja je način zapisa korisničkog sučelja složenog udomljenika koji je opisan u poglavlju 8.2.1. Korisničko sučelje složenog udomljenika definirano je popisom elemenata korisničkog sučelja za preuzimanje s osnovnih udomljenika u složeni udomljenik. Elementi s tog popisa preuzimaju se tijekom učitavanja složenog udomljenika u udomiteljsku stranicu. Kako bi se izbjeglo preuzimanje uklonjenog elementa korisničkog sučelja tijekom sljedećeg učitavanja složenog udomljenika, s popisa elemenata za preuzimanje dovoljno je ukloniti slovčani zapis događaja za preuzimanje uklonjenog elementa.



Slika 8.21. Uklanjanje događaja za preuzimanje elementa korisničkog sučelja s popisa događaja za preuzimanje elemenata korisničkog sučelja

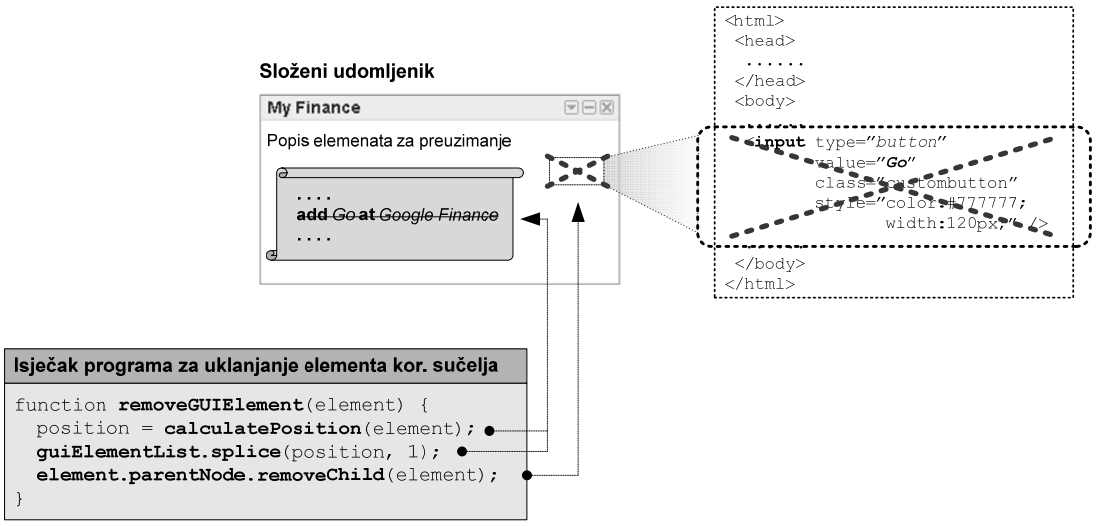
Primjer ažuriranja popisa događaja za preuzimanje elemenata korisničkog sučelja prikazan je slikom 8.21. Primjerom je prikazana promjena popisa elemenata korisničkog sučelja za preuzimanje u složeni udomljenik nakon izgradnje događaja za uklanjanje elementa *Go* koji je u korisničko sučelje složenog udomljenika prethodno preuzet s korisničkog sučelja osnovnog udomljenika *Google Finance*.

Prostor izvođenja

Brisanjem događaja za preuzimanje uklonjenog elementa grafičkog korisničkog sučelja s popisa elemenata za preuzimanje, sprječava se preuzimanje uklonjenog elementa tijekom sljedećeg učitavanja složenog udomljenika u udomiteljsku stranicu. Međutim, kako bi promjene izgleda radne plohe korisničkog sučelja složenog udomljenika odmah postale vidljive potrošaču, iz programske strukture složenog udomljenika uklanjanju se dijelovi zaduženi za prikaz elementa na zaslonu računala.

Prostor izvođenja događaja za uklanjanje elementa grafičkog korisničkog sučelja prikazan je slikom 8.22. Na slici je istaknut isječak programskog kôda u jeziku *Javascript* kojim je ostvareno izvođenje događaja za uklanjanje elementa grafičkog korisničkog sučelja s radne plohe složenog udomljenika. Izborom stavke *Remove* u plivajućem izborniku za

izgradnju programa, poziva se funkcija *removeGUIElement*. Parametar poziva funkcije je pokazivač na element korisničkog sučelja koji je primjenom plivajućeg izbornika označen za uklanjanje s radne plohe složenog udomljenika.



Slika 8.22. Prostor izvođenja događaja za uklanjanje elementa grafičkog korisničkog sučelja

Izvođenje funkcije započinje pronalaskom zapisa događaja na popisu događaja za preuzimanje elemenata korisničkog sučelja. Popis događaja ostvaren je u obliku memorijskog polja. Brisanje događaja s popisa izvodi se primjenom funkcije *splice* koja se u jeziku *Javascript* koristi za rukovanje sadržajem polja. Nakon brisanja događaja za preuzimanje elementa s popisa elemenata za preuzimanje, iz programske strukture složenog udomljenika uklanja se definicija elementa zapisana označnim jezikom *HTML*. Brisanje elementa iz strukture *HTML* dokumenta složenog udomljenika izvodi se pozivom funkcije *removeChild* koja je dio standardne programske knjižnice ugrađene u preglednik *World Wide Web* sadržaja kojom se u jeziku *Javascript* programski rukuje *HTML* dokumentima.

9 Upravljanje radom udomljenika

Oblikovanje i izgradnja radnog tijeka nad skupom udomljenika primjenom programske paradigme za usložnjavanje udomljenika postiže se oblikovanjem logike za programsko upravljanje izvođenjem pojedinačnih udomljenika te logike za uspostavu toka podataka kojom se podaci prenose između različitih udomljenika. U ovom poglavlju opisuje se oblikovanje i izgradnja logike za programsko upravljanje izvođenjem pojedinačnih udomljenika, dok su oblikovanje i izgradnja logike za uspostavu toka podataka opisani u poglavlju 10.

Programsko upravljanje izvođenjem udomljenika ostvaruje se posebno oblikovanim *programskim jezikom za usložnjavanje udomljenika* kojim se oponaša uzajamno djelovanje potrošača i računala putem grafičkog korisničkog sučelja udomljenika. Programski jezik sastoji se od radnji potrošača nad elementima grafičkog korisničkog sučelja. Elementima jezika omogućuje se programsko upisivanje sadržaja u ulazne elemente grafičkog korisničkog sučelja udomljenika i pokretanje izvođenja operacija nad udomljenicima putem aktivacijskih elemenata. Primjerice, tijekom oblikovanja radnog tijeka nad skupom udomljenika, potrošač definira slijed radnji kao što su upisivanje teksta u polje za unos teksta, označavanje označnog polja, izbor stavke iz padajućeg izbornika i pritisak na tipku ili poveznicu kojima se primjenom izabranog skupa udomljenika rješava postavljeni problem. Obrazloženje izbora programskog jezika zasnovanog na radnjama potrošača nad elementima grafičkog korisničkog sučelja udomljenika dano je u poglavlju 7, gdje je pokazano da je takav oblik programskog jezika kognitivno i iskustveno blizak umnom režimu širokog kruga potrošača računala.

U poglavlju 9.1 opisana su svojstva operacija nad podacima na razini grafičkog korisničkog sučelja udomljenika. Pokazano je da je djelovanje bilo koje operacije moguće rastaviti na osnovno i pozadinsko djelovanje. Dok osnovno djelovanje operacije ovisi isključivo o vrsti operacije i vrsti elementa korisničkog sučelja nad kojim se operacija provodi, pozadinsko djelovanje operacije ovisi i o primjenskoj ulozi elementa u promatranom udomljeniku. U poglavlju 9.2 pokazana je metodologija oblikovanja programskog jezika za upravljanje izvođenjem udomljenika razmatranjem isključivo osnovnog djelovanja operacija. Time je omogućeno oblikovanje programskog jezika koji se sastoji od konačnog broja radnji nad elementima korisničkog sučelja, neovisnih o primjenskim svojstvima pojedinih udomljenika. Poglavlje 9.3 donosi opis skupa događaja za programsko upravljanje radom udomljenika koji čine programski jezik za usložnjavanje udomljenika.

9.1 Operacije nad podacima na razini korisničkog sučelja

U poglavlju 7.2.3 pokazana su svojstva programirljivosti grafičkih korisničkih sučelja i usporedba s istovjetnim elementima programskih sučelja. Podaci kojima se rukuje tijekom primjene programske paradigme za usložnjavanje udomljenika pojavljuju se u obliku informacija dostupnih putem grafičkog korisničkog sučelja udomljenika. Takvi podaci se, stoga, nazivaju *podacima predodžbene razine*. Operacije nad podacima pojavljuju se u obliku radnji nad elementima grafičkog korisničkog sučelja udomljenika. S obzirom na to da su pojedinim vrstama elemenata korisničkog sučelja pridružene različite vrste podataka, skup primjenjivih operacija nad podatkom ovisi o pridruženoj vrsti elementa korisničkog sučelja. Tablicom 9.1 prikazani su primjeri najčešće korištenih vrsta elemenata korisničkog sučelja i skup operacija nad podacima pridruženih pojedinim vrstama elemenata.

Tablica 9.1. Primjeri operacija nad podacima na razini grafičkog korisničkog sučelja udomljenika

Element korisničkog sučelja	Operacija nad podatkom
Polje za unos teksta	Upisivanje sadržaja
Tipka	Jednostruki pritisak Dvostruki pritisak
Poveznica	Jednostruki pritisak
Označno polje	Označavanje Odznačavanje
Izborno polje	Obilježavanje
Padajući izbornik	Odabir jedne od ponuđenih stavki

U slučaju ručnog korištenja skupa udomljenika, radni tijek izvodi potrošač uzajamnim djelovanjem putem grafičkog korisničkog sučelja primjenom ulazno-izlaznih naprava računala, poput tipkovnice, miša i zaslona. S druge strane, za programsko izvođenje radnog tijeka potrebno je izgraditi logiku za programsko rukovanje podacima u sustavu povezanih udomljenika. Kako bi postupak izgradnje programske logike bio blizak znanjima i vještinama širokog kruga potrošača, operacije za programsko rukovanje udomljenicima nasljeđuju svojstva uzajamnog djelovanja potrošača i udomljenika putem grafičkog korisničkog sučelja primjenom ulazno-izlaznih naprava računala.

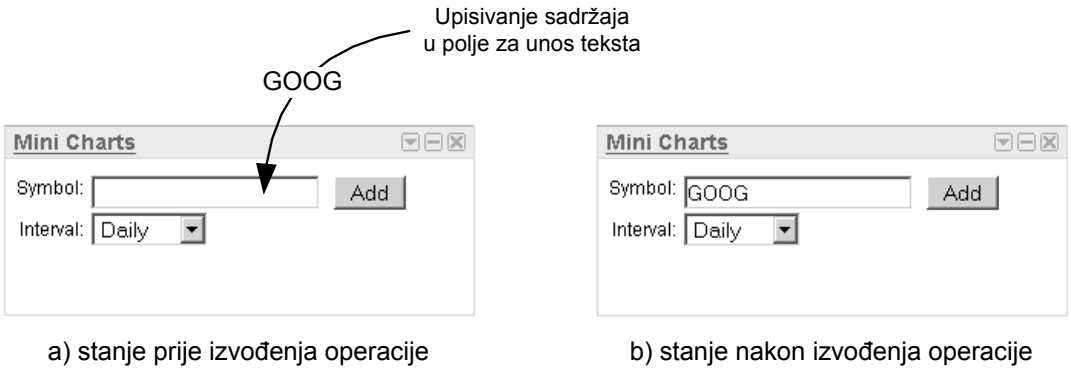
Slika 9.1. Model upravljanja izvođenjem računalnih procesa međudjelovanjem putem grafičkog korisničkog sučelja udomljenika

Slikom 9.1 prikazan je potrošaču blizak model upravljanja izvođenjem računalnih procesa na lokalnom računalu ili u mreži Internet međudjelovanjem putem grafičkog korisničkog sučelja udomljenika. Ponašanje korisnika tijekom upravljanja izvođenjem pozadinskih procesa moguće je modelirati postavljanjem dvaju ključnih pitanja: *što* je potrebno postići izvođenjem procesa i *kako* je to moguće ostvariti. Odgovorom na pitanje *što*, koji proizlazi iz poznavanja značenja računalnog procesa, donosi se odluka o odabiru udomljenika koji se koristi za obavljanje pojedinih dijelova radnog tijeka. Primjerice, za pretvorbu novčane vrijednosti između dviju zadanih valuta potrebno je odabrati udomljenik za pretvorbu valuta. Odgovor na pitanje *kako* ovisi o mogućnostima utjecaja potrošača na izvođenje računalnog procesa. Ako je računalni proces potrošaču izložen u obliku grafičkog korisničkog sučelja udomljenika, onda je zamisli koje proizlaze iz umne aktivnosti potrošača, a posljedica su značenja pozadinskog procesa, moguće ostvariti fizičkim djelovanjem u obliku izvođenja radnji nad elementima grafičkog korisničkog sučelja pridruženog udomljenika. Primjerice, pretvorbu valuta moguće je obaviti upisivanjem brojčane vrijednosti u polje za unos teksta, izborom polazišne i odredišne valute iz padajućeg izbornika te pritiskom na tipku za pokretanje pretvorbe. Unutarnjom povezanošću

korisničkog sučelja udomljenika i pozadinskog procesa postiže se očekivani odziv udomljenika na radnje potrošača i željeni učinak izvođenja procesa.

9.1.1 Osnovno i pozadinsko djelovanje operacija

Djelovanje operacija nad podacima sadržanim u elementima grafičkog korisničkog sučelja udomljenika moguće je razložiti na *osnovno djelovanje* i *pozadinsko djelovanje*. *Osnovno djelovanje operacije* je predvidivo djelovanje čiji učinak ovisi isključivo o vrsti elementa korisničkog sučelja nad kojim se operacija provodi. *Pozadinsko djelovanje operacije* je popratni učinak osnovnog djelovanja operacije koji, osim o vrsti elementa korisničkog sučelja, ovisi i o ulozi elementa u udomljeniku nad kojim se operacija provodi. S obzirom na to da ovisi isključivo o vrsti elementa korisničkog sučelja, osnovno djelovanje bilo koje operacije nad podacima predodžbene razine je unaprijed poznato i neovisno o udomljeniku nad kojim se izvodi operacija. S druge strane, za određivanje učinka pozadinskog djelovanja operacije potrebno je poznavati odziv udomljenika na izvršenu operaciju, odnosno povezanost elementa nad kojim se provodi operacija sa značenjem pozadinskog računalnog procesa.

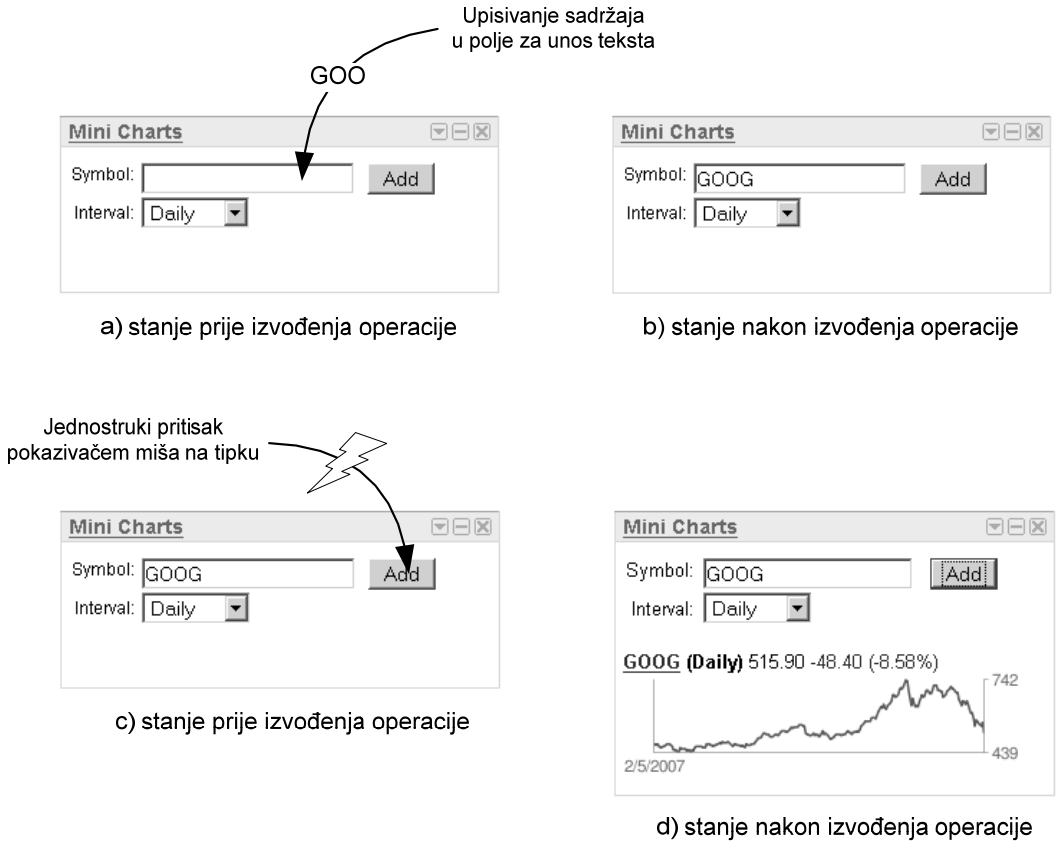


a) stanje prije izvođenja operacije b) stanje nakon izvođenja operacije

Slika 9.2. Rezultat izvođenja operacije s osnovnim djelovanjem nad podacima predodžbene razine

Slikom 9.2 prikazan je primjer izvođenja operacije upisivanja sadržaja u polje za unos teksta. Neovisno o ulozi polja za unos teksta u promatranom udomljeniku, očekivani rezultat izvođenja operacije upisivanja sadržaja u polje za unos teksta jest postavljanje sadržaja polja za unos teksta na upisanu vrijednost. Time je definirano osnovno djelovanje operacije. Osnovnim djelovanjem operacije upisivanja sadržaja u polje za unos teksta, stanja svih ostalih elemenata grafičkog korisničkog sučelja udomljenika ostaju nepromijenjena. Operacija upisivanja sadržaja u polje za unos teksta u prikazanom primjeru nema pozadinsko djelovanje. Ukupni rezultat djelovanja operacije upisivanja sadržaja u polje za unos teksta je, stoga, jednak učinku osnovnog djelovanja operacije.

Za razliku od primjera na slici 9.2, slikom 9.3 prikazani su primjeri izvođenja dviju operacija nad elementima grafičkog korisničkog sučelja udomljenika koje, osim osnovnog, imaju i pozadinsko djelovanje. Gornji primjer prikazuje operaciju upisivanja sadržaja u polje za unos teksta. Osnovno djelovanje te operacije jednako je osnovnom djelovanju operacije prikazane u primjeru na slici 9.2. U ovom primjeru operacija upisivanja sadržaja u polje za unos teksta dodatno je obogaćena pozadinskim djelovanjem koje se očituje u automatskom dovršavanju niza znakova koji se upisuje u polje za unos teksta. Na osnovi prefiksa niza znakova koji je unesen u polje za unos teksta, pozadinska logika udomljenika sposobna je samostalno dovršiti djelomično uneseni niz znakova. Poznavajući vrstu elementa nad kojim se operacija provodi, nije moguće utvrditi stanje podataka zapisanih u elementima grafičkog korisničkog sučelja udomljenika nakon završetka operacije. Za potpuno razumijevanje učinka operacije potrebno je poznavati pozadinski odziv elementa korisničkog sučelja u promatranom udomljeniku na izvedenu operaciju.



Slika 9.3. Rezultat izvođenja operacije s osnovnim i pozadinskim djelovanjem nad podacima predodžbene razine

Donji primjer prikazan na slici 9.3 prikazuje učinak izvođenja operacije pritiska na tipku. Operacija pritiska na tipku tipični je primjer operacije s pozadinskim djelovanjem.

Poznavanje vrste elementa, koji je u promatranom slučaju tipka, te vrste operacije, koja je u promatranom slučaju pritisak na tipku, nije dovoljno za utvrđivanje odziva udomljenika na izvođenje operacije. Za utvrđivanje odziva udomljenika potrebno je poznavati ulogu tipke u promatranom udomljeniku, odnosno značenje pozadinskog procesa pridruženog udomljeniku. Pod pretpostavkom da se pritiskom na tipku preuzima sadržaj upisan u polje za unos teksta i stavka odabrana iz padajućeg izbornika te se na osnovi tih podataka izračunava sadržaj poveznice, odlomka za prikaz teksta i slike, rezultat izvođenja operacije prikazan je slikom 9.3.d. Vidljivi učinak operacije pritiska na tipku jednak je pozadinskom djelovanju operacije, dok osnovno djelovanje operacije ne ostavlja vidljivi učinak. Osnovno djelovanje operacije pritiska na tipku predstavlja aktivnost pritiska na tipku, kojom se pokreće pozadinsko djelovanje operacije.

Djelovanje operacija nad podacima predodžbene razine matematički se opisuje formulom (9.1) kao kompozicija osnovnog i pozadinskog djelovanja

$$O = O_{PD} \circ O_{OD} \tag{9.1}$$

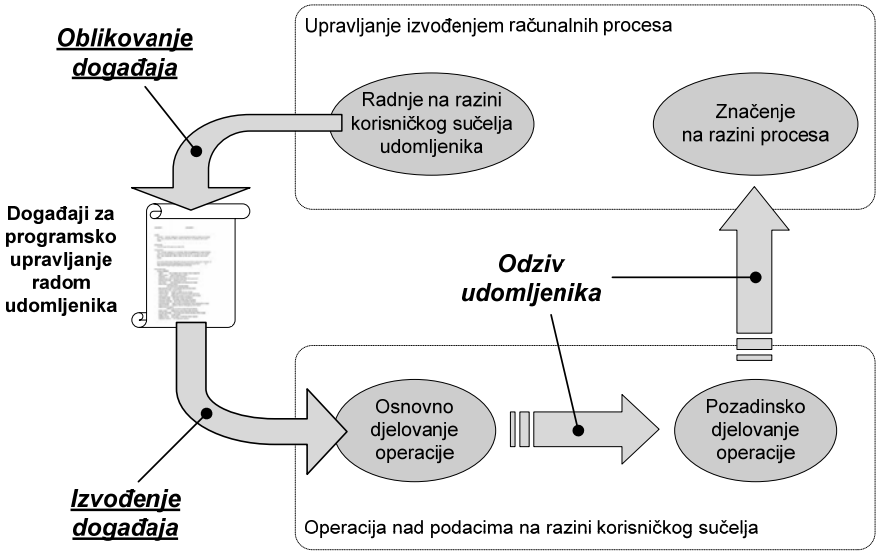
gdje je O_{OD} funkcija kojom se opisuje osnovno djelovanje operacije, dok je O_{PD} funkcija kojom se opisuje pozadinsko djelovanje operacije. U skladu s matematičkom definicijom kompozicije funkcija, formulom (9.1) određen je redoslijed primjene funkcija za osnovno i pozadinsko djelovanje operacije. U prvom koraku nastupa osnovno djelovanje operacije, a na rezultat osnovnog djelovanja primjenjuje se pozadinsko djelovanje.

9.2 Oblikovanje događaja za upravljanje radom udomljenika

Mogućnost rastavljanja djelovanja bilo koje operacije nad podacima predodžbene razine na osnovno i pozadinsko djelovanje koristi se tijekom oblikovanja jezika za programsko upravljanje izvođenjem udomljenika. Ovim svojstvom omogućeno je oblikovanje *općenitog programskog jezika za usložnjavanje udomljenika* koji je primjenjiv za povezivanje bilo koje kombinacije osnovnih udomljenika u složeni udomljenik, a sastoji se od *konačnog skupa radnji nad elementima grafičkog korisničkog sučelja*.

Metodologija oblikovanja programskog jezika prikazana je slikom 9.4. Programski jezik sastoji se od konačnog skupa događaja za programsko upravljanje izvođenjem udomljenika. *Oblikovanje događaja* izvodi se iz radnji koje potrošač poduzima tijekom međudjelovanja s udomljenicima putem grafičkog korisničkog sučelja. Za svaku radnju koju je primjenom ulazno-izlaznih naprava računala moguće izvesti nad elementima grafičkog

korisničkog sučelja udomljenika oblikuje se po jedan događaj programskog jezika. U okviru predloženog ostvarenja programske paradigme prilagođene potrošaču, oblikovani su događaji za programsko izvođenje jednostrukog i dvostrukog pritiska pokazivačem miša na aktivacijske elemente korisničkog sučelja u obliku tipki i poveznica, upisivanje sadržaja u polje za unos teksta, označavanje i odznačavanje označnog polja, obilježavanje izbornog polja te izbor stavke iz padajućeg izbornika.



Slika 9.4. Metodologija oblikovanja programskog jezika za usložnjavanje udomljenika

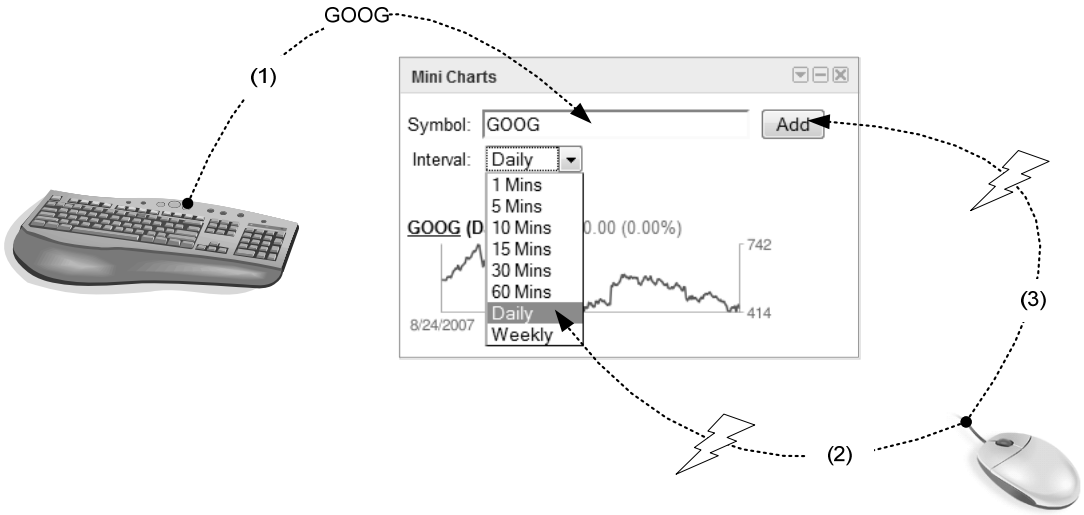
Tijekom *izvođenja događaja* dovoljno je pokrenuti izvođenje osnovnog djelovanja operacije koje je moguće odrediti na osnovi vrste operacije i vrste elementa korisničkog sučelja nad kojim se operacija primjenjuje. Nakon izvođenja osnovnog djelovanja operacije, ciljni udomljenik automatski se odaziva pokretanjem pozadinskog djelovanja operacije koje ovisi o značenju pozadinskog procesa. S obzirom da se pozadinsko djelovanje operacije koje ovisi o značenju pozadinskog procesa pokreće automatskim odzivom udomljenika, za izvođenje događaja nije potrebno poznavati ulogu pojedinih elemenata korisničkog sučelja u promatranom udomljeniku niti značenje izvođenja operacije nad pridruženim pozadinskim procesom. Ovo svojstvo omogućuje izgradnju jezičnog procesora za tumačenje i izvođenje događaja koji je neovisan o značenju pozadinskih procesa pridruženih udomljenicima, već isključivo o vrsti operacije i vrsti elementa korisničkog sučelja. S obzirom na konačni broj različitih vrsta elemenata grafičkog korisničkog sučelja i konačni broj operacija koje je moguće izvoditi nad pojedinom vrstom elemenata, na taj je način programski jezik za usložnjavanje udomljenika moguće oblikovati od konačnog skupa događaja.

9.3 Događaji za upravljanje radom udomljenika

Programska logika za upravljanje izvođenjem udomljenika oblikuje se od niza događaja za programsko oponašanje radnji potrošača tijekom uzajamnog djelovanja putem grafičkog korisničkog sučelja udomljenika primjenom ulazno-izlaznih naprava računala, kao što su tipkovnica, miš i zaslon.

Prostor oblikovanja

Slikom 9.5 prikazan je postupak oblikovanja programske logike za upravljanje izvođenjem udomljenika. U prikazanom primjeru pretpostavlja se da potrošač dio radnog tijeka primjenskog programa ostvaruje primjenom udomljenika *Mini Charts* izvođenjem triju akcija nad njegovim korisničkim sučeljem. Primjenom tipkovnice, u polje za unos teksta *Symbol* unosi se slovčana oznaka dionice kojom se trguje na burzi (1). Pritiskom pokazivačem miša na odgovarajuću stavku padajućeg izbornika *Interval* odabire se vremensko razdoblje za prikaz kretanja cijene dionice (2). Naposljetku, pritiskom pokazivačem miša na tipku *Add*, pokreće se operacija dohvata grafa s podacima o kretanju cijene dionice (3).

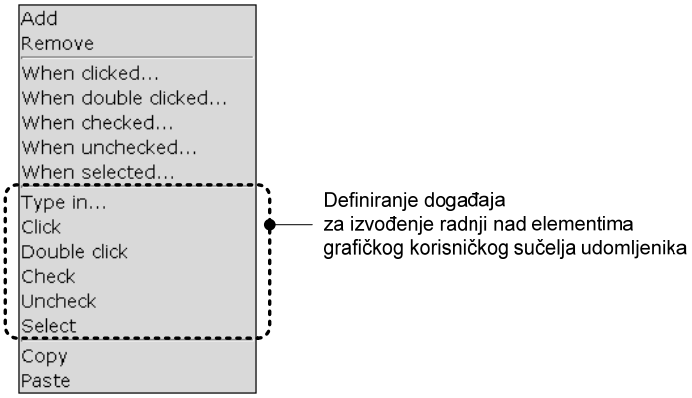


Slika 9.5. Oblikovanje programske logike za upravljanje radom udomljenika

U okviru predloženog ostvarenja programske paradigme prilagođene potrošaču, oblikovani su događaji za programsko izvođenje jednostrukog i dvostrukog pritiska pokazivačem miša na aktivacijske elemente korisničkog sučelja u obliku tipki i poveznica, upisivanje sadržaja u polje za unos teksta, označavanje i odznačavanje označnog polja, obilježavanje izbornog polja te izbor stavke iz padajućeg izbornika.

Prostor izgradnje

Niz događaja za programsko upravljanje radom udomljenika definira se primjenom plivajućeg izbornika, kao što je prikazano slikom 9.6. Plivajući izbornik sadrži šest stavki za izgradnju događaja za programsko izvođenje radnji nad elementima grafičkog korisničkog sučelja udomljenika.



Slika 9.6. Primjena grafičkog izbornika za izgradnju događaja za izvođenje radnji nad elementima grafičkog korisničkog sučelja udomljenika

Stavka *Type in* koristi se za izgradnju događaja za programsko upisivanje sadržaja u polje za unos teksta. Stavka *Click* koristi se za izgradnju događaja za programsko oponašanje jednostrukog pritiska pokazivačem mišem na aktivacijski element korisničkog sučelja. Stavka *Double click* koristi se za izgradnju događaja za programsko oponašanje dvostrukog pritiska pokazivačem mišem na aktivacijski element korisničkog sučelja. Stavke *Check* i *Uncheck* koriste se za izgradnju događaja za programsko upravljanje radom označnog polja. Stavka *Check* koristi se za označavanje, dok se stavka *Uncheck* koristi za odznačavanje označnog polja. Stavka *Select* koristi se za izgradnju događaja za programsko obilježavanje izbornog polja i odabir stavke iz padajućeg izbornika.

Prostor prikaza

Događaji za programsko upravljanje izvođenjem udomljenika koji su izgrađeni u prostoru izgradnje zapisuju se unutar složenog udomljenika u obliku slovčanih zapisa. Primjerice, program za upravljanje izvođenjem udomljenika *Mini Charts* prema scenariju opisanom na slici 9.5 sastoji se od tri događaja koji su prikazani u obliku

type "GOOG" into Symbol at Mini Charts
select "Daily" from Interval at Mini Charts
click Add at Mini Charts

Prvim slovčanim zapisom prikazan je događaj za upisivanje niza znakova *GOOG* u polje za unos teksta *Symbol* na udomljeniku *Mini Charts*. Slovčani zapis ovog događaja izgrađen je primjenom stavke *Type in* iz plivajućeg izbornika. Drugim slovčanim zapisom prikazan je događaj za odabir stavke *Daily* iz padajućeg izbornika *Interval* na udomljeniku *Mini Charts*. Slovčani zapis ovog događaja izgrađen je primjenom stavke *Select* iz plivajućeg izbornika. Konačno, trećim slovčanim zapisom prikazan je događaj za programsko izvođenje jednostrukog pritiska pokazivačem miša na tipku *Add* na udomljeniku *Mini Charts*. Slovčani zapis ovog događaja izgrađen je primjenom stavke *Click* iz plivajućeg izbornika.

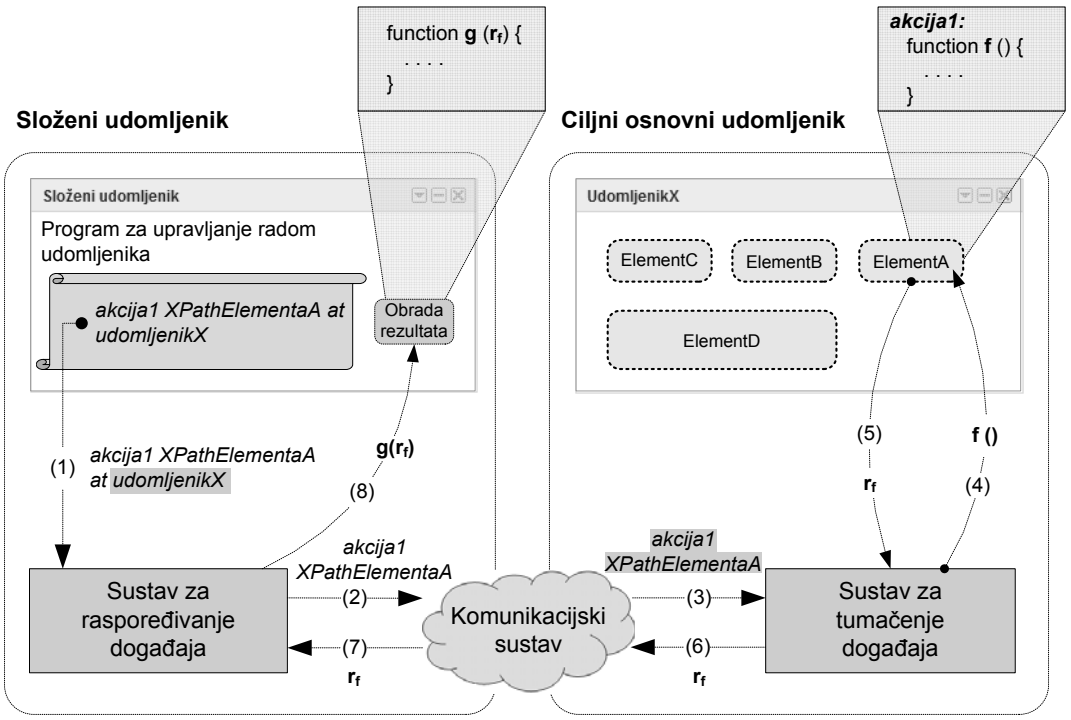
Prostor izvođenja

Prostor izvođenja događaja za programsko upravljanje radom udomljenika sastoji se od osnovnog i složenog udomljenika, kao što je prikazano na slici 9.7. Složeni udomljenik sadrži program za upravljanje radom udomljenika koji se sastoji od skupa slovčanih zapisa događaja. Skup događaja izgrađen je primjenom plivajućeg izbornika u prostoru izgradnje događaja, a unutar složenog udomljenika zapisan je slično kao što je prikazan u prostoru prikaza događaja. U zapisu događaja, simboličko ime ciljnog elementa grafičkog korisničkog sučelja zamijenjeno je *XPath* izrazom kojim je određen položaj elementa unutar strukture *HTML* dokumenta. Osnovni udomljenik sadrži grafičko korisničko sučelje s ciljnim elementom za izvođenje događaja.

Osim slovčanog zapisa programa i grafičkog korisničkog sučelja, osnovni i složeni udomljenik sadrže i programsku infrastrukturu za izvođenje događaja. Složeni udomljenik sadrži sustav za raspoređivanje događaja koji je zadužen je za pronalaženje ciljnog udomljenika nad kojim se obavlja radnja definirana događajem. Osnovni udomljenik sadrži sustav za tumačenje događaja koji je zadužen za programsko pronalaženje naslovljenog elementa korisničkog sučelja i izvođenje definirane radnje. Osnovni i složeni udomljenik povezani su komunikacijskim sustavom. Pojedinosti načela rada i programskog ostvarenja sustava za raspoređivanje događaja, sustava za tumačenje događaja i komunikacijskog sustava opisane su u poglavlju 8.

Izvođenje događaja odvija se u osam koraka. Sustav za raspoređivanje događaja dohvaća slovčani zapis događaja (1). Na osnovi naziva ciljnog udomljenika, zapis događaja prosljeđuje se komunikacijskim sustavom do odredišnog udomljenika (2, 3), gdje ga prihvaća sustav za tumačenje događaja. Na osnovi ključne riječi kojom je određena vrsta radnje koju je potrebno izvršiti nad elementom korisničkog sučelja osnovnog udomljenika,

sustav za tumačenje događaja donosi zaključak o vrsti događaja. Nakon utvrđivanja vrste događaja, sustav za tumačenje događaja parsira *XPath* izraz kojim je određen položaj elementa unutar strukture *HTML* dokumenta osnovnog udomljenika. Pretraživanjem strukture korisničkog sučelja osnovnog udomljenika, sustav za tumačenje događaja pronalazi ciljni element korisničkog sučelja nad kojim je potrebno izvršiti definiranu radnju.



f () Poziv funkcije za oponašanje radnje potrošača nad elementom grafičkog korisničkog sučelja

r_f Povratna vrijednost funcije f()

g (r_f) Poziv funkcije za obradu rezultata izvođenja događaja

Slika 9.7. Prostor izvođenja događaja za programsko upravljanje radom udomljenika

Izvođenje događaja postiže se programskim oponašanjem radnje potrošača nad ciljnim elementom grafičkog korisničkog sučelja osnovnog udomljenika. Svaki element grafičkog korisničkog sučelja udomljenika sadrži programsko sučelje za jezik *Javascript* kojim je moguće programski oponašati radnje potrošača. Pojedinim radnjama potrošača pridružene su funkcije za programsko oponašanje radnji nad elementima. Na osnovi vrste radnje i elementa korisničkog sučelja definiranog u zapisu događaja, sustav za tumačenje događaja poziva odgovarajuću funkciju iz programskog sučelja elementa (4). Na slici 9.7, poziv funkcije za oponašanje radnje potrošača nad elementom grafičkog korisničkog sučelja udomljenika simbolički je označen oznakom *f()*. Povratna vrijednost funkcije *f()*, označena oznakom r_f, vraća se putem komunikacijskog sustava u složeni udomljenik (5, 6, 7), gdje se koristi

tijekom poziva funkcije za obradu rezultata izvođenja događaja (8). Poziv funkcije za obradu rezultata izvođenja događaja simbolički je označen oznakom $g(r_f)$. Točni nazivi i programsko ostvarenje funkcija *f* i *g* ovise o vrsti događaja. U nastavku poglavlja prikazan je skup događaja za programsko upravljanje radom udomljenika te programsko ostvarenje tih dviju funkcija za pojedine vrste događaja.

9.3.1 Upisivanje sadržaja u polje za unos teksta

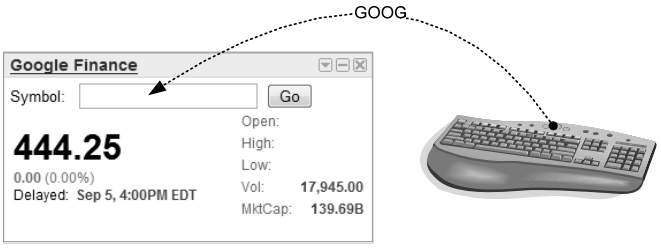
Polje za unos teksta pripada skupini ulaznih elemenata grafičkog korisničkog sučelja. Potrošaču su najčešće predočeno u obliku pravokutnog omeđenog područja unutar kojeg je primjenom tipkovnice moguće upisati proizvoljni niz znakova. Slikom 9.8 prikazan je primjer udomljenika s istaknutim elementom grafičkog korisničkog sučelja u obliku polja za unos teksta. U istaknuto polje za unos teksta upisan je znakovni niz *GOOG*.

Slika 9.8. Element grafičkog korisničkog sučelja udomljenika u obliku polja za unos teksta

Programski jezik za usložnjavanje udomljenika sadrži *događaj za upisivanje sadržaja u polje za unos teksta* kojim se programski oponaša primjena tipkovnice za upisivanje niza znakova u polje za unos teksta. Događaj za upisivanje sadržaja u polje za unos teksta koristi se u slučajevima kada je za postizanje dijela funkcionalnosti složenog udomljenika na korisničkom sučelju nekog od osnovnih udomljenika u polje za unos teksta potrebno upisati zadani niz znakova. Izvođenjem događaja, vrijednost izabranog polja za unos teksta poprima novoupisanu vrijednost.

Prostor oblikovanja

Slikom 9.9 prikazan je prostor oblikovanja događaja za upisivanje sadržaja u polje za unos teksta. Prostor oblikovanja događaja sastoji se od udomljenika čije korisničko sučelje sadrži barem jedno polje za unos teksta te naprave za međudjelovanje korisnika i računala u obliku tipkovnice putem koje je potrošaču omogućeno unošenje proizvoljnih nizova znakova u računalni sustav.

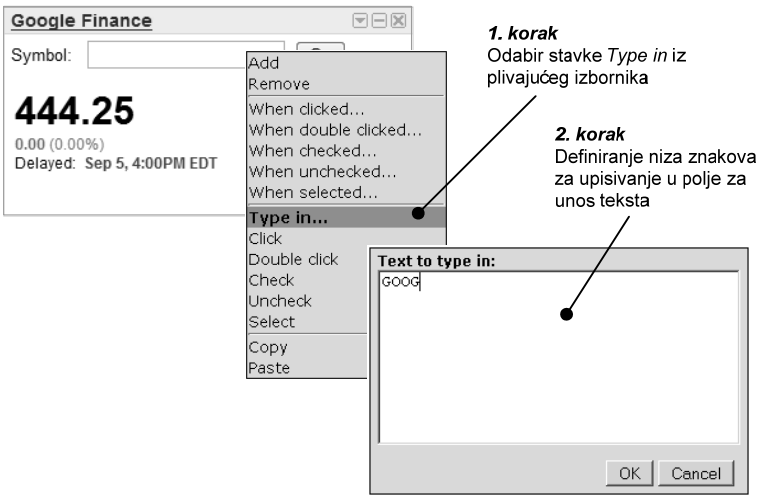


Slika 9.9. Prostor oblikovanja događaja za upisivanje sadržaja u polje za unos teksta

Zamišljajući primjenu tipkovnice, potrošač zamišlja upisivanje proizvoljnog znakovnog sadržaja u izabrano polje za unos teksta na korisničkom sučelju udomljenika. Očekivani rezultat izvođenja događaja je promjena sadržaja polja za unos teksta na novoupisanu vrijednost sadržaja. Ako je neposredno prije izvođenja događaja odredišno polje za unos teksta prazno, neposredno nakon izvođenja događaja sadržaj polja poprima vrijednost koja se upisuje u polje. Ako je neposredno prije izvođenja događaja u odredišnom polju za unos teksta bio upisan određeni sadržaj, neposredno nakon izvođenja događaja prethodni sadržaj polja se briše i poprima novu vrijednost koja se upisuje u polje. U primjeru na slici 9.9, prikazano je oblikovanje događaja za upis znakovnog niza *GOOG* u polje za unos teksta *Symbol* udomljenika *Google Finance*.

Prostor izgradnje

Izgradnja događaja za upisivanje sadržaja u polje za unos teksta prikazana je slikom 9.10. Pritiskom desne tipke miša na izabrano polje za unos teksta pojavljuje se plivajući izbornik za izgradnju programa za usložnjavanje udomljenika. Događaj za upisivanje sadržaja u polje za unos teksta izgrađuje se u dva koraka.



Slika 9.10. Prostor izgradnje događaja za upisivanje sadržaja u polje za unos teksta

U prvom koraku, iz plivajućeg izbornika odabire se stavka *Type in* koja služi za definiranje događaja za programsko upisivanje znakovnih nizova u polje za unos teksta. Odabirom stavke *Type in* iz plivajućeg izbornika, pojavljuje se obrazac za definiranje niza znakova koji je tijekom izvođenja događaja potrebno upisati u izabrano polje za unos teksta. U drugom koraku izgradnje događaja, u prikazani obrazac upisuje se izabrani niz znakova. Primjerom na slici 9.10 prikazan je postupak izgradnje događaja za upisivanje znakovnog niza *GOOG* u polje za unos teksta *Symbol* udomljenika *Google Finance*.

Prostor prikaza

Događaj za upisivanje sadržaja u polje za unos teksta izgrađen primjenom grafičkog izbornika zapisuje se u programsku strukturu složenog udomljenika i sprema za kasnije izvođenje. Primjer slovčanog zapisa događaja za upisivanje sadržaja u polje za unos teksta je

type "GOOG" into Symbol at Google Finance

Izvođenjem zapisanog događaja, znakovni niz *GOOG* upisuje se u polje za unos teksta *Symbol* na udomljeniku *Google Finance*.

Opći oblik zapisa događaja za upisivanje sadržaja u polje za unos teksta prikazan je regularnim definicijama na slici 9.11. Regularnom definicijom *upisSadržajaUPoljeZaUnosTeksta* opisano je pravilo za oblikovanje nizova znakova koji predstavljaju valjano zapisane događaje za upisivanje sadržaja u polje za unos teksta. Regularna definicija *upisSadržajaUPoljeZaUnosTeksta* definirana je ključnim riječima *type*, *into* i *at* te regularnim definicijama *nizZnakova*, *element* i *udomljenik*.

upisSadržajaUPoljeZaUnosTeksta = **type** "*nizZnakova*" **into** *element* **at** *udomljenik*
nizZnakova = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**]+[+,-,_,!,?,$,#,@,...])*
element = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**])*
udomljenik = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**])*

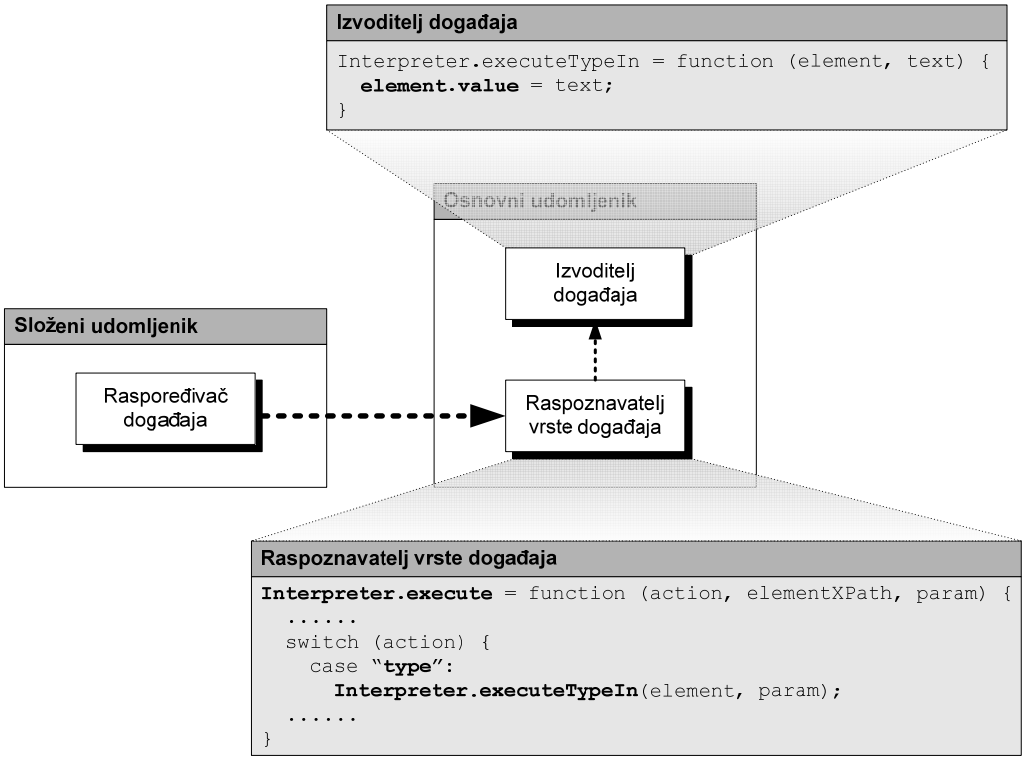
Slika 9.11. Opći oblik zapisa događaja za upisivanje sadržaja u polje za unos teksta

Ključne riječi *type* i *into* koriste se za jednoznačno prepoznavanje događaja za upisivanje sadržaja u polje za unos teksta. Regularnom definicijom *nizZnakova* opisano je pravilo za oblikovanje nizova znakova koji predstavljaju sadržaj za upisivanje u polje za unos teksta. Ti nizovi oblikuju se od bilo koje kombinacije znakova koju je moguće unijeti primjenom tipkovnice, što uključuje velika i mala slova, brojke i posebne znakove.

Regularnom definicijom *element* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje ciljnog polja za unos teksta. Regularnom definicijom *udomljenik* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje udomljenika koji sadrži ciljno polje za unos teksta. Naslovljavanje polja za unos teksta i udomljenika obavlja se nizovima znakova koji se sastoje od brojki te velikih i malih slova abecede. Ključna riječ *at* koristi se u svojstvu graničnika između naziva polja za unos teksta i naziva udomljenika.

Prostor izvođenja

Izvođenje događaja za upisivanje sadržaja u polje za unos teksta prikazano je na slici 9.12. Prikazan je isječak programskog ostvarenja raspoznavatelja vrste događaja koji je zadužen za prepoznavanje događaja za upisivanje sadržaja u polje za unos teksta te programsko ostvarenje izvoditelja događaja. Komunikacija između složenog i osnovnog udomljenika je jednosmjerna jer izvođenjem događaja za upisivanje sadržaja u polje za unos teksta ne nastaju rezultati koje je potrebno vraćati složenom udomljeniku.



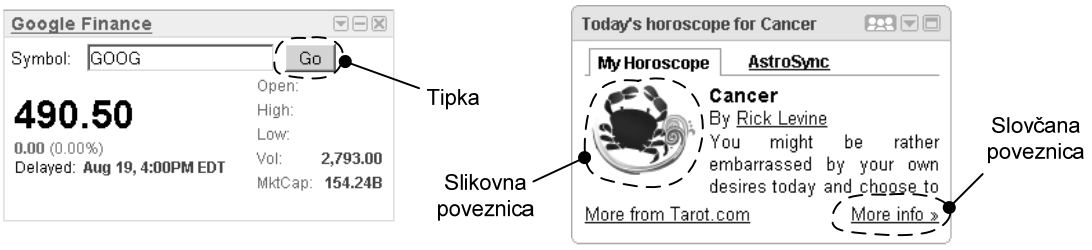
Slika 9.12. Prostor izvođenja događaja za upisivanje sadržaja u polje za unos teksta

Događaj za upisivanje sadržaja u polje za unos teksta raspoznaje se po ključnoj riječi *type*. Nakon prepoznavanja ove vrste događaja, raspoznavatelj vrste događaja pokreće

izvoditelj događaja koji je ostvaren funkcijom *Interpreter.executeTypeIn*. Ulazni parametri funkcije *Interpreter.executeTypeIn* su pokazivač na ciljno polje za unos teksta i slovčani sadržaj koji je potrebno upisati u polje. Upisivanje sadržaja ostvareno je postavljanjem članske varijable *value* programskog objekta za rukovanje poljem za unos teksta na vrijednost slovčanog sadržaja predviđenog za upis u polje. Članska varijabla *value* pripada standardnom programskom sučelju objekta kojim se u jeziku *Javascript* rukuje elementima grafičkog korisničkog sučelja u obliku polja za unos teksta.

9.3.2 Pritisak aktivacijskog elementa

Aktivacijski elementi koriste se za pokretanje izvođenja pozadinskih računalnih procesa putem pridruženog grafičkog korisničkog sučelja. Pokretanje izvođenja računalnih procesa izvodi se pritiskom pokazivačem miša na aktivacijski element korisničkog sučelja. Aktivacijski elementi korisničkog sučelja potrošaču su najčešće predočeni u obliku omeđenog područja na zaslonu računala koje je osjetljivo na pritisak pokazivačem miša. Unutar područja aktivacijskog elementa, slovčanim ili slikovnim sadržajem opisana je aktivnost koja se pokreće pritiskom miša. Najčešći oblici aktivacijskih elemenata su tipke i poveznice. Slikom 9.13 prikazani su primjeri udomljenika s istaknutim aktivacijskim elementima grafičkog korisničkog sučelja u obliku tipke označene slovčanim sadržajem *Go*, slovčane poveznice označene tekstom *More info* te slikovne poveznice označene horoskopskim znakom.

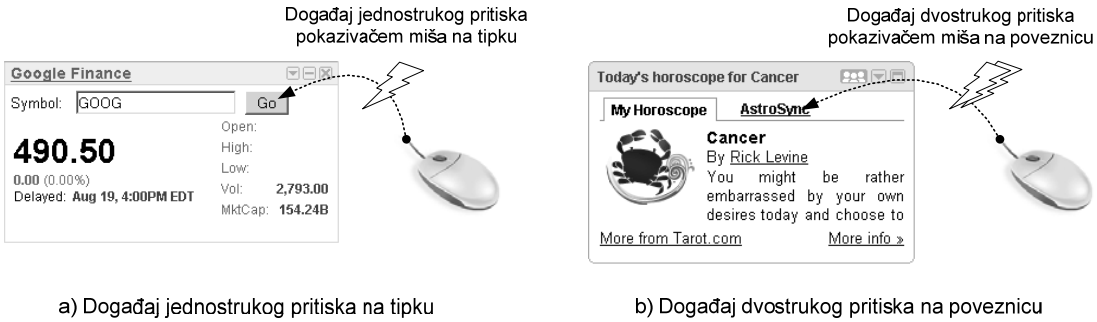


Slika 9.13. Primjeri aktivacijskih elemenata grafičkog korisničkog sučelja udomljenika

Programski jezik za usložnjavanje udomljenika sadrži *događaj za jednostruki pritisak aktivacijskog elementa* i *događaj za dvostruki pritisak aktivacijskog elementa* kojima se programski oponaša primjena miša za obavljanje jednostrukog, odnosno dvostrukog pritiska aktivacijskog elementa. Ova dva događaja koriste se u slučajevima kada je za postizanje dijela funkcionalnosti složenog udomljenika na korisničkom sučelju nekog od osnovnih udomljenika potrebno pokrenuti izvođenje procesa koji se pokreće pritiskom aktivacijskog elementa.

Prostor oblikovanja

Slikom 9.14 prikazan je prostor oblikovanja događaja za izvođenje programskog pritiska na aktivacijski element grafičkog korisničkog sučelja udomljenika. Prostor oblikovanja događaja sastoji se od udomljenika čije korisničko sučelje sadrži barem jedan aktivacijski element te naprave za međudjelovanje korisnika i računala u obliku miša putem koje je korisniku omogućeno unošenje prostorno povezanih podataka u računalni sustav.



a) Događaj jednostrukog pritiska na tipku b) Događaj dvostrukog pritiska na poveznicu

Slika 9.14. Oblikovanje događaja za programski pritisak aktivacijskog elementa

Na slici 9.14.a prikazan je primjer oblikovanja događaja za izvođenje jednostrukog pritiska na aktivacijski element u obliku tipke *Go* udomljenika *Google Finance*, dok je na slici 9.14.b prikazan primjer oblikovanja događaja za izvođenje dvostrukog pritiska na aktivacijski element u obliku poveznice *AstroSync* udomljenika *Today's horoscope*. Zamišljajući primjenu miša, potrošač zamišlja obavljanje radnje jednostrukog, odnosno dvostrukog pritiska na izabrani aktivacijski element korisničkog sučelja udomljenika. Očekivani rezultat izvođenja događaja je pokretanje aktivnosti nad pozadinskim procesom kojom se udomljenik odaziva na pojavu jednostrukog, odnosno dvostrukog pritiska na izabrani aktivacijski element.

Prostor izgradnje

Izgradnja događaja za izvođenje programskog pritiska aktivacijskog elementa grafičkog korisničkog sučelja udomljenika prikazana je slikom 9.15. Pritiskom desne tipke miša na izabrani aktivacijski element pojavljuje se plivajući izbornik za izgradnju programa za usložnjavanje udomljenika.

Događaj za izvođenje jednostrukog pritiska aktivacijskog elementa izgrađuje se odabirom stavke *Click* u plivajućem izborniku. Primjerom na slici 9.15.a prikazan je postupak izgradnje događaja za izvođenje jednostrukog pritiska tipke *Go* udomljenika *Google Finance*. Događaj za izvođenje dvostrukog pritiska aktivacijskog elementa izgrađuje

se odabirom stavke *Double click* u plivajućem izborniku. Primjerom na slici 9.15.b prikazan je postupak izgradnje događaja za izvođenje dvostrukog pritiska na poveznicu *AstroSync* udomljenika *Today's horoscope*.

a) Događaj jednostrukog pritiska na tipku

b) Događaj dvostrukog pritiska na poveznicu

Slika 9.15. Izgradnja događaja za izvođenje programskog pritiska na aktivacijski element grafičkog korisničkog sučelja udomljenika

Prostor prikaza

Događaji za izvođenje jednostrukog i dvostrukog pritiska aktivacijskih elemenata izgrađeni primjenom grafičkog izbornika zapisuju se u programsku strukturu složenog udomljenika i spremaju za kasnije izvođenje. Primjeri slovčanih zapisa događaja za izvođenje jednostrukog i dvostrukog pritiska aktivacijskih elemenata su

click Go at Google Finance
doubleclick AstroSync at Today's Horoscope

Izvođenjem prvog od dvaju zapisanih događaja, programski se oponaša radnja jednostrukog pritiska pokazivačem miša na tipku *Go* udomljenika *Google Finance*, dok se izvođenjem drugog događaja programski oponaša radnja dvostrukog pritiska pokazivačem miša na poveznicu *AstroSync* udomljenika *Today's horoscope*.

Opći oblik zapisa događaja za programsko izvođenje radnje pritiska na aktivacijski element korisničkog sučelja prikazan je regularnim definicijama na slici 9.16. Regularnom definicijom *pritisakNaAktivacijskiElement* opisano je pravilo za oblikovanje nizova znakova koji predstavljaju valjano zapisane događaje za programsko izvođenje radnje pritiska na aktivacijski element korisničkog sučelja. Tom regularnom definicijom definirana su dva moguća oblika događaja.

Prvi oblik događaja je događaj jednostrukog pritiska na aktivacijski element korisničkog sučelja. Pravilo za oblikovanje nizova znakova koji predstavljaju valjano zapisane događaje jednostrukog pritiska na aktivacijski element korisničkog sučelja opisano je regularnom definicijom *jednostrukiPritisak*. Regularna definicija *jednostrukiPritisak* definirana je ključnim riječima *click* i *at* te regularnim definicijama *element* i *udomljenik*. Ključna riječ *click* koristi se za jednoznačno prepoznavanje događaja jednostrukog pritiska na aktivacijski element korisničkog sučelja. Regularnom definicijom *element* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje ciljnog aktivacijskog elementa. Regularnom definicijom *udomljenik* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje udomljenika koji sadrži ciljni aktivacijski element. Naslovljavanje aktivacijskog elementa i udomljenika obavlja se nizovima znakova koji se sastoje od brojki te velikih i malih slova abecede. Ključna riječ *at* koristi se u svojstvu graničnika između naziva aktivacijskog elementa i naziva udomljenika.

pritisakNaAktivacijskiElement = *jednostrukiPritisak* | *dvostrukiPritisak*
jednostrukiPritisak = **click** *element* **at** *udomljenik*
dvostrukiPritisak = **doubleclick** *element* **at** *udomljenik*
element = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**])*
udomljenik = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**])*

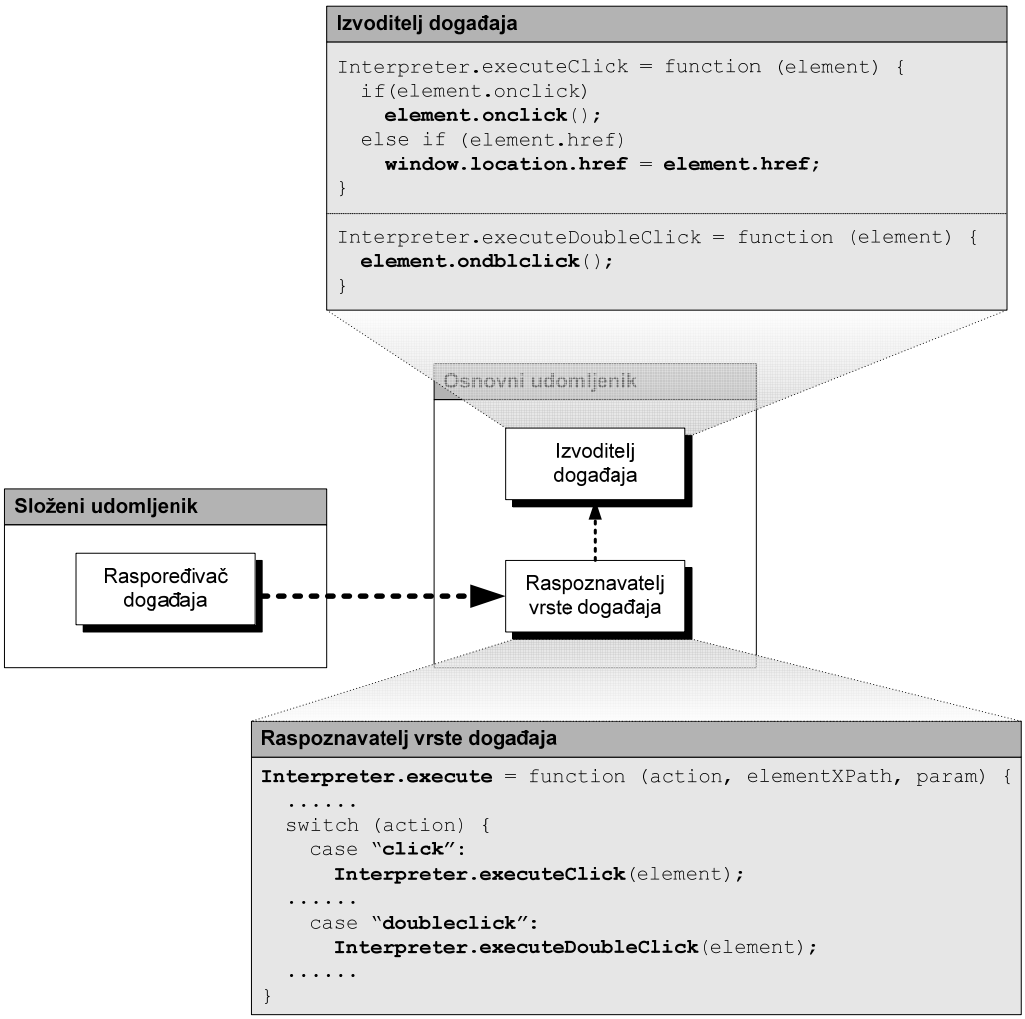
Slika 9.16. Opći oblik zapisa događaja za programsko izvođenje radnje pritiska na aktivacijski element korisničkog sučelja

Drugi oblik događaja je događaj dvostrukog pritiska na aktivacijski element korisničkog sučelja. Pravilo za oblikovanje nizova znakova koji predstavljaju valjano zapisane događaje dvostrukog pritiska na aktivacijski element korisničkog sučelja opisano je regularnom definicijom *dvostrukiPritisak*. Regularna definicija *dvostrukiPritisak* definirana je ključnim riječima *doubleclick* i *at* te regularnim definicijama *element* i *udomljenik*. Ključna riječ *doubleclick* koristi se za jednoznačno prepoznavanje događaja dvostrukog pritiska na aktivacijski element korisničkog sučelja. Uloga ključne riječi *at* te regularnih definicija *element* i *udomljenik* istovjetna je ulozi tih cjelina u regularnoj definiciji *jednostrukiPritisak*.

Prostor izvođenja

Izvođenje događaja pritiska aktivacijskog elementa grafičkog korisničkog sučelja prikazano je na slici 9.17. Prikazan je isječak programskog ostvarenja raspoznavatelja vrste događaja koji je zadužen za prepoznavanje događaja pritiska aktivacijskog elementa te

programsko ostvarenje izvoditelja događaja. Komunikacija između složenog i osnovnog udomljenika je jednosmjerna jer izvođenjem događaja pritiska aktivacijskog elementa ne nastaju rezultati koje je potrebno vraćati složenom udomljeniku.



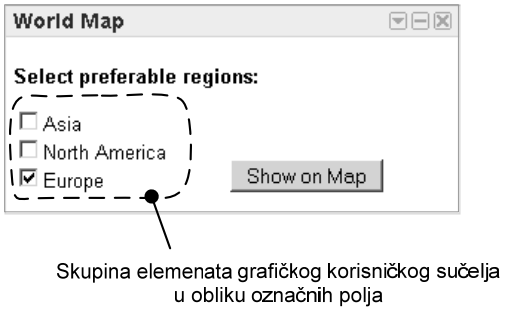
Slika 9.17. Prostor izvođenja događaja za programsko izvođenje radnje pritiska na aktivacijski element grafičkog korisničkog sučelja

Događaj jednostrukog pritiska aktivacijskog elementa raspoznaje se po ključnoj riječi *click*. Nakon prepoznavanja ove vrste događaja, raspoznavatelj vrste događaja pokreće izvoditelj događaja koji je ostvaren funkcijom *Interpreter.executeClick*. Događaj dvostrukog pritiska aktivacijskog elementa raspoznaje se po ključnoj riječi *doubleclick*. Nakon prepoznavanja ove vrste događaja, raspoznavatelj vrste događaja pokreće izvoditelj događaja koji je ostvaren funkcijom *Interpreter.executeDoubleClick*. Ulazni parametar obiju funkcija izvoditelja događaja je pokazivač na ciljni aktivacijski element korisničkog sučelja osnovnog udomljenika.

Programsko oponašanje radnje jednostrukog pritiska na aktivacijski element grafičkog korisničkog sučelja ovisi o vrsti elementa. Izvođenje jednostrukog pritiska na aktivacijski element u obliku tipke i slike postiže se pozivom funkcije *onclick* nad programskim objektom za rukovanje aktivacijskim elementom. Izvođenje jednostrukog pritiska na aktivacijski element u obliku poveznice postiže se postavljanjem članske varijable *location.href* straničnog okvira osnovnog udomljenika na vrijednost internetske adrese na koju pokazuje poveznica, a zapisana je u varijabli *element.href*. Postavljanjem vrijednosti *location.href* straničnog okvira, u stranični okvir učitava se sadržaj s adrese na koju pokazuje poveznica. Programsko oponašanje radnje dvostrukog pritiska na aktivacijski element grafičkog korisničkog sučelja postiže se pozivom funkcije *ondblclick* nad programskim objektom za rukovanje aktivacijskim elementom. Funkcije *onclick* i *ondblclick* pripadaju standardnom programskom sučelju objekta kojim se u jeziku *Javascript* rukuje aktivacijskim elementima grafičkog korisničkog sučelja. Članska varijabla *location.href* pripada standardnom programskom sučelju objekta kojim se u jeziku *Javascript* rukuje elementima grafičkog korisničkog sučelja u obliku poveznica.

9.3.3 Označavanje i odznačavanje označnog polja

Elementi grafičkog korisničkog sučelja udomljenika u obliku označnih polja pripadaju skupini ulaznih elemenata korisničkog sučelja. Koriste se za rukovanje podacima koje je pogodno prikazati u obliku vrijednosti logičke istinitosti. Elementi u obliku označnih polja u svakom se trenutku nalaze u jednom od dva moguća stanja označenosti. Označenim označnim poljem predstavljena je vrijednost logičke istine. Neoznačenim označnim poljem predstavljena je vrijednost logičke laži. Promjena stanja označnog polja obavlja se pritiskom pokazivača miša na područje korisničkog sučelja na kojem je prikazano označno polje. U području oblikovanja grafičkih korisničkih sučelja, skup označnih polja uobičajeno se objedinjava u zajedničku skupinu, čime se ostvaruje element korisničkog sučelja za izbor podskupa vrijednosti iz skupa ponuđenih mogućnosti.



Slika 9.18. Elementi grafičkog korisničkog sučelja udomljenika u obliku označnih polja

Slikom 9.18 prikazan primjer grafičkog korisničkog sučelja udomljenika s istaknutim elementima u obliku označnih polja. Elementi korisničkog sučelja u obliku označnih polja korisniku su najčešće predočeni u obliku kvadratnog polja unutar kojeg je moguće postaviti oznaku vrijednosti logičke istinitosti. Vrijednost logičke laži prikazuje se praznim označnim poljem. Postavljanjem oznake označenosti unutar kvadratnog polja, vrijednost označnog polja postavlja se na vrijednost logičke istine. Uz kvadratno polje uobičajeno se nalazi i popratna slovčana oznaka kojom se opisuje uloga označnog polja u promatranom udomljeniku. U prikazanom primjeru, korisničko sučelje udomljenika sadrži skupinu od tri elementa u obliku označnih polja. Označna polja redom su označena popratnim slovčanim oznakama *Asia*, *North America* i *Europe*. Označno polje *Europe* je označeno, dok su označna polja *Asia* i *North America* neoznačena.

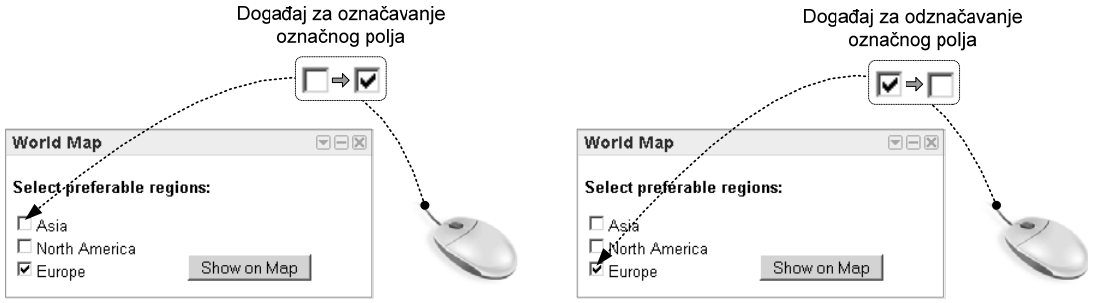
Programski jezik za usložnjavanje udomljenika sadrži dva događaja za programsko upravljanje radom označnih polja. *Događajem za označavanje označnog polja* programski se oponaša primjena miša za izvođenje radnje postavljanja oznake označenosti unutar označnog polja. *Događajem za odznačavanje označnog polja* programski se oponaša primjena miša za izvođenje radnje uklanjanja oznake označenosti iz označnog polja. Događaj za označavanje označnog polja koristi se u slučajevima kada je za postizanje dijela funkcionalnosti složenog udomljenika na korisničkom sučelju nekog od osnovnih udomljenika potrebno izvršiti radnju označavanja označnog polja. Izvođenjem događaja, stanje označenosti izabranog označnog polja poprima oblik označenosti. Događaj za odznačavanje označnog polja koristi se u slučajevima kada je za postizanje dijela funkcionalnosti složenog udomljenika na korisničkom sučelju nekog od osnovnih udomljenika potrebno izvršiti radnju odznačavanja označnog polja. Izvođenjem događaja, stanje označenosti izabranog označnog polja poprima oblik neoznačenosti.

Prostor oblikovanja

Slikom 9.19 prikazan je prostor oblikovanja događaja za označavanje i odznačavanje označnog polja. Prostor oblikovanja događaja sastoji se od udomljenika čije korisničko sučelje sadrži barem jedan element korisničkog sučelja u obliku označnog polja te naprave za međudjelovanje korisnika i računala u obliku miša putem koje je korisniku omogućeno unošenje prostorno povezanih podataka u računalni sustav.

Zamišljajući primjenu miša, potrošač zamišlja obavljanje radnje postavljanja oznake označenosti unutar označnog polja, odnosno obavljanje radnje uklanjanja oznake označenosti iz označnog polja. Očekivani rezultat izvođenja događaja je promjena stanja

označenosti označnog polja iz neoznačenog oblika u označeni oblik ili obrnuto. U primjeru na slici 9.19.a, prikazano je oblikovanje događaja za označavanje označnog polja *Asia* udomljenika *World Map*. Očekivani rezultat izvođenja događaja je promjena stanja označenosti označnog polja iz neoznačenog u označeni oblik. Primjerom na slici 9.19.b, prikazano je oblikovanje događaja za odznačavanje označnog polja *Europe* udomljenika *World Map*. Očekivani rezultat izvođenja događaja je promjena stanja označenosti označnog polja iz označenog u neoznačeni oblik.

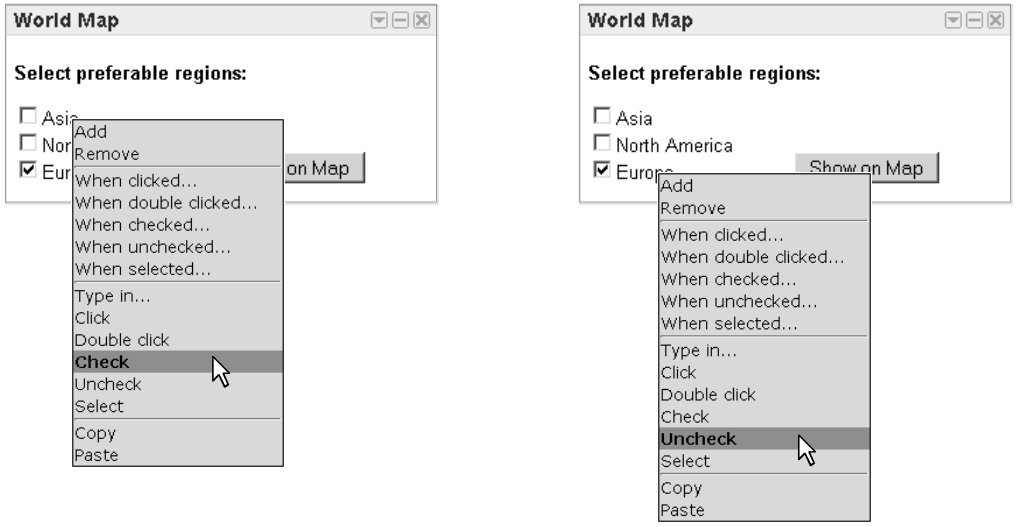


a) Označavanje označnog polja b) Odznačavanje označnog polja

Slika 9.19. Oblikovanje događaja za označavanje i odznačavanje označnog polja

Prostor izgradnje

Izgradnja događaja za označavanje i odznačavanje označnog polja prikazana je slikom 9.20. Pritiskom desne tipke miša na izabrano označno polje pojavljuje se plivajući izbornik za izgradnju programa za usložnjavanje udomljenika.



a) Označavanje označnog polja b) Odznačavanje označnog polja

Slika 9.20. Izgradnja događaja za označavanje i odznačavanje označnog polja

Događaj za označavanje označnog polja izgrađuje se odabirom stavke *Check* u plivajućem izborniku. Primjerom na slici 9.20.a prikazan je postupak izgradnje događaja za označavanje označnog polja *Asia* udomljenika *World Map*. Događaj za odznačavanje označnog polja izgrađuje se odabirom stavke *Uncheck* u plivajućem izborniku. Primjerom na slici 9.20.b prikazan je postupak izgradnje događaja za odznačavanje označnog polja *Europe* udomljenika *World Map*.

Prostor prikaza

Događaji za označavanje i odznačavanje označnog polja izgrađeni primjenom grafičkog izbornika zapisuju se u programsku strukturu složenog udomljenika i spremaju za kasnije izvođenje. Primjeri slovčanih zapisa događaja za označavanje i odznačavanje označnog polja su

check Asia at World Map
uncheck Europe at World Map

Izvođenjem prvog od dvaju zapisanih događaja, programski se oponaša radnja označavanja označnog polja *Asia* udomljenika *World Map*, dok se izvođenjem drugog događaja programski oponaša radnja odznačavanja označnog polja *Europe* udomljenika *World Map*.

Opći oblik zapisa događaja za programsko upravljanje radom označnog polja prikazan je regularnim definicijama na slici 9.21. Regularnom definicijom *upravljanjeOznačnimPoljem* opisano je pravilo za oblikovanje nizova znakova koji predstavljaju valjano zapisane događaje za programsko upravljanje radom označnog polja. Tom regularnom definicijom definirana su dva moguća oblika događaja.

upravljanjeOznačnimPoljem = *označavanjeOznačnogPolja* | *odznačavanjeOznačnogPolja*
označavanjeOznačnogPolja = **check** *element* **at** *udomljenik*
odznačavanjeOznačnogPolja = **uncheck** *element* **at** *udomljenik*
element = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**])*
udomljenik = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**])*

Slika 9.21. Opći oblik zapisa događaja za označavanje i odznačavanje označnog polja

Prvi oblik događaja je događaj za označavanje označnog polja. Pravilo za oblikovanje nizova znakova koji predstavljaju valjano zapisane događaje za označavanje označnog polja opisano je regularnom definicijom *označavanjeOznačnogPolja*. Regularna definicija *označavanjeOznačnogPolja* definirana je ključnim riječima *check* i *at* te regularnim

definicijama *element* i *udomljenik*. Ključna riječ *check* koristi se za jednoznačno prepoznavanje događaja za označavanje označnog polja. Regularnom definicijom *element* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje ciljnog označnog polja. Regularnom definicijom *udomljenik* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje udomljenika koji sadrži ciljno označno polje. Naslovljavanje označnog polja i udomljenika obavlja se nizovima znakova koji se sastoje od brojki te velikih i malih slova abecede. Ključna riječ *at* koristi se u svojstvu graničnika između naziva označnog polja i naziva udomljenika.

Drugi oblik događaja je događaj za odznačavanje označnog polja. Pravilo za oblikovanje nizova znakova koji predstavljaju valjano zapisane događaje za odznačavanje označnog polja opisano je regularnom definicijom *odznačavanjeOznačnogPolja*. Regularna definicija *odznačavanjeOznačnogPolja* definirana je ključnim riječima *uncheck* i *at* te regularnim definicijama *element* i *udomljenik*. Ključna riječ *uncheck* koristi se za jednoznačno prepoznavanje događaja za odznačavanje označnog polja. Uloga ključne riječi *at* te regularnih definicija *element* i *udomljenik* istovjetna je ulozi tih cjelina u regularnoj definiciji *označavanjeOznačnogPolja*.

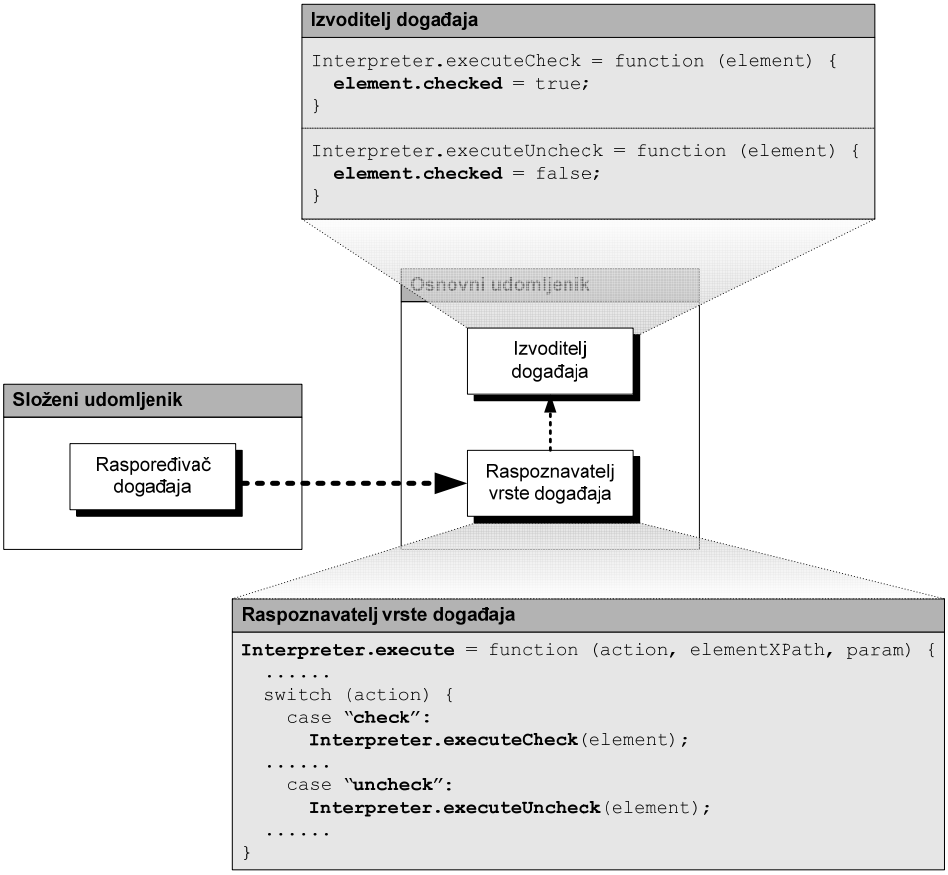
Prostor izvođenja

Izvođenje događaja za označavanje i odznačavanje označnog polja prikazano je na slici 9.22. Prikazan je isječak programskog ostvarenja raspoznavatelja vrste događaja koji je zadužen za prepoznavanje događaja za označavanje i odznačavanje označnog polja te programsko ostvarenje izvoditelja događaja. Komunikacija između složenog i osnovnog udomljenika je jednosmjerna jer izvođenjem događaja za označavanje i odznačavanje označnog polja ne nastaju rezultati koje je potrebno vraćati složenom udomljeniku.

Događaj za označavanje označnog polja raspoznaje se po ključnoj riječi *check*. Nakon prepoznavanja ove vrste događaja, raspoznavatelj vrste događaja pokreće izvoditelj događaja koji je ostvaren funkcijom *Interpreter.executeCheck*. Događaj za odznačavanje označnog polja raspoznaje se po ključnoj riječi *uncheck*. Nakon prepoznavanja ove vrste događaja, raspoznavatelj vrste događaja pokreće izvoditelj događaja koji je ostvaren funkcijom *Interpreter.executeUncheck*. Ulazni parametar obiju funkcija izvoditelja događaja je pokazivač na ciljni element korisničkog sučelja u obliku označnog polja.

Programsko oponašanje radnje označavanja označnog polja postiže se postavljanjem članske varijable *checked* nad programskim objektom za rukovanje označnim poljem na vrijednost logičke istine. Programsko oponašanje radnje odznačavanja označnog polja

postiže se postavljanjem članske varijable *checked* nad programskim objektom za rukovanje označnim poljem na vrijednost logičke laži. Članska varijabla *checked* pripada standardnom programskom sučelju objekta kojim se u jeziku *Javascript* rukuje elementima grafičkog korisničkog sučelja u obliku označnih polja.

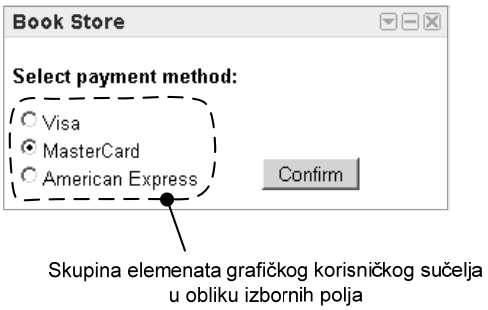


Slika 9.22. Prostor izvođenja događaja za označavanje i odznačavanje označnog polja

9.3.4 Obilježavanje izbornog polja

Elementi grafičkog korisničkog sučelja udomljenika u obliku izbornih polja pripadaju skupini ulaznih elemenata grafičkog korisničkog sučelja. Koriste se za rukovanje skupovima podataka u slučajevima kada je potrebno napraviti izbor točno jednog elementa iz skupa ponuđenih mogućnosti. Skup izbornih polja objedinjuje se u zajedničku skupinu kojom je definiran skup mogućnosti za odabir. Svakim izbornim poljem definira se po jedan element skupa. Elementi u obliku izbornih polja u svakom se trenutku nalaze u jednom od dva moguća stanja obilježenosti. Obilježenim označnim poljem predstavljen je izabrani element u skupu podataka. Neobilježenim izbornim poljem predstavljeni su neizabrani elementi u skupu. U svakom trenutku, isključivo jedno izborno polje koje pripada istoj skupini moguće

je postaviti u stanje obilježenosti, dok su sva ostala izborna polja u skupini neobilježena. Obilježavanjem neobilježenog izbornog polja, izborno polje koje je prethodno bilo obilježeno postaje neobilježeno.



Slika 9.23. Elementi grafičkog korisničkog sučelja udomljenika u obliku izbornih polja

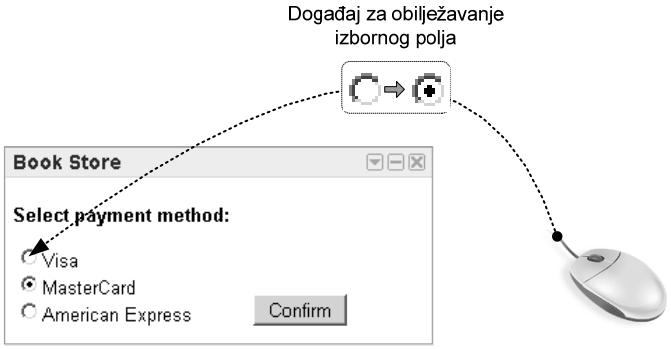
Slikom 9.23 prikazan je primjer grafičkog korisničkog sučelja udomljenika s istaknutim elementima u obliku izbornih polja. Elementi korisničkog sučelja u obliku izbornih polja korisniku su najčešće predočeni u obliku kružnog polja unutar kojeg je moguće postaviti oznaku obilježenosti. Neobilježeno izborno polje prikazuje se praznim kružnim poljem. Postavljanjem oznake obilježenosti unutar kružnog polja, izborno polje postavlja se u oblik obilježenosti. Uz kružno polje obično se nalazi i popratna slovčana oznaka kojom se opisuje uloga označnog polja u promatranom udomljeniku. U prikazanom primjeru, korisničko sučelje udomljenika sadrži skupinu od tri izborna polja koja su redom označena popratnim slovčanim oznakama *Visa*, *MasterCard* i *American Express*. U skupu s mogućnošću izbora između triju ponuđenih mogućnosti, odabir je obavljen obilježavanjem izbornog polja *MasterCard*, čime su izborna polja *Visa* i *American Express* automatski postala neobilježena.

Programski jezik za usložnjavanje udomljenika sadrži *događaj za obilježavanje izbornog polja* kojim se programski oponaša primjena miša za postavljanje oznake obilježenosti unutar izbornog polja. Događaj za obilježavanje izbornog polja koristi se u slučajevima kada je za postizanje dijela funkcionalnosti složenog udomljenika na korisničkom sučelju nekog od osnovnih udomljenika potrebno obilježiti odgovarajuće izborno polje.

Prostor oblikovanja

Slikom 9.24 prikazan je prostor oblikovanja događaja za obilježavanje izbornog polja. Prostor oblikovanja događaja sastoji se od udomljenika čije korisničko sučelje sadrži barem jedan element korisničkog sučelja u obliku izbornog polja te naprave za međudjelovanje

korisnika i računala u obliku miša putem koje je korisniku omogućeno unošenje prostorno povezanih podataka u računalni sustav.



Slika 9.24. Oblikovanje događaja za obilježavanje izbornog polja

Zamišljajući primjenu miša, potrošač zamišlja obavljanje radnje pritiska pokazivačem na neobilježeno izborno polje korisničkog sučelja udomljenika. Očekivani rezultat izvođenja događaja je promjena stanja obilježenosti izbornog polja iz neobilježenog u obilježeni oblik, uz istovremeno postavljanje svih ostalih izbornih polja unutar iste skupine u oblik neobilježenosti. U primjeru na slici 9.24, prikazano je oblikovanje događaja za obilježavanje izbornog polja *Visa* udomljenika *Book Store*.

Prostor izgradnje

Izgradnja događaja za obilježavanje izbornog polja prikazana je slikom 9.25. Pritiskom desne tipke miša na izabrano izborno polje pojavljuje se plivajući izbornik za izgradnju programa za upravljanje radom udomljenika.

Slika 9.25. Izgradnja događaja za obilježavanje izbornog polja

Događaj za obilježavanje izbornog polja izgrađuje se odabirom stavke *Select* u plivajućem izborniku. Primjerom na slici 9.25 prikazan je postupak izgradnje događaja za obilježavanje izbornog polja *Visa* udomljenika *Book Store*.

Prostor prikaza

Događaj za obilježavanje izbornog polja izgrađen primjenom grafičkog izbornika zapisuje se u programsku strukturu složenog udomljenika i sprema za kasnije izvođenje. Primjer slovčanog zapisa događaja za obilježavanje izbornog polja je

select Visa at Book Store

Izvođenjem zapisanog događaja, programski se oponaša radnja obilježavanja izbornog polja *Visa* na udomljeniku *Book Store*.

Opći oblik zapisa događaja za obilježavanje izbornog polja prikazan je regularnim definicijama na slici 9.26. Regularnom definicijom *obilježavanjeIzbornogPolja* opisano je pravilo za oblikovanje nizova znakova koji predstavljaju valjano zapisane događaje za obilježavanje izbornog polja. Regularna definicija *obilježavanjeIzbornogPolja* definirana je ključnim riječima *select* i *at* te regularnim definicijama *element* i *udomljenik*.

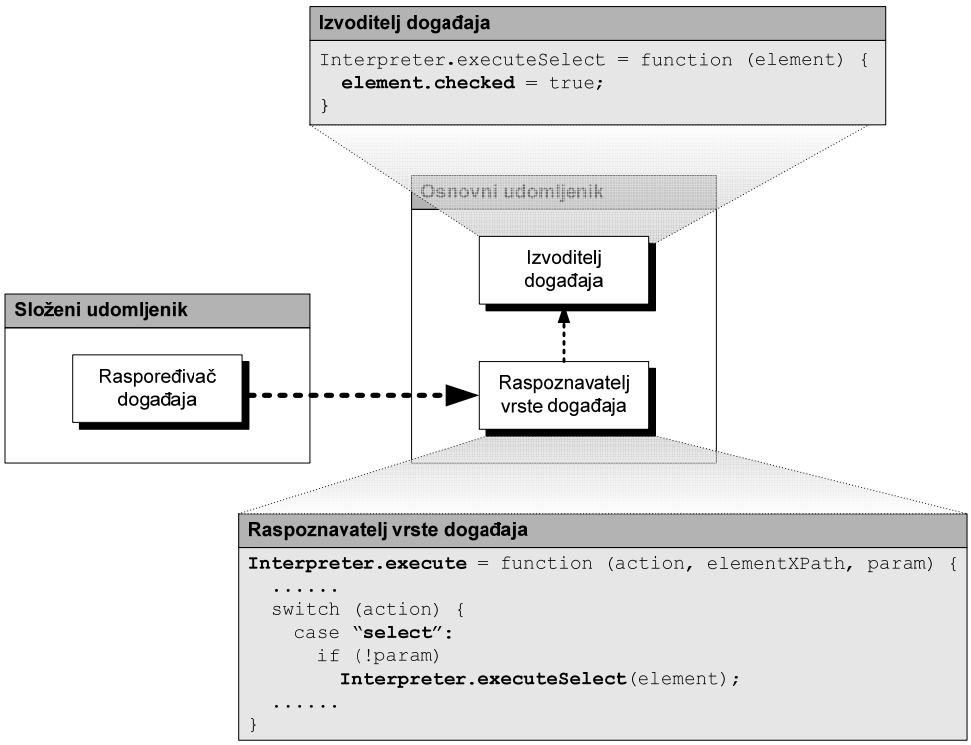
obilježavanjeIzbornogPolja = **select** *element* **at** *udomljenik*
element = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**])*
udomljenik = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**])*

Slika 9.26. Opći oblik zapisa događaja za obilježavanje izbornog polja

Ključna riječ *select* koristi se za jednoznačno prepoznavanje događaja za obilježavanje izbornog polja. Regularnom definicijom *element* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje izbornog polja nad kojim se izvodi operacija obilježavanja. Regularnom definicijom *element* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje ciljnog izbornog polja. Regularnom definicijom *udomljenik* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje udomljenika koji sadrži ciljno izborno polje. Naslovljavanje izbornog polja i udomljenika obavlja se nizovima znakova koji se sastoje od brojki te velikih i malih slova abecede. Ključna riječ *at* koristi se u svojstvu graničnika između naziva izbornog polja i naziva udomljenika.

Prostor izvođenja

Izvođenje događaja za obilježavanje izbornog polja prikazano je na slici 9.27. Prikazan je isječak programskog ostvarenja raspoznavatelja vrste događaja koji je zadužen za prepoznavanje događaja za obilježavanje izbornog polja te programsko ostvarenje izvoditelja događaja. Komunikacija između složenog i osnovnog udomljenika je jednosmjerna jer izvođenjem događaja za obilježavanje izbornog polja ne nastaju rezultati koje je potrebno vraćati složenom udomljeniku.



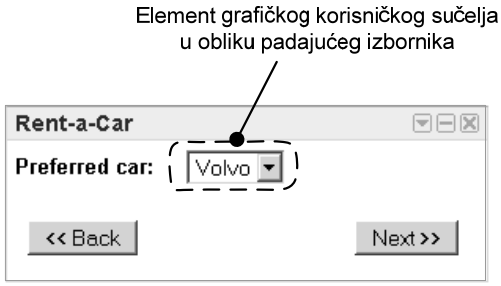
Slika 9.27. Prostor izvođenja događaja za obilježavanje izbornog polja

Događaj za obilježavanje izbornog polja raspoznaje se po ključnoj riječi *select*. Međutim, budući da je istom ključnom riječi označen i događaj za odabir stavke iz padajućeg izbornika koji je opisan u poglavlju 9.3.5, prepoznavanje vrste događaja izvodi se na osnovi parametara poziva funkcije *Interpreter.execute*. Događaj za obilježavanje izbornog polja razlikuje se od događaja za odabir stavke iz padajućeg izbornika po tome što, osim ključne riječi i *XPath* izraza za označavanje položaja izbornog polja unutar strukture *HTML* dokumenta osnovnog udomljenika, ne sadrži dodatne parametre. Nakon prepoznavanja događaja za obilježavanje izbornog polja, raspoznavatelj vrste događaja pokreće izvoditelj događaja koji je ostvaren funkcijom *Interpreter.executeSelect*. Ulazni parametar funkcije izvoditelja događaja je pokazivač na ciljni element grafičkog korisničkog sučelja u obliku izbornog polja.

Programsko oponašanje radnje obilježavanja izbornog polja postiže se postavljanjem članske varijable *checked* nad programskim objektom za rukovanje izbornim poljem na vrijednost logičke istine. Postavljanjem te varijable na vrijednost logičke istine, prethodno izabrano izborno polje postavlja se u stanje neoznačenosti. Članska varijabla *checked* pripada standardnom programskom sučelju objekta kojim se u jeziku *Javascript* rukuje elementima grafičkog korisničkog sučelja u obliku izbornih polja.

9.3.5 Odabir stavke iz padajućeg izbornika

Elementi grafičkog korisničkog sučelja udomljenika u obliku padajućih izbornika pripadaju skupini ulaznih elemenata korisničkog sučelja. Primjena ove vrste elemenata slična je primjeni elemenata u obliku izbornih polja. Padajući izbornici koriste se za oblikovanje korisničkog sučelja koje omogućava izbor točno jednog elementa iz skupa ponuđenih mogućnosti. Padajući izbornik sastoji se od konačnog skupa izborničkih stavki. Svakom stavkom izbornika definira se po jedan element skupa. Izbor jednog elementa iz skupa ponuđenih mogućnosti obavlja se odabirom odgovarajuće stavke iz padajućeg izbornika. U svakom trenutku, iz padajućeg izbornika odabrana je točno jedna stavka.



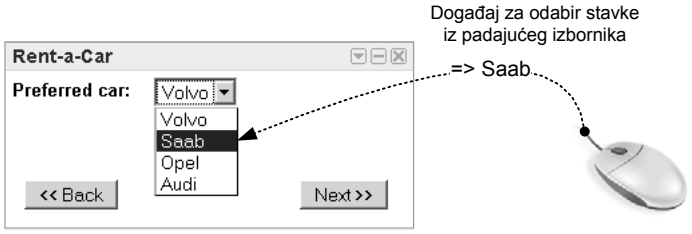
Slika 9.28. Elementi grafičkog korisničkog sučelja udomljenika u obliku padajućeg izbornika

Slikom 9.28 prikazan primjer grafičkog korisničkog sučelja udomljenika s istaknutim elementom u obliku padajućeg izbornika. Elementi korisničkog sučelja u obliku padajućih izbornika korisniku su najčešće predočeni u obliku pravokutnog polja unutar kojeg je nizom znakova upisan naziv odabrane stavke. U prikazanom primjeru, korisničko sučelje udomljenika sadrži padajući izbornik u kojem je odabrana stavka pod nazivom *Volvo*.

Programski jezik za usložnjavanje udomljenika sadrži *događaj za odabir stavke iz padajućeg izbornika*. Događajem za odabir stavke iz padajućeg izbornika programski se oponaša primjena miša ili tipkovnice tijekom odabira jedne od ponuđenih stavki s popisa stavki padajućeg izbornika.

Prostor oblikovanja

Slikom 9.29 prikazan je prostor oblikovanja događaja za odabir stavke iz padajućeg izbornika. Prostor oblikovanja događaja sastoji se od udomljenika čije korisničko sučelje sadrži barem jedan element korisničkog sučelja u obliku padajućeg izbornika te naprave za međudjelovanje korisnika i računala u obliku miša putem koje je korisniku omogućeno unošenje prostorno povezanih podataka u računalni sustav.

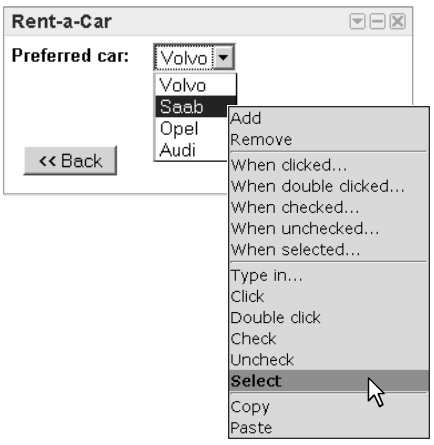


Slika 9.29. Oblikovanje događaja za odabir stavke iz padajućeg izbornika

Zamišljajući primjenu miša, potrošač zamišlja obavljanje radnje pritiska pokazivačem na izabranu stavku padajućeg izbornika korisničkog sučelja udomljenika. Očekivani rezultat izvođenja događaja je postavljanje odabrane stavke padajućeg izbornika u stanje obilježenosti. U primjeru na slici 9.29, prikazano je oblikovanje događaja za odabir stavke *Saab* iz padajućeg izbornika *Prefered car* udomljenika *Rent-a-Car*.

Prostor izgradnje

Izgradnja događaja za odabir stavke iz padajućeg izbornika slična je izgradnji događaja za obilježavanje izbornog polja i prikazana je slikom 9.30. Pritiskom desne tipke miša na izabranu stavku u popisu stavki padajućeg izbornika pojavljuje se plivajući izbornik za izgradnju programa za usložnjavanje udomljenika.



Slika 9.30. Izgradnja događaja za odabir stavke iz padajućeg izbornika

Događaj za odabir stavke iz padajućeg izbornika izgrađuje se odabirom stavke *Select* u plivajućem izborniku. Primjerom na slici 9.30 prikazan je postupak izgradnje događaja za odabir stavke *Saab* iz padajućeg izbornika *Preferred car* udomljenika *Rent-a-Car*.

Prostor prikaza

Događaj za odabir stavke iz padajućeg izbornika izgrađen primjenom grafičkog izbornika zapisuje se u programsku strukturu složenog udomljenika i sprema za kasnije izvođenje. Primjer slovčanog zapisa događaja za odabir stavke iz padajućeg izbornika je

$$select \;\; Saab \;\; from \;\; Preferred\ car \;\; at \;\; Rent-a-Car$$

Izvođenjem zapisanog događaja, programski se oponaša radnja odabira stavke *Saab* iz padajućeg izbornika *Preferred car* udomljenika *Rent-a-Car*.

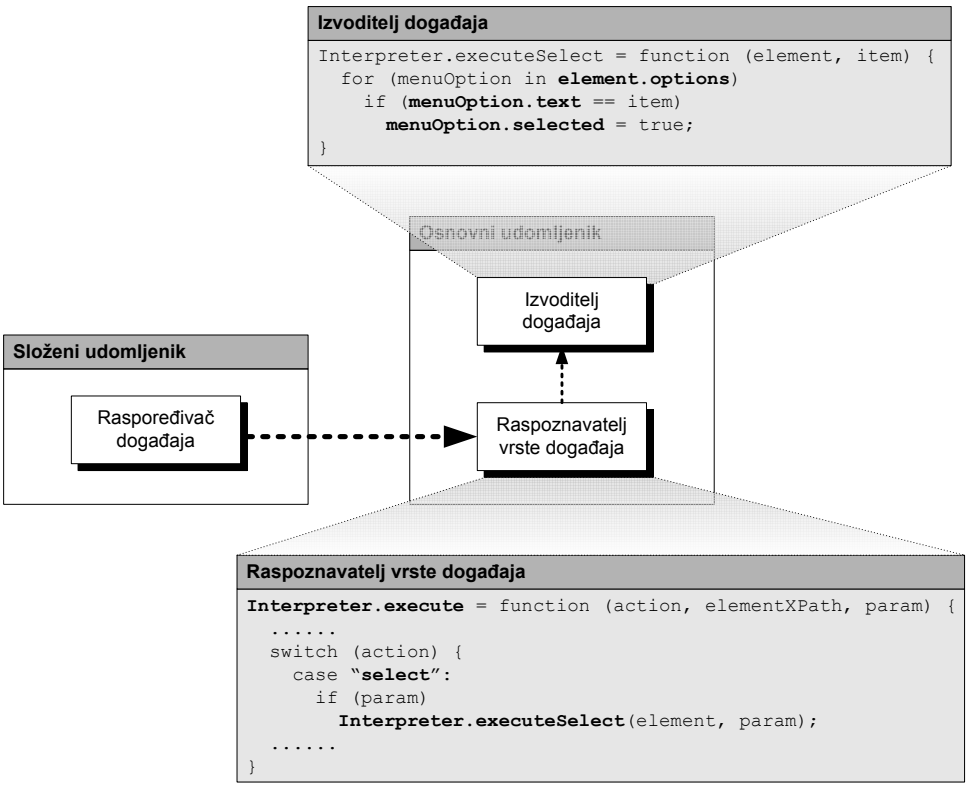
odabirStavke = **select** *stavka* **from** *element* **at** *udomljenik*
stavka = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**])*
element = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**])*
udomljenik = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**])*

Slika 9.31. Opći oblik zapisa događaja za odabir stavke iz padajućeg izbornika

Opći oblik zapisa događaja za odabir stavke iz padajućeg izbornika prikazan je regularnim definicijama na slici 9.31. Regularnom definicijom *odabirStavke* opisano je pravilo za oblikovanje nizova znakova koji predstavljaju valjano zapisane događaje za odabir stavke iz padajućeg izbornika. Regularna definicija *odabirStavke* definirana je ključnim riječima *select*, *from* i *at* te regularnim definicijama *stavka*, *element* i *udomljenik*. Ključne riječi *select* i *from* koriste se za jednoznačno prepoznavanje događaja za odabir stavke iz padajućeg izbornika. Regularnom definicijom *stavka* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje stavke padajućeg izbornika. Regularnom definicijom *element* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje padajućeg izbornika nad kojim se izvodi operacija odabira stavke. Regularnom definicijom *udomljenik* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje udomljenika nad kojim se izvodi operacija odabira stavke. Naslovljavanje stavke, padajućeg izbornika i udomljenika obavlja se nizovima znakova koji se sastoje od brojki te velikih i malih slova abecede. Ključna riječ *at* koristi se u svojstvu graničnika između naziva padajućeg izbornika i naziva udomljenika.

Prostor izvođenja

Izvođenje događaja za odabir stavke iz padajućeg izbornika prikazano je na slici 9.32. Prikazan je isječak programskog ostvarenja raspoznavatelja vrste događaja koji je zadužen za prepoznavanje događaja za odabir stavke iz padajućeg izbornika te programsko ostvarenje izvoditelja događaja. Komunikacija između složenog i osnovnog udomljenika je jednosmjerna jer izvođenjem događaja za odabir stavke iz padajućeg izbornika ne nastaju rezultati koje je potrebno vraćati složenom udomljeniku.



Slika 9.32. Prostor izvođenja događaja za odabir stavke iz padajućeg izbornika

Događaj za odabir stavke iz padajućeg izbornika raspoznaje se po ključnoj riječi *select*. Od događaja za obilježavanje izbornog polja, koji je označen istom ključnom riječi, razlikuje se po tome što, osim ključne riječi i *XPath* izraza za označavanje položaja izbornika unutar strukture *HTML* dokumenta osnovnog udomljenika, sadrži i dodatni parametar s nazivom izborničke stavke koja se odabire izvođenjem događaja. Nakon prepoznavanja događaja za odabir stavke iz padajućeg izbornika, raspoznavatelj vrste događaja pokreće izvoditelj događaja koji je ostvaren funkcijom *Interpreter.executeSelect*. Ulazni parametri funkcije izvoditelja događaja su pokazivač na ciljni element korisničkog sučelja u obliku padajućeg izbornika i naziv izborničke stavke koju je potrebno odabrati izvođenjem događaja.

Programsko oponašanje radnje odabira stavke iz padajućeg izbornika započinje pronalaženjem stavke zadanog naziva među stavkama izbornika. Pristup stavkama izbornika omogućen je putem programskog polja *options* koje sadrži popis izborničkih stavki. Usporedba naziva pojedinih izborničkih stavki sa zadanim nazivom stavke provodi se primjenom članske varijable *text* nad programskim objektom za rukovanje stavkama izbornika. Odabir pronađene izborničke stavke provodi se postavljanjem članske varijable *selected* nad programskim objektom za rukovanje stavkama izbornika na vrijednost logičke istine. Postavljanjem te varijable na vrijednost logičke istine, prethodno odabrana stavka izbornika postavlja se u stanje neoznačenosti. Programsko polje *options* te članske varijable *text* i *selected* pripadaju standardnom programskom sučelju objekta kojim se u jeziku *Javascript* rukuje elementima grafičkog korisničkog sučelja u obliku padajućih izbornika.

10 Upravljanje tokom podataka

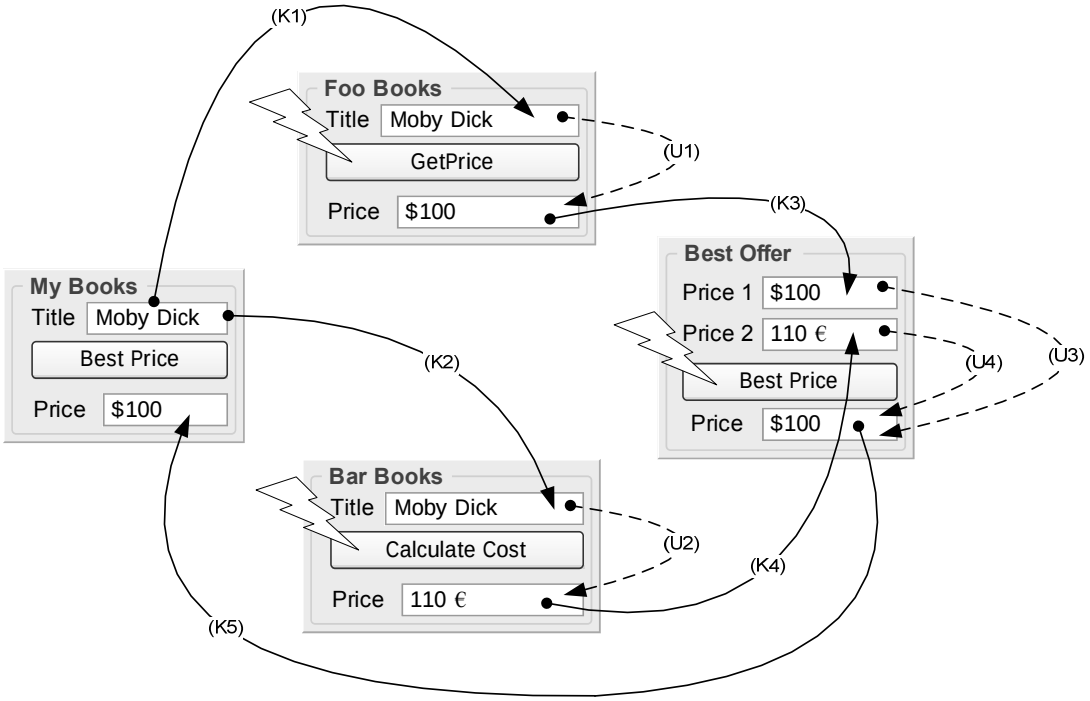
Događajima za pokretanje radnji nad udomljenicima koji su opisani u poglavlju 9, potrošačima je omogućeno programsko upravljanje izvođenjem pojedinačnih udomljenika. Međutim, za izgradnju primjenskih programa u kojima se informacije dobivene izvođenjem nekog od udomljenika koriste kao ulazni podaci u drugim udomljenicima, potrošaču je potrebno omogućiti oblikovanje programskog prijenosa podataka između različitih udomljenika. Programska paradigma za usložnjavanje udomljenika, stoga, sadrži mehanizme za uspostavu toka podataka (engl. *data flow*) između dvaju ili više različitih udomljenika.

U ovom poglavlju opisuje se postupak oblikovanja i skup događaja za uspostavu toka podataka nad udomljenicima. Poglavlje 10.1 donosi definiciju toka podataka u sustavu povezanih udomljenika. Definiran je ugrađeni tok podataka kojim se opisuje povezanost ulaznih i izlaznih elemenata grafičkog korisničkog sučelja unutar pojedinačnih udomljenika te korisnički tok podataka kojim se opisuje povezanost izlaznih elemenata korisničkog sučelja izvorišnih udomljenika s ulaznim elementima korisničkog sučelja odredišnih udomljenika. U poglavlju 10.2 opisana je potrošaču prilagođena metodologija za programsko ostvarenje toka podataka u sustavu udomljenika. S obzirom na to da je uobičajeni način prenošenja podataka između elemenata grafičkog korisničkog sučelja *World Wide Web* stranica i udomljenika primjena mehanizma preuzimanja i preslikavanja sadržaja (engl. *copy/paste*), potrošaču prilagođena metodologija za programsko ostvarenje toka podataka u sustavu udomljenika također je zasnovana na tom mehanizmu. U poglavlju 10.3 prikazana je analiza podudarnosti podataka sadržanih u različitim vrstama elemenata grafičkog

korisničkog sučelja udomljenika i mogućnost uspostave toka podataka između različitih vrsta elemenata.

10.1 Tok podataka u sustavu povezanih udomljenika

Tokom podataka u sustavu povezanih udomljenika određuje se način prijenosa informacija između pojedinih udomljenika tijekom izvođenja radnog tijeka definiranog od strane potrošača. Tokom podataka obuhvaćen je prijenos ulaznih podataka definiranih od strane potrošača od složenog udomljenika prema osnovnim udomljenicima, prijenos međurezultata između osnovnih udomljenika i prijenos završnih rezultata izvođenja radnog tijeka od osnovnih udomljenika prema složenom udomljeniku. Uspostavom toka podataka u sustavu udomljenika, međusobno nezavisni udomljenici postaju dio povezane cjeline.



Slika 10.1. Tok podataka u sustavu povezanih udomljenika

Slikom 10.1 prikazan je primjer toka podataka u sustavu povezanih udomljenika. U prikazanom primjeru, skup koji čine tri osnovna udomljenika, čiji su nazivi redom *Foo Books*, *Bar Books* i *Best Offer*, povezuje se u složeni udomljenik *My Books* s ciljem pronalaska najpovoljnije cijene knjige na tržištu. Udomljenici *Foo Books* i *Bar Books* koriste se za pristup udaljenim elektroničkim knjižarama. Putem tih dvaju udomljenika, potrošač za zadani naslov knjige dobiva informaciju o cijeni knjige u pojedinoj knjižari. Pod pretpostavkom da dvije elektroničke knjižare posluju na područjima gdje se platni promet

obavlja u različitim novčanim valutama, u radni tijek primjenskog programa potrebno je uključiti i udomljenik *Best Offer* za pretvorbu i usporedbu vrijednosti valuta koja na osnovi trenutnog tečaja određuje povoljniju cijenu knjige.

Složena funkcionalnost koja je postignuta objedinjavanjem funkcionalnosti triju osnovnih udomljenika izložena je na korištenje putem složenog udomljenika *My Books*. Korisničko sučelje složenog udomljenika sadrži polje za unos teksta *Title* za unos naslova knjige, tipku *Best Price* za pokretanje radnog tijeka za pretraživanje povoljnije cijene knjige te polje za prikaz teksta *Price* za prikaz konačnog rezultata izvođenja radnog tijeka u obliku povoljnije cijene za zadanu knjigu.

Tok podataka u sustavu povezanih udomljenika prikazan je usmjerenim linijama na slici 10.1. Naslov knjige koji je zapisan u obliku sadržaja polja za unos teksta *Title* složenog udomljenika *My Books* prenosi se do dvaju osnovnih udomljenika za pretraživanje elektroničkih knjižara, *Foo Books* i *Bar Books* (K1, K2). U svakom od tih dvaju udomljenika, podatak o naslovu knjige upisuje se u polje za unos teksta *Title*. Na osnovi primljenog podatka o naslovu knjige, svaki od dvaju udomljenika nezavisno pretražuje i izračunava cijenu knjige. Pokretanje procesa za pretraživanje i izračunavanje cijene knjige obavlja se pritiskom na tipku *Get Price* udomljenika *Foo Books*, odnosno *Calculate Cost* udomljenika *Bar Books*. Rezultati pretrage prikazuju se u obliku sadržaja polja za prikaz teksta *Price* (U1, U2). Podaci o cijenama knjige iz dvaju udomljenika prenose se do udomljenika za izračun povoljnije cijene knjige *Best Offer* (K3, K4). Cijene knjige koje su izražene u dvjema različitim novčanim valutama upisuju se u polja za unos teksta *Price 1* i *Price 2*. Na osnovi primljenih podataka o cijenama knjige, udomljenik *Best Offer* izračunava povoljniju cijenu s obzirom na trenutno važeći omjer vrijednosti dviju valuta. Pokretanje usporedbe vrijednosti valuta obavlja se pritiskom na tipku *Best Price* udomljenika *Best Offer*. Rezultat usporedbe prikazuje se u polju za prikaz teksta *Price* (U3, U4), nakon čega se prenosi do složenog udomljenika *My Books* i upisuje u polje za prikaz teksta *Price* kako bi se prikazao potrošaču kao krajnji rezultat izvođenja radnog tijeka (K5).

Složeni udomljenik koji nastaje oblikovanjem radnog tijeka nad skupom osnovnih udomljenika formalno se opisuje formulom (10.1)

$$SU = f(OU, TP) \tag{10.1}$$

gdje je *SU* funkcionalnost složenog udomljenika koja nastaje povezivanjem funkcionalnosti skupa osnovnih udomljenika, *OU* je skup funkcionalnosti međusobno nezavisnih i nepovezanih osnovnih udomljenika koji se koriste u postupku usložnjavanja, dok je *TP* tok

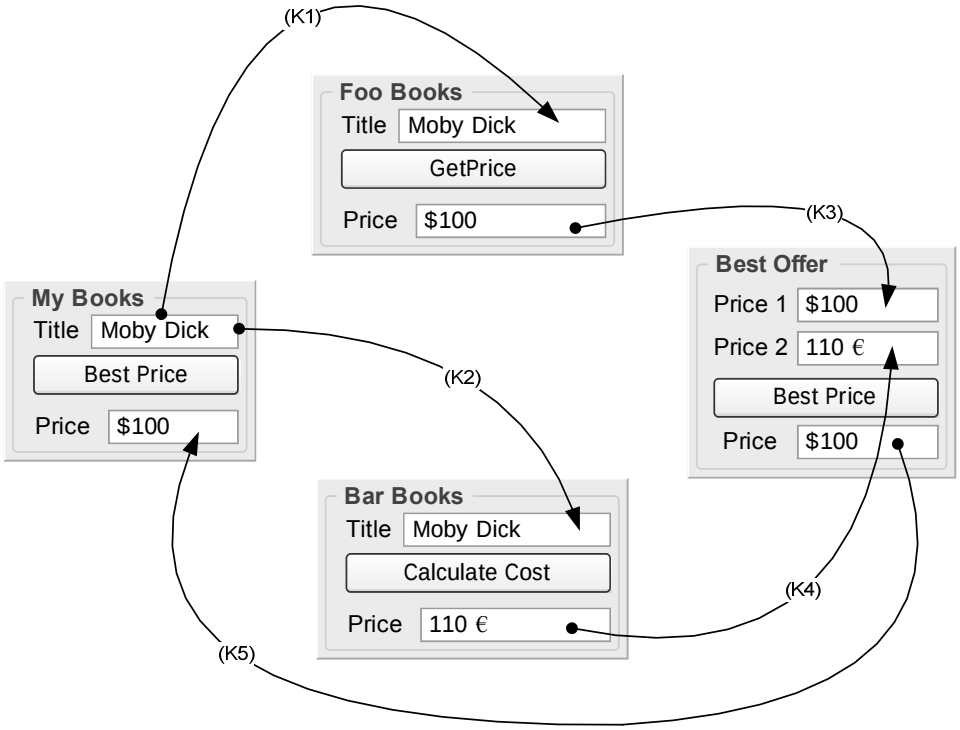
podataka definiran nad skupom udomljenika. Objedinjena funkcionalnost skupa osnovnih udomljenika koja je izložena putem složenog udomljenika funkcija je izabranog skupa osnovnih udomljenika i definiranog toka podataka. Uspostavom toka podataka *TP*, skup nezavisnih i nepovezanih osnovnih udomljenika *OU* povezuje se u cjelinu koja zajednički obavlja složeni proces po volji potrošača, a na korištenje je izložen putem složenog udomljenika *SU*.

Prema formuli 10.1, na funkcionalnost poosobljenog složenog udomljenika moguće je utjecati promjenom izbora osnovnih udomljenika ili promjenom toka podataka koji je definiran nad skupom izabranih osnovnih udomljenika. Iako korisnici programske paradigme za usložnjavanje udomljenika nemaju utjecaja na funkcionalnosti pojedinačnih osnovnih udomljenika, mogućnošću izbora skupa udomljenika i samostalnog definiranja toka podataka nad izabranim skupom udomljenika omogućeno je ostvarenje različitih funkcionalnosti složenog udomljenika. Mogućnost izbora skupa udomljenika i mogućnost definiranja proizvoljnog toka podataka nad izabranim skupom udomljenika predstavljaju ključne elemente izražajnosti programske paradigme za usložnjavanje udomljenika.

Tok podataka u sustavu povezanih udomljenika sastoji se dviju vrsta toka podataka. Prvu vrstu toka podataka čini *korisnički tok podataka* kojim se definira prijenos podataka između različitih udomljenika, a na slici 10.1 označen je oznakama K1 do K5. Drugu vrstu toka podataka čini *ugrađeni tok podataka* koji je određen podatkovnim zavisnostima elemenata grafičkog korisničkog sučelja koji pripadaju istom udomljeniku. Svojstva i razlike između ovih dviju vrsta toka podataka opisane su u nastavku poglavlja. Korisnički tok podataka opisan je u poglavlju 10.1.1, dok je ugrađeni tok podataka opisan u poglavlju 10.1.2.

10.1.1 Korisnički tok podataka

Povezivanje nezavisnih udomljenika u složeni udomljenik prema potrebama potrošača omogućeno je definiranjem korisničkog toka podataka. S obzirom na to da su pojedini udomljenici koji se koriste za oblikovanje i izgradnju složenog udomljenika međusobno nezavisni, od potrošača se očekuje da ih poveže samostalnim definiranjem korisničkog toka podataka. Korisničkim tokom podataka definira se način prijenosa informacija između različitih i međusobno nezavisnih udomljenika koji se koriste za izgradnju složenog udomljenika. Korisnički tok podataka definira se prema zakonitostima radnog tijeka definiranog od strane potrošača u skladu s ulogom pojedinog udomljenika tijekom izvođenja složenog udomljenika.

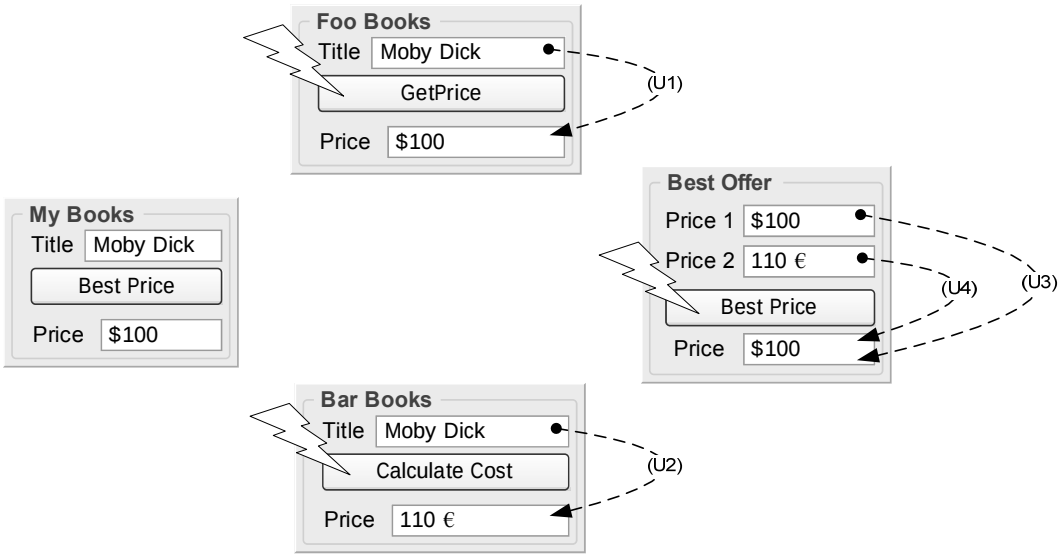


Slika 10.2. Korisnički tok podataka u sustavu povezanih udomljenika

Slikom 10.2 prikazan je primjer usložnjavanja udomljenika sa slike 10.1 s izdvojenim korisničkim tokom podataka. Korisničkim tokom podataka određen je prijenos informacije o naslovu knjige od složenog udomljenika *My Books* prema osnovnim udomljenicima *Foo Books* (K1) i *Bar Books* (K2), prijenos informacija o cijenama knjiga od osnovnih udomljenika *Foo Books* i *Bar Books* prema osnovnom udomljeniku *Best Offer* (K3, K4) te prijenos informacije o povoljnijoj cijeni knjige od osnovnog udomljenika *Best Offer* prema složenom udomljeniku *My Books* (K5). Korisnički tok podataka oblikuje se događajima za uspostavu toka podataka koji su opisani u poglavlju 10.2.

10.1.2 Ugrađeni tok podataka

Ugrađenim tokom podataka naziva se tok podataka unutar pojedinog udomljenika te tok podataka između udomljenika i pozadinskog procesa izloženog putem udomljenika. Ugrađeni tok podataka posljedica je djelovanja operacija nad aktivacijskim elementima grafičkog korisničkog sučelja udomljenika. Određen je unutarnjom primjenskom logikom pojedinog udomljenika, a djelovanje mu je ograničeno na elemente korisničkog sučelja pojedinog udomljenika. S obzirom na to da se pokreće odzivom udomljenika na radnje izvršene putem aktivacijskih elemenata korisničkog sučelja, ugrađeni tok podataka osnovnih udomljenika vidljiv je potrošaču, ali nije pod njegovim izravnim nadzorom niti utjecajem.



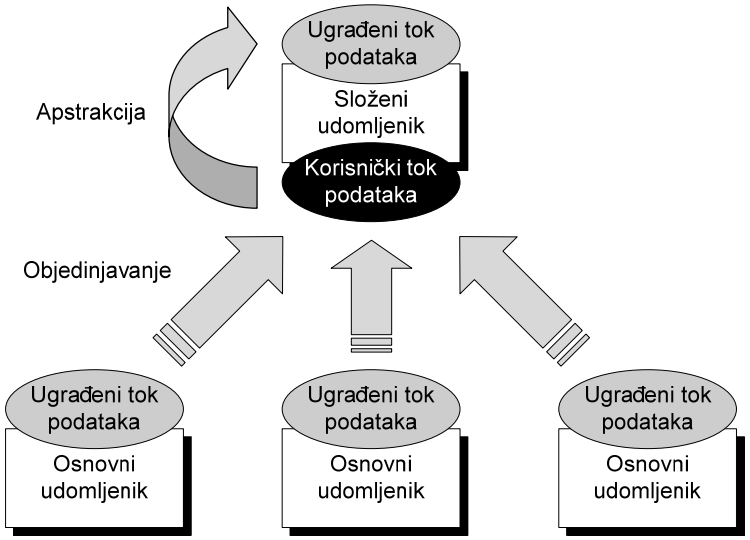
Slika 10.3. Ugrađeni tok podataka u sustavu povezanih udomljenika

Slikom 10.3 prikazan je primjer usložnjavanja udomljenika sa slike 10.1 s izdvojenim ugrađenim tokom podataka. Ugrađenim tokom podataka unutar osnovnih udomljenika *Foo Books* i *Bar Books* određena je zavisnost podatka zapisanog u polju za prikaz teksta *Price* o podatku zapisanom u polju za unos teksta *Title* (U1, U2). Ugrađeni tok podataka udomljenika *Foo Books* pokreće se djelovanjem radnje pritiska na tipku *Get Price*, dok se ugrađeni tok podataka udomljenika *Bar Books* pokreće djelovanjem radnje pritiska na tipku *Calculate Cost*. Djelovanje ugrađenog toka podataka u oba udomljenika očituje se u osvježavanju sadržaja odlomka za prikaz teksta *Price* podatkom o cijeni knjige čiji je naslov zadan u obliku sadržaja polja za unos teksta *Title*. Ugrađenim tokom podataka unutar osnovnog udomljenika *Best Offer* određena je zavisnost podatka zapisanog u polju za prikaz teksta *Price* o podacima zapisanim u poljima za unos teksta *Price 1* i *Price 2* (U3, U4). Ugrađeni tok podataka pokreće se djelovanjem radnje pritiska na tipku *Best Price*. Djelovanje ugrađenog toka podataka očituje se u osvježavanju sadržaja polja za prikaz teksta *Price* podatkom o povoljnijoj cijeni knjige.

10.1.3 Ugrađeni tok podataka složenog udomljenika

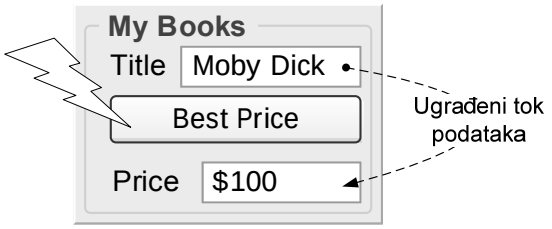
Objedinjavanjem korisničkog i ugrađenog toka podataka na razini osnovnih udomljenika nastaje ugrađeni tok podataka na razini složenog udomljenika, kao što je prikazano slikom 10.4. Korisničkim tokom podataka, koji je dio definicije radnog tijeka složenog udomljenika, postiže se objedinjavanje međusobno nezavisnih ugrađenih tokova podataka osnovnih udomljenika. Učinci toka podataka na razini skupa povezanih osnovnih

udomljenika izloženi su potrošaču putem složenog udomljenika. Ugrađeni tok podataka složenog udomljenika je apstrakcija korisničkog i ugrađenog toka podataka na razini skupa osnovnih udomljenika.



Slika 10.4. Objedinjavanje i apstrakcija toka podataka u sustavu povezanih udomljenika

Slikom 10.5 prikazan je ugrađeni tok podataka unutar složenog udomljenika *My Books*. Ugrađeni tok podataka tog udomljenika je apstrakcija korisničkog toka podataka prikazanog slikom 10.2 koji je definiran nad skupom osnovnih udomljenika *Foo Books*, *Bar Books* i *Best Offer* te nad složenim udomljenikom *My Books* te ugrađenih tokova podataka unutar osnovnih udomljenika *Foo Books*, *Bar Books* i *Best Offer* koji su prikazani slikom 10.3. Ugrađenim tokom podataka unutar složenog udomljenika *My Books* određena je zavisnost podatka prikazanog u polju za prikaz teksta *Price* o sadržaju polja za unos teksta *Title*. Ugrađeni tok podataka složenog udomljenika *My Books* pokreće se pritiskom na aktivacijski element korisničkog sučelja u obliku tipke *Best Price*. Djelovanje ugrađenog toka podataka očituje se u osvježavanju sadržaja polja za prikaz teksta *Price* podatkom o povoljnijoj cijeni knjige čiji je naslov zadan u obliku sadržaja polja za unos teksta *Title*.



Slika 10.5. Ugrađeni tok podataka složenog udomljenika

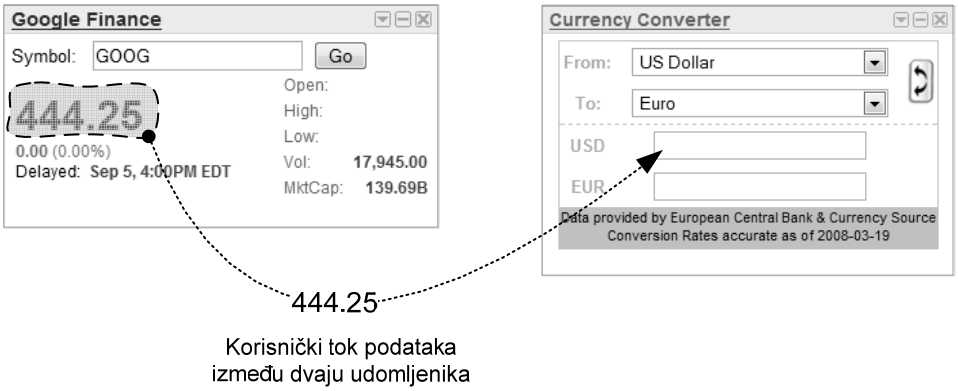
Načelom objedinjavanja i apstrakcije toka podataka omogućeno je višerazinsko usložnjavanje udomljenika. Iako su stvarne podatkovne zavisnosti složenog udomljenika određene složenim tokom podataka koji se sastoji od korisničkog i ugrađenog toka podataka u povezanom skupu udomljenika, ugrađeni tok podataka složenog udomljenika predstavljen je potrošaču apstraktnim ugrađenim tokom podataka na razini složenog udomljenika. Apstrakcija korisničkog i ugrađenog toka podataka u sustavu povezanih udomljenika pojednostavljenim ugrađenim tokom podataka složenog udomljenika omogućuje primjenu složenog udomljenika bez poznavanja pojedinosti njegova programskog ostvarenja. Time je omogućeno izjednačavanje svojstava osnovnih i složenih udomljenika s gledišta primjene. Složeni udomljenik moguće je koristiti za krajnju primjenu na jednak način kako i bilo koji drugi osnovni udomljenik. Osim toga, apstrakcijom toka podataka omogućeno je višerazinsko usložnjavanje udomljenika, odnosno primjena složenog udomljenika u svojstvu osnovnog udomljenika na višoj razini usložnjavanja.

10.2 Programsko ostvarenje toka podataka

Ugrađeni tok podataka pokreće se izvođenjem radnji nad aktivacijskim elementima grafičkog korisničkog sučelja udomljenika. U programskoj paradigmi za usložnjavanje udomljenika, takve se radnje opisuju događajima za upravljanje radom udomljenika koji su opisani u poglavlju 9 pa za programsko ostvarenje ugrađenog toka podataka nije potrebno koristiti posebno oblikovane događaje programskog jezika. Za programsko ostvarenje toka podataka u okviru programske paradigme za usložnjavanje udomljenika je, stoga, dovoljno oblikovati događaj za uspostavu korisničkog toka podataka.

Prostor oblikovanja

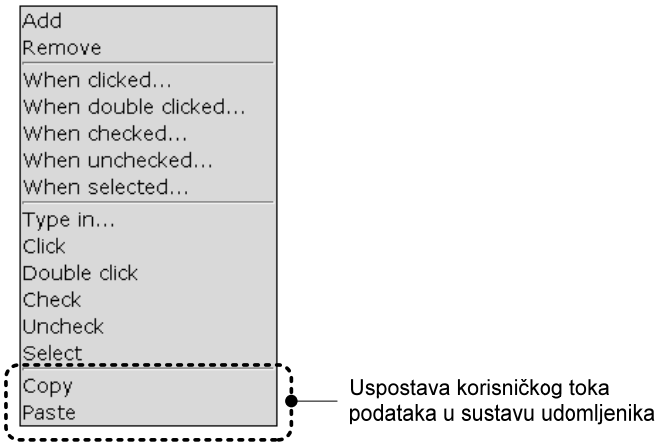
Događaj za uspostavu toka podataka između dvaju nezavisnih udomljenika primjenjuje se za prijenos informacija dostupnih putem grafičkog korisničkog sučelja između tih dvaju udomljenika. Oblikovanje događaja za uspostavu toka podataka sastoji se od određivanja dvaju elementa grafičkog korisničkog sučelja između kojih se uspostavlja tok podataka te smjera prijenosa informacije. U primjeru na slici 10.6 prikazano je oblikovanje događaja za prijenos sadržaja odlomka za prikaz teksta s cijenom dionice sa udomljenika *Google Finance* u polje za unos teksta *USD* udomljenika *Currency Converter*.



Slika 10.6. Oblikovanje događaja za uspostavu korisničkog toka podataka

Prostor izgradnje

Uobičajen, prihvatljiv i potrošaču razumljiv način prijenosa informacija između elemenata grafičkog korisničkog sučelja tijekom ručne uporabe udomljenika je primjena mehanizma preuzimanja sadržaja iz izvorišnog elementa (engl. *copy*) i njegova preslikavanja u ciljni element korisničkog sučelja (engl. *paste*) posredstvom privremenog međuspremnika sadržaja (engl. *clipboard*). Kako bi se programsko ostvarenje toka podataka oslonilo na iskustva stečena tijekom ručne uporabe udomljenika, uspostava toka podataka u okviru programske paradigme za usložnjavanje udomljenika također se zasniva na mehanizmu preuzimanja i preslikavanja sadržaja.

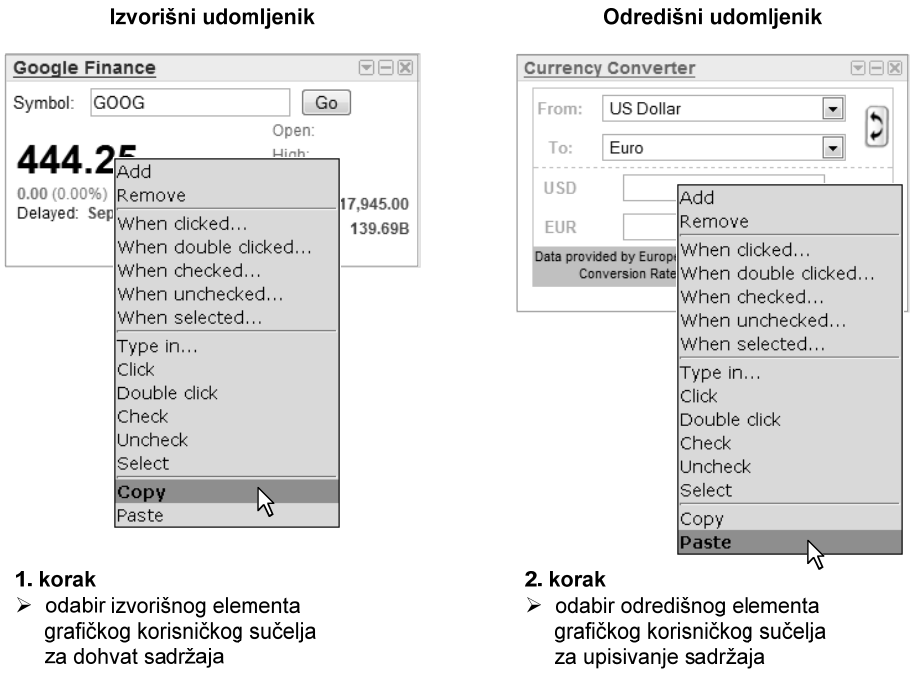


Slika 10.7. Primjena grafičkog izbornika za uspostavu korisničkog toka podataka u sustavu udomljenika

Događaji za programsko ostvarenje toka podataka u sustavu udomljenika izgrađuju se primjenom grafičkog plivajućeg izbornika. Prostor izgradnje događaja za uspostavu toka podataka u sustavu udomljenika prikazan je slikom 10.7. Plivajući izbornik sadrži dvije

stavke za izgradnju događaja za uspostavu toka podataka. Stavka *Copy* koristi se za odabir izvorišnog elementa grafičkog korisničkog sučelja udomljenika čiji je sadržaj potrebno preuzeti u privremeni međuspremnik. Stavka *Paste* koristi se za odabir odredišnog elementa grafičkog korisničkog sučelja udomljenika u koji je potrebno upisati prethodno preuzeti sadržaj.

Izgradnja događaja za uspostavu toka podataka prikazana je slikom 10.8. Prostor izgradnje događaja čine izvorišni i odredišni udomljenik te potpomagana okolina za izgradnju događaja u obliku plivajućeg izbornika. Događaj za uspostavu toka podataka u sustavu udomljenika izgrađuje se u dva koraka.



Slika 10.8. Izgradnja događaja za uspostavu korisničkog toka podataka u sustavu udomljenika

U prvom se koraku odabire izvorišni element grafičkog korisničkog sučelja udomljenika iz kojeg se tijekom izvođenja događaja preuzima sadržaj. Tijekom odabira izvorišnog elementa, koristi se grafičko korisničko sučelje izvorišnog udomljenika. Odabir elementa za dohvat sadržaja obavlja se odabirom stavke *Copy* iz plivajućeg izbornika.

U drugom se koraku odabire odredišni element grafičkog korisničkog sučelja udomljenika u koji se tijekom izvođenja događaja upisuje sadržaj preuzet u prvom koraku. Tijekom upisivanja sadržaja, koristi se grafičko korisničko sučelje odredišnog udomljenika. Odabir elementa za upisivanje sadržaja obavlja se odabirom stavke *Paste* iz plivajućeg izbornika.

Prostor prikaza

Događaj za uspostavu toka podataka izgrađen u prostoru izgradnje zapisuje se unutar složenog udomljenika i prikazuje potrošaču slovčanim zapisom u obliku

copy Price at Google Finance to USD at Currency Converter

Prikazanim slovčanim zapisom opisan je događaj kojim se sadržaj elementa grafičkog korisničkog sučelja *Price* udomljenika *Google Finance* prenosi do udomljenika *Currency Converter*, gdje se upisuje u element grafičkog korisničkog sučelja *USD*. Slovčani zapis za prikaz događaja za uspostavu toka podataka u sustavu udomljenika sastoji se od podataka o izvorišnom i odredišnom elementu korisničkog sučelja te smjeru prijenosa sadržaja. Podaci o nazivu izvorišnog, odnosno odredišnog elementa korisničkog sučelja sastoje se od naziva elementa i naziva udomljenika iz kojeg se preuzima, odnosno u koji se prenosi sadržaj.

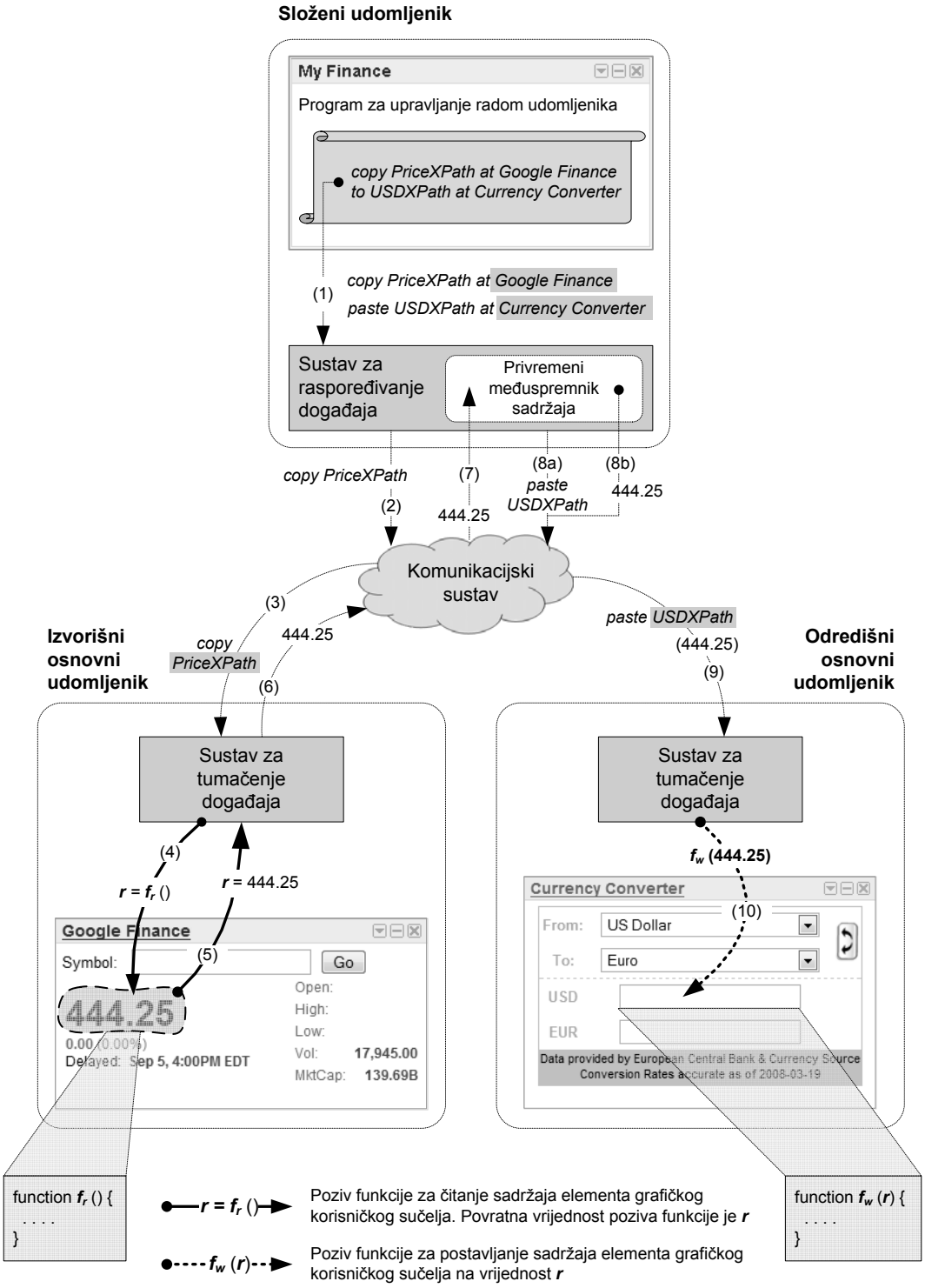
tokPodataka = **copy** *element* **at** *udomljenik* **to** *element* **at** *udomljenik*
element = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**])*
udomljenik = ([**A**..**Z**]+[**a**..**z**]+[**0**..**9**])*

Slika 10.9. Opći oblik zapisa događaja za uspostavu toka podataka u sustavu udomljenika

Opći oblik zapisa događaja za uspostavu toka podataka u sustavu udomljenika prikazan je regularnim definicijama na slici 10.9. Regularnom definicijom *tokPodataka* opisano je pravilo za oblikovanje nizova znakova koji predstavljaju valjano zapisane događaje za uspostavu toka podataka u sustavu udomljenika. Regularna definicija *tokPodataka* definirana je ključnim riječima *copy*, *to* i *at* te regularnim definicijama *element* i *udomljenik*. Ključne riječi *copy* i *to* koriste se za jednoznačno prepoznavanje događaja za uspostavu toka podataka. Regularnom definicijom *element* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje izvorišnog i odredišnog elementa korisničkog sučelja. Regularnom definicijom *udomljenik* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje udomljenika između kojih se uspostavlja tok podataka. Naslovljavanje elemenata korisničkog sučelja i udomljenika obavlja se nizovima znakova koji se sastoje od brojki te velikih i malih slova abecede. Ključna riječ *at* koristi se u svojstvu graničnika između naziva elementa korisničkog sučelja i naziva udomljenika.

Prostor izvođenja

Prostor izvođenja događaja za uspostavu toka podataka u sustavu udomljenika sastoji se od složenog udomljenika te izvorišnog i odredišnog osnovnog udomljenika, kao što je prikazano na slici 10.10.



Slika 10.10. Prostor izvođenja događaja za uspostavu toka podataka u sustavu udomljenika

Složeni udomljenik sadrži program za upravljanje radom udomljenika koji sadrži događaj za uspostavu toka podataka. Unutar složenog udomljenika, događaj je zapisan slično kao što je prikazan u prostoru prikaza događaja. U zapisu događaja, simbolička imena izvorišnog i odredišnog elementa grafičkog korisničkog sučelja zamijenjena su *XPath* izrazima kojima su određeni položaji tih elemenata unutar strukture *HTML* dokumenata izvorišnog i odredišnog osnovnog udomljenika. Izvorišni osnovni udomljenik sadrži grafičko korisničko sučelje s izvorišnim elementom od kojeg je potrebno dohvatiti sadržaj i prenijeti ga do odredišnog osnovnog udomljenika. Odredišni osnovni udomljenik sadrži grafičko korisničko sučelje s odredišnim elementom u koji je potrebno upisati preneseni sadržaj.

Osim grafičkog korisničkog sučelja, udomljenici sadrže i programsku infrastrukturu za izvođenje događaja za uspostavu toka podataka koja se sastoji od sustava za raspoređivanje događaja, sustava za tumačenje događaja i komunikacijskog sustava. Pojedinosti načela rada i programskog ostvarenja tih sustava opisane su u poglavlju 8.

Izvođenje događaja za uspostavu toka podataka odvija se u deset koraka. Sustav za raspoređivanje događaja dohvaća slovčani zapis događaja (1). Nakon dohvata, događaj se rastavlja na dva jednostavna pomoćna događaja, od kojih se prvi koristi za čitanje sadržaja izvorišnog elementa, a drugi za upisivanje sadržaja u odredišni element korisničkog sučelja. Na osnovi naziva izvorišnog udomljenika u pomoćnom događaju za čitanje sadržaja, zapis događaja se komunikacijskim sustavom prosljeđuje do izvorišnog osnovnog udomljenika (2, 3), gdje ga prihvaća sustav za tumačenje događaja. Na osnovi ključne riječi *copy* kojom je jednoznačno određen pomoćni događaj za čitanje sadržaja, sustav za tumačenje događaja donosi zaključak o vrsti događaja. Nakon utvrđivanja vrste događaja, sustav za tumačenje događaja izvorišnog osnovnog udomljenika parsira *XPath* izraz kojim je određen položaj izvorišnog elementa grafičkog korisničkog sučelja unutar strukture *HTML* dokumenta izvorišnog osnovnog udomljenika. Dohvat sadržaja obavlja se pozivom funkcije f_r za čitanje sadržaja zadanog elementa grafičkog korisničkog sučelja udomljenika (4). Pročitana vrijednost sadržaja prosljeđuje se komunikacijskim sustavom do složenog udomljenika (5, 6), gdje se sprema u privremeni međuspremnik sadržaja (7). Upisivanjem sadržaja u privremeni međuspremnik unutar složenog udomljenika završava izvođenje pomoćnog događaja za čitanje sadržaja izvorišnog elementa grafičkog korisničkog sučelja.

Izvođenje drugog pomoćnog događaja kojim se prethodno dohvaćena vrijednost upisuje u odredišni element grafičkog korisničkog sučelja udomljenika započinje osmim korakom. Sustav za raspoređivanje događaja složenog udomljenika dohvaća slovčani zapis pomoćnog događaja za upisivanje sadržaja (8a). Istovremeno, iz privremenog

međuspremnika sadržaja dohvaća se vrijednost sadržaja koju je potrebno upisati u odredišni element korisničkog sučelja, a prethodno je preuzeta iz izvorišnog elementa (8b). Na osnovi naziva odredišnog udomljenika u zapisu pomoćnog događaja, zapis događaja i vrijednost sadržaja zapisana u međuspremniku komunikacijskim se sustavom prosljeđuju do odredišnog osnovnog udomljenika (9), gdje ih prihvaća sustav za tumačenje događaja. Na osnovi ključne riječi *paste* kojom je jednoznačno određen pomoćni događaj za upisivanje sadržaja, sustav za tumačenje događaja donosi zaključak o vrsti događaja. Nakon utvrđivanja vrste događaja, sustav za tumačenje događaja odredišnog osnovnog udomljenika parsira *XPath* izraz kojim je određen položaj odredišnog elementa grafičkog korisničkog sučelja unutar strukture *HTML* dokumenta odredišnog osnovnog udomljenika. Upisivanje sadržaja obavlja se pozivom funkcije f_w za postavljanje sadržaja odredišnog elementa grafičkog korisničkog sučelja udomljenika na zadanu vrijednost (10).

10.3 Podudarnost oblika podataka

Događajem za uspostavu toka podataka u sustavu povezanih udomljenika određeni su izvorišni i odredišni element grafičkog korisničkog sučelja između kojih je potrebno obaviti prijenos sadržaja. S obzirom na podudarnost vrste izvorišnog i odredišnog elementa korisničkog sučelja, tijekom izvođenja događaja moguća je pojava dvaju slučaja. Ako su izvorišni i odredišni element korisničkog sučelja istovrsni, onda ta dva elementa sadrže isti oblik sadržaja. Primjerice, ako se izvorišni i odredišni element korisničkog sučelja pojavljuju u obliku polja za unos teksta, onda oba elementa sadrže informacije u obliku niza znakova. S druge strane, ako su izvorišni i odredišni element korisničkog sučelja raznovrsni, onda su sadržaji tih dvaju elemenata podudarni ili nepodudarni. Primjerice, ako se jedan od elemenata korisničkog sučelja pojavljuje u obliku poveznice, a drugi u obliku polja za unos teksta, onda ta dva elementa sadrže podudarni oblik sadržaja u obliku niza znakova. Međutim, ako se jedan od elemenata korisničkog sučelja pojavljuje u obliku slikovnog elementa, a drugi u obliku polja za unos teksta, onda ta dva elementa sadrže nepodudarne oblike sadržaja.

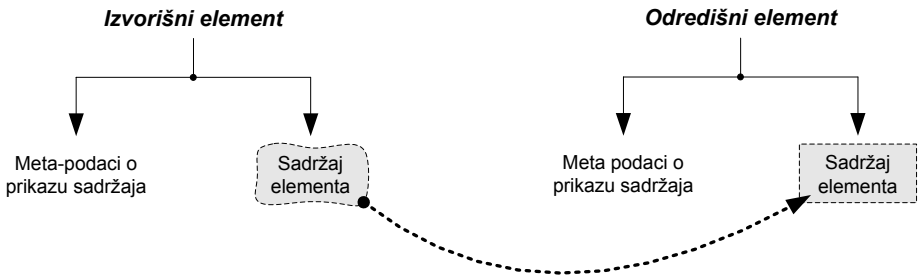
Tok podataka u sustavu udomljenika moguće je uspostaviti između istovrsnih elemenata grafičkog korisničkog sučelja te između raznovrsnih elemenata s podudarnim oblikom sadržaja. Tablicom 10.1 prikazana je povezanost oblika sadržaja s vrstama elemenata grafičkog korisničkog sučelja. Događajima programskog jezika za uspostavu toka podataka u sustavu udomljenika moguće je prenositi sadržaj u obliku niza znakova između polja za unos teksta, odjeljaka za prikaz teksta, slovčanih oznaka, poveznica i padajućih izbornika. Slikovni sadržaj moguće je prenositi između elemenata grafičkog korisničkog

sučelja u obliku slika. Vrijednosti logičke istinitosti moguće je prenositi između elemenata grafičkog korisničkog sučelja u obliku označnih i izbornih polja. Događaji za uspostavu toka podataka koji sadrže raznovrsne elemente grafičkog korisničkog sučelja s nepodudarnim oblicima sadržaja nisu elementi programskog jezika za usložnjavanje udomljenika i smatraju se neispravnim događajima.

Tablica 10.1. Povezanost oblika sadržaja i elemenata grafičkog korisničkog sučelja udomljenika

Oblik sadržaja	Element korisničkog sučelja
Niz znakova	Polje za unos teksta Odjeljak za prikaz teksta Slovčana oznaka Poveznica Padajući izbornik
Slikovni sadržaj	Slika
Logička istina/Logička laž	Označno polje Izborno polje
∅	Tipka

Na osnovi podudarnosti oblika sadržaja i vrste elemenata grafičkog korisničkog sučelja, definirano je pravilo za programsko ostvarenje funkcija za dohvat i postavljanje sadržaja elemenata grafičkog korisničkog sučelja udomljenika. Te dvije funkcije koriste se tijekom izvođenja događaja za uspostavu toka podataka u sustavu udomljenika, a na slici 10.10 označene su oznakama f_r i f_w. Slika 10.11 prikazuje shematski prikaz programskog ostvarenja tih dviju funkcija.



Slika 10.11. Programsko ostvarenje funkcija za dohvat i postavljanje sadržaja elemenata grafičkog korisničkog sučelja udomljenika

Bilo koji element grafičkog korisničkog sučelja udomljenika moguće je rastaviti na dva skupa podataka. Prvi skup podataka sadrži meta-podatke o prikazu primjenskog sadržaja

koji je predstavljen promatranim elementom grafičkog korisničkog sučelja. Drugi skup podataka sadrži primjenski sadržaj koji se putem elementa grafičkog korisničkog sučelja prikazuje prema pravilima koja su definirana meta-podacima. Funkciju f_r za dohvat sadržaja izvorišnog elementa grafičkog korisničkog sučelja potrebno je ostvariti na način da se njezinim izvođenjem obavlja dohvat isključivo primjenskog sadržaja ciljnog elementa, bez dohvaćanja meta-podataka o prikazu sadržaja. Funkciju f_w za postavljanje sadržaja odredišnog elementa grafičkog korisničkog sučelja potrebno je ostvariti na način da se njezinim izvođenjem postavlja vrijednost primjenskog sadržaja ciljnog elementa, dok meta-podaci o prikazu sadržaja ostaju nepromijenjeni.

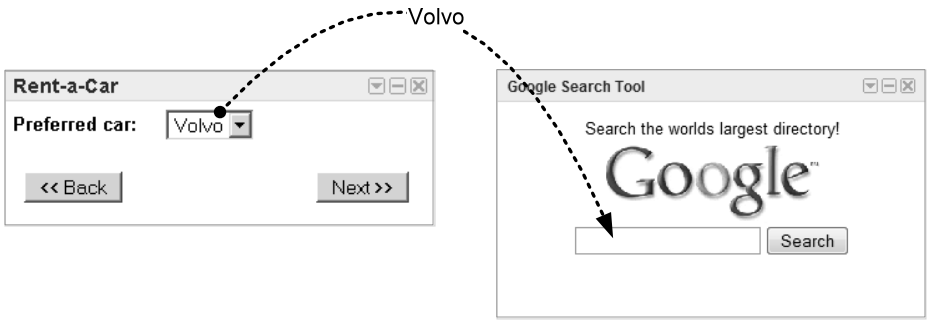
`<input id="GF185symbol" type="text" value="GOOG"/>`	polje za unos teksta
`<span id="vGF185_1">444.25</span>`	odjeljak za prikaz teksta
`<p id="GF185desc" class="desc">Sep 5, 5:00PM EST</p>`	odjeljak za prikaz teksta
`<label id="GF185item" for="GF185symbol">Symbol</label>`	slovčana oznaka
`<a href="/finance?client=ig&q=GOOG" target="_blank">GOOG</a>`	poveznica

`<select id="RC190carType" size="1">`
`  <option value="1">Opel</option>`
`  <option value="2" selected="yes">Volvo</option>`
`  <option value="3">Saab</option>`
`  <option value="4">BMW</option>`
`</select>`
	padajući izbornik

`<img height"75" width="250"` `  src="http://chart.apis.google.com/chart.gif"/>`	slika
`<input id="cardType" type="radio" value="Visa" checked="yes"/>`	izborno polje
`<input id="region" type="checkbox" value="Europe" checked="no"/>`	označno polje

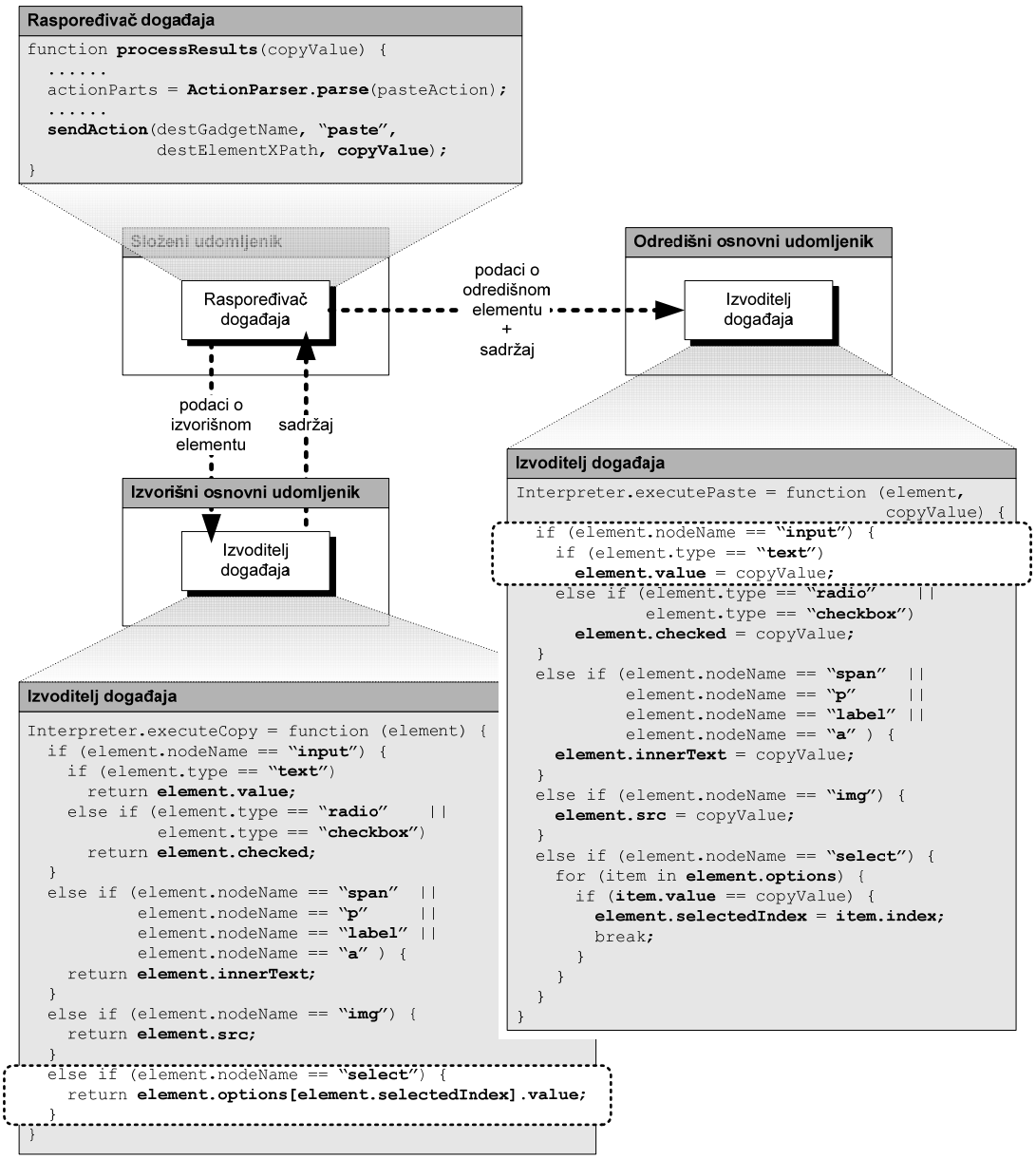
Slika 10.12. Zapis elemenata grafičkog korisničkog sučelja označnim jezikom *HTML*

Slikom 10.12 prikazan je zapis elemenata grafičkog korisničkog sučelja označnim jezikom *HTML* koji se koristi za izgradnju udomljenika. Normalnim tekstom prikazani su meta-podaci koje preglednik *World Wide Web* sadržaja koristi za prikaz elemenata na zaslonu računala. Masno otisnutim slovima prikazan je sadržaj pojedinog elementa grafičkog korisničkog sučelja.



Slika 10.13. Tok podataka između raznovrsnih elemenata grafičkog korisničkog sučelja udomljenika

Programsko ostvarenje funkcija za dohvat i postavljanje sadržaja elemenata grafičkog korisničkog sučelja udomljenika prema shematskom prikazu prikazanom na slici 10.11 omogućava uspostavu toka podataka između raznovrsnih elemenata grafičkog korisničkog sučelja udomljenika pod uvjetom da elementi sadrže podudarni oblik sadržaja. Slikom 10.13 prikazan je primjer prijenosa sadržaja, odnosno uspostave toka podataka između elementa korisničkog sučelja u obliku padajućeg izbornika i polja za unos teksta. Iako su to dva raznovrsna elementa, među njima je moguće uspostaviti tok podataka jer oba elementa sadrže podudarni primjenski sadržaj u obliku niza znakova.



Slika 10.14. Programsko ostvarenje podsustava za izvođenje događaja za uspostavu toka podataka u sustavu udomljenika

Slikom 10.14 prikazano je programsko ostvarenje funkcije za dohvat sadržaja izvorišnog elementa grafičkog korisničkog sučelja i funkcije za postavljanje sadržaja odredišnog elementa grafičkog korisničkog sučelja. Dohvat sadržaja izvorišnog elementa ostvaren je funkcijom *Interpreter.executeCopy* koja je dio programskog ostvarenja izvoditelja događaja izvorišnog osnovnog udomljenika. Postavljanje sadržaja odredišnog elementa ostvareno je funkcijom *Interpreter.executePaste* koja je dio programskog ostvarenja izvoditelja događaja odredišnog osnovnog udomljenika. S obzirom na to da bilo koji udomljenik može poprimiti ulogu izvorišnog i odredišnog udomljenika, obje funkcije ugrađene su u svaki osnovni udomljenik. Funkcije su ostvarene u programskom jeziku *Javascript*, koristeći standardno programsko sučelje za rukovanje *HTML* dokumentima.

Funkcija *Interpreter.executeCopy* za dohvat sadržaja izvorišnog elementa kao ulazni parametar prima pokazivač na izvorišni element grafičkog korisničkog sučelja udomljenika. Izvođenje funkcije započinje ispitivanjem vrste elementa. Za ispitivanje vrste elementa koristi se članska varijabla *nodeName* kojom se u jeziku *Javascript* pristupa nazivu oznake elementa grafičkog korisničkog sučelja u označnom jeziku *HTML* (engl. *HTML tag*). Oznake pojedinih vrsta elemenata grafičkog korisničkog sučelja u označnom jeziku *HTML* prikazane su slikom 10.12. Nakon utvrđivanja vrste elementa, koristi se odgovarajuće programsko sučelje za pristup sadržaju elementa. Za pristup sadržaju polja za unos teksta koristi se članska varijabla *value*. Za pristup sadržaju izbornog i označnog polja koristi se članska varijabla *checked*. Za pristup sadržaju odjeljka za prikaz teksta, slovčane oznake i poveznice koristi se članska varijabla *innerText*. Za pristup sadržaju elementa u obliku slike koristi se članska varijabla *src*. U toj varijabli sadržana je adresa poslužitelja s koje preglednik *World Wide Web* sadržaja učitava slikovni sadržaj. Za pristup stavkama padajućeg izbornika koristi se memorijsko polje *option*, odabranoj stavci pristupa se putem varijable *selectedIndex*, dok se pristup nazivu odabrane stavke ostvaruje primjenom članske varijable *value*.

Povratna vrijednost funkcije *Interpreter.executeCopy* je sadržaj izvorišnog elementa grafičkog korisničkog sučelja, bez meta-podataka za prikaz sadržaja na zaslonu računala. Sadržaj pojedinih vrsta elemenata grafičkog korisničkog sučelja istaknut je na slici 10.12 masno otisnutim slovima. Rezultat izvođenja funkcije vraća se mehanizmom povratnog poziva (engl. *callback*) raspoređivaču događaja u složeni udomljenik pozivom funkcije *processResults*. Ulazni parametar povratnog poziva je vrijednost sadržaja izvorišnog elementa grafičkog korisničkog sučelja dobiven izvođenjem funkcije *Interpreter.executeCopy*. Razlog primjene i način rada mehanizma povratnog poziva opisan je u poglavlju 8 u okviru opisa programskog ostvarenja komunikacijskog sustava za komunikaciju udomljenika unutar udomiteljske stranice.

Nakon što od izvoditelja događaja izvorišnog osnovnog udomljenika primi sadržaj pročitan iz izvorišnog elementa korisničkog sučelja, raspoređivač događaja unutar složenog udomljenika započinje izvođenje drugog dijela događaja za uspostavu toka podataka. S popisa događaja dohvaća se pomoćni događaj za upisivanje sadržaja u odredišni element korisničkog sučelja, parsira se te se zajedno sa sadržajem pročitanim iz izvorišnog elementa prosljeđuje u odredišni osnovni udomljenik.

Upisivanje sadržaja u odredišni element grafičkog korisničkog sučelja ostvareno je funkcijom *Interpreter.executePaste*. Ulazni parametar funkcije je pokazivač na odredišni element grafičkog korisničkog sučelja udomljenika i sadržaj koji je potrebno upisati u taj element. Slično kao i kod funkcije *Interpreter.executeCopy*, izvođenje funkcije *Interpreter.executePaste* započinje ispitivanjem vrste odredišnog elementa. Nakon utvrđivanja vrste elementa, koristi se odgovarajuće programsko sučelje za pristup sadržaju elementa i postavljanje sadržaja na vrijednost pročitanu iz izvorišnog elementa.

Na slici 10.14 posebno su istaknuti dijelovi funkcija *Interpreter.executeCopy* i *Interpreter.executePaste* koji se izvode tijekom prijenosa podataka između raznovrsnih elemenata korisničkog sučelja s podudarnim oblikom sadržaja. Primjerom je prikazan prijenos naziva stavke odabrane u padajućem izborniku izvorišnog udomljenika u polje za unos teksta odredišnog udomljenika. Istaknuti dijelovi programa obavljaju tok podataka između dvaju udomljenika prema scenariju prikazanom na slici 10.13.

11 Upravljanje tijekom izvođenja programa

Radni tijek nad skupom osnovnih udomljenika koji se sastoji od događaja za upravljanje izvođenjem udomljenika i događaja za uspostavu toka podataka, a pokreće se putem grafičkog korisničkog sučelja složenog udomljenika, određen je redoslijedom izvođenja događaja. Uspostavom vremenskih odnosa nad skupom događaja za usložnjavanje udomljenika, potrošač određuje tijek izvođenja primjenskog programa. U ovom poglavlju opisani su postupci za definiranje vremenskih odnosa koji određuju redoslijed izvođenja događaja za usložnjavanje udomljenika.

Upravljanje tijekom izvođenja programa u okviru programske paradigme za usložnjavanje udomljenika postiže se prostornim razmještajem događaja unutar ćelija dvodimenzionalne tablične plohe. Za razliku od većine programskih jezika opće namjene u kojima postoji isključivo jedan smjer napredovanja vremena koji je usmjeren odozgo prema dolje, u programskom jeziku za usložnjavanje udomljenika vrijeme istovremeno napreduje odozgo prema dolje i slijeva na desno. Modeliranje vremenskih odnosa unutar dvodimenzionalne tablične plohe koristi se u svrhu jednostavnijeg i preglednijeg isticanja vremenski zavisnih i vremenski nezavisnih događaja. Dok je vremenski zavisne događaje potrebno izvoditi slijedno jedan iza drugog, vremenski nezavisne događaje moguće je izvoditi proizvoljnim redoslijedom, uključujući i istodobno.

Osnovna zamisao dvodimenzionalne tablične plohe je pružanje čovjeku prirodnog načina izražavanja relacije slijednosti i relacije vremenske neodređenosti, odnosno istodobnosti. S obzirom da je za većinu ljudi uobičajeni način prikaza istodobnih aktivnosti

njihov usporedni prikaz u prostoru, modeliranje vremenskih odnosa primjenom dvodimenzionalne tablične plohe slično je postupku skiciranja uz pomoć papira i olovke. Prikaz vremenskih odnosa primjenom dvodimenzionalnog prostora s dvosmjernim napredovanjem vremena odabrano je na osnovi *teorije jezične predodređenosti* kojom se opisuje utjecaj govornog jezika na doživljaj vremena. Istraživanje utjecaja govornog jezika na doživljaj vremensko-prostornih odnosa opisano je u poglavlju 3.

U poglavlju 11.1 opisana su vremenska svojstva udomljenika koja proizlaze iz raspodijeljenosti pozadinskih računalnih procesa čijim se izvođenjem upravlja putem grafičkog korisničkog sučelja udomljenika. U poglavlju 11.2 opisana su svojstva vremenske relacije za oblikovanje radnog tijeka nad skupom udomljenika te je obrazložena potreba za mogućnošću modeliranja vremenske zavisnosti i vremenske neodređenosti tijekom oblikovanja potrošačkih programa paradigmom usložnjavanja udomljenika. U poglavlju 11.3 opisan je način upravljanja tijekom izvođenja programa primjenom prostornog razmještaja događaja unutar ćelija dvodimenzionalne tablične plohe. Poglavlje 11.4 donosi pregled programskog ostvarenja najčešćih oblikovnih obrazaca tijeka izvođenja programa u programskom jeziku za usložnjavanje udomljenika. U poglavlju 11.5 opisano je programsko upravljanje odzivom složenog udomljenika na radnje potrošača putem grafičkog korisničkog sučelja za pokretanje radnog tijeka oblikovanog nad skupom osnovnih udomljenika.

11.1 Vremenska svojstva skupa udomljenika

Svojstvo računalnih procesa raspodijeljenih u mreži Internet koji su putem grafičkog korisničkog sučelja udomljenika izloženi korisniku programske paradigme za usložnjavanje udomljenika jest njihova međusobna prostorna razdvojenost i vremenska nezavisnost koja im omogućuje istodobno postojanje i izvođenje. Vremenskom nezavisnošću tih računalnih procesa i mogućnošću istovremenog udomljavanja pripadajućih udomljenika unutar udomiteljske stranice potrošača, omogućeno je njihovo istodobno korištenje putem grafičkih korisničkih sučelja.

Tijekom oblikovanja radnog tijeka koji nastaje usložnjavanjem udomljenika, svaki od osnovnih udomljenika koristi se za obavljanje dijela cjelokupne logike primjenskog programa. Zakonitosti primjenskog programa ponekad dopuštaju istodobno izvođenje pojedinih dijelova programa, uz vremensko usklađivanje izvođenja u odgovarajućim točkama radnog tijeka. Iako svjestan vremenske nezavisnosti i mogućnosti istodobnog izvođenja pojedinih dijelova radnog tijeka, tehnika međudjelovanja putem ulazno-izlaznih naprava računala tijekom ručne uporabe udomljenika ograničava potrošača na slijedno izvođenje radnji putem grafičkog korisničkog sučelja. Međutim, oblikovanjem programskog

jezika za automatizirano upravljanje izvođenjem udomljenika, mogućnost modeliranja istodobnog izvođenja vremenski nezavisnih dijelova radnog tijeka postaje važno svojstvo programske paradigme. Svojstvo vremenske nezavisnosti i mogućnost istodobnog izvođenja pojedinih udomljenika razlog su za osmišljavanje prikladnog načina kojim potrošač tu nezavisnost i istodobnost može iskazati programskim zapisom.

Zbog mogućnosti istodobnog izvođenja pojedinih dijelova primjenskog programa, potrošač tijekom oblikovanja radnog tijeka ima određeni stupanj slobode u definiranju vremenskih odnosa nad skupom događaja za usložnjavanje udomljenika. Sloboda u definiranju vremenskih odnosa javlja se u slučajevima kada je radom skupa nezavisnih osnovnih udomljenika moguće upravljati na način da više različitih sljedova izvođenja programa daje jednakovrijedan krajnji rezultat. Programska paradigma za usložnjavanje udomljenika potrošaču omogućava modeliranje vremenske neodređenosti u programu za usložnjavanje udomljenika, uz uvođenje nužnog stupnja određenosti na mjestima gdje to zahtijevaju zakonitosti područja primjene za koje se gradi program.

11.2 Vremenska relacija nad skupom događaja za usložnjavanje udomljenika

Vremenski odnosi nad skupom događaja za usložnjavanje udomljenika uspostavljaju se definiranjem *relacije vremenskog slijeđenja* i *relacije vremenske nezavisnosti*. Relacije se definiraju nad uređenim parom događaja za usložnjavanje udomljenika. Za bilo koji uređeni par događaja vrijedi jedna od tih dviju relacija.

Relacija vremenskog slijeđenja

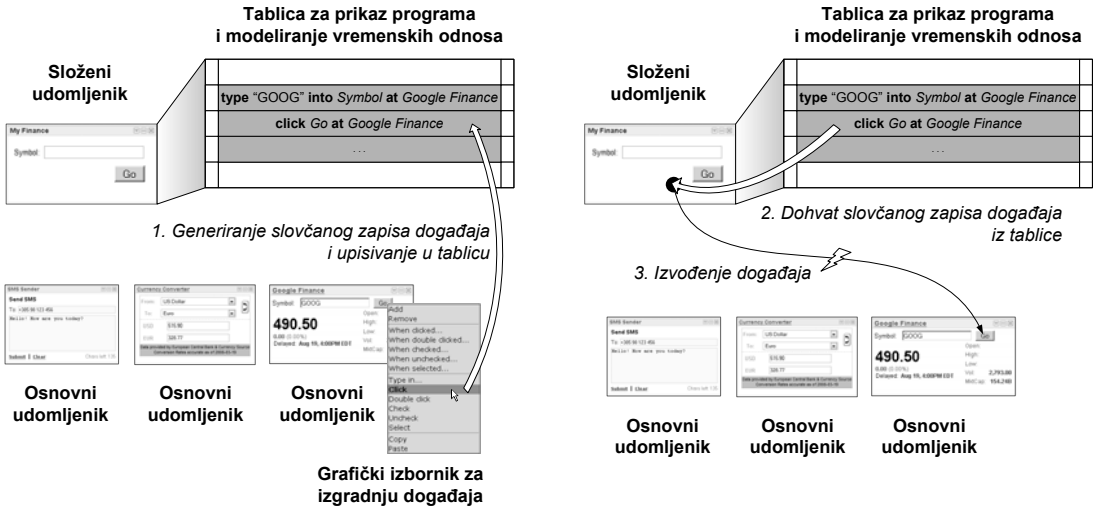
Ako su dva događaja za usložnjavanje udomljenika dovedena u relaciju vremenskog slijeđenja, onda izvođenje prvog događaja mora završiti prije nego što započne izvođenje drugog događaja. Relacija vremenskog slijeđenja ima svojstva antisimetričnosti i tranzitivnosti. Ako je relacijom vremenskog slijeđenja određeno da početak izvođenja događaja B slijedi nakon završetka izvođenja događaja A, onda istodobno vrijedi da završetak izvođenja događaja A prethodi početku izvođenja događaja B. Ako je relacijom vremenskog slijeđenja određeno da početak izvođenja događaja C slijedi nakon završetka izvođenja događaja B, a početak izvođenja događaja B slijedi nakon završetka izvođenja događaja A, onda istodobno vrijedi da početak izvođenja događaja C slijedi nakon izvođenja događaja A.

Relacija vremenske nezavisnosti

Ako dva događaja nisu dovedena u relaciju vremenskog slijeđenja, onda su ta dva događaja u relaciji vremenske nezavisnosti. U tom slučaju, promatrane događaje moguće je izvoditi proizvoljnim redoslijedom, uključujući i istodobno. Relacija vremenske nezavisnosti između dvaju događaja isključuje postojanje relacije vremenskog slijeđenja između tih dvaju događaja i obrnuto.

11.3 Dvodimenzionalna tablična ploha za modeliranje vremenskih odnosa

Modeliranje vremenskih odnosa među događajima programa za usložnjavanje udomljenika obavlja se u prostoru diskretne dvodimenzionalne plohe u obliku tablice [210]. Događaji za programsko upravljanje izvođenjem udomljenika i uspostavu toka podataka koji su opisani u poglavljima 9 i 10 razmještaju se po površini dvodimenzionalne tablične plohe. Cjelokupni prostor tablice podijeljen je na diskretne plošne elemente koji se nazivaju ćelijama. Položaj pojedinih događaja za usložnjavanje udomljenika određen je diskretnim koordinatama ćelije u kojoj je događaj zapisan. Vremenski odnosi između događaja određeni su njihovim relativnim prostornim položajem unutar ćelija tablice.



a) Izgradnja primjenskog programa b) Izvođenje primjenskog programa

Slika 11.1. Uloga tablice tijekom izgradnje i izvođenja primjenskog programa zasnovanog na usložnjavanju udomljenika

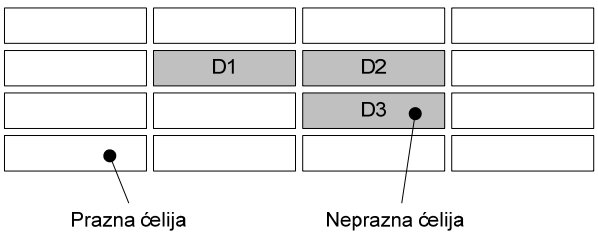
Slikom 11.1 prikazana je uloga tablice tijekom izgradnje i izvođenja primjenskog programa zasnovanog na usložnjavanju udomljenika. Tablična ploha za modeliranje vremenskih odnosa dio je podatkovne strukture složenog udomljenika. Primjenom grafičkog

izbornika za izgradnju događaja za usložnjavanje udomljenika, potrošač obilazi skup udomljenika i definira skup događaja kojima se programski oponašaju radnje nad elementima grafičkog korisničkog sučelja, kao što je opisano u poglavljima 9 i 10. Događaji izgrađeni primjenom grafičkog izbornika zapisuju se unutar ćelija tablice (1). Događaji se zapisuju u obliku slijednog uzorka izvođenja programa onim redoslijedom kojim ih potrošač definira primjenom grafičkog izbornika. Nakon što je završio s izgradnjom događaja, potrošač promjenom prostornog razmještaja događaja unutar ćelija tablice utječe na njihove međusobne vremenske odnose, odnosno slijedni uzorak izvođenja događaja pretvara u uzorak koji odgovara zahtjevima primjenskog programa. Tijekom izvođenja primjenskog programa, složeni udomljenik čita sadržaj tablice (2) i izvodi skup zapisanih događaja u skladu s definiranim vremenskim odnosima (3).

11.3.1 Vremenski uređaj nad ćelijama tablice

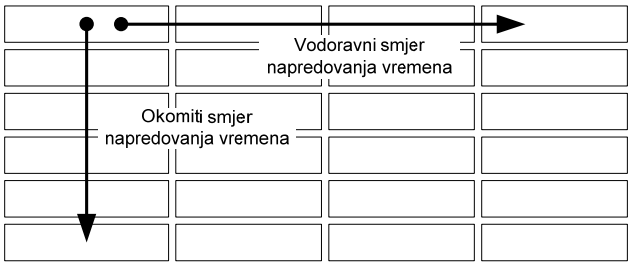
Tablica je naziv za radnu površinu dvodimenzionalne tablične plohe unutar koje se zapisuju događaji programa za usložnjavanje udomljenika. Tablična ploha koristi se za prikaz događaja izgrađenih primjenom grafičkog izbornika, kao što je opisano u poglavljima 9 i 10, te za definiranje vremenskih odnosa nad skupom izgrađenih događaja. Cjelokupni program izgrađen od strane potrošača primjenom grafičkog izbornika smješten je unutar tablice. S obzirom na tabličnu organizaciju programa i modeliranje vremenskih odnosa unutar tablične plohe, postupak programiranja složenog udomljenika naziva se tabličnim programiranjem (engl. *tabular programming*, *table-based programming*) [107].

Osnovna prostorna jedinica tablice naziva se ćelijom. Ćelija tablice služi za zapis događaja za usložnjavanje udomljenika koji su izgrađeni primjenom grafičkog izbornika. Pojedini događaj za usložnjavanje udomljenika zapisan je u točno jednoj ćeliji tablice. Nadalje, unutar jedne ćelije tablice moguće je zapisati najviše jedan događaj za usložnjavanje udomljenika.



Slika 11.2. Dvodimenzionalna tablica s praznim i nepraznim ćelijama

Za bilo koju od ćelija tablice vrijedi da se nalazi u jednom od dva moguća stanja. Ako ćelija ne sadrži zapis događaja za usložnjavanje udomljenika, ćelija je prazna. Ako ćelija sadrži zapis događaja za usložnjavanje udomljenika, ćelija je neprazna. Neposredno prije izgradnje složenog udomljenika, sve ćelije tablice su prazne. Upisivanjem zapisa događaja za usložnjavanje udomljenika u zadanu ćeliju, promatrana ćelija postaje neprazna. Slikom 11.2 prikazana je tablica dvodimenzionalne tablične plohe s istaknutim pojmovima prazne i neprazne ćelije. Prikazana tablica sadrži tri neprazne ćelije u kojima su zapisani događaji *D1*, *D2* i *D3*, dok su ostale ćelije tablice prazne.



Slika 11.3. Tablica za modeliranje vremenskih odnosa s dvosmjernim napredovanjem vremena

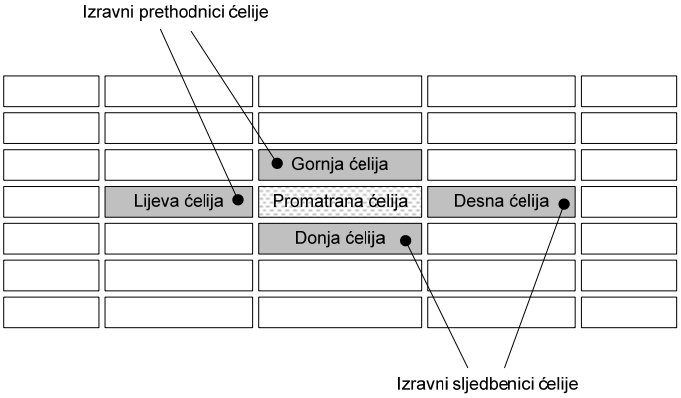
Vremenski uređaj nad ćelijama tablice definiran je nad nepraznim ćelijama tablice, dok se prazne ćelije koriste za razdvajanje vremenski nezavisnih dijelova programa. Unutar tablice vrijeme istovremeno i nezavisno napreduje u dva smjera: slijeva na desno i odozgo prema dolje, kao što je prikazano slikom 11.3. To znači da izvođenje događaja koji je zapisan u ćeliji koja se nalazi ispod ili desno od promatrane ćelije može započeti tek nakon što završi izvođenje događaja zapisanog u promatranoj ćeliji. S druge strane, izvođenje događaja koji je zapisan u ćeliji koja se nalazi iznad ili lijevo od promatrane ćelije mora završiti prije nego što započne izvođenje događaja koji je zapisan u promatranoj ćeliji.

Praznom ćelijom prekida se tijek vremena i omogućuje modeliranje vremenski nezavisnih tijekova izvođenja programa. Unutar nakupina nepraznih ćelija koje su međusobno razdvojene praznim ćelijama vrijeme napreduje samostalno i nezavisno od vremena u ostalim ćelijama tablice. Događaji koji su zapisani u ćelijama koje pripadaju nakupinama nepraznih ćelija koje su međusobno razdvojene praznim ćelijama smatraju se vremenski nezavisnim događajima.

Okolina ćelije

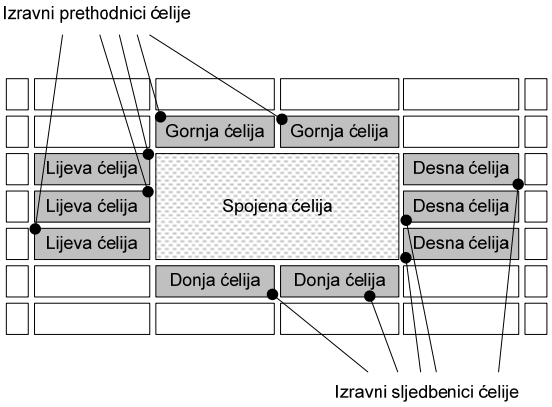
S obzirom na definirani vremenski uređaj nad ćelijama tablice, definira se skup ćelija koje je moguće koristiti za uspostavu izravnog vremenskog odnosa s promatranom ćelijom.

Svaka ćelija tablice omeđena je skupom ostalih ćelija. Skup ćelija koje su u izravnom vremenskom odnosu s promatranom ćelijom naziva se okolinom ćelije.



Slika 11.4. Okolina ćelije

Slikom 11.4 prikazana je tablica s istaknutim skupom ćelija koji čini okolinu promatrane ćelije. Okolinu ćelije čine ćelije koje se nalaze neposredno slijeva, zdesna, iznad i ispod promatrane ćelije. Ćelije koje se nalaze neposredno lijevo i iznad promatrane ćelije nazivaju se izravnim prethodnicima ćelije. Ćelije koje se nalaze neposredno desno i ispod promatrane ćelije nazivaju se izravnim sljedbenicima ćelije. Zapisivanjem događaja u ćeliju koja pripada okolini promatrane ćelije, događaj se dovodi u izravnu vremensku vezu s događajem zapisanim u promatranoj ćeliji.



Slika 11.5. Okolina spojene ćelije

Izvođenje događaja zapisanih u svim nepraznim ćelijama koje su izravni prethodnici promatrane ćelije nužan je i dovoljan uvjet za izvođenje događaja zapisanog u promatranoj ćeliji. Izvođenje događaja zapisanog u promatranoj ćeliji nužan je, ali ne i dovoljan uvjet za izvođenje događaja koji je zapisan u bilo kojoj od ćelija koja je izravni sljedbenik promatrane ćelije. Izvođenje događaja zapisanog u promatranoj ćeliji nije dovoljan uvjet za

izvođenje događaja zapisanog u ćeliji koja je izravni sljedbenik promatrane ćelije jer svaki od izravnih sljedbenika, osim promatrane ćelije, može imati i drugog izravnog prethodnika.

Osim pravilne tablične strukture, programskom paradigmom za usložnjavanje udomljenika dopušta se uporaba spojenih ćelija. Spojena ćelija sastoji se od skupa osnovnih ćelija tablice koje su spojene u zajedničku ćeliju. Spajanje ćelija dopušteno je uzduž redaka i stupaca tablice, uz zadržavanje četverokutnog oblika spojene ćelije. Primjena spojene ćelije omogućava više od dva izravna prethodnika, odnosno više od dva izravna sljedbenika promatrane ćelije. Slikom 11.5 prikazana je primjena spojene ćelije za modeliranje vremenskih odnosa u programu za upravljanje radom udomljenika.

Lokalnost i diskretnost napredovanja vremena

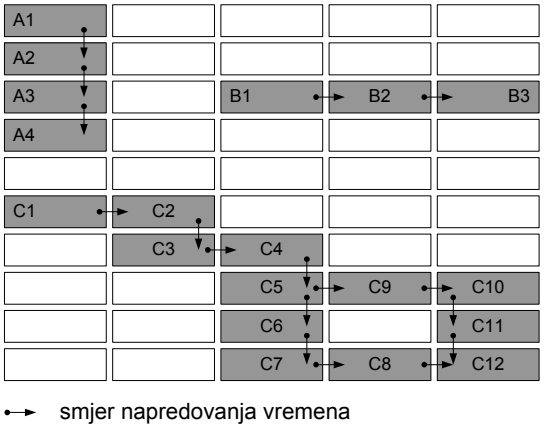
Vremenski uređaj nad ćelijama tablice s dvosmjernim napredovanjem vremena omogućuje jednostavan, pregledan i prirodan način izražavanja vremenski zavisnih i vremenski nezavisnih događaja za usložnjavanje udomljenika. Modeliranje vremenskih odnosa unutar ćelija tablice zasnovano je na svojstvima *lokalnosti* i *diskretnosti napredovanja vremena*.

Lokalnost napredovanja vremena

Svojstvom lokalnosti napredovanja vremena omogućuje se izražavanje neodređenosti za vrijeme izvođenja događaja za usložnjavanje udomljenika. Napredovanje vremena unutar tablice definirana je unutar nakupine nepraznih ćelija. Unutar pojedine nakupine nepraznih ćelija vrijeme napreduje samostalno i nezavisno od vremena u svim ostalim nakupinama nepraznih ćelija. Odvajanjem nakupina nepraznih ćelija skupom praznih ćelija prekida se napredovanje vremena iz jedne u drugu nakupinu. Vremenski odnosi između događaja zapisanih u različitim nakupinama nepraznih ćelija nisu definirani i predstavljaju element za modeliranje neodređenosti u primjenskom programu za usložnjavanje udomljenika.

Primjena svojstva lokalnosti napredovanja vremena tijekom izgradnje programa za usložnjavanje udomljenika prikazana je slikom 11.6. Na slici su prikazane tri nakupine nepraznih ćelija. Unutar nakupine nepraznih ćelija koja sadrži događaje A1 do A4, vrijeme napreduje odozgo prema dolje. Unutar te nakupine, prvi se izvodi događaj A1, a posljednji događaj A4. Unutar nakupine nepraznih ćelija koja sadrži događaje B1 do B3, vrijeme napreduje slijeva na desno. Unutar te nakupine, prvi se izvodi događaj B1, a posljednji događaj B3. Napredovanje vremena unutar nakupine nepraznih ćelija koja sadrži događaje C1 do C12 određeno je istovremenim postojanjem višestrukih vremenskih tijekova.

Izvođenje tog dijela programa započinje događajima C1 do C5. Nakon izvođenja događaja C5 dolazi do razdvajanja vremenskog tijeka na dva smjera, od kojih jedan napreduje prema dolje prema ćeliji s događajem C6, a drugi u desno prema ćeliji s događajem C9. Dva vremenska toka napreduju samostalno i nezavisno i ponovno se spajaju u zajednički vremenski tijek u ćeliji s događajem C12. Događaj C12 ne može biti izveden prije nego što su izvedeni događaji C8 i C11.



Slika 11.6. Lokalnost napredovanja vremena unutar ćelija tablice

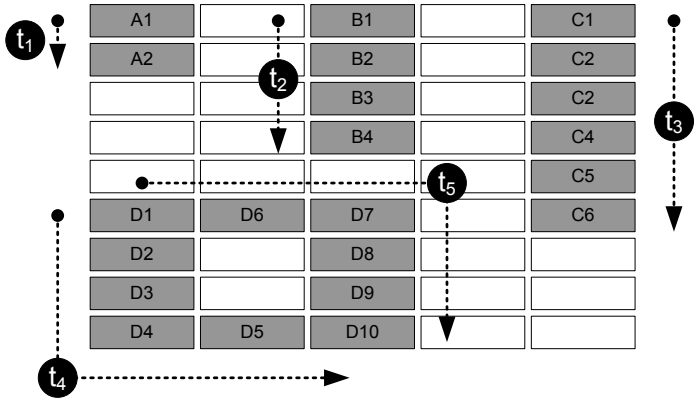
Izvođenje programa za usložnjavanje udomljenika započinje događajem zapisanim u ćeliji tablice koja nema niti jednog prethodnika. Ako u tablici postoji više nepraznih ćelija koje nemaju niti jednog prethodnika, izvođenje programa može započeti bilo kojim od događaja zapisanih u tim ćelijama. Sve ćelije koje u promatranom trenutku vremena imaju ispunjene uvjete za izvođenje, moguće je izvesti istodobno. U primjeru prikazanom na slici 11.6, izvođenje primjenskog programa moguće je započeti bilo kojim od događaja A1, B1 i C1. Nadalje, izvođenje programa moguće je započeti i istodobnim izvođenjem svih triju događaja.

Diskretnost napredovanja vremena

Napredovanje vremena unutar ćelija tablice zasniva se na diskretnim vremenskim trenucima. U svakom trenutku diskretnog vremena izvodi se jedan događaj unutar vremenskog slijeda. Napredovanje diskretnog vremena unutar vremenskog slijeda poticano je završecima izvođenja događaja zapisanih u ćelijama tablice, a ne napredovanjem apsolutnog vremena.

Program za usložnjavanje udomljenika prikazan slikom 11.7 sastoji se od četiri nakupine nepraznih ćelija. Nakupina koju čine ćelije s događajima A1 i A2 sastoji se od dva događaja, gdje se prvi izvodi događaj A1, a posljednji događaj A2. Nakupina koju čine ćelije

s događajima B1 do B4 sastoji se od četiri događaja, gdje se prvi izvodi događaj B1, a posljednji događaj B4. Nakupina koju čine ćelije s događajima C1 do C6 sastoji se od šest događaja, gdje se prvi izvodi događaj C1, a posljednji događaj C6. Bez obzira na najveću duljinu vremenskog slijeda koji čine događaji C1 do C6, izvođenje slijeda t_3 ne mora nužno trajati dulje od sljedova t_1 i t_2 jer se napredovanje vremena u pojedinim nakupinama odvija u diskretnim vremenskim trenucima čije je napredovanje uzrokovano isključivo brzinom izvođenja događaja zapisanih u ćelijama unutar promatrane nakupine.



Slika 11.7. Diskretnost napredovanja vremena unutar ćelija tablice

Svojstvo diskretnosti napredovanja vremena čini ključnu razliku između vremenskog uređaja nad tablicom programske paradigme za usložnjavanje udomljenika i Ganttovog dijagrama [225]. U Ganttovom dijagramu duljina lanca povezanih aktivnosti ujedno određuje i trajanje pojedinih aktivnosti u lancu te lanca kao cjeline u apsolutnom vremenu, dok prostorne oznake početaka i završetaka aktivnosti određuju vremenske trenutke njihovih početaka i završetaka. U tablici dvodimenzionalne tablične plohe koja se koristi za zapis programa za usložnjavanje udomljenika, duljina lanca nepraznih ćelija određuje samo broj i redoslijed događaja koje je potrebno izvoditi slijedno. Duljina lanca nepraznih ćelija nije povezana s trajanjem izvođenja pojedinih događaja, kao ni trajanjem izvođenja lanca kao cjeline.

Svojstvo diskretnosti napredovanja vremena omogućava usklađivanje početaka i završetaka izvođenja pojedinih događaja bez poznavanja vremena potrebnog za izvođenje pojedinih događaja. Značaj svojstva diskretnosti napredovanja vremena tijekom oblikovanja programa za usložnjavanje udomljenika prikazan je nakupinom nepraznih ćelija D1 do D10 na slici 11.7. Tom nakupinom prikazan je slučaj u kojem se prvi izvodi događaj D1, a posljednji događaj D10. Nakon izvođenja događaja D1, izvođenje programa se razdvaja u dva istodobna i nezavisna vremenska tijeka, od kojih jedan napreduje prema ćeliji s

događajem D2, a drugi prema ćeliji s događajem D6. Dva nezavisna vremenska tijeka samostalno napreduju u diskretnom vremenu i ponovno se susreću na mjestu ćelije s događajem D10, gdje se spajaju u jedan vremenski tijek. Događaj D10, stoga, ne smije biti izveden prije nego što su izvedeni događaji D5 i D9. Spajanje vremenskih tijekova t_4 i t_5 u ćeliji D10, međutim, ne zahtijeva njihovo jednako trajanje u apsolutnom vremenu.

Diskretnošću napredovanja vremena, potrošaču je omogućeno oblikovanje vremenskih odnosa između događaja poznavajući isključivo vremenske zavisnosti početaka i završetaka izvođenja događaja. Trajanje izvođenja pojedinih događaja, koje je nepredvidivo tijekom izgradnje programa, nema utjecaja na oblikovanje i izgradnju programa.

11.4 Ostvarenje uzoraka tijeka izvođenja programa primjenom tabličnog programiranja

Vremenski uređaj nad ćelijama tablice omogućava pojednostavljeno i pregledno modeliranje vremenske slijednosti i vremenske nezavisnosti u programu za usložnjavanje udomljenika. U ovom poglavlju pokazano je kako je različitim kombinacijama prostornog razmještaja događaja po ćelijama tablice moguće ostvariti različite uzorke tijeka izvođenja programa za usložnjavanje udomljenika.

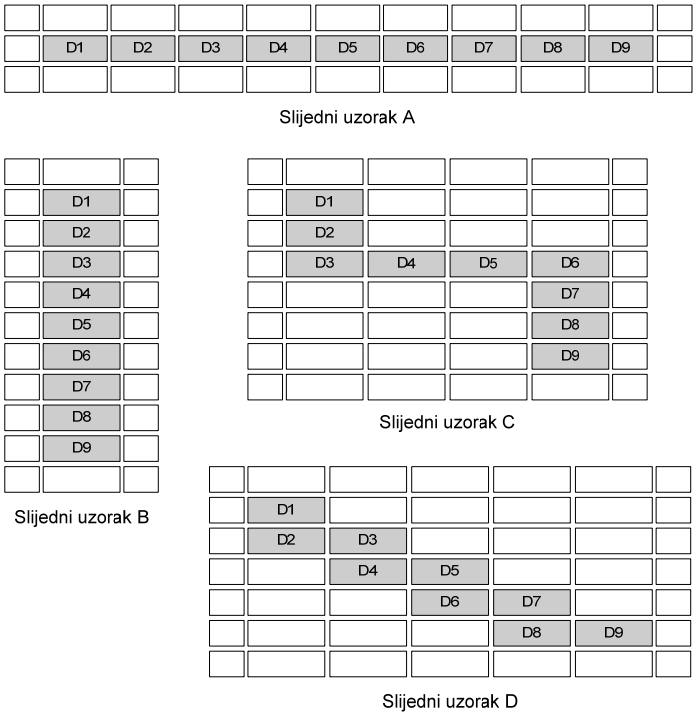
11.4.1 Slijedni uzorak izvođenja programa

Neka se radni tijek za usložnjavanje udomljenika ostvaruje skupom događaja koje je potrebno obaviti slijedno jedan iza drugog. Program za usložnjavanje udomljenika kojim se modelira opisani radni tijek postiže se slijednim uzorkom izvođenja programa. Slikom 11.8 prikazano je nekoliko različitih načina programskog ostvarenja slijednog uzorka izvođenja programa u programskom jeziku za usložnjavanje udomljenika.

Slijedni uzorak izvođenja programa postiže se prostornim razmještajem skupa događaja u susjedne ćelije tablice. Razmještanje događaja obavlja se na način da svaka neprazna ćelija tablice, osim ćelije koja sadrži događaj kojim započinje izvođenje programa, ima isključivo jednog izravnog prethodnika. U odnosu na svog izravnog prethodnika, promatrana naredba upisuje se u donju ili desnu ćeliju tablice.

S obzirom na dvosmjerno napredovanje vremena unutar nepraznih ćelija tablice, postoji više različitih mogućnosti zapisa slijednog uzorka izvođenja programa. Slikom 11.8 prikazane su četiri različite, ali jednakovrijedne mogućnosti zapisa slijednog uzorka izvođenja programa. Za zapis slijednog uzorka *A*, koristi se isključivo vodoravno napredovanje vremena slijeva na desno. Za zapis slijednog uzorka *B*, koristi se isključivo

okomito napredovanje vremena odozgo prema dolje. Slijednim uzorcima *C* i *D* prikazana su dva načina zapisa slijednog uzorka uz istodobno korištenje vodoravnog i okomitog smjera napredovanja vremena.

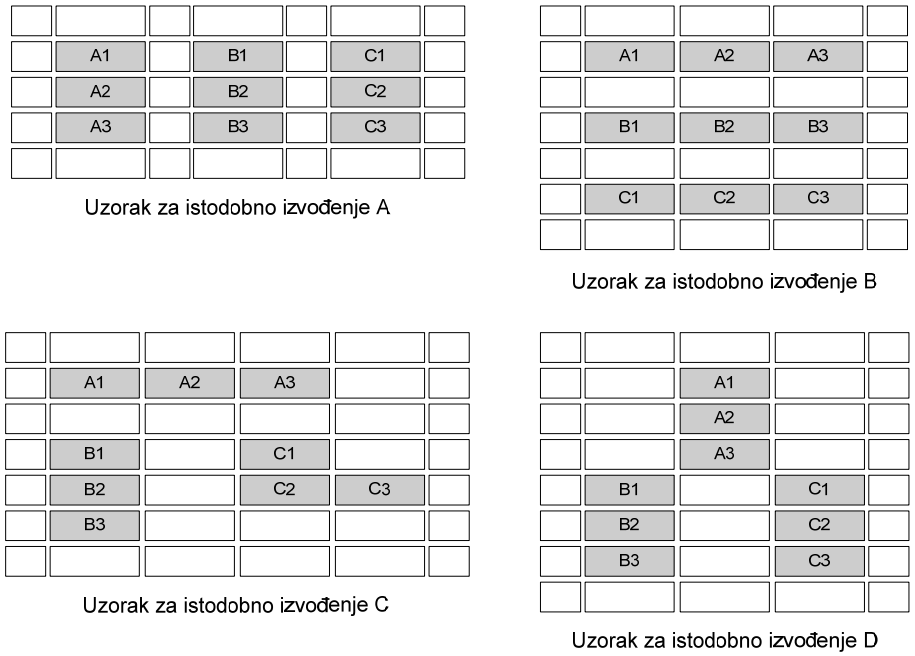


Slika 11.8. Programsko ostvarenje slijednog uzorka izvođenja programa u programskom jeziku za usložnjavanje udomljenika

11.4.2 Uzorak za istodobno izvođenje programa

Neka se radni tijek za usložnjavanje udomljenika ostvaruje skupom događaja od kojih je pojedine skupine događaja moguće obaviti proizvoljnim redoslijedom, uključujući i istodobno. Program za upravljanje radom udomljenika kojim se modelira opisani radni tijek postiže se uzorkom za istodobno izvođenje programa. Slikom 11.9 prikazano je nekoliko različitih načina programskog ostvarenja uzorka za istodobno izvođenje programa u programskom jeziku za usložnjavanje udomljenika.

Program za upravljanje radom udomljenika prikazan slikom 11.9 sastoji se od tri cjeline (A, B i C) koje je moguće izvoditi vremenski nezavisno. Pojedina cjelina sastoji se od tri slijedna događaja za usložnjavanje udomljenika. Događaji koji pripadaju istoj cjelini zapisuju se u susjedne ćelije tablice. Događaji koji pripadaju različitim cjelinama odvajaju se praznim ćelijama, čime postaju vremenski nezavisni od događaja iz drugih cjelina te ih je moguće izvoditi istodobno.



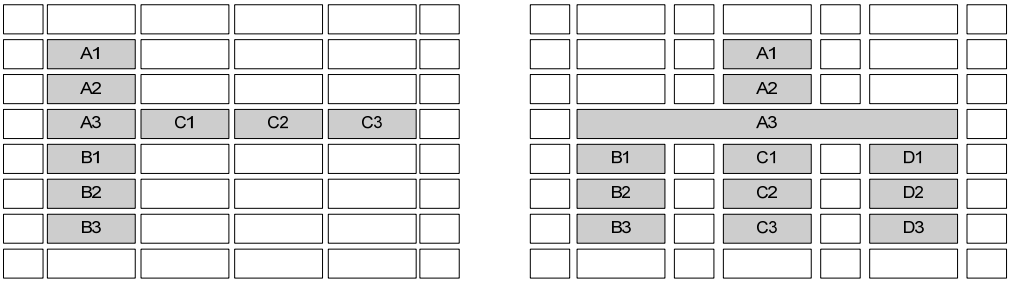
Slika 11.9. Programsko ostvarenje uzorka za istodobno izvođenje programa u programskom jeziku za usložnjavanje udomljenika

Slično ostvarenju slijednog uzorka, dvosmjerno napredovanje vremena unutar nepraznih ćelija tablice omogućava više različitih načina ostvarenja uzorka za istodobno izvođenje programa. Slikom 11.9 prikazane su četiri različite, ali jednakovrijedne mogućnosti ostvarenja uzorka za istodobno izvođenje programa. Za ostvarenje uzorka *A*, unutar svih međusobno nezavisnih programskih cjelina koristi se okomito napredovanje vremena odozgo prema dolje. Za ostvarenje uzorka *B*, unutar svih međusobno nezavisnih programskih cjelina koristi se vodoravno napredovanje vremena slijeva na desno. Za ostvarenje uzorka *C*, unutar cjeline A koristi se vodoravno napredovanje vremena, unutar cjeline B okomito napredovanje vremena, dok se unutar cjeline C koristi kombinacija okomitog i vodoravnog napredovanja vremena. Za ostvarenje uzorka *D*, unutar svih međusobno nezavisnih programskih cjelina koristi se okomito napredovanje vremena, slično uzorku *A*. Svojstvo uzorka *D* je razmještaj vremenski nezavisnih cjelina na način da se one dodiruju isključivo u jednoj točki koja se nalazi u kutu jedne od nepraznih ćelija. Budući da vremenskim uređajem nad nepraznim ćelijama tablice nije definirano dijagonalno napredovanje vremena, programske cjeline ostaju vremenski nezavisne.

11.4.3 Uzorak za razdvajanje vremenskih tijekova

Neka izvođenje radnog tijeka za usložnjavanje udomljenika započinje skupinom slijednih događaja, nakon čega se izvođenje grana na nekoliko slijednih skupina događaja

koje su sve međusobno vremenski nezavisne. Iako međusobno vremenski nezavisne, izvođenje tih skupina događaja ne smije započeti prije nego što završi izvođenje prve skupine događaja. Program za usložnjavanje udomljenika kojim se modelira opisani radni tijek postiže se uzorkom za razdvajanje vremenskih tijekova. Slikom 11.10 prikazan je primjer programskog ostvarenja uzorka za razdvajanje vremenskih tijekova u programskom jeziku za usložnjavanje udomljenika. Dvosmjerno napredovanje vremena unutar nepraznih ćelija tablice omogućava ostvarenje uzorka za razdvajanje vremenskih tijekova bez korištenja dodatnih mehanizama za usklađivanje tijeka izvođenja programa, kao što su semafori, redovi poruka ili posebne naredbe za pokretanje istodobnih računalnih procesa ili dretvi.

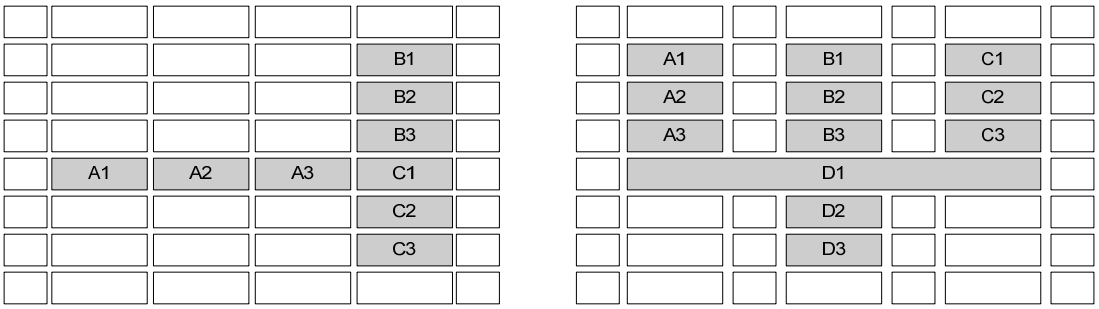


Slika 11.10. Programsko ostvarenje uzorka za razdvajanje vremenskih tijekova u programskom jeziku za usložnjavanje udomljenika

Primjenom pravilne tablične strukture omogućeno je razdvajanje početnog vremenskog tijeka na najviše dva nezavisna vremenska tijeka, kao što je prikazano na lijevoj strani slike 11.10. Primjenom spojenih ćelija, početni vremenski tijek moguće je razdvojiti na proizvoljni broj nezavisnih vremenskih tijekova, kao što je prikazano na desnoj strani slike 11.10. Pritom spojena ćelija sadrži završni događaj programske cjeline kojom započinje izvođenje radnog tijeka, odnosno događaj čije izvođenje završava neposredno prije razdvajanja vremenskih tijekova.

11.4.4 Uzorak za spajanje vremenskih tijekova

Neka izvođenja radnog tijeka za usložnjavanje udomljenika započinje s nekoliko skupina slijednih događaja koje su sve međusobno vremenski nezavisne. Nakon završetka izvođenja svih nezavisnih skupina događaja, izvođenje programa nastavlja se odgovarajućim brojem slijednih događaja. Program za usložnjavanje udomljenika kojim se modelira opisani radni tijek postiže se uzorkom za spajanje vremenskih tijekova. Slikom 11.11 prikazan je primjer programskog ostvarenja uzorka za spajanje vremenskih tijekova u programskom jeziku za usložnjavanje udomljenika.



Slika 11.11. Programsko ostvarenja uzorka za spajanje vremenskih tijekova u programskom jeziku za usložnjavanje udomljenika

Slično kao i kod programskog ostvarenja uzorka za razdvajanje vremenskih tijekova, dvosmjerno napredovanje vremena unutar nepraznih ćelija tablice omogućava ostvarenje uzorka za spajanje vremenskih tijekova bez korištenja dodatnih mehanizama za usklađivanje tijeka izvođenja programa, kao što su semafori ili redovi poruka. Primjenom pravilne tablične strukture omogućeno je spajanje najviše dvaju nezavisnih vremenskih tijekova u zajednički vremenski tijek, kao što je prikazano na lijevoj strani slike 11.11. Primjenom spojenih ćelija, proizvoljni broj nezavisnih vremenskih tijekova moguće je spojiti u zajednički vremenski tijek, kao što je prikazano na desnoj strani slike 11.11. Pritom spojena ćelija sadrži početni događaj programske cjeline kojom se nastavlja izvođenje programa nakon spajanja vremenskih tijekova.

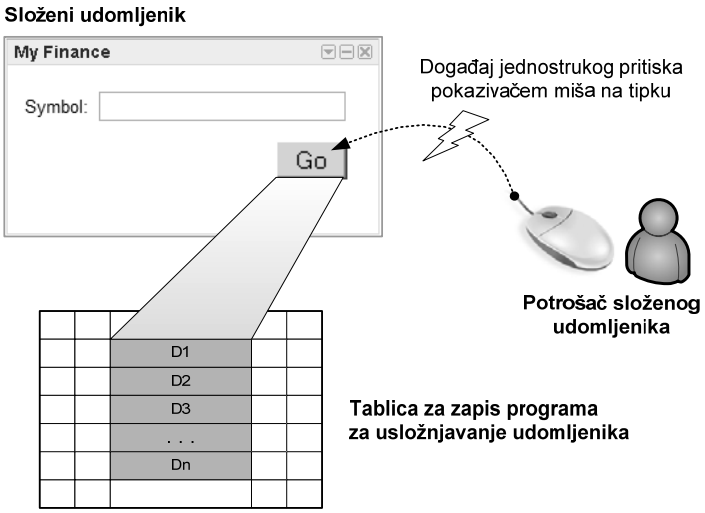
11.5 Odziv složenog udomljenika na radnje potrošača

Osim oblikovanja uzoraka tijeka izvođenja programa primjenom prostornog razmještaja događaja unutar ćelija tablice, upravljanje tijekom izvođenja programa podrazumijeva i upravljanje odzivom složenog udomljenika na radnje potrošača putem pripadajućeg grafičkog korisničkog sučelja. Obavljanjem radnji nad elementima grafičkog korisničkog sučelja složenog udomljenika, potrošač pokreće radni tijek definiran skupom događaja za usložnjavanje udomljenika koji su zapisani u ćelijama tablice.

Oblikovanje i izgradnja grafičkog korisničkog sučelja složenog udomljenika prikazani su u poglavlju 8. Oblikovanje i izgradnja programa kojim složeni udomljenik upravlja radom skupa osnovnih udomljenika primjenom događaja za upravljanje izvođenjem udomljenika i uspostavu toka podataka opisani su u poglavlju 9 i 10. Veza između korisničkog sučelja složenog udomljenika i programa za usložnjavanje udomljenika koji je zapisan u ćelijama tablice ostvaruje se *događajima za odziv složenog udomljenika na radnje potrošača*.

Prostor oblikovanja

Odziv složenog udomljenika na radnje potrošača u programskom jeziku za usložnjavanje udomljenika programski se ostvaruje skupom događaja koji se sastoji od *događaja za odziv složenog udomljenika na jednostruki pritisak aktivacijskog elementa*, *događaja za odziv složenog udomljenika na dvostruki pritisak aktivacijskog elementa*, *događaja za odziv složenog udomljenika na označavanje označnog polja*, *događaja za odziv složenog udomljenika na odznačavanje označnog polja*, *događaja za odziv složenog udomljenika na obilježavanje izbornog polja* i *događaja za odziv složenog udomljenika na izbor stavke iz padajućeg izbornika*. Prostor oblikovanja događaja za odziv složenog udomljenika na radnje potrošača prikazan je slikom 11.12 na primjeru događaja za odziv složenog udomljenika na jednostruki pritisak aktivacijskog elementa grafičkog korisničkog sučelja u obliku tipke.



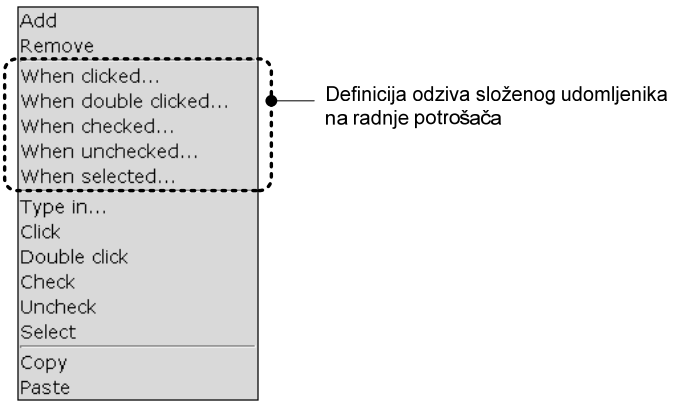
Slika 11.12. Oblikovanje događaja za odziv složenog udomljenika na radnje potrošača

U tablici za zapis programa za usložnjavanje udomljenika zapisan je program koji je potrošač samostalno izgradio primjenom događaja za upravljanje izvođenjem pojedinačnih udomljenika i događaja za uspostavu toka podataka između različitih udomljenika. Program je zapisan kao niz slijednih događaja označenih oznakama *D1* do *Dn*. Prema zamislima potrošača, prikazani slijed događaja potrebno je izvesti svaki put kada potrošač složenog udomljenika obavi radnju pritiska pokazivačem miša na izabranu tipku korisničkog sučelja. Ako tijekom izvođenja složenog udomljenika potrošač nijednom ne obavi radnju pritiska na zadanu tipku korisničkog sučelja, dio programa koji se odaziva na radnju pritiska na tu tipku neće se nijednom izvesti. Ako tijekom izvođenja složenog udomljenika potrošač obavi više

radnji pritiska na zadanu tipku korisničkog sučelja, dio programa koji se odaziva na radnju pritiska na tu tipku izvest će se onoliko puta koliko puta je pritisnuta tipka.

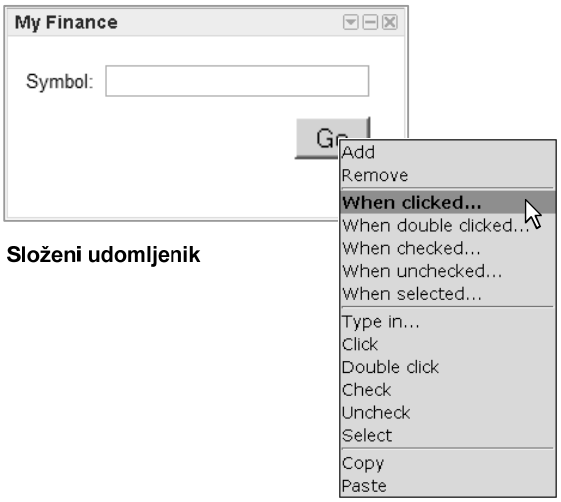
Prostor izgradnje

Grafički plivajući izbornik za izgradnju događaja za usložnjavanje udomljenika sadrži skup stavki za definiranje odziva složenog udomljenika na radnje potrošača. Stavke plivajućeg izbornika koje se koriste za izgradnju ove vrste događaja prikazane su slikom 11.13. Događaji za odziv složenog udomljenika na radnje potrošača izgrađuju se odabirom jedne od stavki plivajućeg izbornika koje započinju ključnom riječju *When*. Stavka izbornika pod nazivom *When clicked* koristi se za izgradnju događaja za odziv složenog udomljenika na jednostruki pritisak aktivacijskog elementa. Stavka izbornika pod nazivom *When double clicked* koristi se za izgradnju događaja za odziv složenog udomljenika na dvostruki pritisak aktivacijskog elementa. Stavka izbornika pod nazivom *When checked* koristi se za izgradnju događaja za odziv složenog udomljenika na radnju označavanja označnog polja. Stavka izbornika pod nazivom *When unchecked* koristi se za izgradnju događaja za odziv složenog udomljenika na radnju odznačavanja označnog polja. Stavka izbornika pod nazivom *When selected* koristi se za izgradnju događaja za odziv složenog udomljenika na radnju obilježavanja izbornog polja i radnju izbora stavke iz padajućeg izbornika.



Slika 11.13. Grafički izbornik za definiranje odziva složenog udomljenika na radnje potrošača

Primjena grafičkog izbornika tijekom izgradnje događaja za odziv složenog udomljenika na radnje potrošača prikazana je slikom 11.14. Prostor izgradnje događaja čini složeni udomljenik i plivajući izbornik. Pritiskom desne tipke miša na izabrani element korisničkog sučelja složenog udomljenika pojavljuje se plivajući izbornik za izgradnju programa za usložnjavanje udomljenika.



Slika 11.14. Izgradnja događaja za odziv udomljenika na radnje potrošača

Događaj za odziv složenog udomljenika na radnje potrošača izgrađuje se odabirom odgovarajuće stavke plivajućeg izbornika koja započinje ključnom riječju *When*. Primjerom na slici 11.14 prikazan je postupak izgradnje događaja za odziv složenog udomljenika na jednostruki pritisak tipke *Go* složenog udomljenika *My Finance*. Na sličan način moguće je izgraditi i događaje za odziv složenog udomljenika na ostale vrste radnji potrošača.

Prostor prikaza

Događaji za odziv složenog udomljenika na radnje potrošača izgrađeni primjenom grafičkog izbornika zapisuju se u tablicu unutar složenog udomljenika u obliku slovčanog zapisa. Primjer slovčanog zapisa događaja za odziv složenog udomljenika na radnje potrošača je

when clicked Go at My Finance

Izvođenjem zapisanog događaja, stvara se programska zamka (engl. *software trap*) u kojoj ostaje zaglavljen dio programa za usložnjavanje udomljenika koji se pokreće pritiskom potrošača na tipku *Go* složenog udomljenika *My Finance*. Oslobađanje zaglavljenog programa iz programske zamke postiže se mehanizmom programskog prekida (engl. *software interrupt*) kojeg uzrokuje potrošač pritiskom na tipku *Go* složenog udomljenika *My Finance*.

Opći oblik zapisa događaja za odziv složenog udomljenika na radnje potrošača prikazan je regularnim definicijama na slici 11.15. Regularnom definicijom *odzivNaRadnjePotrošača* opisano je pravilo za oblikovanje nizova znakova koji

predstavljaju valjano zapisane događaje za odziv složenog udomljenika na radnje potrošača. Regularna definicija *odzivNaRadnjePotrošača* definirana je ključnim riječima *when* i *at* te regularnim definicijama *radnja*, *element* i *udomljenik*.

odzivNaRadnjePotrošača = **when** *radnja element* **at** *udomljenik*
radnja = **clicked** | **doubleclicked** | **checked** | **unchecked** | **selected**
element = ([**A**..**Z**] + [**a**..**z**] + [**0**..**9**]) $^{+}$
udomljenik = ([**A**..**Z**] + [**a**..**z**] + [**0**..**9**]) $^{+}$

Slika 11.15. Opći oblik zapisa događaja za odziv složenog udomljenika na radnje potrošača

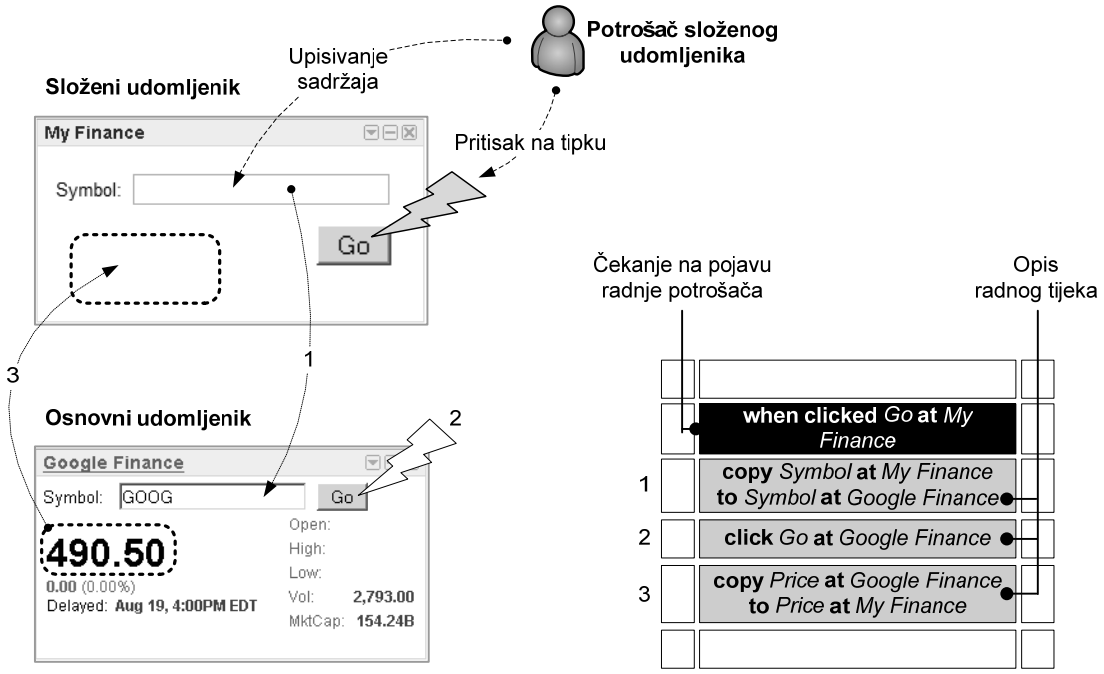
Ključna riječ *when* koristi se za jednoznačno prepoznavanje događaja za odziv složenog udomljenika na radnje potrošača. Regularnom definicijom *radnja* definirana je vrsta radnje kojom potrošač uzrokuje odziv složenog udomljenika, odnosno pokretanje izvođenja radnog tijeka. Vrsta radnje definirana je skupom ključnih riječi kojima se opisuju različite radnje za pokretanje radnog tijeka. Regularnom definicijom *element* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje elementa korisničkog sučelja složenog udomljenika koji je osjetljiv na pojavu radnje potrošača. Regularnom definicijom *udomljenik* opisano je pravilo za oblikovanje nizova znakova pomoću kojih se obavlja naslovljavanje složenog udomljenika putem čijeg se korisničkog sučelja pokreće radni tijek nad osnovnim udomljenicima. Naslovljavanje ciljnog elementa korisničkog sučelja i složenog udomljenika obavlja se nizovima znakova koji se sastoje od brojki te velikih i malih slova abecede. Ključna riječ *at* koristi se u svojstvu graničnika između naziva elementa korisničkog sučelja i naziva složenog udomljenika.

Sprega s događajima za upravljanje izvođenjem osnovnih udomljenika i događajima za uspostavu toka podataka

Događaji za odziv složenog udomljenika na radnje potrošača poprimaju puni smisao kada se koriste u sprezi s događajima za upravljanje izvođenjem osnovnih udomljenika i događajima za uspostavu toka podataka u skupu udomljenika. Događajima za upravljanje izvođenjem osnovnih udomljenika i događajima za uspostavu toka podataka u skupu udomljenika oblikuje se radni tijek za programsko upravljanje radom skupa udomljenika. Događajima za odziv složenog udomljenika na radnje potrošača, izvođenje tog radnog tijeka moguće je vezati uz pojavljivanje događaja uzrokovanih radnjama potrošača putem grafičkog korisničkog sučelja složenog udomljenika. Time je omogućeno pokretanje izvođenja radnog tijeka od strane potrošača putem grafičkog korisničkog sučelja složenog

udomljenika. Primjena događaja za odziv složenog udomljenika na radnje potrošača u sprezi s događajima za upravljanje izvođenjem osnovnih udomljenika i događajima za uspostavu toka podataka prikazana je slikom 11.16.

Na lijevoj strani slike 11.16 istaknute su radnje koje izvodi potrošač složenog udomljenika kako bi pokrenuo radni tijek te koraci koje poduzima složeni udomljenik kako bi izveo pokrenuti radni tijek. Putem korisničkog sučelja složenog udomljenika, potrošač u polje za unos teksta *Symbol* unosi početne vrijednosti za izvođenje radnog tijeka i pokreće njegovo izvođenje pritiskom na tipku *Go*. Nakon pokretanja radnog tijeka od strane potrošača, složeni udomljenik izvodi radni tijek koji se sastoji od triju koraka. U prvom koraku programski se obavlja prijenos sadržaja polja za unos teksta *Symbol* iz složenog udomljenika u istoimeno polje za unos teksta osnovnog udomljenika. U drugom koraku programski se obavlja radnja pritiska na tipku *Go* na osnovnom udomljeniku. U trećem koraku programski se obavlja prijenos sadržaja polja za prikaz teksta *Price* iz osnovnog udomljenika u istoimeni element složenog udomljenika. Tri koraka od kojih se sastoji radni tijek potrebno je obaviti svaki put kada potrošač obavi radnju pritiska na tipku *Go* složenog udomljenika.



Slika 11.16. Primjena događaja za odziv složenog udomljenika na radnje potrošača u programu za usložnjavanje udomljenika

Program za usložnjavanje udomljenika kojim se postiže opisana funkcionalnost prikazan je na desnoj strani slike. Program se sastoji od tri događaja kojima se opisuju tri

slijedna koraka za izvođenje radnog tijeka. Zbog slijednog izvođenja, ta tri događaja zapisana su u tri susjedne ćelije tablice, koristeći se okomitim smjerom napredovanja vremena. U ćeliji čije izvođenje prethodi izvođenju početnog događaja u radnom tijeku zapisan je događaj za odziv složenog udomljenika na radnju potrošača. Tim je događajem definirano da je izvođenje događaja koji čine radni tijek vezano uz pojavu radnje pritiska na tipku *Go* na korisničkom sučelju složenog udomljenika *My Finance*. S obzirom da nema definiranog prethodnika, uvjeti za izvođenje događaja za odziv složenog udomljenika na radnju potrošača moguće je izvesti svaki put kada potrošač obavi zadanu radnju. Izvođenjem tog događaja, stvaraju se uvjeti za izvođenje događaja koji su njegovi sljedbenici, a to su događaji koji čine radni tijek primjenskog programa.

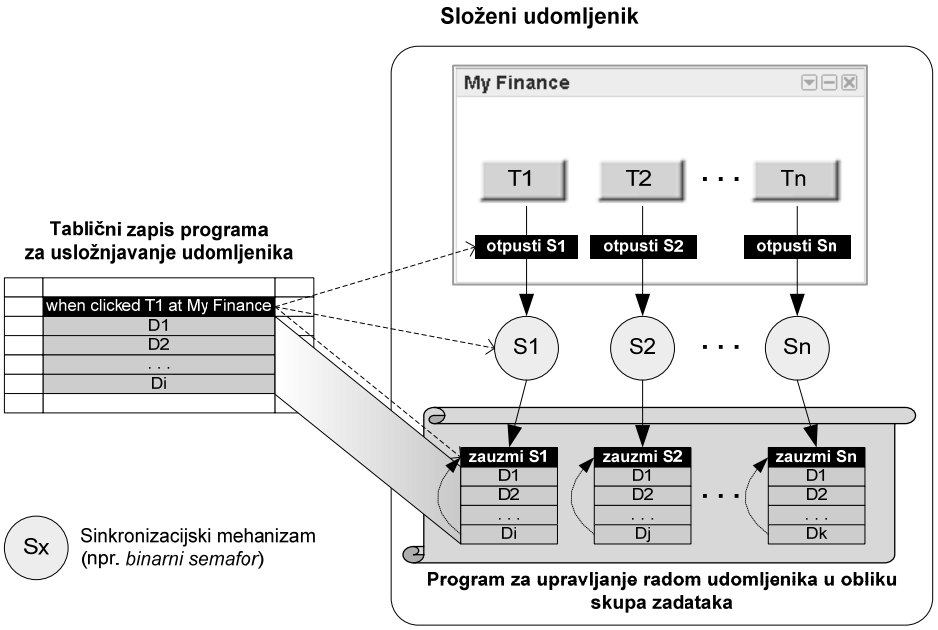
Dijelovi programa za usložnjavanje udomljenika koji se pokreću odzivom na radnje potrošača putem grafičkog korisničkog sučelja složenog udomljenika grade se u dva koraka. Najprije se primjenom grafičkog izbornika definira događaj za odziv složenog udomljenika na izabranu radnju potrošača. Nakon toga se primjenom grafičkog izbornika definira slijed događaja od kojih se sastoji radni tijek primjenskog programa, a izvode se kao odziv na prethodno definiranu radnju potrošača. Skup događaja kojima se složeni udomljenik odaziva na radnju potrošača sastoji se od događaja za upravljanje izvođenjem osnovnih udomljenika i uspostavu toka podataka.

Prostor izvođenja

Prostor izvođenja događaja za odziv složenog udomljenika na radnje potrošača prikazan je slikom 11.17. Složeni udomljenik sastoji se od programa za usložnjavanje udomljenika kojim je definiran radni tijek nad skupom udomljenika te odgovarajućeg broja elemenata korisničkog sučelja za pokretanje tog radnog tijeka. Pojedinim elementima korisničkog sučelja složenog udomljenika pokreću se različiti, vremenski nezavisni dijelovi radnog tijeka koji su zapisani u ćelijama tablice.

U prostoru izvođenja, dijelovi radnog tijeka čije je izvođenje vezano uz pojavu radnji potrošača preslikavaju se u zasebne zadatke. Počeci izvođenja pojedinih zadataka usklađuju se primjenom sinkronizacijskih mehanizama, primjerice binarnih semafora. Za svaki zadatak koji se odaziva na radnju potrošača stvara se po jedan sinkronizacijski mehanizam. Pojedini zadatak sastoji se od događaja za ostvarenje radnog tijeka, dok se na početak zadatka dodaje naredba za sinkronizaciju putem sinkronizacijskog mehanizma. Na početku rada, svi sinkronizacijski mehanizmi nalaze se u neprolaznom stanju te su svi zadaci koji se pokreću odzivom na radnje potrošača zaglavljeni u pripadajućim redovima čekanja. Pojava radnje

potrošača uzrokuje programski prekid, a rutina za obradu prekida oslobađa zaglavljeni zadatak iz reda čekanja odgovarajućeg sinkronizacijskog mehanizma. Nakon završetka izvođenja svih događaja iz tijela zadatka, zadatak se vraća u red čekanja sinkronizacijskog mehanizma kako bi bio spreman odazvati se na pojavu sljedeće radnje potrošača.



Slika 11.17. Prostor izvođenja događaja za odziv složenog udomljenika na radnje potrošača

11.5.1 Uzorak za deterministički odabir tijeka izvođenja

Programsko upravljanje odzivom udomljenika na radnje potrošača proširuje mogućnosti upravljanja tijekom izvođenja programa na dodatni skup uzoraka koje nije moguće postići isključivo prostornim razmještajem događaja unutar ćelija tablice za modeliranje vremenskih odnosa. Jedan od uzoraka izvođenja programa koji se ostvaruje primjenom događaja za odziv složenog udomljenika na radnje potrošača je uzorak za deterministički odabir tijeka izvođenja.

Neka se radni tijek za usložnjavanje udomljenika sastoji od skupa vremenski nezavisnih dijelova. Neka je izvođenje pojedinog dijela radnog tijeka moguće započeti samo ako su ispunjeni određeni uvjeti. U ovisnosti o stanju okoline primjenskog programa u promatranom trenutku izvođenja, dijelovi radnog tijeka dijele se na dvije skupine: one koji zadovoljavaju uvjete za izvođenje i one koji te uvjete ne zadovoljavaju. Izvođenjem jednog dijela radnog tijeka onemogućava se izvođenje svih ostalih dijelova. Program za usložnjavanje udomljenika kojim se modelira opisani radni tijek postiže se uzorkom za deterministički odabir tijeka izvođenja. Slikom 11.18 prikazan je primjer programskog

ostvarenja uzorka za deterministički odabir tijeka izvođenja u programskom jeziku za usložnjavanje udomljenika.

Slika 11.18. Programsko ostvarenje uzorka za deterministički odabir tijeka izvođenja u programskom jeziku za usložnjavanje udomljenika

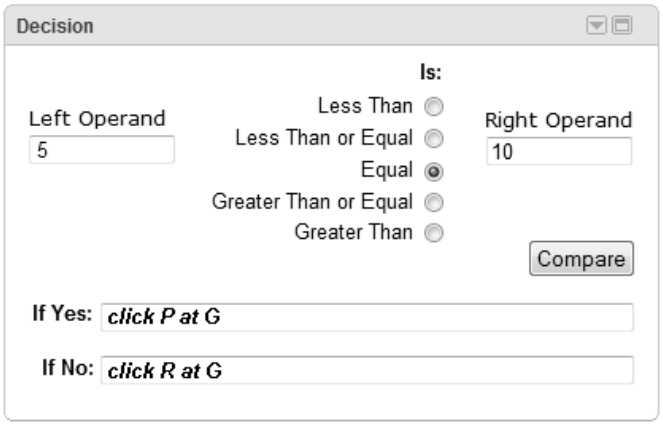
Uzorak za deterministički odabir tijeka izvođenja u programskom jeziku za usložnjavanje udomljenika postiže se spregom događaja za odziv složenog udomljenika na radnje potrošača, događaja za upravljanje izvođenjem osnovnih udomljenika i uspostavu toka podataka te tablice za modeliranje vremenskih odnosa. Slikom 11.18 prikazan je program za usložnjavanje udomljenika koji se sastoji od dviju nakupina nepraznih ćelija. Obje nakupine nepraznih ćelija ostvaruju slijedne uzorke izvođenja programa. Izvođenje nakupine nepraznih ćelija A1-A3 vezano je uz pojavu radnje pritiska na tipku *P* na složenom udomljeniku *G*. S druge strane, izvođenje nakupine nepraznih ćelija B1-B3 vezano je uz pojavu radnje pritiska na tipku *R* na složenom udomljeniku *G*. Ovisno o tome koju radnju obavi potrošač putem korisničkog sučelja složenog udomljenika *G*, tijekom izvođenja programa stvaraju se uvjeti za izvođenje jedne ili druge nakupine nepraznih ćelija. S obzirom na to da potrošač složenog udomljenika *G* ima izravni utjecaj na pokretanje izvođenja pojedinih dijelova radnog tijeka, uzorak pokazuje svojstva determinističkog odabira.

11.5.2 Uzorak za grananje tijeka izvođenja programa

Uzorak za grananje tijeka izvođenja koristi se kada je tijekom izvođenja programa potrebno donijeti odluku o izboru jednog od dva ili više mogućih smjerova napredovanja programa. Programski jezik za usložnjavanje udomljenika ne sadrži posebne događaje za upravljanje grananjem tijeka izvođenja programa, već se za tu svrhu koristi udomljenik posebne namjene.

Programsko ostvarenje uzorka za grananje tijeka izvođenja programa u programskom jeziku za usložnjavanje udomljenika slično je ostvarenju uzorka za deterministički odabir tijeka izvođenja. Kod uzorka za deterministički odabir tijeka izvođenja, odabir programske cjeline kojom se nastavlja izvođenje programa određeno je djelovanjem potrošača putem

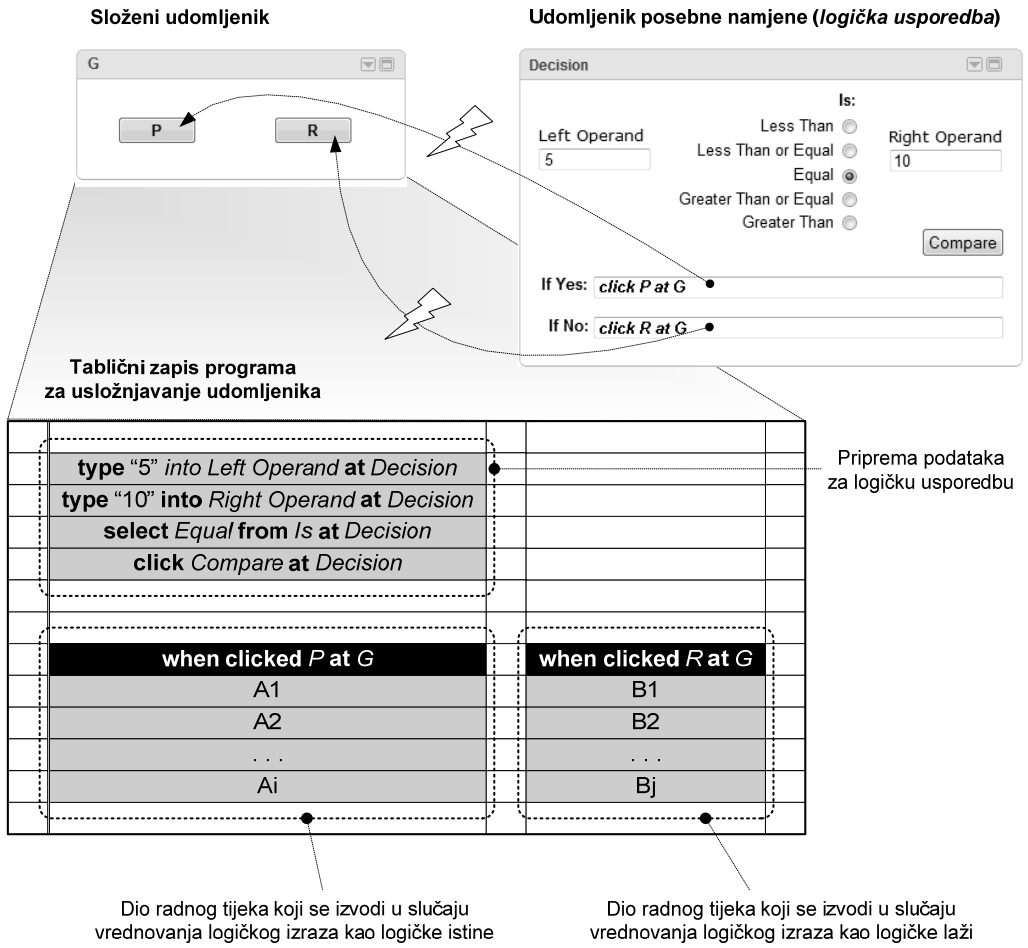
grafičkog korisničkog sučelja složenog udomljenika. Kod uzorka za grananje tijeka izvođenja, odabir programske cjeline kojom se nastavlja izvođenje programa također je određeno pojavom radnje nad elementom grafičkog korisničkog sučelja složenog udomljenika, ali izvor radnje nije djelovanje korisnika, nego udomljenik posebne namjene.



Slika 11.19. Primjer udomljenika posebne namjene za programsko upravljanje grananjem tijeka izvođenja programa za usložnjavanje udomljenika

Slikom 11.19 prikazan je primjer udomljenika posebne namjene za upravljanje grananjem tijeka izvođenja programa za usložnjavanje udomljenika. Udomljenik omogućava unošenje vrijednosti i definiranje uvjeta za vrednovanje logičkih izraza te pokretanje postupka vrednovanja. Vrednovanjem unesenih vrijednosti s obzirom na postavljene logičke uvjete, ispunjena je jedna od dviju vrijednosti logičke istinitosti. Za pojedinu vrijednost logičke istinitosti, unutar udomljenika posebne namjene definira se događaj nad korisničkim sučeljem složenog udomljenika na koji se odaziva dio radnog tijeka zapisanog u ćelijama tablice.

Primjerom na slici 11.20 prikazana je primjena udomljenika posebne namjene za programsko upravljanje grananjem tijeka izvođenja programa za usložnjavanje udomljenika. U slučaju vrednovanja logičkog uvjeta kao logičke istine, udomljenik posebne namjene uzrokuje događaj pritiska na tipku *P* složenog udomljenika *G*. U slučaju vrednovanja logičkog uvjeta kao logičke laži, udomljenik posebne namjene uzrokuje događaj pritiska na tipku *R* složenog udomljenika *G*. Unutar složenog udomljenika *G* definiran je radni tijek koji se sastoji od tri cjeline. Prvom cjelinom obavlja se priprema podataka za logičku usporedbu, definiraju se uvjeti usporedbe te se pokreće usporedba pritiskom na tipku *Compare* udomljenika posebne namjene. Preostale dvije cjeline radnog tijeka ostvaruju uzorak za deterministički odabir tijeka izvođenja koji se odaziva na jedan od dva moguća događaja koji pristižu od udomljenika posebne namjene.



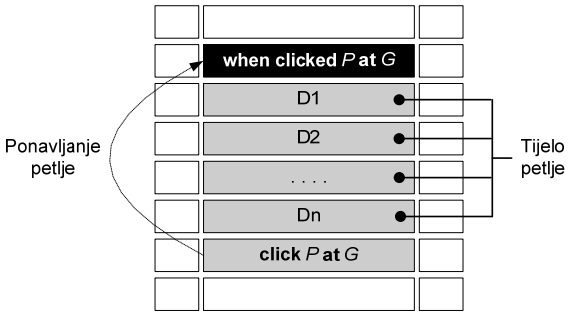
Slika 11.20. Programsko ostvarenje uzorka za grananje tijeka izvođenja programa u programskom jeziku za usložnjavanje udomljenika

11.5.3 Uzorak za ponavljanje tijeka izvođenja programa

Uzorak za ponavljanje tijeka izvođenja, odnosno rad programa u ponavljajućoj petlji koristi se kada je tijekom izvođenja programa iste dijelove programa potrebno izvesti u nekoliko uzastopnih koraka. Programski jezik za usložnjavanje udomljenika ne sadrži posebne događaje za upravljanje ponavljanjem tijeka izvođenja programa. Za tu svrhu koriste se događaji za upravljanje odzivom udomljenika na radnje potrošača u sprezi s udomljenikom posebne namjene za vrednovanje istinitosti logičkih izraza koji je opisan u poglavlju 11.5.2, a koristi se za ostvarenje uzorka za grananje tijeka izvođenja.

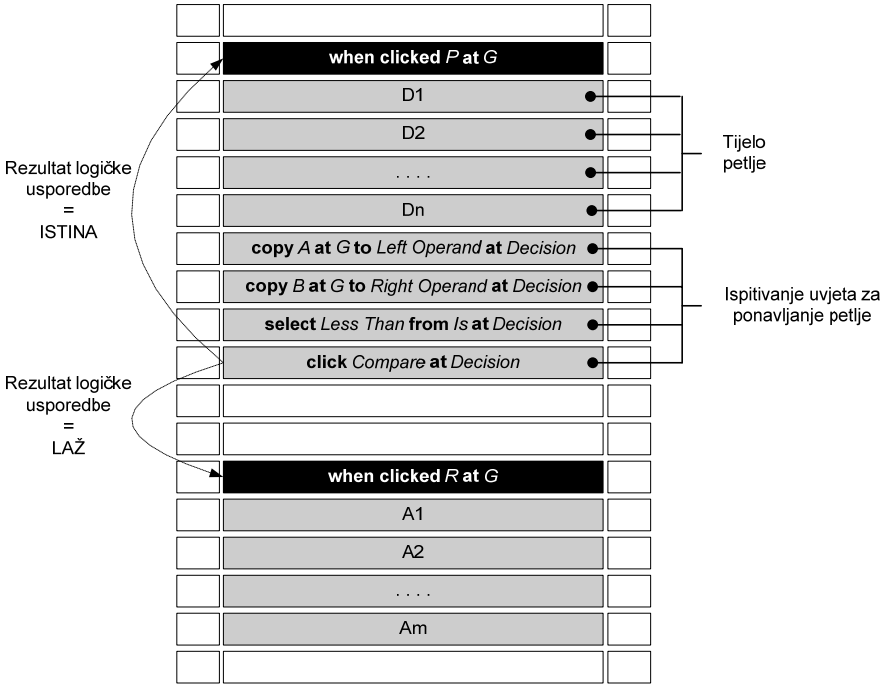
Slikom 11.21 prikazano je programsko ostvarenje uzorka za bezuvjetno ponavljanje tijeka izvođenja programa u programskom jeziku za usložnjavanje udomljenika. Dio radnog tijeka koji predstavlja cjelinu za ponavljanje, odaziva se na pojavu pritiska na element korisničkog sučelja *P* složenog udomljenika *G*. Tijelo petlje čini skup slijednih događaja D1

do Dn. Nakon izvođenja posljednjeg događaja iz tijela petlje, program uzrokuje događaj pritiska na element korisničkog sučelja *P* složenog udomljenika *G*, čime su stvoreni uvjeti za novi korak izvođenja događaja iz tijela petlje.



Slika 11.21. Programsko ostvarenje uzorka za bezuvjetno ponavljanje tijeka izvođenja programa u programskom jeziku za usložnjavanje udomljenika

Slikom 11.22 prikazano je programsko ostvarenje uzorka za uvjetno ponavljanje tijeka izvođenja programa u programskom jeziku za usložnjavanje udomljenika. Uzorak za uvjetno ponavljanje tijeka izvođenja programa koristi se udomljenikom posebne namjene za usporedbu logičkih izraza, slično kao što je prikazano kod programskog ostvarenja uzorka za grananje tijeka izvođenja programa. Udomljenik za usporedbu logičkih izraza koristi se za ispitivanje uvjeta za ponavljanje na kraju petlje.



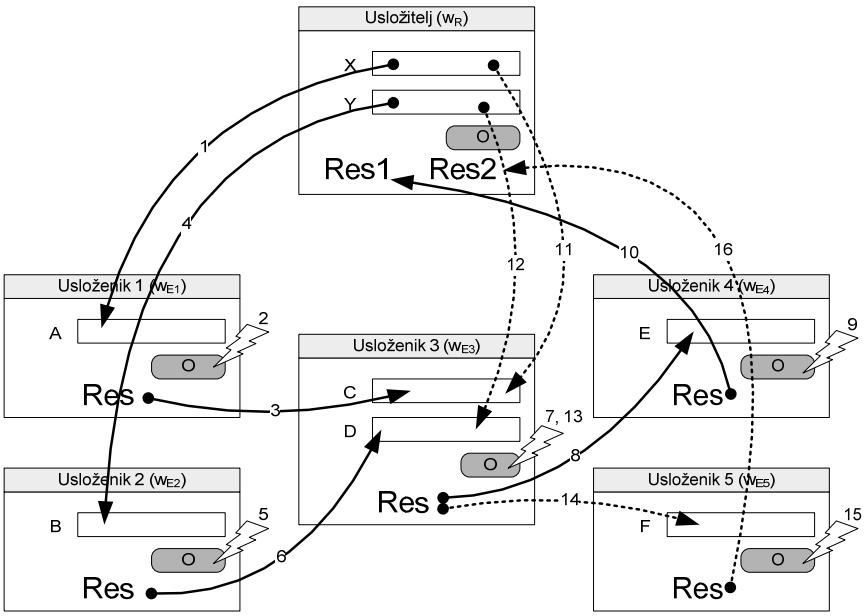
Slika 11.22. Programsko ostvarenje uzorka za uvjetno ponavljanje tijeka izvođenja programa u programskom jeziku za usložnjavanje udomljenika

Dio programa koji predstavlja cjelinu za ponavljanje odaziva se na pojavu pritiska na element korisničkog sučelja *P* složenog udomljenika *G*. Tijelo petlje čini skup slijednih događaja D1 do Dn. Nakon izvođenja posljednjeg događaja iz tijela petlje, primjenom udomljenika za usporedbu logičkih izraza pod nazivom *Decision* obavlja se ispitivanje uvjeta za ponavljanje petlje. Ako je uvjet za ponavljanje petlje ispunjen, udomljenik *Decision* uzrokuje događaj pritiska na element korisničkog sučelja *P* složenog udomljenika *G*, čime su stvoreni uvjeti za novi korak izvođenja događaja iz tijela petlje. Ako uvjet za ponavljanje petlje nije ispunjen, udomljenik *Decision* uzrokuje događaj pritiska na element korisničkog sučelja *R* složenog udomljenika *G*. U dijelu programa koji se odziva na pojavu tog događaja definira se ponašanje programa nakon izlaska iz tijela petlje. U primjeru na slici 11.22, nakon završetka izvođenja petlje izvodi se slijed događaja A1 do Am koji se odaziva na pojavu pritiska na element korisničkog sučelja *R* složenog udomljenika *G*.

12 Formalni opis primjenskog programa

Primjenski programi izgrađeni od strane potrošača paradigmom usložnjavanja udomljenika u ovom su poglavlju formalno opisani matematičkim modelima teorije grafova. Formalni opis primjenskih programa primjenom teorije grafova omogućava provedbu analize povezanosti udomljenika, analize podatkovnih zavisnosti među udomljenicima i analize tijeka upravljanja nad skupom udomljenika primjenom postojećih algoritama nad grafovima. Rezultate tih analiza moguće je iskoristiti za izradu statistike najkorisnijih udomljenika za izradu primjenskih programa, najčešće korištenih uzoraka usložnjavanja udomljenika, sličnih primjenskih programa koje su izgradili različiti potrošači i slično.

Primjenski programi opisani su grafovima u kojima čvorovi grafa predstavljaju udomljenike povezane u primjenski program, dok grane grafa predstavljaju strukturne, podatkovne i upravljačke veze među udomljenicima. Prikaz formalnog opisa primjenskih programa izgrađenih primjenom programske paradigme za usložnjavanje udomljenika pokazan je na primjeru primjenskog programa na slici 12.1. Primjenski program izložen u obliku složenog udomljenika pod nazivom *Usložitelj* izgrađen je povezivanjem pet osnovnih udomljenika čiji su nazivi redom *Usloženik 1*, *Usloženik 2*, *Usloženik 3*, *Usloženik 4* i *Usloženik 5*. Za potrebe daljnjeg opisa, složeni udomljenik naziva se *usložiteljem* (engl. *widgeter*), dok se osnovni udomljenici nazivaju *usloženicima* (engl. *widgetee*). Dvodimenzionalna tablica s prikazom programa prikazana je slikom 12.2. U tabličnom prikazu programa iskorištena je mogućnost istodobnog izvođenja pojedinih dijelova radnog tijeka kako bi se prikazali njegovi učinci na pravila za stvaranje formalnog opisa.



Slika 12.1. Primjenski program izgrađen od strane potrošača paradigmom usložnjavanja udomljenika

U poglavlju 12.1 formalno je opisana dvodimenzionalna tablica koja sadrži zapis primjenskog programa izgrađenog od strane potrošača. Na osnovi formalnog opisa tablice, prikazani su postupci izgradnje skupine grafova kojima se opisuju različite vrste relacija nad udomljenicima povezanim u radni tijek. U poglavlju 12.2 prikazan je *graf upravljačke uključenosti*. U poglavlju 12.3 prikazan je *graf podatkovne uključenosti*. *Graf podatkovne povezanosti* prikazan je u poglavlju 12.4, dok je *graf upravljačke povezanosti* prikazan u poglavlju 12.5.

wait for click O at w_R	4) copy Y at w_R to B at w_{E2}	5) click O at w_{E2}	
1) copy X at w_R to A at w_{E1}		6) copy Res at w_{E2} to D at w_{E3}	
2) click O at w_{E1}			
3) copy Res at w_{E1} to C at w_{E3}	7) click O at w_{E3}		
		8) copy Res at w_{E3} to E at w_{E4}	
		9) click O at w_{E4}	
		10) copy Res at w_{E4} to Res1 at w_R	12) copy Y at w_R to D at w_{E3}
		11) copy X at w_R to C at w_{E3}	13) click O at w_{E3}
			14) copy Res at w_{E3} to F at w_{E5}
			15) click O at w_{E5}
			16) copy Res at w_{E5} to Res2 at w_R

Slika 12.2. Tablični zapis primjenskog programa

12.1 Formalni opis tablice za prikaz programa

Dvodimenzionalna tablica koja sadrži zapis primjenskog programa izgrađenog od strane potrošača sadrži četiri skupine informacija. Prva informacija je cjelokupna tablica koja predstavlja složeni udomljenik izgrađen povezivanjem skupa osnovnih udomljenika. Druga informacija je skup osnovnih udomljenika koji se koriste za izgradnju složenog udomljenika.

Treća informacija je skup događaja nad elementima grafičkog korisničkog sučelja udomljenika čijim se izvođenjem postiže povezivanje udomljenika u primjenski program. Četvrta informacija je prostorni razmještaj događaja po ćelijama tablice kojim se definiraju njihovi vremenski odnosi, odnosno redoslijed njihova izvođenja.

U [107], gdje je tablično programiranje uvedeno za izradu kompozicije usluga (engl. *service composition*), sadržaj tablice formalno je opisan Petrijevim mrežama (engl. *Petri nets*). Za potrebe formalnog opisa programa za usložnjavanje udomljenika, u ovom je poglavlju sadržaj tablice formalno opisan primjenom teorije skupova i teorije grafova. Ovi oblici formalnog opisa uobičajeno se koriste za prikaz podatkovnih i upravljačkih zavisnosti u računalnih programima. Primjenski program zapisan u tablici formalno se opisuje uređenom četvorkom prema formuli 12.1.

$$P = (w_R, W_E, D_P, S) \tag{12.1}$$

Oznakom P označava se primjenski program koji potrošač gradi usložnjavanjem udomljenika. Oznakom w_R označava se usložitelj, odnosno složeni udomljenik koji objedinjuje funkcionalnosti skupa povezanih osnovnih udomljenika i izlaže ih na korištenje potrošaču. Oznakom W_E označen je skup usloženika, odnosno osnovnih udomljenika korištenih za izgradnju usložitelja. Oznakom D_P označava se skup događaja koje nad elementima grafičkog korisničkog sučelja usloženika provodi usložitelj. Ti događaji zapisani su u ćelijama tablice. Oznakom S označava se relacija prostornog razmještaja događaja iz skupa D_P po ćelijama tablice.

Neka je oznakom G označen skup elemenata grafičkog korisničkog sučelja usložitelja i skupa usloženika, dok je oznakom C označen skup konstantnih vrijednosti koje predstavljaju sadržaj podatkovnih elemenata grafičkog korisničkog sučelja, kao što je tekst upisan u polje za unos teksta ili stavka odabrana u padajućem izborniku. Skup događaja koje nad elementima grafičkog korisničkog sučelja usloženika provodi usložitelj podijeljen je u četiri skupine i formalno opisan formulama 12.2, 12.3, 12.4, 12.5 i 12.6.

$$D = D_R \cup D_C \cup D_{CD} \cup D_{DF} \tag{12.2}$$

$$D_R = \{wait\ for\ click\ x\ at\ w_R\}, \forall x \in G \tag{12.3}$$

$$D_C = \left\{ \begin{matrix} click\ x\ at\ t, \\ double\ click\ x\ at\ t \end{matrix} \right\}, \forall t \in W_E, \forall x \in G \tag{12.4}$$

$$D_{CD} = \left\{ \begin{array}{l} type\ c\ into\ x\ at\ t, \\ check\ x\ at\ t, \\ uncheck\ x\ at\ t, \\ select\ x\ at\ t, \\ select\ c\ \ from\ x\ at\ t \end{array} \right\}, \forall t \in W_E, \forall x \in G, \forall c \in C \tag{12.5}$$

$$D_{DF} = \left\{ copy\ x\ at\ s\ to\ y\ at\ t \right\}, \forall s,t \in w_R \cup W_E, \forall x,y \in G \tag{12.6}$$

Prvu skupinu događaja koja je opisana formulom 12.3 čine događaji za upravljanje odzivom usložitelja na radnje potrošača (*wait for click*). Drugu skupinu događaja koja je opisana formulom 12.4 čine događaji kojima usložitelj pokreće izvođenje usloženika. Ovu skupinu čine događaji jednostrukog (*click*) i dvostrukog (*double click*) pritiska na aktivacijski element korisničkog sučelja usloženika. Treću skupinu događaja koja je opisana formulom 12.5 čine događaji kojima usložitelj priprema podatke koji se koriste tijekom izvođenja usloženika. Ovu skupinu čine događaji upisivanja sadržaja u polje za unos teksta (*type in*), događaji označavanja (*check*) i odznačavanja (*uncheck*) označnog polja te događaji obilježavanja izbornog polja ili odabira stavke iz padajućeg izbornika (*select*). Četvrtu skupinu događaja koja je opisana formulom 12.6 čine događaji za uspostavu toka podataka između dvaju usloženika ili između usložitelja i usloženika (*copy to*).

Formulama 12.3, 12.4, 12.5 i 12.6 opisani su svi mogući oblici događaja čija je uporaba dozvoljena tijekom izgradnje primjenskog programa. Skup događaja D_P koji je korišten za izgradnju primjenskog programa je podskup skupa svih mogućih događaja nad izabranim skupom udomljenika, a formalno je opisan formulom 12.7.

$$D_P \subseteq D \tag{12.7}$$

Prostorni razmještaj događaja po ćelijama tablice koji određuje redoslijed njihova izvođenja opisan je prostornom relacijom *S*. Relacija *S* formalno je opisana formulom 12.8.

$$S : D_P \times D_P \rightarrow \left\{ lijevo, iznad, \varnothing \right\} \tag{12.8}$$

Relacijom *S* definirani su prostorni odnosi između svakog pojedinog para događaja zapisanih u tablici. Postoje tri moguće vrijednosti relacije *S* za promatrani par događaja: između dvaju događaja vrijedi pomoćna relacija *lijevo*, između dvaju događaja vrijedi pomoćna relacija *iznad* ili dva događaja nisu u međusobnoj prostornoj relaciji. Pomoćne relacije *lijevo* i *iznad* opisane su formulama 12.9 i 12.10.

$$lijevo : D_P \times D_P \rightarrow \{ISTINA, LAŽ\} \tag{12.9}$$

$$lijevo(d_i, d_j) = \begin{Bmatrix} ISTINA \mid d_i \text{ je zapisan u ćeliji neposredno lijevo od } d_j \\ LAŽ \mid inače \end{Bmatrix}$$

$$iznad : D_P \times D_P \rightarrow \{ISTINA, LAŽ\} \tag{12.10}$$

$$iznad(d_i, d_j) = \begin{Bmatrix} ISTINA \mid d_i \text{ je zapisan u ćeliji neposredno iznad } d_j \\ LAŽ \mid inače \end{Bmatrix}$$

Kao što je opisano formulom 12.9, između dvaju događaja vrijedi relacija *lijevo* ako je prvi događaj u tablici zapisan u ćeliji koja je neposredno lijevo od ćelije u kojoj je zapisan drugi događaj. Slično, između dvaju događaja vrijedi relacija *iznad* ako je prvi događaj u tablici zapisan u ćeliji koja je neposredno iznad ćelije u kojoj je zapisan drugi događaj, kao što je opisano formulom 12.10.

Na osnovi prostorne relacije *S* definira se vremenska relacija *T* koja je formalno opisana formulom 12.11.

$$T = f(S) : D_P \times D_P \rightarrow \{ISTINA, LAŽ\} \tag{12.11}$$

Relacijom *T* definirani su vremenski odnosi između svakog pojedinog para događaja zapisanih u tablici. Ako između dvaju događaja vrijedi relacija *T*, onda izvođenje prvog događaja prethodi izvođenju drugog događaja. Ako između dvaju događaja ne vrijedi relacija *T*, onda je događaje moguće izvoditi proizvoljnim redoslijedom, uključujući i istodobno.

Proračun vremenske relacije *T* na osnovi prostorne relacije *S* definiran je formulom 12.12.

$$d_i \rightarrow d_j = T(d_i, d_j) = \begin{Bmatrix} ISTINA \mid lijevo(d_i, d_j) \ \vee \ iznad(d_i, d_j) \\ LAŽ \mid inače \end{Bmatrix} \tag{12.12}$$

Izvođenje događaja d_i prethodi izvođenju događaja d_j ako je događaj d_i zapisan u ćeliji koja se nalazi neposredno lijevo ili neposredno iznad ćelije u kojoj je zapisan događaj d_j.

Vremenska relacija T ima svojstvo asimetričnosti koje je opisano formulom 12.13. Za vremensku relaciju T definirana je relacija T^+ koja predstavlja tranzitivno okruženje relacije T. Relacija T^+ definirana je formulom 12.14.

$$d_i \to d_j \in T \Rightarrow d_j \to d_i \notin T \tag{12.13}$$

$$\begin{aligned} &d_i \to d_j \in T \Rightarrow d_i \to d_j \in T^+ \\ &d_i \to d_j \in T^+ \wedge d_j \to d_k \in T^+ \Rightarrow d_i \to d_k \in T^+ \end{aligned} \tag{12.14}$$

Svojstvom asimetričnosti određeno je jednosmjerno djelovanje vremenske relacije. Ako izvođenje događaja d_i prethodi izvođenju događaja d_j, onda događaj d_j ne smije biti izveden prije ni istodobno s izvođenjem događaja d_i. Tranzitivnim okruženjem vremenske relacije određeno je njeno ulančano djelovanje. Ako izvođenje događaja d_i prethodi izvođenju događaja d_j, a izvođenje događaja d_j prethodi izvođenju događaja d_k, onda izvođenje događaja d_i prethodi izvođenju događaja d_k.

12.2 Graf upravljačke uključenosti

Grafom upravljačke uključenosti (engl. *control inclusion graph*) opisuje se stupanj oslonjenosti usložitelja na funkcije usloženika. Stupnjem oslonjenosti usložitelja na funkcije usloženika smatra se broj operacija koje usložitelj pokreće nad usloženicima kako bi se obavila funkcija primjenskog programa. U primjenskom programu za usložnjavanje udomljenika, pokretanje operacija nad usloženicima obavlja se izvođenjem događaja jednostrukog (*click*) ili dvostrukog (*double click*) pritiska na aktivacijski element grafičkog korisničkog sučelja. Graf upravljačke uključenosti formalno je opisan formulom 12.15.

$$G_{CI} = (V, E)$$

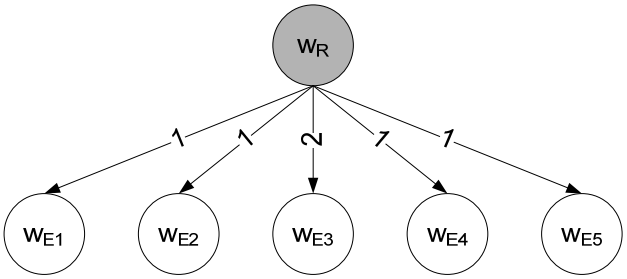
$$V = w_R \cup W_E$$

$$E = \left\{ (w_R, t) \middle| \exists d \in D_P \wedge t \in W_E \wedge \begin{pmatrix} d = click\ x\ at\ t\ \vee \\ d = double\ click\ x\ at\ t \end{pmatrix} \right\} \tag{12.15}$$

$$D_{control}(t) = \left\{ d \in D_P \middle| \exists t \in W_E \wedge \begin{pmatrix} d = click\ x\ at\ t\ \vee \\ d = double\ click\ x\ at\ t \end{pmatrix} \right\}, \forall t \in W_E$$

$$težina\left((w_R, t)\right) = \left|D_{control}(t)\right|, \forall t \in W_E$$

Graf upravljačke uključenosti je usmjereni težinski graf. Čvorove grafa čini usložitelj i skup usloženika. Grane grafa čine događaji za pokretanje aktivnosti nad usloženicima koje usložitelj pokreće izvođenjem operacija jednostrukog ili dvostrukog pritiska na element korisničkog sučelja usloženika. Usložitelj se povezuje usmjerenom granom sa svim usloženicima nad kojima izvodi barem jednu operaciju pritiska na element korisničkog sučelja. Grane grafa usmjerene su od usložitelja prema usloženicima i predstavljaju smjer upravljanja. Težina pojedine grane jednaka je broju operacija koje usložitelj izvodi nad usloženikom koji je tom granom povezan s usložiteljem.



Slika 12.3. Graf upravljačke uključenosti

Slikom 12.3 prikazan je graf upravljačke uključenosti za primjenski program prikazan slikom 12.1 i tablicom na slici 12.2. Graf se sastoji od šest čvorova, od kojih vršni čvor predstavlja usložitelj, dok preostalih pet čvorova predstavlja usloženike. U zadanom primjenskom programu, usložitelj pokreće ukupno šest operacija nad pet usloženika. Nad usloženicima w_{E1}, w_{E2}, w_{E4} i w_{E5} pokreće po jednu operaciju koje su u tablici na slici 12.2 redom označene oznakama 2, 5, 9 i 15, dok nad usloženikom w_{E3} pokreće dvije operacije koje su označene oznakama 7 i 13.

12.3 Graf podatkovne uključenosti

Grafom podatkovne uključenosti (engl. *data inclusion graph*) opisuje se stupanj oslonjenosti usložitelja na podatke usloženika. Stupnjem oslonjenosti usložitelja na podatke usloženika smatra se zbroj podataka koje usložitelj čita s usloženika i podataka koje usložitelj upisuje u usloženike. U primjenskom programu za usložnjavanje udomljenika, upisivanje podataka u usloženike predstavljeno je događajem za upisivanje sadržaja u polje za unos teksta (*type in*), događajem za označavanje označnog polja (*check*), događajem za odznačavanje označnog polja (*uncheck*) i događajem za obilježavanje izbornog polja i izbor stavke iz padajućeg izbornika (*select*). Čitanje podataka s usloženika predstavljeno je

događajem za prijenos podataka između elemenata korisničkog sučelja udomljenika (*copy to*). Događaj za prijenos podataka je složeni događaj koji se sastoji od operacije čitanja i operacije upisivanja podataka. Graf podatkovne uključenosti formalno je opisan formulom 12.16.

$$G_{DI} = (V, E)$$

$$V = w_R \cup W_E$$

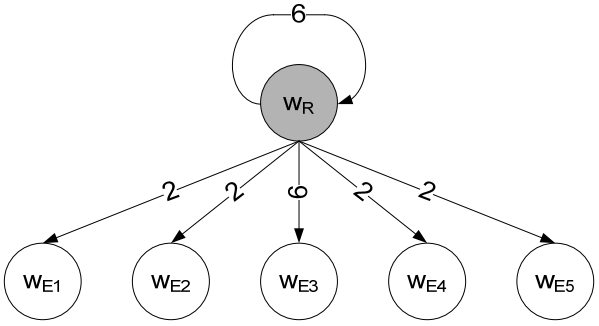
$$E = \left\{ (w_R, t) \middle| \exists d \in D_P \wedge \exists s, t \in w_R \cup W_E \wedge \begin{pmatrix} d = type\ c\ into\ x\ at\ t\ \vee \\ d = check\ x\ at\ t\ \vee \\ d = uncheck\ x\ at\ t\ \vee \\ d = select\ x\ at\ t\ \vee \\ d = select\ c\ from\ x\ at\ t\ \vee \\ d = copy\ x\ at\ s\ to\ y\ at\ t\ \vee \\ d = copy\ x\ at\ t\ to\ y\ at\ s \end{pmatrix} \right\} \tag{12.16}$$

$$D_{data}(t) = \left\{ d \in D_P \middle| \exists t \in w_R \cup W_E \wedge \begin{pmatrix} d = type\ c\ into\ x\ at\ t\ \vee \\ d = check\ x\ at\ t\ \vee \\ d = uncheck\ x\ at\ t\ \vee \\ d = select\ x\ at\ t\ \vee \\ d = select\ c\ from\ x\ at\ t\ \vee \\ d = copy\ x\ at\ s\ to\ y\ at\ t\ \vee \\ d = copy\ x\ at\ t\ to\ y\ at\ s \end{pmatrix} \right\}, \forall t \in w_R \cup W_E$$

$$težina\left((w_R, t)\right) = \left|D_{data}(t)\right|, \forall t \in w_R \cup W_E$$

Graf podatkovne uključenosti je usmjereni težinski graf. Čvorove grafa čini usložitelj i skup usloženika. Grane grafa čine podatkovne operacije za čitanje i upisivanje sadržaja u elemente grafičkog korisničkog sučelja udomljenika. Usložitelj se povezuje usmjerenom granom sa svim usloženicima nad kojima izvodi barem jednu podatkovnu operaciju. Događaji za upisivanje sadržaja u polje za unos teksta (*type in*), za označavanje označnog polja (*check*), za odznačavanje označnog polja (*uncheck*) te za obilježavanje izbornog polja i izbor stavke iz padajućeg izbornika (*select*) su operacije upisivanja sadržaja. Za ovu vrstu događaja usmjerena grana postavlja se od čvora usložitelja prema čvoru odredišnog usloženika. Događaj za prijenos podataka između elemenata korisničkog sučelja udomljenika (*copy to*) sastoji od operacije čitanja i operacije upisivanja podataka. Za ovu

vrstu događaja u grafu postoje dvije usmjerene grane od kojih je jedna usmjerena od usložitelja prema izvorišnom usloženiku, a druga od usložitelja prema odredišnom usloženiku. Ako usložitelj izvodi podatkovnu operaciju nad samim sobom, usmjerena grana započinje i završava na čvoru usložitelju. Podatkovne operacije koje usložitelj izvodi nad samim sobom su događaji za prijenos podataka iz ulaznih elemenata usložitelja u odredišne elemente usloženika i događaji za prijenos rezultata iz izlaznih elemenata usloženika u izlazne elemente usložitelja. Težina pojedine grane grafa jednaka je zbroju operacija čitanja i upisivanja sadržaja koje usložitelj izvodi nad usloženikom koji je tom granom povezan s usložiteljem. Za grane koje počinju i završavaju u čvoru usložitelju, težina grane jednaka je zbroju operacija čitanja i upisivanja sadržaja koje usložitelj izvodi nad samim sobom.



Slika 12.4. Graf podatkovne uključenosti

Slikom 12.4 prikazan je graf podatkovne uključenosti za primjenski program prikazan slikom 12.1 i tablicom na slici 12.2. Graf se sastoji od šest čvorova, od kojih vršni čvor predstavlja usložitelj, dok preostalih pet čvorova predstavlja usloženike. U zadanom primjenskom programu, usložitelj izvodi ukupno 20 podatkovnih operacija. Nad usloženicima w_{E1}, w_{E2}, w_{E4} i w_{E5} izvodi po dvije podatkovne operacije od kojih je jedna operacija čitanja, a druga operacija upisivanja sadržaja. Te su operacije u tablici na slici 12.2 redom označene oznakama 1 i 3 za usloženik w_{E1}, oznakama 4 i 6 za usloženik w_{E2}, oznakama 8 i 10 za usloženik w_{E4} te oznakama 14 i 16 za usloženik w_{E5}. Nad usloženikom w_{E3} usložitelj izvodi šest podatkovnih operacija, od kojih su dvije operacije čitanja, a četiri operacije upisivanja sadržaja. Te su operacije u tablici na slici 12.2 redom označene oznakama 3, 6, 8, 11, 12 i 14. Osim operacija nad usloženicima, usložitelj izvodi šest podatkovnih operacija nad samim sobom. Četiri operacije koriste se za prijenos ulaznih podataka s usložitelja na usloženike w_{E1}, w_{E2} i w_{E3}. Te su operacije u tablici na slici 12.2 redom označene oznakama 1, 4, 11 i 12. Dvije operacije koriste se za prijenos rezultata s usloženika w_{E4} i w_{E5} na usložitelj. Te su operacije u tablici na slici 12.2 redom označene oznakama 10 i 16.

12.4 Graf podatkovne povezanosti

Grafom podatkovne povezanosti (engl. *data connectivity graph*) opisuje se tok podataka između usložitelja i usloženika te između pojedinih usloženika. U primjenskom programu za usložnjavanje udomljenika, tok podataka od usložitelja prema usloženiku uspostavlja se izvođenjem događaja za upisivanje sadržaja u polje za unos teksta (*type in*), događajem za označavanje označnog polja (*check*), događajem za odznačavanje označnog polja (*uncheck*), događajem za obilježavanje izbornog polja i izbor stavke iz padajućeg izbornika (*select*) i događajem za prijenos podataka (*copy to*). Tok podataka od usloženika prema usložitelju, kao i tok podataka između dvaju usloženika uspostavlja se isključivo događajem za prijenos podataka (*copy to*). Graf podatkovne povezanosti formalno je opisan formulom 12.17.

$$G_{DC} = (V, E)$$

$$V = w_R \cup W_E$$

$$E = E_1 \cup E_2$$

$$E_1 = \left\{ (w_R, t) \middle| \exists d \in D_P \wedge \exists t \in W_E \wedge \begin{pmatrix} d = type\ c\ into\ x\ at\ t\ \vee \\ d = check\ x\ at\ t\ \vee \\ d = uncheck\ x\ at\ t\ \vee \\ d = select\ x\ at\ t\ \vee \\ d = select\ c\ \ from\ x\ at\ t \end{pmatrix} \right\} \tag{12.17}$$

$$E_2 = \left\{ (s,t) \middle| \exists d \in D_P \wedge \exists s,t \in w_R \cup W_E \wedge d = copy\ x\ at\ s\ to\ y\ at\ t \right\}$$

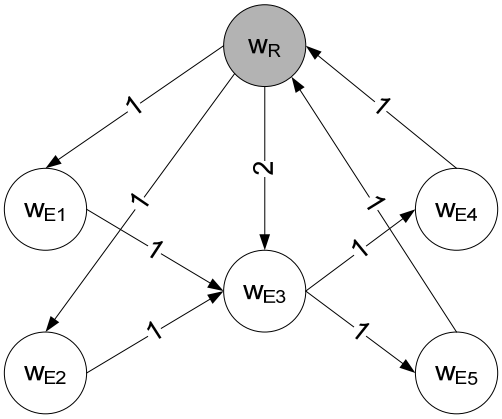
$$D_{DC1}(w_R, t) = \left\{ d \in D_P \middle| \exists t \in W_E \wedge \begin{pmatrix} d = type\ c\ into\ x\ at\ t\ \vee \\ d = check\ x\ at\ t\ \vee \\ d = uncheck\ x\ at\ t\ \vee \\ d = select\ x\ at\ t\ \vee \\ d = select\ c\ \ from\ x\ at\ t \end{pmatrix} \right\}, \forall t \in W_E$$

$$D_{DC2}(s,t) = \left\{ d \in D_P \middle| \exists s,t \in w_R \cup W_E \wedge d = copy\ x\ at\ s\ to\ y\ at\ t \right\}, \forall s,t \in w_R \cup W_E$$

$$težina\left((w_R, t)\right) = \left| D_{DC1}(w_R, t) \right| + \left| D_{DC2}(w_R, t) \right|, \forall t \in W_E$$

$$težina\left((s,t)\right) = \left|D_{DC2}(s,t)\right|, \forall s \in W_E, t \in w_R \cup W_E$$

Graf podatkovne povezanosti je usmjereni težinski graf. Čvorove grafa čini usložitelj i skup usloženika. Grane grafa čine operacije za uspostavu toka podataka između usložitelja i usloženika te između dvaju usloženika. Usložitelj se povezuje usmjerenom granom sa svim usloženicima prema kojima prenosi podatke ili s kojih podatke prenosi prema sebi. Dva usloženika povezuju se usmjerenom granom ako postoji definirani događaj za prijenos podataka između tih dvaju usloženika. Grane grafa usmjerene su u smjeru prijenosa podataka. Težina pojedine grane jednaka je broju operacija prijenosa podataka između usložitelja i usloženika, odnosno između dvaju usloženika.



Slika 12.5. Graf podatkovne povezanosti

Slikom 12.5 prikazan je graf podatkovne povezanosti za primjenski program prikazan slikom 12.1 i tablicom na slici 12.2. Graf se sastoji od šest čvorova, od kojih vršni čvor predstavlja usložitelj, dok preostalih pet čvorova predstavlja usloženike. U zadanom primjenskom programu, usložitelj izvodi ukupno 10 operacija za prijenos podataka. Operacijom 1 u tablici na slici 12.2 obavlja se prijenos ulaznog parametra s usložitelja w_R na usloženik w_{E1}. Operacijom 4 obavlja se prijenos drugog ulaznog parametra s usložitelja w_R na usloženik w_{E2}. Na osnovi tih operacija, u grafu podatkovne povezanosti postoje usmjerene grane od usložitelja w_R prema usloženicima w_{E1} i w_{E2}. Operacijama 3 i 6 obavlja se prijenos rezultata izvođenja s usloženika w_{E1} i w_{E2} u ulazna polja usloženika w_{E3}. Na osnovi tih operacija, u grafu podatkovne povezanosti postoje usmjerene grane od usloženika w_{E1} i w_{E2} prema usloženiku w_{E3}. Operacijama 8 i 14 obavlja se prijenos rezultata izvođenja s usloženika w_{E3} u ulazna polja usloženika w_{E4} i w_{E5}. Na osnovi tih operacija, u grafu podatkovne povezanosti postoje usmjerene grane od usloženika w_{E3} prema usloženicima w_{E4} i w_{E5}. Operacijama 10 i 16 obavlja se prijenos rezultata izvođenja s usloženika w_{E4} i w_{E5} do

usložitelja w_R, gdje se ti podaci prikazuju potrošaču. Na osnovi tih operacija, u grafu podatkovne povezanosti postoje usmjerene grane od usloženika w_{E4} i w_{E5} prema usložitelju w_R. Operacijama 11 i 12 obavlja se prijenos dvaju ulaznih parametara s usložitelja w_R do usloženika w_{E3}. Na osnovi tih operacija, u grafu podatkovne povezanosti postoji usmjerena grana od usložitelja w_R prema usloženiku w_{E3}. Dvije operacije označene su jednom granom čija je težina, zbog dviju operacija, postavljena na vrijednost 2.

12.5 Graf upravljačke povezanosti

Grafom upravljačke povezanosti (engl. *control connectivity graph*) opisuje se tijek upravljanja izvođenjem skupa usloženika. U primjenskom programu za usložnjavanje udomljenika, upravljanje izvođenjem skupa usloženika postiže se izvođenjem događaja jednostrukog (*click*) ili dvostrukog (*double click*) pritiska na aktivacijski element grafičkog korisničkog sučelja usloženika. Grafom upravljačke povezanosti opisuje se prijenos upravljačkog tijeka između usloženika u skladu s vremenskom relacijom definiranom nad programom zapisanim u ćelijama tablice.

$$G_{CC} = (V, E)$$

$$V = W_E \tag{12.18}$$

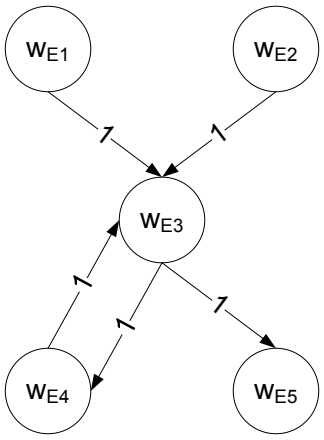
$$E = \left\{ (s,t) \,\middle|\, \exists d_i, d_j \in D_P \wedge s,t \in W_E \wedge \left(\begin{array}{l} \left(d_i = click\ x\ at\ s\ \vee\ d_i = double\ click\ x\ at\ s\right) \\ \wedge \\ \left(d_j = click\ y\ at\ t\ \vee\ d_j = double\ click\ y\ at\ t\right) \\ \wedge \\ d_i \to d_j \wedge \nexists d_k \mid d_i \to d_k \to d_j \end{array} \right) \right\}$$

$$D_{CC}(s,t) = \left\{ d_i \in D_P \,\middle|\, \exists d_j \in D_P \wedge s,t \in W_E \wedge \left(\begin{array}{l} \left(d_i = click\ x\ at\ s\ \vee\ d_i = double\ click\ x\ at\ s\right) \\ \wedge \\ \left(d_j = click\ y\ at\ t\ \vee\ d_j = double\ click\ y\ at\ t\right) \\ \wedge \\ d_i \to d_j \wedge \nexists d_k \mid d_i \to d_k \to d_j \end{array} \right) \right\}$$
$$, \forall s,t \in W_E$$

$$težina\left((s,t)\right) = \left|D_{CC}(s,t)\right|, \forall s,t \in W_E$$

Graf upravljačke povezanosti formalno je opisan formulom 12.18. Graf upravljačke povezanosti je usmjereni težinski graf. Čvorove grafa čini skup usloženika. Grane grafa predstavljaju relacije prethođenja tijekom izvođenja događaja za pokretanje operacija nad usloženicima. Ako uslożitelj pokreće dvije uzastopne operacije nad dva različita usloženika, ta se dva usloženika povezuju usmjerenom granom. Ako uslożitelj pokreće dvije uzastopne operacije nad istim usloženikom, usloženik se usmjerenom granom povezuje sa samim sobom. Dva usloženika ili usloženik sa samim sobom povezuju se usmjerenom granom i ako dvije operacije nisu uzastopne, ali se između njih ne izvodi niti jedna druga operacija za pokretanje usloženika, već isključivo podatkovne operacije. Grana kojom su povezani čvorovi grafa usmjerena je od usloženika nad kojim uslożitelj operaciju izvodi prije prema usloženiku nad kojim uslożitelj operaciju izvodi kasnije. Težina pojedine grane jednaka je broju slučajeva u kojima se dva promatrana usloženika pokreću neposredno jedan iza drugog, odnosno u kojima se isti usloženik pokreće više puta uzastopno.

S obzirom na to da je moguće da uslożitelj nad istim usloženicima pokreće veći broj operacija, u grafu je moguća pojava petlji. Ako se između pokretanja dviju operacija nad istim usloženikom pokreće skup operacija nad ostalim usloženicima, promatrani usloženik predstavljen je čvorom koji je početni i završni čvor petlje. Ostali usloženici čije se izvođenje pokreće između dviju operacija nad istim usloženikom predstavljaju čvorove grafa na putu koji čini petlju.



Slika 12.6. Graf upravljačke povezanosti

Slikom 12.6 prikazan je graf upravljačke povezanosti za primjenski program prikazan slikom 12.1 i tablicom na slici 12.2. Graf se sastoji od pet čvorova koji predstavljaju pet usloženika od kojih je izgrađen uslożitelj. U zadanom primjenskom programu, uslożitelj pokreće ukupno šest operacija nad pet usloženika. Nakon početne pripreme podataka,

usložitelj pokreće izvođenje usloženika w_{E1} i w_{E2}. Pokretanje tih dvaju usloženika izvodi se međusobno nezavisno u vremenu, što je u tablici prikazano operacijama 2 i 5. Na osnovi tih dviju operacija, u grafu upravljačke povezanosti usloženici w_{E1} i w_{E2} nemaju čvorova prethodnika i nisu međusobno povezani. Nakon pokretanja usloženika w_{E1} i w_{E2}, usložitelj izvođenjem operacije 7 pokreće izvođenje usloženika w_{E3}. S obzirom na to da su usloženici w_{E1} i w_{E2} vremenski prethodnici usloženika w_{E3}, u grafu upravljačke povezanosti usloženici w_{E1} i w_{E2} povezuju se usmjerenim granama s usloženikom w_{E3}. Nakon pokretanja usloženika w_{E3}, usložitelj izvođenjem operacije 9 pokreće izvođenje usloženika w_{E4}. Zbog toga se u grafu upravljačke povezanosti usloženik w_{E3} povezuje usmjerenom granom s usloženikom w_{E4}. Nakon pokretanja usloženika w_{E4}, usložitelj izvođenjem operacije 13 po drugi puta pokreće izvođenje usloženika w_{E3}. Zbog toga se u grafu upravljačke povezanosti usloženik w_{E4} povezuje usmjerenom granama s usloženikom w_{E3}. Budući da se izvođenje usloženika w_{E3} pokreće dva puta, od kojih prvo izvođenje prethodi, a drugo slijedi neposredno nakon pokretanja izvođenja usloženika w_{E4}, usloženici w_{E3} i w_{E4} u grafu upravljačke povezanosti čine petlju. Nakon drugog pokretanja usloženika w_{E3}, usložitelj izvođenjem operacije 15 pokreće izvođenje usloženika w_{E5}. Zbog toga se u grafu upravljačke povezanosti usloženik w_{E3} povezuje usmjerenom granama s usloženikom w_{E5}.

13 Ocjena izražajnosti programske paradigme

Programiranje prilagođeno potrošaču zasnovano na programskoj paradigmi za usložnjavanje udomljenika omogućuje širokom krugu korisnika računala bez formalnog obrazovanja u području programskog inženjerstva samostalno uključivanje u razvoj primjenskih programa. Kako bi se stvaralačke mogućnosti programske paradigme za usložnjavanje udomljenika usporedile s programskim paradigmama koje se koriste u grani profesionalnog razvoja programske potpore i u grani razvoja prilagođenog krajnjim korisnicima u pojedinim područjima primjene, u ovom je poglavlju definirana mjera *stvaralačkih mogućnosti programske paradigme*. Pokazano je da zbog kratkog vremena potrebnog za razvoj programa i velikog broja korisnika kojima je omogućeno sudjelovanje u njihovu razvoju, stvaralačke mogućnosti programske paradigme za usložnjavanje udomljenika mnogostruko premašuju stvaralačke mogućnosti strojnog programiranja te objektno-orijentiranih i primjenski usmjerenih programskih paradigmi. Međutim, zbog primjetnog nedostatka kritičnog broja udomljenika, stvaralačke mogućnosti programske paradigme za usložnjavanje udomljenika danas nije moguće u potpunosti iskoristiti. Prema izvršenim procjenama porasta broja udomljenika, punu uporabljivost programske paradigme moguće je postići za tri do devet godina. Kako bi se utvrdila *potpunost paradigme*, njena izražajnost ispitana je s obzirom na Turingovu potpunost (engl. *Turing completeness*). Pokazano je preslikavanje pojedinih elemenata Turingove potpunosti na elemente programske paradigme za usložnjavanje udomljenika te istovjetnost paradigme s modelom Turingovog stroja. Ocjena stvaralačkih mogućnosti programske paradigme opisana je u poglavlju 13.1, dok je ocjena izražajnosti paradigme s obzirom na Turingovu potpunost opisana u poglavlju 13.2.

13.1 Ocjena stvaralačkih mogućnosti programske paradigme

Mjerom stvaralačkih mogućnosti programske paradigme procjenjuje se ukupna količina novih primjenskih programa koju zajednica korisnika može razviti u određenom razdoblju primjenom određene programske paradigme. Mjera je zasnovana na tri veličine: vremenu razvoja primjenskih programa, količini i raznovrsnosti elemenata raspoloživih za izgradnju primjenskih programa te broju korisnika koji ovladavaju znanjima i vještinama za primjenu programske paradigme.

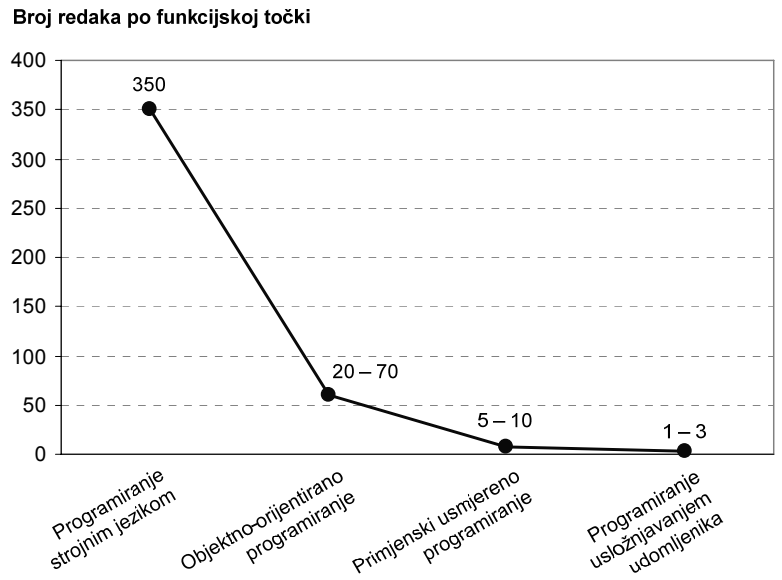
13.1.1 Vrijeme razvoja primjenskih programa

Prva veličina koja se koristi za procjenu stvaralačkih mogućnosti programske paradigme je vrijeme razvoja primjenskih programa. Što je vrijeme potrebno za razvoj programa kraće, to je veći broj programa moguće razviti u promatranom razdoblju. Za određivanje vremena razvoja primjenskih programa koristi se procjena količine programske logike koja je potrebna za razvoj programa. Manja količina potrebne programske logike podrazumijeva brži razvoj programa.

S obzirom na to da različite programske paradigme i jezici zahtijevaju različite količine programske logike za ostvarenje istih programskih funkcionalnosti, razvijeni su posebni postupci za procjenu veličine programa neovisno o korištenoj programskoj paradigmi i jeziku. Jedna od najčešće primjenjivanih tehnika za procjenu veličine programa je *mjerenje funkcijskih značajki programa* (engl. *functional size measurement – FSM*) [238, 239]. Mjerenje funkcijskih značajki programa je međunarodno prihvaćena i od strane organizacije ISO standardizirana tehnika za procjenu veličine programske potpore.

Tehnikom mjerenja funkcijskih značajki programa, veličina programa izražava se brojem Albrechtovih funkcijskih točaka (engl. *Albrecht's function point*) [240, 241]. Tehniku je osmislio znanstvenik Allan Albrecht u kasnim 1970. godinama za vrijeme dok je radio u tvrtci IBM. Albrechtova funkcijska točka je osnovna jedinica programske potpore koja za korisnika predstavlja nedjeljivu funkcijsku cjelinu računalnog programa. Primjeri programskih funkcionalnosti koji predstavljaju tipične oblike Albrechtovih funkcijskih točaka su obrada ulaznih podataka preuzetih od korisnika putem korisničkog sučelja, prikaz rezultata izvođenja programa te obavljanje transakcije između korisničkog i poslužiteljskog dijela programa. S obzirom na to da je nezavisna o programskoj paradigmi, jeziku i razvojnom alatu korištenom za razvoj programa, tehnika mjerenja funkcijskih značajki programa zamijenila je prethodno korištenu tehniku u kojoj se veličina programa izražavala brojem redaka programske logike (engl. *lines of code*) u ostvarenju programa. Dok je

veličina programa izražena brojem Albrechtovih funkcijskih točaka odraz količine funkcionalnosti ugrađenih u promatrani program, veličina programa izražena brojem redaka programske logike svojstvo je programskog ostvarenja tih funkcionalnosti u promatranom programskom jeziku. S obzirom na to da se primjenski programi različitog opsega programskih funkcionalnosti sastoje od različitog broja funkcijskih točaka, za određivanje vremena razvoja primjenskog programa koristi se količina programske logike koja je potrebna za razvoj jedne funkcijske točke.



Slika 13.1. Količina programske logike potrebne za razvoj Albrechtove funkcijske točke primjenom pojedinih oblika programskih paradigmi

Broj redaka programske logike koja je potrebna za ostvarenje funkcijske točke utvrđuje se empirijskim ispitivanjima pojedinih programskih paradigmi i programskih jezika. Rezultati empirijskih ispitivanja prikazani su slikom 13.1. Četiri istaknute točke grafa prikazuju prosječne vrijednosti broja redaka programske logike koja je potrebna za ostvarenje Albrechtove funkcijske točke primjenom strojnog jezika te objektno-orijentirane, primjenski usmjerene i potrošaču prilagođene programske paradigme zasnovane na usložnjavanju udomljenika.

Empirijskom analizom programskih jezika utvrđeno je da programiranje strojnim jezikom prosječno zahtijeva nekoliko stotina strojnih naredbi za ostvarenje jedne funkcijske točke [242]. Objektno-orijentirani programski jezici, kao što su Ada, Smalltalk, C++ i Java, tipično zahtijevaju nekoliko desetaka redaka programske logike za ostvarenje jedne funkcijske točke [242]. Primjenski usmjereni programski jezici i specijalizirani razvojni alati posebno prilagođeni pojedinim područjima primjene, kao što su jezici za izradu tabličnih

proračuna ili upitni jezici za rukovanje bazama podataka, tipično zahtijevaju desetak redaka programske logike za ostvarenje jedne funkcijske točke [242]. Programiranje zasnovano na usložnjavanju udomljenika omogućava ostvarenje programske logike za rukovanje korisniku vidljivim ulaznim podacima, rezultatima izvođenja programa ili transakcijama između korisničkih i poslužiteljskih dijelova programa, što predstavlja tipične primjere Albrechtovih funkcijskih točaka, u svega jednom ili nekoliko redaka.

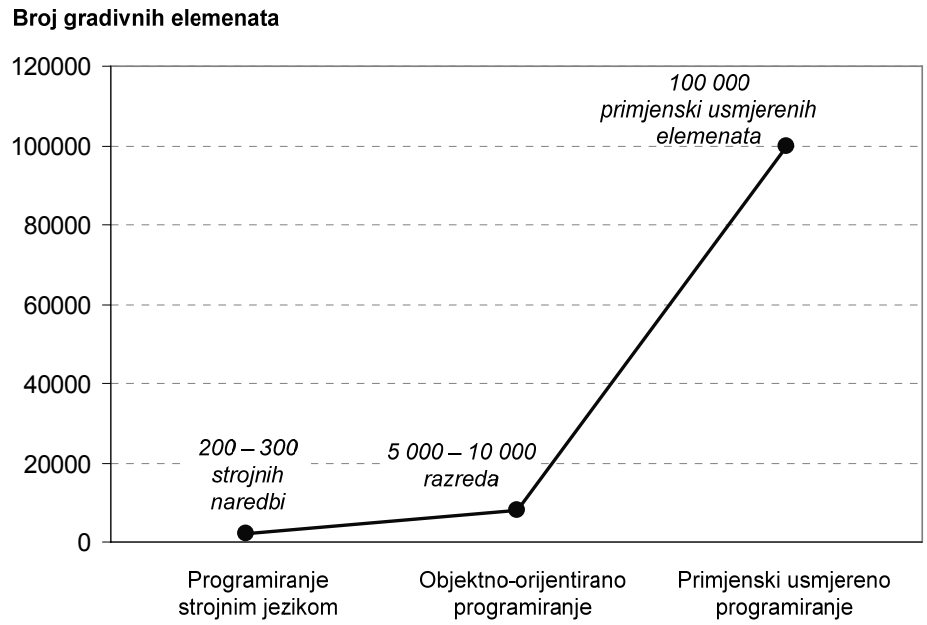
Ovi podaci pokazuju da se podizanjem razine apstrakcije programskih elemenata od kojih se grade primjenski programi postiže približno eksponencijalni pad količine programske logike potrebne za programsko ostvarenje Albrechtovih funkcijskih točaka. Time se istodobno postiže približno eksponencijalno skraćivanje vremena razvoja programske potpore. Unutar predviđenog područja primjene, primjenski usmjereni programski jezici skraćuju vrijeme razvoja programske potpore za red veličine u odnosu na objektno-orijentirane programske jezike, dok su objektno-orijentirane programske paradigme zasnovane na višim programskim jezicima za red veličine učinkovitije od programiranja strojnim jezikom. Programiranje zasnovano na usložnjavanju udomljenika, gdje je izgradnja primjenskih programa zasnovana na povezivanju gotovih i potpuno funkcionalnih programskih blokova u radne tijekove prema zamislima potrošača, skraćuje vrijeme razvoja primjenskih programa za dodatni red veličine u odnosu na primjenski usmjereno programiranje.

13.1.2 Dostupnost gradivnih elemenata

Druga veličina koja se koristi za procjenu stvaralačkih mogućnosti programske paradigme je izražajnost. Izražajnost programske paradigme uobičajeno se definira na dva načina. Formalni način utvrđivanja izražajnosti programske paradigme je usporedba mogućnosti računanja s obzirom na teoriju izračunljivosti, odnosno na Turingovu potpunost (engl. *Turing completeness*) [264, 265]. Ovim oblikom izražajnosti utvrđuje se skup izračunljivih funkcija koje je moguće opisati primjenom promatrane programske paradigme, neovisno o tome kolika je složenost, opsežnost i razumljivost programskog zapisa. Ocjena izražajnosti programske paradigme za usložnjavanje udomljenika s obzirom na Turingovu potpunost opisana je u poglavlju 13.2. Neformalni način utvrđivanja izražajnosti programske paradigme je procjena uporabljivosti paradigme za razvoj ciljne skupine primjenskih programa od strane ciljne skupine korisnika. Ovim oblikom izražajnosti ocjenjuje se jednostavnost, razumljivost i sažetost strukture primjenskog programa. Za ocjenu stvaralačkih mogućnosti programske paradigme za usložnjavanje udomljenika, u ovom je poglavlju korišten neformalni način procjene izražajnosti.

Programska paradigma za usložnjavanje udomljenika pripada skupini paradigmi za izgradnju primjenskih programa povezivanjem programskih komponenata. Za ovu vrstu programskih paradigmi svojstveno je premještanje naglaska s ostvarenja jezgrenih programskih funkcionalnosti na iskorištavanje gotovih funkcionalnosti dostupnih u obliku programskih komponenata. Izgradnja primjenskih programa zasniva se na povezivanju skupa gotovih komponenata u radni tijek prema zakonitostima primjenskog programa.

Izražajnost programske paradigme za izgradnju primjenskih programa povezivanjem programskih komponenata u pravilu je usko povezana s brojnošću i raznolikošću dostupnih komponenata te njihovom raspršenošću po različitim područjima primjene. Povećanjem broja i raznolikosti gradivnih elemenata, povećava se raznovrsnost programskih funkcionalnosti raspoloživih za povezivanje u nove primjenske programe. Nadalje, povećanje raznovrsnosti dostupnih programskih funkcionalnosti dovodi do širenja područja primjene unutar kojeg je za izradu primjenskih programa pogodna primjena promatrane paradigme. Što je broj dostupnih gradivnih elemenata veći, to je područje primjene programske paradigme šire, a programska paradigma izražajnija.



Slika 13.2. Količina gradivnih elemenata kojom se postiže visoka učinkovitost pojedinih oblika programskih paradigmi

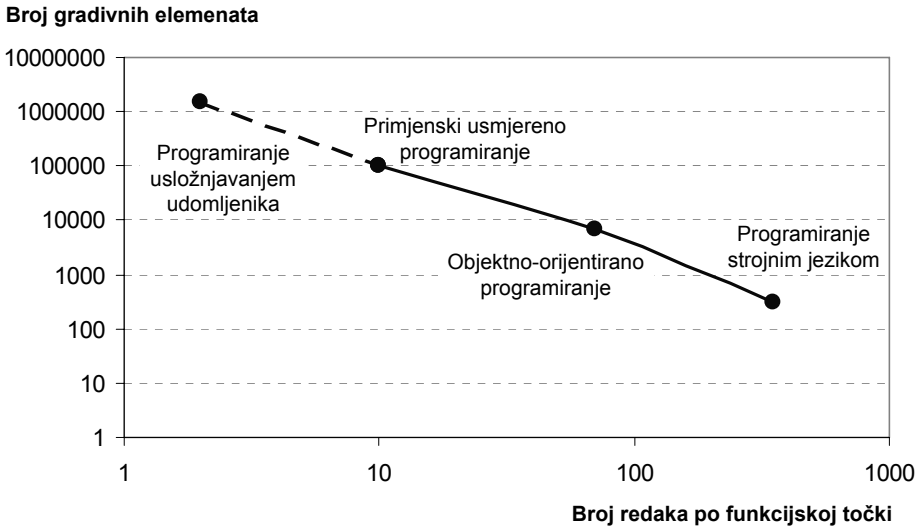
Tijekom ocjene izražajnosti pojedinih oblika programskih paradigmi, analizirana je veličina skupa gradivnih elemenata koji programeru stoje na raspolaganju tijekom izgradnje primjenskih programa. Rezultati su prikazani slikom 13.2. Tijekom programiranja strojnim jezikom, osnovnim elementima za izgradnju primjenskih programa smatraju se strojne

naredbe. Suvremeni procesori raspolažu skupom od nekoliko stotina strojnih naredbi, od osnovnog skupa naredbi za obavljanje aritmetičkih i logičkih operacija, do naprednih naredbi za računalnu grafiku i obradu višemedijskih podataka [244, 245]. Suvremene razvojne okoline zasnovane na objektno-orijentiranoj programskoj paradigmi raspolažu programskim knjižnicama od desetak tisuća razreda kako bi se postigao visoki stupanj učinkovitosti razvojnog postupka. Primjerice, najnovija inačica razvojne okoline *Java Development Kit* za programski jezik Java sadrži oko 4000 razreda [246], dok programska okolina *Microsoft .NET Framework* sadrži oko 11000 razreda [247].

Za razliku od programiranja strojnim jezikom i objektno-orijentiranih programskih paradigmi, gdje je zbog programskih jezika opće namjene za razvoj primjenskih programa dovoljan jedan programski jezik, primjenski usmjereno programiranje sadrži zasebne programske jezike za pojedina područja primjene. Procjenjuje se da se u današnje vrijeme koristi oko tisuću različitih primjenski usmjerenih programskih jezika, od široko primjenjivih jezika za izradu tabličnih proračuna, jezika za izradu *World Wide Web* stranica ili upitnih jezika za pristup bazama podataka, do usko specijaliziranih jezika za projektiranje sklopovlja ili izradu jezičnih procesora [248]. Analizom gradivnih elemenata u pojedinim primjenski usmjerenim jezicima, pokazalo se da takvi jezici tipično sadrže stotinjak primjenski usmjerenih programskih elemenata. Primjerice, jezici za izradu tabličnih proračuna prosječno sadrže nekoliko stotina funkcija, označni jezik *HTML* sadrži oko 80 elemenata za izradu *World Wide Web* stranica, upitni jezik *SQL* za pristup bazama podataka sadrži približno 20 naredbi za postavljanje upita, jezik *Risla* za izradu financijskih primjenskih programa sadrži stotinjak elemenata za opis financijskih proizvoda, jezik *SIAL* za opis paralelnih kemijskih procesa sadrži oko 40 elemenata, dok jezik *Alice* za izradu 3D animacija sadrži oko 250 elemenata za definiranje i upravljanje objektima scene. Iz analize broja primjenski usmjerenih programskih jezika i količine gradivnih elemenata u pojedinim jezicima, proizlazi da je pokrivenost različitih područja primjene postignuta skupom od približno sto tisuća primjenski usmjerenih programskih elemenata.

Kako bi se procijenila količina udomljenika kojom se postiže prag visoke učinkovitosti programske paradigme za usložnjavanje udomljenika, izvršena je analiza utjecaja zrnatosti gradivnih elemenata na veličinu skupa tih elemenata. Slikom 13.3 prikazana je zavisnost broja redaka programske logike potrebne za ostvarenje jedne Albrechtove funkcijske točke u pojedinoj programskoj paradigmi i broja gradivnih elemenata kojima se postiže visoki stupanj učinkovitosti promatrane paradigme. Zbog velikog raspona vrijednosti, obje osi grafičkog prikaza prikazane su u logaritamskom mjerilu.

Iz prikazanog grafičkog prikaza uočljiva je približno eksponencijalna zavisnost broja redaka programske logike potrebne za ostvarenje jedne Albrechtove funkcijske točke i broja gradivnih elemenata kojima se postiže visoki stupanj učinkovitosti programske paradigme. Smanjenjem broja redaka programske logike po funkcijskoj točki, dolazi do eksponencijalnog porasta broja potrebnih gradivnih elemenata jer se smanjenje količine programske logike po funkcijskoj točki postiže povećanjem zrnatosti gradivnih elemenata. S druge strane, povećanjem zrnatosti gradivnih elemenata raste broj ograničenja o mogućnostima njihova međusobnog povezivanja pa je za održavanje razine izražajnosti paradigme potrebno povećati njihov broj.



Slika 13.3. Procjena praga učinkovitosti programske paradigme za usložnjavanje udomljenika

Procjena količine udomljenika potrebnih za postizanje zadovoljavajuće razine izražajnosti programske paradigme za usložnjavanje udomljenika izvršena je ekstrapolacijom krivulje prikazane slikom 13.3 na osnovi triju poznatih točaka. Ekstrapolacija krivulje po vodoravnoj osi izvršena je do točke koja označava broj redaka programske logike po funkcijskoj točki uz primjenu programske paradigme za usložnjavanje udomljenika. Pod pretpostavkom da se trend porasta količine potrebnih gradivnih elemenata nastavi po približno istom zakonu, smanjenje količine programske logike po funkcijskoj točki na svega nekoliko redaka zahtijevalo bi skup gradivnih elemenata reda veličine milijun udomljenika, uz njihovu približno jednoliku raspršenost po pojedinim područjima primjene.

Analiza današnje količine udomljenika

Broj udomljenika raspoloživih za izgradnju primjenskih programa u današnje je vrijeme značajno ispod zahtijevanog praga učinkovitosti od milijun udomljenika.

Najopsežniji javni spremnik dostupnih udomljenika je *Google Gadgets Directory* imenik [251] koji okuplja oko 140 tisuća udomljenika. Zajedno s ostalim javno dostupnim spremnicima, broj udomljenika danas ne prelazi 200 tisuća. Udomljenici dostupni u *Google Gadgets Directory* imeniku zasnovani su na *Google Gadgets* specifikaciji [211]. Predviđeni su za postavljanje na poosobljene *iGoogle* udomiteljske stranice [252], ali ih je moguće udomiti i unutar bilo koje druge *World Wide Web* stranice.

Tablica 13.1. Razdioba udomljenika po područjima primjene u *Google Gadgets Directory* imeniku

Područje primjene	Ukupni broj udomljenika	Broj raznolikih udomljenika
Igre i zabava	25612	850
Pomoćni alati	9874	330
Informiranje	4827	160
Komunikacija i druženje	3900	130
Tehnologija	2752	90
Sport	1928	65
Financije	1505	50

Tablicom 13.1 prikazan je broj udomljenika dostupnih u *Google Gadgets Directory* imeniku razvrstan prema najzastupljenijim područjima primjene. Iz tablice je vidljiva pokrivenost nekoliko područja primjene razmjerno velikim skupom udomljenika, ali i nedostatak udomljenika u većini preostalih područja, kao što su trgovina, industrija, obrazovanje, zdravstvo, javna uprava i slično te stručnim područjima, kao što su prirodne, društvene, tehničke, medicinske i ostale vrste znanosti.

Unatoč razmjerno velikom broju udomljenika u malom broju područja primjene, zbog slobodnog i nenadziranog razvoja udomljenika od strane otvorene zajednice programera, primjetna je velika zastupljenost udomljenika s istim ili vrlo sličnim funkcionalnostima. Kako je za ocjenu stvaralačkih mogućnosti programske paradigme za usložnjavanje udomljenika važna količina raznolikih udomljenika i njihova raspodjela po različitim područjima primjene, obavljena je procjena sličnosti udomljenika u *Google Gadgets Directory* imeniku. S obzirom na to da je pretraživanje udomljenika u imeniku organizirano po ključnim riječima, procjena sličnosti udomljenika izvršena je na osnovi količine dobivenih rezultata za zadane ključne riječi u nazivu udomljenika. Analiza sličnosti pokazala je da za najpopularnije vrste primjenskih funkcionalnosti, kao što je osobni organizator, telefonski imenik, slanje elektroničke pošte, pretvorba valuta, pretraživanje *World Wide Web* sadržaja te pretraživanje vijesti, glazbe, filmova, zemljovida i vremenske prognoze u imeniku postoji i do nekoliko stotina istovjetnih ili sličnih udomljenika. Pretraživanjem

imenika za više desetaka ključnih riječi i pojmova iz različitih područja primjene, utvrđeno je da za pojedine primjenske funkcionalnosti *Google Gadgets Directory* imenik prosječno sadrži približno 30 istih ili sličnih udomljenika. Trećim stupcem tablice 13.1 prikazana je procjena broja raznolikih udomljenika po područjima primjene koja su najzastupljenija u *Google Gadgets Directory* imeniku.

Procjena daljnjeg povećanja broja udomljenika potaknuta primjenom programske paradigme za usložnjavanje udomljenika prikazana je u poglavlju 13.1.4. Proširivost skupa udomljenika novim udomljenicima i njihovo uključivanje u sustav programske paradigme omogućeni su univerzalnim jezikom za povezivanje udomljenika koji nije ovisan o području primjene pojedinih udomljenika. Kao što je opisano u poglavlju 7.2, programski jezik za povezivanje udomljenika u radni tijek zasnovan je na općenito primjenjivim radnjama potrošača putem grafičkog korisničkog sučelja udomljenika. Oblikovanje primjenski neovisnog programskog jezika zahtijeva se *svojstvom nezavisnosti povezujućih elemenata i značenja blokova* koje je opisano u poglavlju 6.2.6 u okviru modela programiranja prilagođenog potrošaču.

13.1.3 Brojnost korisnika programske paradigme

Treća veličina koja se koristi za procjenu stvaralačkih mogućnosti programske paradigme je veličina ukupne svjetske zajednice korisnika koji ovladavaju znanjima i vještinama za primjenu paradigme. Što je broj korisnika programske paradigme veći, to je veći ukupni broj primjenskih programa koje je zajednica korisnika sposobna izgraditi u promatranom razdoblju. Brojnost korisnika pojedinih programskih paradigmi određena je na osnovi dostupnih statističkih podataka o kretanju broja korisnika računala, internetskih korisnika i školovanih programera u pojedinim dijelovima svijeta. Rezultati su prikazani tablicom 13.2.

Prema statističkim podacima dostupnim u izvještajima o kretanju broja programera [254], u 2010. godini diljem svijeta na poslovima razvoja programske potpore očekuje se približno 20 milijuna školovanih programera koji ovladavaju znanjima i vještinama potrebnim za primjenu objektno-orijentiranih programskih jezika. Iako u grani profesionalnog razvoja programske potpore objektno-orijentirani programski jezici danas zauzimaju prevladavajuću ulogu, nešto manje od dva posto, odnosno oko 350 tisuća programera još uvijek gradi programsku potporu posebne namjene koja zahtijeva primjenu strojnih jezika [255, 256]. Istodobno se očekuje približno 500 milijuna krajnjih korisnika računalnih primjenskih programa koji nisu školovani u području programskog inženjerstva,

ali samostalno sudjeluju u izradi programske potpore u području vlastite stručnosti [71]. Tijekom razvoja primjenskih programa, krajnji se korisnici koriste posebno prilagođenim razvojnim alatima za pojedina područja primjene.

Tablica 13.2. Procjena broja korisnika koji ovladavaju vještinama za primjenu pojedinih oblika programskih paradigmi

Programiranje strojnim jezikom	Objektno-orijentirano programiranje	Primjenski usmjereno programiranje	Programiranje usložnjavanjem udomljenika
350 tisuća školovanih programera	20 milijuna školovanih programera	500 milijuna krajnjih korisnika	2 milijarde *web* korisnika

Nasuprot školovanim programerima i krajnjim korisnicima u pojedinim stručnim područjima, tijekom 2010. godine diljem svijeta se očekuje približno dvije milijarde internetskih korisnika [257] koji ovladavaju znanjima i vještinama potrebnim za primjenu udomljenika i programske paradigme za usložnjavanje udomljenika. Osim što već danas značajno premašuje broj školovanih programera i krajnjih korisnika, broj korisnika mreže Internet u snažnom je porastu, osobito u azijskim, afričkim i zemljama Bliskog istoka.

13.1.4 Stvaralačke mogućnosti programske paradigme

Na osnovi podataka o vremenu razvoja primjenskih programa, količini dostupnih gradivnih elemenata i brojnosti korisnika programske paradigme koji su prikazani u poglavljima 13.1.1, 13.1.2 i 13.1.3, definirana je mjera *stvaralačkih mogućnosti programske paradigme* (*SMPP*). Tom mjerom procjenjuje se ukupna količina novih primjenskih programa koju zajednica korisnika može razviti u određenom razdoblju primjenom određene programske paradigme.

$$SMPP = \frac{broj\ korisnika}{vrijeme\ razvoja\ \ funkcijske\ točke} \cdot faktor\ uporabljivosti \qquad (13.1)$$

Procjena stvaralačkih mogućnosti programske paradigme izvodi se primjenom formule 13.1, gdje je *SMPP* pokazatelj stvaralačkih mogućnosti programske paradigme. Stvaralačke mogućnosti programske paradigme izražavaju se *brojem Albrechtovih funkcijskih točaka koji zajednica korisnika može razviti u jedinici vremena*. Kao što je vidljivo iz formule 13.1, stvaralačke mogućnosti programske paradigme povećavaju se porastom broja korisnika koji ovladavaju znanjima i vještinama za primjenu paradigme i skraćivanjem vremena potrebnog

za razvoj Albrechtovih funkcijskih točaka. Faktorom uporabljivosti u obzir se uzima dostupnost i raspršenost gradivnih elemenata koje pojedina programska paradigma zahtijeva kako bi se postigla zadovoljavajuća razina učinkovitosti razvojnog postupka.

Vrijeme potrebno za razvoj jedne Albrechtove funkcijske točke proporcionalno je broju redaka programske logike koji su prosječno potrebni da bi se primjenom promatrane paradigme ostvarila funkcionalnost funkcijske točke. Proporcionalna ovisnost vremena potrebnog za razvoj funkcijske točke i broja redaka programske logike potrebnog za njezino ostvarenje prikazana je formulom 13.2

$$vrijeme\ razvoja\ \ funkcijske\ točke = T_{LOC} \cdot broj\ redaka\ \ po\ \ funkcijskoj\ točki \quad (13.2)$$

gdje je T_{LOC} vrijeme koje je prosječno potrebno utrošiti za oblikovanje i ostvarenje jednog retka programske logike.

Faktor uporabljivosti je bezdimenzionalna vrijednost iz intervala [0, 1]. Ako je vrijednost faktora uporabljivosti jednaka nuli, to znači da za promatranu programsku paradigmu ne postoji niti jedan gradivni element. S obzirom da primjenom takve paradigme nije moguće graditi primjenske programe, stvaralačke mogućnosti takve paradigme jednake su nuli. Ako vrijednost faktora uporabljivosti iznosi jedan, onda promatrana programska paradigma sadrži dovoljan broj gradivnih elemenata jednoliko raspršenih po svim područjima primjene koji je nužan za postizanje praga učinkovitosti.

Za procjenu faktora uporabljivosti programske paradigme koristi se Metcalfeov zakon [258]. Taj je zakon izvorno osmišljen za procjenu uporabljivosti komunikacijskog sustava. Prema tom zakonu, uporabljivost komunikacijskog sustava, primjerice telefonske mreže, raste s kvadratom broja sudionika u sustavu. Metcalfeovim zakonom matematički se opisuje stanje niske početne uporabljivosti komunikacijskog sustava za vrijeme dok mu se ne priključi kritični broj sudionika. Kasniji priljev novih sudionika uhodanom sustavu daje sve veći doprinos jer se broj mogućih međusobnih veza povećava po približno kvadratnom zakonu. Osim za procjenu uporabljivosti komunikacijskog sustava, Metcalfeov zakon često se primjenjuje za procjenu uporabljivosti različitih oblika informacijskih sustava, kao što su *World Wide Web* sustav i društvene mreže. Primjena Metcalfeovog zakona na komponentno-usmjerene programske paradigme ima sljedeće značenje: linearnim porastom broja raspoloživih gradivnih komponenti, uporabljivost programske paradigme kvadratno raste. Prema tome, povećanje broja gradivnih elemenata u početku ima mali, a kasnije sve značajniji doprinos uporabljivosti paradigme.

$$faktor\ uporabljivosti = \left(\frac{1}{n} \sum_{i=1}^{n} \left(\frac{broj\ dostupnih\ elemenata_i}{broj\ elemenata\ na\ pragu\ učinkovitosti_i} \right)^2 \right)^2 \qquad (13.3)$$

Faktor uporabljivosti programske paradigme procjenjuje se primjenom formule 13.3. Oznakom *n* označen je broj ciljnih područja primjene programske paradigme. U proračunu faktora uporabljivosti, Metcalfeov zakon primijenjen je na dvije razine. Na prvoj razini, Metcalfeov zakon primjenjuje se za procjenu uporabljivosti paradigme unutar pojedinih područja primjene. Za pojedino područje primjene, koje je označeno oznakom *i*, faktor uporabljivosti definiran je kao kvadrat omjera broja dostupnih gradivnih elemenata (*broj dostupnih elemenata*$_i$) i broja gradivnih elemenata koji su potrebni za postizanje praga učinkovitosti u tom području primjene (*broj elemenata na pragu učinkovitosti*$_i$). Na drugoj razini, Metcalfeov zakon primjenjuje se za procjenu uporabljivosti paradigme u cjelini. Tu se izračunava daljnji kvadratni porast faktora uporabljivosti koji je uzrokovan porastom mogućnosti razvoja višedisciplinarnih primjenskih programa koji zahtijevaju istodobnu dostupnost gradivnih elemenata u različitim područjima primjene. Faktor uporabljivosti paradigme u cjelini definiran je kao kvadrat srednje vrijednosti faktora uporabljivosti po pojedinim područjima primjene.

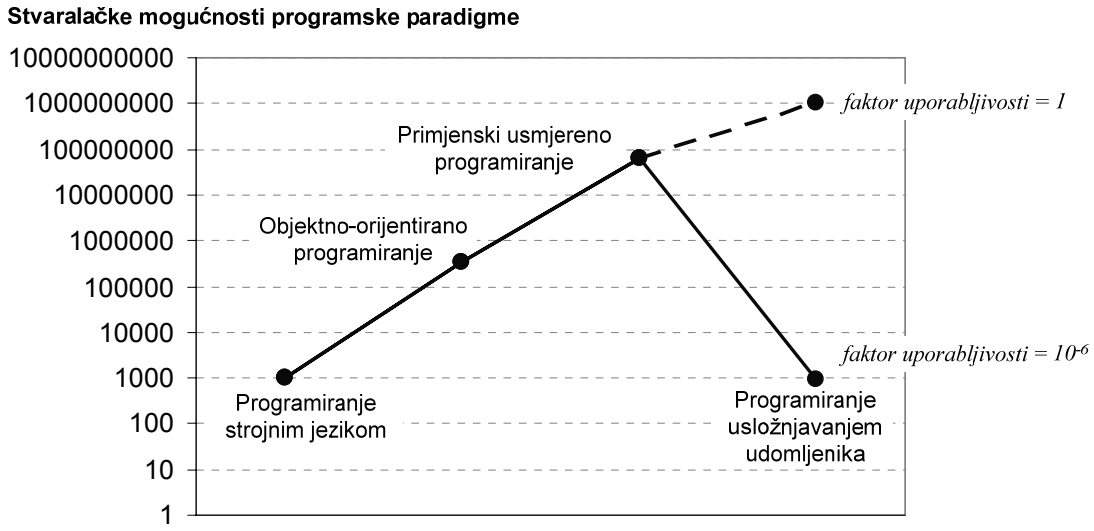
$$\begin{aligned} &n = 1000 \\ &broj\ elemenata\ na\ pragu\ učinkovitosti_i = 1000, \forall i \in [1, 1000] \\ &broj\ dostupnih\ elemenata_i = \begin{cases} vrijednost\ iz\ tablice\ 13.1 & za\ 1 \le i \le 7 \\ 0 & inače \end{cases} \end{aligned} \qquad (13.4)$$

Formulom 13.4 prikazan je način određivanja parametara formule 13.3 za proračun faktora uporabljivosti programske paradigme za usložnjavanje udomljenika. Na osnovi podataka o približno tisuću primjenski usmjerenih programskih jezika kojima je danas pokriven vrlo širok skup područja primjene, procijenjeno je da je visoku razinu programske paradigme za usložnjavanje udomljenika moguće postići s približno tisuću skupova udomljenika (n = 1000), od kojih je po jedan namijenjen za pojedino područje primjene. Na osnovi podataka o prosječno stotinjak gradivnih elemenata u pojedinim primjenski usmjerenim jezicima te odnosu zrnatosti primjenski usmjerenih elemenata i udomljenika, gdje se za zadržavanje iste razine izražajnosti paradigme za jedan primjenski usmjereni element zahtijeva približno deset udomljenika, procijenjeno je da bi svako područje primjene trebalo biti zastupljeno s približno tisuću udomljenika (*broj elemenata na pragu učinkovitosti*$_i$ = 1000, $\forall i \in [1, 1000]$). Kao što je prikazano tablicom 13.1, u današnje je

vrijeme svega nekoliko područja primjene pokriveno brojem udomljenika koji je usporediv s pragom učinkovitosti od tisuću udomljenika, dok za ostala područja primjene udomljenici još uvijek nisu razvijeni ili je njihov broj zanemariv. Uvrštavanjem vrijednosti iz trećeg stupca tablice 13.1 u formulu 13.3, faktor uporabljivosti programske paradigme za usložnjavanje udomljenika danas iznosi približno 10^{-6}, odnosno tek milijunti dio maksimalne vrijednosti.

Grafičkim prikazom na slici 13.4 prikazane su usporedne vrijednosti stvaralačkih mogućnosti programske paradigme za programiranje strojnim jezikom, objektno-orijentirane i primjenski-usmjerene programske paradigme te programsku paradigmu za usložnjavanje udomljenika. Zbog velikog raspona vrijednosti stvaralačkih mogućnosti za pojedine oblike programskih paradigmi, vrijednosti na okomitoj osi prikazane su u logaritamskom mjerilu.

Stvaralačke mogućnosti programiranja strojnim jezikom te objektno-orijentirane i primjenski-usmjerene programske paradigme izračunate su uz pretpostavku maksimalne vrijednosti faktora uporabljivosti (*faktor uporabljivosti = 1*). Iz prikazanih rezultata je vidljivo da, počevši od programiranja strojnim jezikom, preko objektno-orijentiranog programiranja, do primjenski usmjerenog programiranja, svaka nova programska paradigma povećava stvaralačke mogućnosti za dva do tri reda veličine.



Slika 13.4. Procjena stvaralačkih mogućnosti pojedinih oblika programskih paradigmi

Stvaralačke mogućnosti programske paradigme za usložnjavanje udomljenika prikazane su dvjema točkama koje predstavljaju dvije različite vrijednosti faktora uporabljivosti. U današnje vrijeme, kada je faktor uporabljivosti programske paradigme za usložnjavanje udomljenika zbog nedovoljne količine udomljenika približno jednak nuli

(*faktor uporabljivosti* = 10^{-6}), stvaralačke mogućnosti približno su jednake stvaralačkim mogućnostima strojnog programiranja. Međutim, kada se faktor uporabljivosti programske paradigme za usložnjavanje udomljenika približi maksimalnoj vrijednosti (*faktor uporabljivosti = 1*), očekuje se da njene stvaralačke mogućnosti zbog kratkog vremena potrebnog za razvoj programa, velikog broja korisnika i velike količine dostupnih udomljenika premaše stvaralačke mogućnosti primjenski-usmjerenih programskih paradigmi za dodatni red veličine.

Procjena budućeg rasta broja udomljenika

Kako bi se procijenilo vrijeme potrebno za porast faktora uporabljivosti programske paradigme za usložnjavanje udomljenika s današnje vrijednosti koja je približno jednaka nuli na maksimalnu vrijednost koja iznosi jedan, izvršena je procjena daljnjeg povećanja broja udomljenika. Tijekom procjene razmatrana su dva slučaja. U prvom slučaju je, na osnovi dostupnih podataka o kretanju broja udomljenika u prošlosti i očekivanom priljevu novih programera koji će sudjelovati u razvoju udomljenika u budućnosti, izvršena procjena porasta količine udomljenika uz sudjelovanje školovanih programera. U drugom slučaju, u model za procjenu porasta broja udomljenika uključene su nove stvaralačke mogućnosti koje nastaju primjenom programske paradigme za usložnjavanje udomljenika. U tom slučaju, školovanim programerima u razvoju udomljenika pridružuju se potrošači.

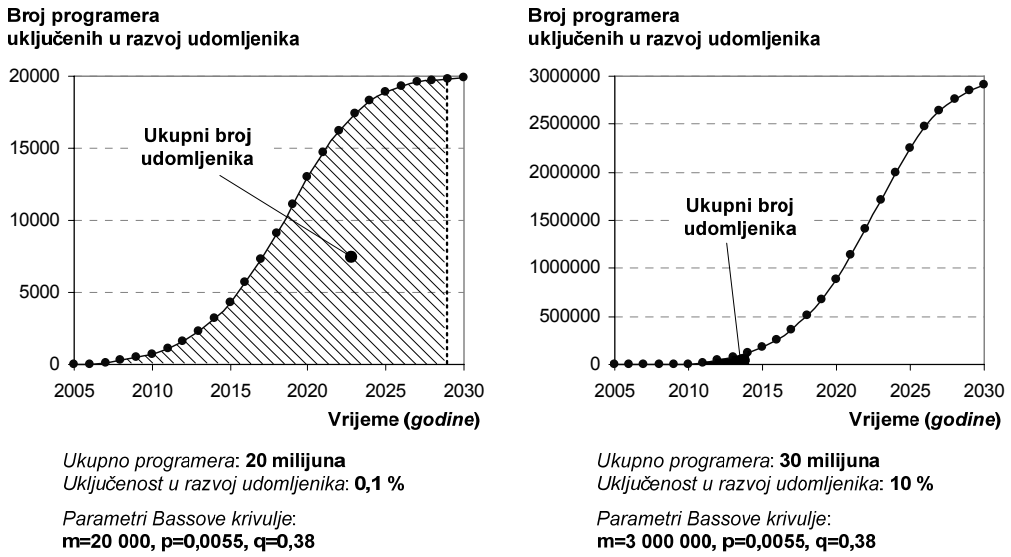
Procjena rasta broja udomljenika uz sudjelovanje školovanih programera

Procjena kretanja broja udomljenika koje razvijaju školovani programeri izvršena je primjenom Bassove formule [259, 260] koju je 1969. godine osmislio američki znanstvenik Frank Bass. Bassova formula primjenjuje se za predviđanje prihvaćanja tehnologijskih inovacija među korisnicima. Ovisnost broja prodanih primjeraka novog tehnologijskog proizvoda, odnosno broja korisnika nove tehnologijske usluge i vremena tijekom kojeg su proizvod ili usluga prisutni na tržištu opisana je formulom 13.5

$$N_t = N_{t-1} + p\left(m - N_{t-1}\right) + q\frac{N_{t-1}}{m}\left(m - N_{t-1}\right) \tag{13.5}$$

gdje je N_t broj prodanih primjeraka proizvoda, odnosno broj korisnika usluge u trenutku t, a N_{t-1} broj prodanih primjeraka proizvoda, odnosno broj korisnika usluge u trenutku $t-1$. Oznakom m označen je potencijal tržišta. Izražava se veličinom ciljne skupine korisnika za koju se očekuje da će u doglednom vremenu započeti s primjenom tehnologijske usluge ili

brojem primjeraka proizvoda koji je moguće rasprodati na određenom tržištu. Oznakama *p* i *q* označeni su faktori vanjskog i unutarnjeg utjecaja na širenje tehnologijske inovacije. Faktor *p* naziva se faktorom inovacije (engl. *coefficient of innovation*). Označava vjerojatnost širenja tehnologijske inovacije potaknute vanjskim utjecajima. Primjeri vanjskih utjecaja na širenje tehnologijske inovacije su oglašavanje i promidžba proizvoda ili usluge putem masovnih medija, vladine mjere koje potiču primjenu proizvoda ili usluge, tehnologijski napredak koji otvara nove mogućnosti za plasman i primjenu proizvoda ili usluge i slično. Faktor *q* naziva se faktorom oponašanja (engl. *coefficient of imitation*). Označava vjerojatnost širenja tehnologijske inovacije potaknute unutarnjim utjecajima. Primjer unutarnjeg utjecaja na širenje tehnologijske inovacije je prihvaćanje proizvoda ili usluge od strane novih korisnika pod utjecajem korisnika koji već koriste proizvod ili uslugu.



Slika 13.5. Bassova krivulja uključenosti školovanih programera u razvoj udomljenika

Bassova krivulja prihvaćenosti tehnologijske inovacije ima oblik S-krivulje, kao što je prikazano slikom 13.5. Životni vijek tehnologijske inovacije započinje sporim priljevom korisnika jer su korisnici u početku skeptični prema novom proizvodu ili usluzi. Nakon privikavanja i stvaranja kritičnog broja korisnika, dolazi razdoblje masovnog priljeva novih korisnika. Iscrpljivanjem potencijala tržišta dolazi do zasićenja i usporavanja priljeva novih korisnika. Za većinu tehnologijskih inovacija, tipično vrijeme od trenutka plasmana proizvoda ili usluge na tržište do zasićenja tržišta iznosi između 20 i 30 godina. Empirijskim ispitivanjima utvrđene su tipične vrijednosti faktora *p* i *q* u formuli 13.5 koje vjerno opisuju postupak prihvaćanja tehnologijskih inovacija. Prosječna vrijednost faktora inovacije *p* iznosi 0,03, ali je nerijetko i manja od 0,01. Tipični raspon vrijednosti faktora oponašanja *q* nalazi se između 0,3 i 0,5 s prosječnom vrijednošću 0,38.

Slikom 13.5 prikazane su dvije Bassove krivulje za procjenu porasta broja školovanih programera koji sudjeluju u razvoju udomljenika. Pomoću dviju različitih krivulja opisana su dva različita scenarija uključivanja programera u razvoj udomljenika. Vodoravnim osima grafova prikazano je vrijeme u godinama, počevši od 2005. godine kada su se udomljenici pojavili na tržištu. Okomite osi grafova prikazuju broj programera uključenih u razvoj udomljenika do promatranog trenutka.

Tijekom procjene broja programera koji su mogući budući razvijatelji udomljenika, pretpostavljeno je da će zbog razvoja programske potpore u drugim područjima primjene, samo dio ukupne zajednice programera biti uključen u razvoj udomljenika. Osim toga, ukupni broj programera u svijetu u stalnom je porastu pa su kod procjene broja mogućih razvijatelja udomljenika u obzir uzeti i statistički podaci o povećanju broja programera. Prema dostupnim izvještajima [254], 2007. godine je širom svijeta bilo približno 15 milijuna programera, dok je do 2010. godine taj broj porastao na približno 20 milijuna. Prema tim statističkim podacima, ukupni broj programera godišnje se povećava za približno 1,5 milijuna.

Lijevom krivuljom na slici 13.5 prikazan je slučaj u kojem je pretpostavljeno da će se samo 0,1 posto od ukupno 20 milijuna programera, koliko ih danas ima u svijetu, postupno uključiti u razvoj udomljenika. To je slučaj koji vjerno opisuje prošlost i današnje stanje u području razvoja udomljenika. Desnom krivuljom prikazan je optimistični slučaj u kojem je pretpostavljeno da će u sljedećih desetak godina ukupni broj programera u svijetu dodatno porasti na 30 milijuna, od čega će se 10 posto postupno uključiti u razvoj udomljenika. Postupno uključivanje programera u razvoj udomljenika odvija se prema Bassovoj formuli. Vrijednost parametra m koji određuje potencijal tržišta predstavlja udio ukupnog broja programera koji su mogući budući razvijatelji udomljenika, dok vrijednosti parametara p i q za obje krivulje iznose p=0,0055 i q=0,38, što su tipične vrijednosti faktora inovacije i faktora oponašanja za programske proizvode.

$$U_t = \int_{t_0}^{t} k \cdot N_t dt \tag{13.6}$$

Tijekom razvoja udomljenika, programeri primjenjuju objektno-orijentirane ili s njima po učinkovitosti izjednačene programske paradigme. Pretpostavljena brzina razvoja udomljenika primjenom te programske paradigme kreće se u rasponu od 100 do 150 udomljenika godišnje po jednom programeru, što znači da programer prosječno utroši dva do tri radna dana za razvoj jednog udomljenika. Ukupni broj udomljenika razvijenih do trenutka

t izračunava se primjenom formule 13.6 kao integral ukupne brzine razvoja udomljenika od strane cjelokupne zajednice programera uključenih u razvoj udomljenika. Oznakom U_t označen je ukupni broj udomljenika razvijen do trenutka t, k je pretpostavljena brzina razvoja udomljenika od strane pojedinačnih programera, N_t je funkcija uključivanja programera u razvoj udomljenika koja je prikazana Bassovom krivuljom na slici 13.5, dok je t_0 trenutak plasmana udomljenika na tržište. Podintegralna funkcija koja je umnožak brzine razvoja udomljenika od strane pojedinačnih programera i broja programera uključenih u razvoj udomljenika predstavlja ukupnu brzinu razvoja udomljenika od strane cjelokupne zajednice programera koji su do promatranog trenutka uključeni u razvoj udomljenika.

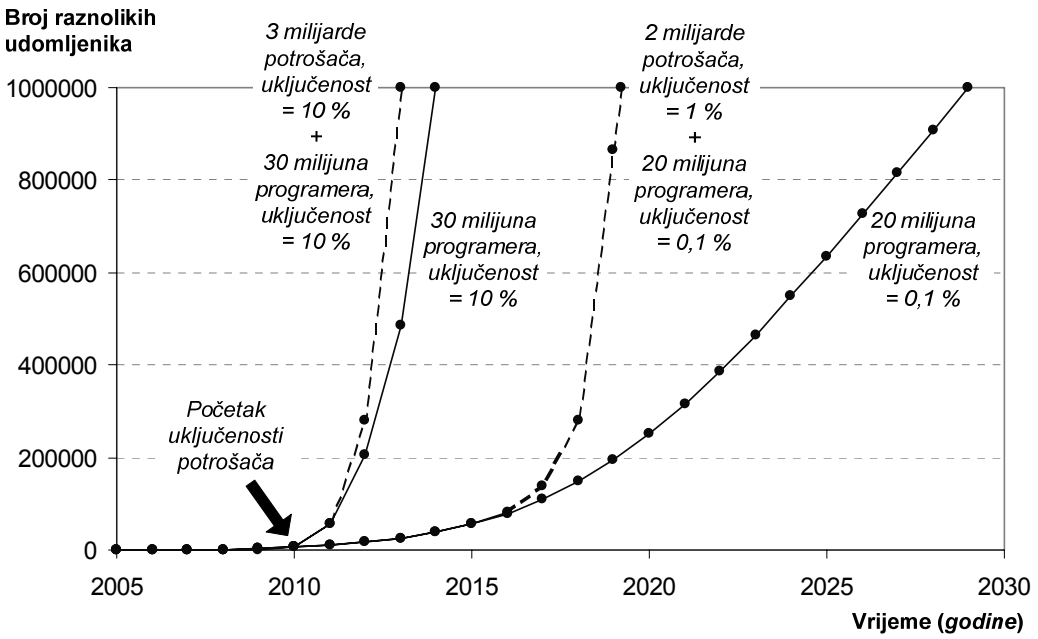
Ukupni broj udomljenika razvijenih do trenutka t proporcionalan je površini ispod Bassove krivulje na slici 13.5. Uz pretpostavljenu brzinu razvoja od približno 150 udomljenika godišnje po jednom programeru, određene su točke u kojima ukupna količina udomljenika dostiže prag učinkovitosti programske paradigme od milijun raznolikih udomljenika. Trenutak dostizanja tog praga za lijevu krivulju je 2029. godina, dok je desnom krivuljom to moguće postići već 2014. godine. U tim dvjema točkama, površine ispod Bassovih krivulja odgovaraju količini od približno 30 milijuna udomljenika, što uz stopu sličnosti od 30 istih ili sličnih udomljenika iznosi približno milijun raznolikih udomljenika.

Procjena rasta broja udomljenika uz sudjelovanje školovanih programera i potrošača

Slikom 13.6 prikazana je procjena porasta broja udomljenika pod pretpostavkom da se školovanim programerima u razvoju udomljenika postupno priključuju i potrošači koji nove udomljenike razvijaju primjenom programske paradigme za usložnjavanje udomljenika. Vodoravnom osi grafa prikazano je vrijeme u godinama, počevši od 2005. godine kada su se udomljenici pojavili na tržištu. Okomita os grafa prikazuje ukupnu količinu raznolikih udomljenika. Na slici su prikazane četiri krivulje. Punim krivuljama prikazana su dva scenarija sa slike 13.5 gdje u razvoju udomljenika sudjeluju isključivo školovani programeri. Isprekidanim krivuljama prikazan je dodatni porast broja udomljenika uz istodobno sudjelovanje školovanih programera i potrošača

Pune krivulje dobivene su primjenom formule 13.6 na Bassove krivulje prikazane slikom 13.5. Položenija puna krivulja prikazuje porast ukupnog broja raznolikih udomljenika za slučaj u kojem se u razvoj udomljenika postupno uključuje 0,1 posto od ukupno 20 milijuna programera. Strmija puna krivulja prikazuje porast ukupnog broja raznolikih udomljenika za slučaj u kojem se u razvoj udomljenika postupno uključuje 10 posto od ukupno 30 milijuna programera.

Za svaki od dva prikazana slučaja razvoja udomljenika uz sudjelovanje isključivo školovanih programera, isprekidanim krivuljama prikazano je dodatno ubrzanje razvoja udomljenika uzrokovano postupnim uključivanjem potrošača. Uključivanje potrošača u razvoj udomljenika započinje 2010. godine pojavom potrošaču prilagođene programske paradigme za usložnjavanje udomljenika.



Slika 13.6. Procjena porasta količine udomljenika zajedničkim djelovanjem programera i potrošača

Slično kao i kod školovanih programera, tijekom procjene broja potrošača uključenih u razvoj udomljenika razmatrane su različite razine uključenosti. Analize ponašanja korisnika internetskih zajednica pokazale su da se uključenost potrošača u aktivno stvaranje sadržaja odvija po pravilu 90-9-1 [262]. U bilo kojoj zajednici korisnika, samo jedan posto članova aktivno i neprekidno doprinosi stvaranju novog sadržaja, devet posto članova uključuje se povremeno, dok se djelovanje preostalih devedeset posto članova uglavnom svodi na korištenje sadržaja koji stvara manjina. Slikom 13.6 prikazani su slučajevi očekivane jedanpostotne i optimistične desetpostotne uključenosti potrošača u razvoj udomljenika.

Položenija isprekidana krivulja prikazuje porast ukupnog broja raznolikih udomljenika koji nastaju zajedničkim djelovanjem programera i potrošača, uz pretpostavku niže razine uključenosti za obje skupine razvijatelja. To je slučaj u kojem se počevši od 2005. godine u razvoj udomljenika postupno uključuje 0,1 posto od ukupno 20 milijuna programera, dok im se počevši od 2010. godine postupno počinje pridruživati 1 posto od ukupno 2 milijarde potrošača. Ovom dinamikom razvoja udomljenika, prag uporabljivosti programske paradigme koji iznosi milijun raznolikih udomljenika moguće je postići do 2019. godine. U

odnosu na slučaj niske razine uključenosti školovanih programera, uključivanje potrošača skraćuje vrijeme do pune uporabljivosti programske paradigme za deset godina.

Strmijom isprekidanom krivuljom prikazan je optimistični slučaj uz pretpostavku visoke razine uključenosti obiju skupina razvijatelja u razvoj udomljenika. Ovdje je pretpostavljeno da će se, počevši od 2005. godine, u razvoj udomljenika postupno uključivati 10 posto od budućih 30 milijuna programera, dok će se ukupni broj potrošača povećati s dvije na tri milijarde, od čega će se 10 posto postupno uključiti u razvoj udomljenika. Ovom dinamikom razvoja udomljenika, prag uporabljivosti programske paradigme koji iznosi milijun raznolikih udomljenika moguće je postići već 2013. godine. U odnosu na slučaj visoke razine uključenosti školovanih programera, visokom razinom uključenosti potrošača skraćuje se vrijeme do pune uporabljivosti programske paradigme za dodatnih godinu dana.

Proračun količine udomljenika koje zajednički razvijaju školovani programeri i potrošači izvršen je primjenom formule 13.7

$$U_t = \int_{t_0}^{t} k \cdot N_{PROG,t} \cdot \left(1 + \frac{SMPP_{POTR,t}}{SMPP_{PROG,t}}\right) dt \tag{13.7}$$

gdje je U_t ukupni broj udomljenika razvijenih do trenutka t. Ukupni broj udomljenika izračunava se kao integral ukupne brzine razvoja udomljenika kojoj zajednički pridonose programeri i potrošači. Brzina razvoja udomljenika od strane programera $k \cdot N_{PROG,t}$ vremenski je promjenjiva veličina koja ovisi o brzini razvoja udomljenika od strane pojedinačnih programera k i broja programera $N_{PROG,t}$ koji su do trenutka t uključeni u razvoj udomljenika. Toj brzini pribraja se brzina razvoja udomljenika od strane potrošača. Brzina razvoja udomljenika od strane potrošača također je vremenski promjenjiva veličina koja se izračunava na osnovi omjera stvaralačkih mogućnosti programera $SMPP_{PROG,t}$ i potrošača $SMPP_{POTR,t}$. Kako broj programera i potrošača uključenih u razvoj udomljenika postupno raste u skladu s Bassovom formulom prihvaćenosti tehnologijskih inovacija, stvaralačke mogućnosti programera i potrošača vremenski su rastuće veličine. Dok su promjene stvaralačkih mogućnosti programera uzrokovane postupnim povećanjem broja programera uključenih u razvoj udomljenika, stvaralačke mogućnosti potrošača mijenjaju se pod utjecajem postupnog priljeva novih potrošača i postupnog povećanja raspoloživog broja udomljenika. Povećanjem broja udomljenika raspoloživih za izgradnju novih udomljenika povećava se faktor uporabljivosti programske paradigme za usložnjavanje udomljenika, čime se postupno povećava i doprinos potrošača razvoju udomljenika.

13.2 Ocjena potpunosti programske paradigme

Formalni oblik utvrđivanja izražajnosti programske paradigme je definicija izražajnosti primjenom teorije izračunljivosti, odnosno Chomskyjeve hijerarhije formalnih jezika, automata i gramatika [264]. Izražajnost programske paradigme definira se u odnosu na Turingovu potpunost (engl. *Turing completeness*) [264, 265]. Programska paradigma najvećeg mogućeg stupnja izražajnosti izjednačava se s definicijom Turingovog stroja. U ovom je poglavlju pokazana istovjetnost programske paradigme za usložnjavanje udomljenika i Turingovog stroja, odnosno pokazano je da je primjenom te programske paradigme moguće ostvariti bilo koju funkciju koja je izračunljiva Turingovim strojem.

Tablica 13.3. Ostvarenje Turingove potpunosti programskom paradigmom za usložnjavanje udomljenika

Element Turingove potpunosti	**Element programske paradigme za usložnjavanje udomljenika**
Slijedno izvođenje programa	Razmještanje skupa događaja za opis radnji nad elementima grafičkog korisničkog sučelja udomljenika u susjedne ćelije tablice
Pridruživanje	Događaj za uspostavu toka podataka nad elementima grafičkog korisničkog sučelja udomljenika (*copy/paste*)
Grananje tijeka izvođenja programa	Udomljenik za usporedbu logičkih izraza
Ponavljanje tijeka izvođenja programa	Događaji za odziv složenog udomljenika na radnje nad elementima grafičkog korisničkog sučelja (*when clicked*, *when doubleclicked*, *when checked*, *when unchecked*, *when selected*) + Događaji za izvođenje radnji nad elementima grafičkog korisničkog sučelja udomljenika (*click*, *doubleclick*, *check*, *uncheck*, *select*)
Rad s cijelim brojevima	Udomljenik za izvođenje aritmetičkih operacija

Programski jezik zadovoljava svojstvo Turingove potpunosti ako ima sljedeća svojstva: (1) podržava rad s cijelim brojevima i cjelobrojne aritmetičke operacije, (2) omogućava slijedno izvođenje programa, (3) sadrži naredbu pridruživanja te podržava (4) grananje i (5) ponavljanje tijeka izvođenja programa [265]. U različitim programskim paradigmama i jezicima, pojedini elementi Turingove potpunosti postižu se na različite načine. Primjerice, uobičajeni mehanizam za ostvarenje ponavljanja tijeka izvođenja programa u imperativnim programskim jezicima je primjena programske petlje, dok se u funkcijskim programskim jezicima ponavljanje ostvaruje rekurzijom [265].

Turingova potpunost programske paradigme za usložnjavanje udomljenika postiže se kombinacijom složenog udomljenika čije je ponašanje definirano skupom događaja za opis radnji nad elementima grafičkog korisničkog sučelja i konačnog skupa pomoćnih udomljenika. Tablicom 13.3 prikazani su elementi programske paradigme za usložnjavanje udomljenika kojima se postižu pojedini elementi Turingove potpunosti.

Slijedno izvođenje programa

Slijedno izvođenje programa omogućeno je razmještanjem skupa događaja za opis radnji nad elementima grafičkog korisničkog sučelja udomljenika u susjedne ćelije tablice kojom se definiraju vremenski odnosi nad skupom događaja. Primjena tablice za uređivanje vremenskih odnosa nad skupom događaja opisana je u poglavlju 11.3, dok je programsko ostvarenje slijednog uzorka izvođenja programa opisano u poglavlju 11.4.1.

Pridruživanje

U programskoj paradigmi za usložnjavanje udomljenika, pridruživanje sadržaja jednog elementa grafičkog korisničkog sučelja drugom elementu postiže se događajem za uspostavu toka podataka (*copy/paste*). Događaj za uspostavu toka podataka omogućuje proizvoljno premještanje sadržaja između pojedinih elemenata grafičkog korisničkog sučelja. Primjena događaja za uspostavu toka podataka opisana je u poglavlju 10.

Grananje tijeka izvođenja programa

U programskoj paradigmi za usložnjavanje udomljenika, grananje tijeka izvođenja programa postiže se pomoćnim udomljenikom za usporedbu logičkih izraza. S obzirom na to da je za utvrđivanje Turingove potpunosti programske paradigme dovoljno pokazati rad s cijelim brojevima, udomljenikom za usporedbu logičkih izraza dovoljno je osigurati mogućnost usporedbe cijelih brojeva. Udomljenik za usporedbu logičkih izraza sastavljenih od cijelih brojeva i njegova primjena za programsko ostvarenje grananja tijeka izvođenja programa prikazani su u poglavlju 11.5.2.

Ponavljanje tijeka izvođenja programa

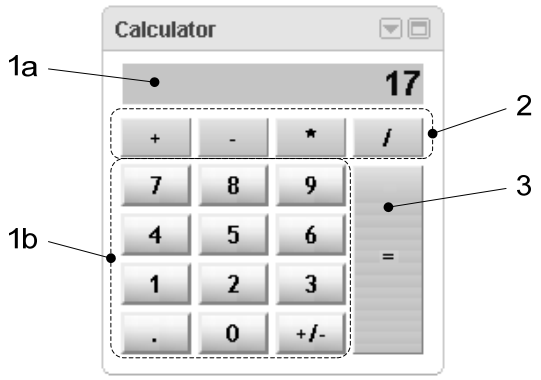
U programskoj paradigmi za usložnjavanje udomljenika, ponavljanje tijeka izvođenja programa postiže se mogućnošću odziva složenog udomljenika na radnje nad elementima vlastitog korisničkog sučelja. Taj odziv definira se događajima označenim ključnim riječima *when clicked*, *when doubleclicked*, *when checked*, *when unchecked* i *when selected*, kao što

je opisano u poglavlju 11.5. Pokretanje dijela programa pridruženog odzivu složenog udomljenika na radnje nad elementima njegova korisničkog sučelja postiže se izvođenjem događaja označenih ključnim riječima *click*, *doubleclick*, *check*, *uncheck* i *select*, kao što je opisano u poglavlju 9. Odziv složenog udomljenika na radnje nad elementima vlastitog korisničkog sučelja moguće je pokrenuti iz istog složenog udomljenika, iz drugog složenog udomljenika ili iz pomoćnog udomljenika za usporedbu logičkih izraza.

Događaji za definiranje odziva i pokretanje radnji nad elementima korisničkog sučelja složenog udomljenika omogućuju programsko ostvarenje bezuvjetnog ponavljanja tijeka izvođenja. Za programsko ostvarenje uvjetnog ponavljanja tijeka izvođenja, uz ove dvije vrste događaja, koristi se i pomoćni udomljenik za usporedbu logičkih izraza. Programsko ostvarenje bezuvjetnog i uvjetnog ponavljanja tijeka izvođenja programa prikazani su u poglavlju 11.5.3.

Rad s cijelim brojevima

Slično kao što se za ostvarenje grananja tijeka izvođenja programa koristi sprega složenog udomljenika i pomoćnog udomljenika za usporedbu logičkih izraza, za rad s cijelim brojevima koristi se sprega složenog udomljenika i pomoćnog udomljenika za izvođenje aritmetičkih operacija. Udomljenik za izvođenje aritmetičkih operacija je programsko ostvarenje ručnog računala (engl. *calculator*) u obliku udomljenika. Prikazan je slikom 13.7.



Slika 13.7. Udomljenik za izvođenje aritmetičkih operacija

Uloga udomljenika za izvođenje aritmetičkih operacija u programskoj paradigmi za usložnjavanje udomljenika jednaka je ulozi ugrađenih aritmetičkih operatora u programskim jezicima opće namjene. Iako je za svojstvo Turingove potpunosti programske paradigme dovoljan udomljenik za cjelobrojno računanje, zbog jednostavnijeg izvođenja aritmetičkih proračuna koristi se udomljenik koji podržava rad s brojevima s pomičnom točkom.

Upravljanje radom udomljenika za izvođenje aritmetičkih operacija obavlja složeni udomljenik. Unošenje operanada obavlja se izravnim upisom brojčanih vrijednosti u polje koje predstavlja zaslon računala primjenom događaja za upis sadržaja u polje za unos teksta (*type in*) (1a), prenošenjem vrijednosti s drugih udomljenika primjenom događaja za uspostavu toka podataka (*copy/paste*) (1a) ili primjenom brojčane tipkovnice kojom se upravlja izvođenjem događaja jednostrukog pritiska na brojčane tipke (*click*) (1b). Izbor aritmetičkih operacija obavlja se izvođenjem događaja jednostrukog pritiska na tipke za odabir vrste operacija (*click*) (2). Pokretanje aritmetičkih proračuna obavlja se izvođenjem događaja jednostrukog pritiska na tipku za pokretanje aritmetičkih operacija (*click*) (3). Preuzimanje rezultata aritmetičkih operacija obavlja se izvođenjem događaja za uspostavu toka podataka (*copy/paste*), čime se rezultati s udomljenika za izvođenje aritmetičkih operacija prenose u složeni udomljenik.

Primjena elemenata Turingove potpunosti tijekom izgradnje programa

Primjena programske paradigme za usložnjavanje udomljenika tijekom programskog ostvarenja algoritma koji zahtijeva elemente Turingove potpunosti pokazana je na primjeru složenog kamatnog računa. Složeni kamatni račun koristi se za obračun prinosa ostvarenih na uložena novčana sredstva. Podjednako je koristan matematički alat za ulagače, bankovne službenike i financijske savjetnike. Primjenom složenog kamatnog računa, prinos, odnosno kamata na uložena novčana sredstva, odnosno glavnicu, obračunava se prema formuli 13.8.

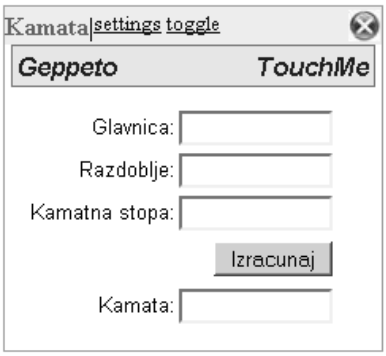
$$kamata = glavnica\left(r^{razdoblje} - 1\right)$$

$$r = 1 + \frac{kamatna\ stopa}{100} \tag{13.8}$$

Za programsko ostvarenje složenog kamatnog računa zahtijeva se programski jezik sa svojstvom Turingove potpunosti. Iako su s mreže Internet danas dobavljivi udomljenici za izračun kamata primjenom složenog kamatnog računa koje je moguće koristiti kao osnovne blokove za izgradnju primjenskih programa, ovdje je pokazano kako je algoritam za izračun kamata moguće ostvariti primjenom osnovnih elemenata programske paradigme za usložnjavanje udomljenika.

Slikom 13.8 prikazan je udomljenik za izračun kamata primjenom složenog kamatnog računa koji je potrebno izgraditi kao pokazni primjer svojstva Turingove potpunosti programske paradigme za usložnjavanje udomljenika. Korisničko sučelje udomljenika sastoji

se od ulaznih polja *Glavnica* za unos glavnice, *Razdoblje* za unos razdoblja ulaganja i *Kamatna stopa* za unos kamatne stope, tipke *Izracunaj* za pokretanje izračuna te izlaznog polja *Kamata* za prikaz rezultata.



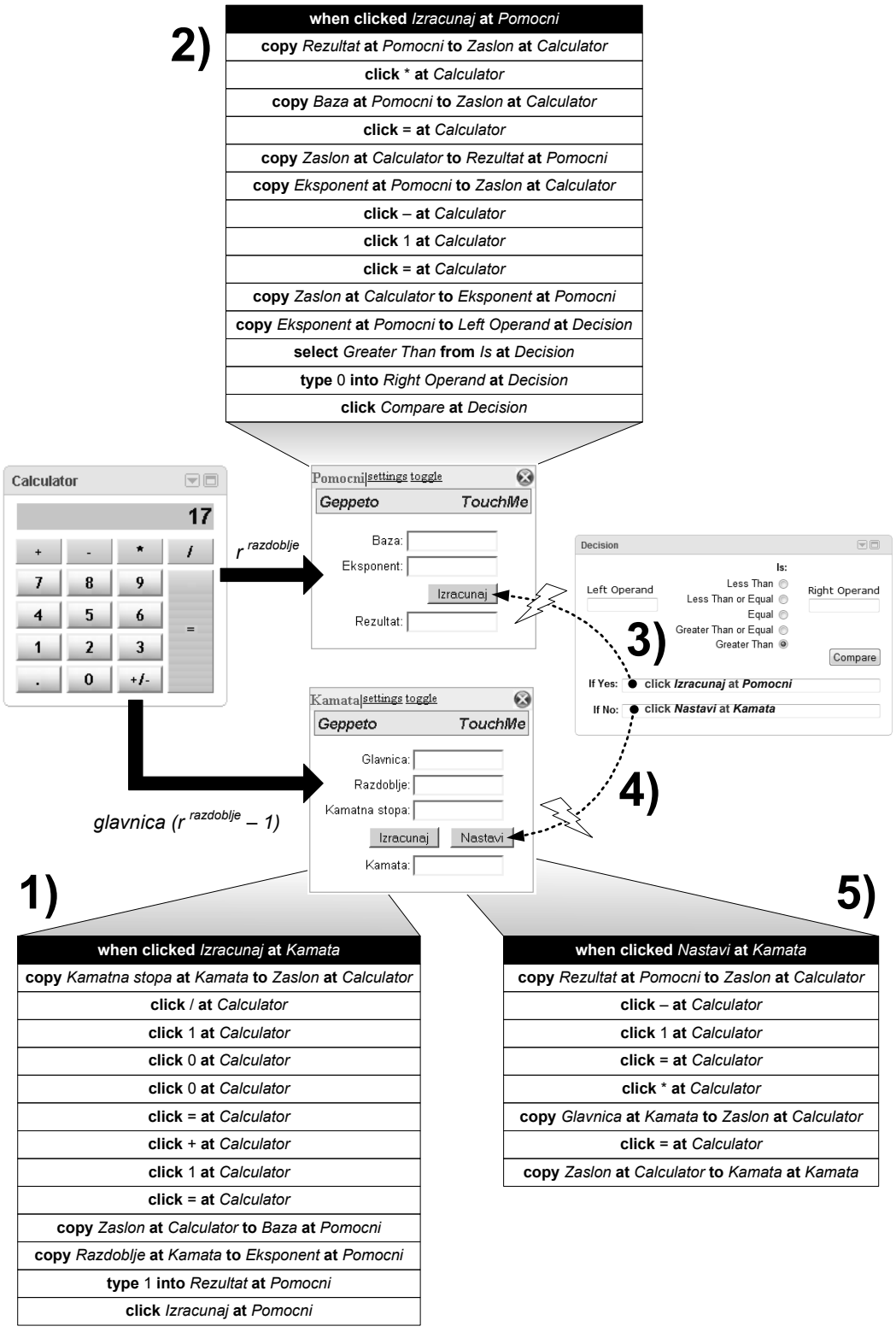
Slika 13.8. Udomljenik za proračun složenog kamatnog računa

Algoritam za izračun kamata ostvaren je primjenom četiriju udomljenika, kao što je prikazano na slici 13.9. Za izračun kamata koriste se udomljenik za izvođenje aritmetičkih operacija, udomljenik za usporedbu logičkih izraza te dva složena udomljenika programirljiva od strane potrošača. Uz svaki složeni udomljenik, prikazana je i programska logika kojom taj udomljenik upravlja radom ostalih udomljenika.

Udomljenik za izvođenje aritmetičkih operacija koristi se za aritmetičke proračune prikazane formulom 13.8. Složeni udomljenik *Pomocni* upravlja radom udomljenika za izvođenje aritmetičkih operacija tijekom izračunavanja izraza $r^{razdoblje}$. S obzirom na to da udomljenik za izvođenje aritmetičkih operacija podržava samo osnovne aritmetičke operacije, potenciranje se izvodi odgovarajućim brojem uzastopnih množenja. Udomljenik za usporedbu logičkih izraza koristi se za zaustavljanje programske petlje koja je potrebna za izvođenje operacije potenciranja uzastopnim množenjem. Za izračunavanje preostalih dijelova formule 13.8, zadužen je složeni udomljenik *Kamata* u sprezi s udomljenikom za izvođenje aritmetičkih operacija. Taj se udomljenik ujedno koristi i za unošenje ulaznih podataka i prikaz rezultata korisniku primjenskog programa.

Izvođenje programa započinje unosom ulaznih podataka u ulazna polja udomljenika *Kamata* i pritiskom potrošača na tipku *Izracunaj*. Pritiskom na tipku *Izracunaj*, pokreće se dio programa koji se odaziva na događaj pritiska na tipku *Izracunaj* (1). U tom dijelu programa, složeni udomljenik *Kamata* najprije uz pomoć udomljenika za izvođenje aritmetičkih operacija izračunava vrijednost izraza *r*. Nakon toga, udomljenik *Kamata* priprema podatke za udomljenik *Pomocni* koji se koristi za izračunavanje izraza $r^{razdoblje}$. Izračunata vrijednost izraza *r* prenosi se s udomljenika za izvođenje aritmetičkih operacija u

ulazno polje *Baza* udomljenika *Pomocni*, dok se vrijednost iz polja *Razdoblje* udomljenika *Kamata* upisuje u polje *Eksponent* udomljenika *Pomocni*.



Slika 13.9. Programsko ostvarenje primjenskog programa za proračun složenog kamatnog računa primjenom programske paradigme za usložnjavanje udomljenika

S obzirom na to da udomljenik *Pomocni* operaciju potenciranja izračunava kao skup slijednih operacija množenja baze i rezultata množenja iz prethodnog koraka, u polje *Rezultat* udomljenika *Pomocni* upisuje se vrijednost 1 kao početni rezultat operacije potenciranja. Izračunavanje izraza $r^{razdoblje}$ pokreće složeni udomljenik *Kamata* pritiskom na tipku *Izracunaj* udomljenika *Pomocni*, čime završava izvođenje dijela programa označenog oznakom (1).

Pritiskom složenog udomljenika *Kamata* na tipku *Izracunaj* složenog udomljenika *Pomocni*, pokreće se dio programa označen oznakom (2). U tom dijelu programa, složeni udomljenik *Pomocni* najprije uz pomoć udomljenika za izvođenje aritmetičkih operacija obavlja operaciju množenja baze i trenutnog rezultata potenciranja. Nakon toga obavlja se provjera da li je za izračun potencije potrebno nastaviti s operacijama množenja. U tu svrhu, složeni udomljenik *Pomocni* uz pomoć udomljenika za izvođenje aritmetičkih operacija smanjuje vrijednost eksponenta za jedan. Nakon toga se, uz pomoć udomljenika za usporedbu logičkih izraza, provjerava da li je vrijednost eksponenta smanjena na nulu. Tijekom usporedbe vrijednosti, umanjena vrijednost eksponenta koristi se u svojstvu lijevog operanda, dok se u svojstvu desnog operanda koristi konstantna vrijednost jednaka nuli. Te dvije vrijednosti udomljenik *Pomocni* upisuje u odgovarajuća polja udomljenika za usporedbu logičkih izraza, uz označavanje operacije *Greater Than* u izbornom polju *Is* za odabir vrste logičke usporedbe. Logičku usporedbu vrijednosti pokreće složeni udomljenik *Pomocni* pritiskom na tipku *Compare* udomljenika za usporedbu logičkih izraza, čime završava izvođenje dijela programa označenog oznakom (2).

Izvođenje programa nastavlja udomljenik za usporedbu logičkih izraza. Ako je trenutna vrijednost eksponenta još uvijek veća od nule, rezultat logičke usporedbe vrednuje se kao logička istina. U tom slučaju, udomljenik za usporedbu logičkih izraza obavlja operaciju pritiska na tipku *Izracunaj* udomljenika *Pomocni* (3). Time se pokreće novi korak izračunavanja potencije $r^{razdoblje}$ ponovnim izvođenjem dijela programa označenog oznakom (2). Ako je trenutna vrijednost eksponenta jednaka nuli, rezultat logičke usporedbe vrednuje se kao logička laž. To znači da su obavljeni svi koraci u izračunavanju potencije $r^{razdoblje}$ uzastopnim množenjem i da je potrebno zaustaviti daljnje množenje. U tom slučaju, udomljenik za usporedbu logičkih izraza obavlja operaciju pritiska na tipku *Nastavi* udomljenika *Kamata* (4), čime se nastavlja s izračunom izraza $glavnica(r^{razdoblje} - 1)$ koji čini preostali dio formule 13.8.

Pritiskom udomljenika za usporedbu logičkih izraza na tipku *Nastavi* složenog udomljenika *Kamata*, pokreće se završni dio programa označen oznakom (5). Polje *Rezultat*

udomljenika *Pomocni* u tom trenutku sadrži izračunatu vrijednost izraza $r^{razdoblje}$. Složeni udomljenik *Kamata* koristi tu vrijednost kako bi uz pomoć udomljenika za izvođenje aritmetičkih operacija izračunao vrijednost izraza *glavnica*($r^{razdoblje} - 1$). Krajnji rezultat izvođenja programa preuzima se s udomljenika za izvođenje aritmetičkih operacija te se prikazuje korisniku u polju *Kamata* istoimenog udomljenika.

14 Zaključak

Istraživanje obuhvaćeno ovom doktorskom disertacijom bavi se proučavanjem postupaka, tehnologija i programskih alata za uključivanje širokog kruga potrošača računala u razvoj računalnih primjenskih programa. Cilj istraživanja je uspostava skupa pravila za oblikovanje potrošaču prilagođene programske paradigme te izbor programskih elemenata i tehnika programiranja za njezino ostvarenje. Istraživanje je potaknuto potrebama suvremenog čovjeka za samostalnim upravljanjem vlastitim životnim i radnim okolinama pogonjenim informacijsko-komunikacijskom tehnologijom za čiju se prilagodbu poosobljenim potrebama pojedinaca zahtijeva određeni oblik programiranja. S druge strane, istraživanje je potaknuto težnjom pojedinaca za stvaralačkim doprinosom razvoju digitalnog društva. Istraživanje je provedeno višedisciplinarnim pristupom koji je obuhvatio područje kognitivne znanosti s naglaskom na psihologiju programiranja, područje marketinga i planiranja tržišta te granu programskog inženjerstva koja se bavi razvojem primjenskih programa od strane krajnjih korisnika.

U okviru doktorske disertacije, definirano je područje programiranja prilagođenog potrošaču. Definicija nove grane programskog inženjerstva zasnovana je na uočljivim razlikama u znanjima, vještinama, potrebama i očekivanjima između školovanih programera koji primjenske programe grade za naručitelja, krajnjih korisnika koji primjenske programe grade u području vlastite stručnosti te potrošača kao najšire skupine korisnika računala koji primjenske programe grade i koriste za širok opseg svakodnevnih osobnih potreba. Programiranje prilagođeno potrošaču je programska paradigma koja, zbog oslonjenosti na opća znanja i iskustvo stečeno prethodnom primjenom informacijsko-komunikacijskih

proizvoda i usluga, dopušta trenutno upuštanje potrošača u postupak izrade primjenskih programa, bez potrebe za dodatnim oblicima školovanja.

Na osnovi definicije, uspostavljen je model potrošaču prilagođene programske paradigme sa skupom smjernica za izbor gradivnih elemenata, programskog jezika, tehnike izgradnje programa i predodžbe programa potrošaču. Model programiranja prilagođenog potrošaču definiran je na osnovi rezultata istraživanja u području kognitivne znanosti te iskustava potrošača stečenih primjenom informacijsko-komunikacijske tehnologije. Na osnovi teorijskih modela kognitivne znanosti kojima se objašnjava umna aktivnost čovjeka tijekom učenja, pamćenja i zaključivanja, definirana su svojstva koja programiranje čine kognitivno bliskim umnom režimu potrošača. Na osnovi analize odnosa potrošača prema tehnologiji, definirana su svojstva koja programiranje čine iskustveno bliskim umnom režimu potrošača. Primjenom definiranog modela, ocjenjene su različite vrste elemenata za izgradnju primjenskih programa i različite vrste programskih jezika za povezivanje elemenata u radni tijek te su izdvojeni oni čija su semantička obilježja i način predodžbe bliski znanju i iskustvu potrošača.

Ostvarenje potrošaču prilagođene programske paradigme predloženo u okviru doktorske disertacije zasnovano je na usložnjavanju udomljenika. Udomljenici su primjenski programi koji ostvaruju potrošaču usmjerene primjenske funkcije te ih putem minimalističkog grafičkog korisničkog sučelja prikazanog u pregledniku *World Wide Web* sadržaja izlažu potrošačima. Minijaturizacijom korisničkog sučelja postiže se mogućnost istodobnog izvođenja proizvoljnog skupa međusobno nezavisnih udomljenika unutar iste *World Wide Web* stranice i njihovo povezivanje u složenije primjenske programe prema zamislima potrošača. Udomljenici su višestruko iskoristivi elementi za izgradnju primjenskih programa proizašlih iz zamisli komponentnog *World Wide Web* sustava, slično kao što su objekti kod objektno-orijentiranog programiranja, programske komponente kod razvoja zasnovanog na komponentama, odnosno usluge kod uslužno-usmjerenog računarstva.

U programskoj paradigmi za usložnjavanje udomljenika, udomljenici se koriste kao osnovne gradivne komponente potrošačkih primjenskih programa, a udomiteljska stranica prikazana u pregledniku *World Wide Web* sadržaja kao potrošaču prilagođena razvojna okolina. Primjenom predložene paradigme, primjenski se program gradi povezivanjem skupa udomljenika u radni tijek na predodžbenoj razini, odnosno na razini grafičkih korisničkih sučelja. Programski jezik za izgradnju radnog tijeka primjenskih programa čine radnje potrošača nad elementima grafičkog korisničkog sučelja udomljenika. U programskoj paradigmi za usložnjavanje udomljenika, tradicionalni oblik programiranja nad apstraktnim

programskim sučeljima objekata, komponenti ili usluga zamijenjen je programiranjem nad potrošaču bliskim i razumljivim grafičkim korisničkim sučeljima udomljenika. Prikaz programa zasnovan je na opisu radnji nad elementima grafičkog korisničkog sučelja udomljenika i njihovom razmještaju u dvodimenzionalnom tabličnom prostoru. Razmještaj programskog zapisa unutar ćelija tablice s dvosmjernim napredovanjem vremena omogućuje jednostavno oblikovanje složenih uzoraka tijeka izvođenja programa, kao što je slijedno i istodobno izvođenje. Primjenski programi izgrađeni od strane potrošača formalno su opisani teorijom grafova.

Kako bi se stvaralačke mogućnosti programske paradigme za usložnjavanje udomljenika usporedile s ostalim oblicima programskih paradigmi, definirana je mjera *stvaralačkih mogućnosti programske paradigme*. Pokazano je da zbog kratkog vremena potrebnog za razvoj programa i velikog broja korisnika kojima je omogućeno sudjelovanje u njihovu razvoju, stvaralačke mogućnosti programske paradigme za usložnjavanje udomljenika mnogostruko premašuju stvaralačke mogućnosti strojnog programiranja te objektno-orijentiranih i primjenski usmjerenih programskih paradigmi. Iako zbog primjetnog nedostatka udomljenika stvaralačke mogućnosti predložene programske paradigme danas nije moguće u potpunosti iskoristiti, izvršene su procjene budućeg porasta broja udomljenika koje pokazuju da je njenu punu uporabljivost moguće postići za tri do devet godina. Potpunost paradigme pokazana je istovjetnošću s modelom Turingovog stroja.

Programska paradigma za usložnjavanje udomljenika programski je ostvarena proširenjem sustava *Apache Shindig* [222]. *Apache Shindig* je programski sustav otvorenog kôda koji pruža funkcionalnosti udomiteljske stranice za izvođenje udomljenika. Izvorno je pokrenut od strane tvrtke *Google Inc.* kao referentni sustav za izvođenje udomljenika usklađenih s *OpenSocial* [253] i *Google Gadgets* [211] specifikacijama. Izvorni oblik udomiteljske stranice sustava *Apache Shindig* proširen je plivajućim izbornikom za definiranje radnog tijeka nad skupom osnovnih udomljenika te programirljivim složenim udomljenikom za spremanje i pokretanje izgrađenih programa. U programskoj paradigmi za usložnjavanje udomljenika, bilo koji udomljenik izgrađen prema *Google Gadgets* specifikaciji moguće je koristiti kao osnovni blok za izgradnju primjenskog programa. Sustav je u obliku web usluge dostupan na adresi *http://geppeto.fer.hr*.

Rezultati istraživanja i izgrađeni programski alat predstavljeni su na američkim sveučilištima *Stanford University*, Palo Alto, CA, *University of California at Irvine*, Irvine, CA, *University of Southern California*, Los Angeles, CA i *Naval Postgraduate School*, Monterey, CA, tvrtci *Google Inc.*, Mountain View, CA, državnoj agenciji za istraživanje

mora i atmosfere *NOAA*, Pacific Grove, CA te na *University of Toronto* u Kanadi. Programska paradigma za usložnjavanje udomljenika planira se iskoristiti u dva područja primjene s naglašenim potrebama za poosobljavanje primjenskih programa zahtjevima pojedinaca. S jedne strane, programska paradigma planira se primijeniti za oblikovanje primjenskih programa za praćenje koncentracije klorofila te morskih i zračnih strujanja uz obale Tihog oceana. S druge strane, planirana je primjena paradigme za izradu primjenskih programa za praćenje financijskih pokazatelja dionica na burzi, izradu tehničke analize i potporu ulagačkom odlučivanju.

Oblikovanje potrošaču prilagođene programske paradigme prvi je korak prema iskorištavanju stvaralačkih mogućnosti širokog kruga korisnika računala i mreže Internet za razvoj primjenskih programa. Programskom paradigmom za usložnjavanje udomljenika definiran je potrošaču prilagođen postupak za oblikovanje tijeka izvođenja i toka podataka nad skupom računalnih procesa izloženih putem udomljenika. Istraživanja je moguće nastaviti u smjeru proširenja osnovnog oblika programske paradigme elementima za izgradnju primjenskih programa posebne namjene, kao što su programi poticani događajima iz okoline i programi za obradu dinamički promjenjivih informacija u stvarnom vremenu. Nadalje, istraživanja je moguće nastaviti u smjeru pronalaska metodologije za razmjenu potrošačkih programa sa zajednicom potrošača i udruženog sudjelovanja u njihovu razvoju. Usmjerene grafove za formalni opis potrošačkih programa moguće je primijeniti za prikupljanje podataka i dubinsku analizu strukturne, podatkovne i vremenske povezanosti udomljenika uključenih u radni tijek primjenskih programa. Rezultati dubinske analize primjenjivi su za izradu statistike najkorisnijih udomljenika, najčešće korištenih uzoraka usložnjavanja udomljenika, sličnih primjenskih programa koje su izgradili različiti potrošači i slično. Na osnovi tih informacija, moguće je oblikovati algoritme za ponudu udomljenika s ciljem izgradnje potpomagane razvojne okoline koja potrošača vodi kroz postupak razvoja primjenskih programa.

15 Literatura

[1] S. D. Reese, L. Rutigliano, K. Hyun, J. Jeong: “Mapping the Blogosphere: Professional and Citizen-Based Media in the Global News Arena”, Journalism, Vol. 8, No. 3, 2007, pp. 235-261

[2] C. Marlow: “Audience, Structure and Authority in the Weblog Community”, International Communication Association Conference, May 2004

[3] B. Coffey, S. Woolworth: “Destroy the Scum, and Then Neuter Their Families: The Web Forum as a Vehicle for Community Discourse?”, The Social Science Journal, Vol. 41, No. 1, 2004, pp. 1-14

[4] L. Denoyer, P. Gallinari: “The Wikipedia XML Corpus”, ACM SIGIR Forum, Vol. 40, No. 1, June 2006, pp. 64-69

[5] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, S. Moon: “I Tube, You Tube, Everybody Tubes: Analyzing the World’s Largest User Generated Content Video System”, Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, San Diego, CA, USA, 2007, pp. 1-14

[6] A. N. Joinson: “Looking At, Looking Up or Keeping Up with People?: Motives and Use of Facebook”, Proceeding of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems, Florence, Italy, 2008, pp. 1027-1036

[7] M. Thelwall: “Social Networks, Gender, and Friending: An Analysis of MySpace Member Profiles”, Journal of the American Society for Information Science and Technology, Vol. 59, No. 8, June 2008, pp. 1321-1330

[8] A. Java, X. Song, T. Finin, B. Tseng: “Why We Twitter: Understanding Microblogging Usage and Communities”, Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis, San Jose, CA, USA, 2007, pp. 56-65

[9] S. J. Andriole, E. Roberts: “Point/Counterpoint: Technology Curriculum for the Early 21st Century”, Communications of the ACM, Vol. 51, No. 7, July 2008, pp. 27-32

[10] T. Tullis, B. Albert: “Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics”, Morgan Kaufmann, 2008

[11] J. Van Dijk, K. Hacker: “The Digital Divide as a Complex and Dynamic Phenomenon”, Proceedings of the 50th Annual Conference of the International Communication Association, Acapulco, Mexico, June 2000

[12] M. S. Eastin, R. LaRose: “Internet Self-Efficacy and the Psychology of the Digital Divide”, Journal of Computer-Mediated Communication, Vol. 6, No. 1, September 2000

[13] A. Molina: “The Digital Divide: The Need for a Global e-Inclusion Movement”, Technology Analysis & Strategic Management, Vol. 15, No. 1, March 2003, pp. 137-152

[14] One Laptop Per Child (OLPC), World Wide Web, http://laptop.org

[15] 50x15, World Wide Web, http://50x15.org

[16] B. A. Myers: “A Brief History of Human Computer Interaction Technology”, ACM Interactions, Vol. 5, No. 2, March 1998, pp. 44-54

[17] M. H. Chignell, J. A. Waterworth: “Wimps and Nerds: An Extended View of the User Interface”, ACM SIGCHI Bulletin, Vol. 23, No. 2, 1991, pp. 15-21

[18] S. H. Han, M. H. Yun, J. Kwahk, S. W. Hong: “Usability of Consumer Electronic Products”, International Journal of Industrial Ergonomics, Vol. 28, No. 3-4, September-October 2001, pp. 143-151

[19] A. J. Ko, R. Abraham, M. M. Burnett, B. A. Myers: “End-User Software Engineering”, IEEE Software, Vol. 26, No. 5, September/October 2009, pp. 16-17

[20] M. Burnett, G. Rothermel, C. Cook: “Software Engineering for End-User Programmers”, Workshop on End User Development, Fort Lauderdale, FL, USA, April 2003, pp. 12-15

[21] H. Prahofer, D. Hurnaus, H. Mossenbock: “Building End-User Programming Systems Based on a Domain-Specific Language”, White Paper, ZDNet, World Wide Web, http://www.dsmforum.org/events/DSM06/Papers/4-Praehofer.pdf

[22] R. Guo, B. B. Zhu, M. Feng, A. Pan, B. Zhou: “Compoweb: A Component-Oriented Web Architecture”, Proceedings of the 17th International Conference on World Wide Web, April 2008, Beijing, China, pp. 545-554

[23] A. F. Blackwell: “What is programming?”, Proceedings of the 14th Workshop of the Psychology of Programming Interest Group (PPIG 2002), London, UK, June 2002, pp. 204-218

[24] T. R. G. Green, M. Petre: “Usability Analysis of Visual Programming Environments: A Cognitive Dimensions Framework”, Journal of Visual Languages and Computing, Vol. 7, No. 2, June 1996, pp. 131-174

[25] J. P. Kenny, C. H. Waddington, H. C. Longuet-Higgins, J. R. Lucas: “The Nature of Mind”, Gifford Lectures, 1971-1973

[26] P. Thagard: “Mind: Introduction to Cognitive Science”, Second Edition, MIT Press, Cambridge, MA, USA, 2005

[27] P. Thagard: “Cognitive Science”, Stanford Encyclopedia of Philosophy, April 2007, World Wide Web, http://plato.stanford.edu/entries/cognitive-science

[28] G. M. Weinberg: “The Psychology of Computer Programming”, Silver Anniversary Edition, Dorset House Publishing, New York, NY, USA, 1998

[29] T. T. Carey, M. M. Shepherd: “Towards Empirical Studies of Programming in New Paradigms”, Proceedings of the 16th ACM Annual Conference on Computer Science, Atlanta, GA, USA, 1988, pp. 72-78

[30] J. Fodor, Z. Pylyshyn: “Connectionism and Cognitive Architecture: A Critical Analysis”, Cognition, Vol. 28, 1988, pp. 3-71

[31] J. Garson: “Connectionism”, Stanford Encyclopedia of Philosophy, March 2007, World Wide Web, http://plato.stanford.edu/entries/connectionism

[32] J. Watson, “Psychology as a Behaviorist Views It”, Psychological Review, Vol. 20, 1913, pp. 158-77

[33] G. Graham: “Behaviorism”, Stanford Encyclopedia of Philosophy, July 2007, World Wide Web, http://plato.stanford.edu/entries/behaviorism

[34] A. F. Blackwell, K. N. Whitley, J. Good, M. Petre: “Cognitive Factors in Programming with Diagrams”, Artificial Intelligence Review, Vol. 15, No. 1-2, 2001, pp. 95-114

[35] S. Xu: “A Cognitive Model for Program Comprehension”, Proceedings of the 2005 Third ACIS International Conference on Software Engineering Research, Management, and Applications (SERA’05), Mt. Pleasant, MI, USA, August 2005, pp. 392-398

[36] M. Petre, A. F. Blackwell: “Mental Imagery in Program Design and Visual Programming”, International Journal of Human Computer Studies, Vol. 51, No. 1, July 1999, pp. 7-30

[37] R. Navarro-Prieto, J. J. Canas: “Are Visual Programming Languages Better? The Role of Imagery in Program Comprehension”, International Journal of Human Computer Studies, Vol. 54, No. 6, June 2001, pp. 799-829

[38] R. Navarro-Prieto, J. J. Canas: “Mental Representation and Imagery in Program Comprehension”, Proceedings of the 11th Workshop of the Psychology of Programming Interest Group (PPIG 1999), Leeds, UK, January 1999, pp. 1-6

[39] R. Mancy, N. Reid: “Aspects of Cognitive Style and Programming”, Proceedings of the 16th Workshop of the Psychology of Programming Interest Group (PPIG 2004), Carlow, Ireland, April 2004, pp. 1-9

[40] A. J. Ko, B. A. Myers, H. H. Aung: “Six Learning Barriers in End-User Programming Systems”, Proceedings of the 2004 IEEE International Symposium on Visual Languages and Human Centric Computing, Rome, Italy, September 2004, pp. 199-206

[41] N. Pennington: “Stimulus Structures and Mental Representations in Expert Comprehension of Computer Programs”, Cognitive Psychology, Vol. 19, 1987, pp. 295-341

[42] G. A. Miller: “The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information”, The Psychological Review, Vol. 63, 1956, pp. 81-97

[43] B. Bell, W. Citrin, C. Lewis, J. Rieman, R. Weaver, N. Wilde, B. Zorn: “Using the Programming Walkthrough to Aid in Programming Language Design”, Software – Practice and Experience, Vol. 24, No. 1, January 1994, pp. 1-25

[44] A. F. Blackwell: “See What You Need: Helping End-users to Build Abstractions”, Journal of Visual Languages and Computing, Vol. 12, No. 5, October 2001, pp. 475-499

[45] C. Digiano, K. Kahn, A. Cypher, D. C. Smith: “Integrating Learning Supports into the Design of Visual Programming Systems”, Journal of Visual Languages and Computing, Vol. 12, No. 5, October 2001, pp. 501-524

[46] J. Rode, M. B. Rosson: “Programming at Runtime: Requirements & Paradigms for Nonprogrammer Web Application Development”, IEEE HCC 2003, Auckland, New Zealand, 2003

[47] K. Stenning, J. Oberlander: “A Cognitive Theory of Graphical and Linguistic Reasoning: Logic and implementation”, Cognitive Science, Vol. 19, No. 1, 1995, pp. 97-140

[48] B. Bell, J. Rieman, C. Lewis: “Usability Testing of a Graphical Programming System: Things We Missed in a Programming Walkthrough”, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Reaching Through Technology, New Orleans, LA, USA, 1991, pp. 7-12

[49] S. Clarke: “Evaluating a New Programming Language”, Proceedings of the 13th Workshop of the Psychology of Programming Interest Group (PPIG 2001), Bournemouth, UK, April 2001, pp. 275-289

[50] C. Austin: “A Human Factors Analysis of a Functional Shading Language”, Chapter 4 in “Renaissance: A Functional Shading Language”, M.Sc. Thesis, Iowa State University, Ames, IA, USA, 2005

[51] B. Shneiderman: “Response Time and Display Rate in Human Performance with Computers”, ACM Computing Surveys, Vol. 16, No. 3, September 1984, pp. 265-285

[52] M. Kavakli, J. S. Gero: “The Structure of Concurrent Cognitive Actions: A Case Study on Novice and Expert Designers”, Design Studies, Vol. 23, No. 1, January 2002, pp. 25-40

[53] M. Kavakli, M. Suwa, J. S. Gero, T. Purcell: “Sketching Interpretation in Novice and Expert Designers”, Visual Reasoning in Design, Key Centre of Design Computing and Cognition, University of Sydney, Sydney, NSW, Australia, 1999, pp. 209-219

[54] F. Turbak, C. Royden, J. Stephan, J. Herbst: “Teaching Recursion Before Loops In CS1”, Journal of Computing in Small Colleges, Vol. 14, No. 4, May 1999, pp. 86-101

[55] D. Ginat, E. Shifroni: “Teaching Recursion in a Procedural Environment – How Much Should We Emphasize the Computing Model?”, ACM SIGCSE Bulletin, Vol. 31, No. 1, March 1999, pp. 127-131

[56] T. Gotschi, I. Sanders, V. Galpin: “Mental Models of Recursion”, ACM SIGCSE Bulletin, Vol. 35, No. 1, January 2003, pp. 346-350

[57] I. Sanders, V. Galpin, T. Gotschi: “Mental Models of Recursion Revisited”, ACM SIGCSE Bulletin, Vol. 38, No. 3, September 2006, pp. 138-142

[58] D. M. Kurland, R. D. Pea: “Children’s Mental Models of Recursive Logo Programs”, Journal of Educational Computing Research, Vol. 1, No. 2, 1985, pp. 235-243

[59] K. Ueda: “Concurrent Logic/Constraint Programming: The Next 10 Years”, The Logic Programming Paradigm: A 25-Year Perspective, Springer-Verlag, 1999, pp. 53-71

[60] P. Koopman, R. R. Hoffman: “Work-arounds, Make-work, and Kludges”, IEEE Intelligent Systems, Vol. 18, No. 6, November 2003, pp. 70-75

[61] M. Shanahan: “The Frame Problem”, Stanford Encyclopedia of Philosophy, November 2009, World Wide Web, http://plato.stanford.edu/entries/frame-problem

[62] L. Boroditsky: “Does Language Shape Thought?: Mandarin and English Speakers’ Conceptions of Time”, Cognitive Psychology, Vol. 43, 2001, pp. 1-22

[63] B. Goertzel: “On the Physics and Phenomenology of Time”, World Wide Web, http://www.goertzel.org/papers/timepap.html

[64] R. Le Poidevin: “The Experience and Perception of Time”, Stanford Encyclopedia of Philosophy, November 2009, World Wide Web, http://plato.stanford.edu/entries/time-experience

[65] T. W. Malone, K. Crowston: “The Interdisciplinary Study of Coordination”, ACM Computing Surveys, Vol. 26, No. 1, March 1994, pp. 87-119

[66] E. M. Roggers: “Diffusion of Innovations”, Fourth Edition, Free Press, 1995

[67] E. M. Rogers: “New Product Adoption and Diffusion”, Journal of Consumer Research, Vol. 2, No. 4, March 1976, pp. 290-301

[68] C. Jones: “End-User Programming”, IEEE Computer, Vol. 28, No. 9, September 1995, pp. 68-70

[69] J. Rode, M. B. Rosson, M. A. Perez Quinones: “End User Development of Web Applications”, End User Development, Kluwer/Springer, 2005

[70] J. Rode: “Nonprogrammer Web Application Development”, Conference on Human Factors in Computing Systems (CHI2004), Doctoral Consortium, Vienna, Austria, April 2004, pp. 1055-1056

[71] C. Scaffidi, M. Shaw, B. Myers: “Estimating the Numbers of End Users and End User Programmers”, Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing, Dallas, TX, USA, September 2005, pp. 207-214

[72] R. Saint-Paul, B. Benatallah, J. Vayssiere: “Data Services in your Spreadsheet”, Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology, Nantes, France, 2008, pp. 690-694

[73] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, S. R. Klemmer: “Opportunistic Programming: Writing Code to Prototype, Ideate, and Discover”, IEEE Software, Vol. 26, No. 5, September/October 2009, pp. 18-24

[74] B. A. Myers, J. F. Pane, A. Ko: “Natural Programming Languages and Environments”, Communications of the ACM, Vol. 47, No. 9, September 2004, pp. 47-52

[75] D. A. Carr: “End-User Programmers Need Improved Development Support”, Workshop on End User Development, Fort Lauderdale, FL, USA, April 2003, pp. 16-18

[76] M. F. Costabile, A. Piccinno, D. Fogli, P. Mussio: “Software Shaping Workshops: Environments to Support End-User Development”, Workshop on End User Development, Fort Lauderdale, FL, USA, April 2003, pp. 19-22

[77] S. D. J. Barbosa, P. Palanque, R. Bastide: “Some Generic Mechanisms for Increasing the Usability of EUD Environments”, Workshop on End User Development, Fort Lauderdale, FL, USA, April 2003, pp. 27-32

[78] J. H. Jahnke, M. d’Entremont, M. Lavender, A. McNair: “User-Programming of Net-Centric Embedded Control Software”, Workshop on End User Development, Fort Lauderdale, FL, USA, April 2003, pp. 39-42

[79] R. C. Miller: “End User Programming for Web Users”, Workshop on End User Development, Fort Lauderdale, FL, USA, April 2003, pp. 61-64

[80] F. Cox, T. Smedley: “Special Issue on Visual Languages for End-user and Domain-specific Programming”, Journal of Visual Languages and Computing, Vol. 12, No. 5, October 2001, pp. 473-474

[81] I. F. Cruz, P. S. Leveille: “As You Like It: Personalized Database Visualization Using a Visual Language”, Journal of Visual Languages and Computing, Vol. 12, No. 5, October 2001, pp. 525-549

[82] M. Erwig: “Semantics of Visual Languages”, Proceedings of the 13th IEEE Symposium on Visual Languages (VL’97), Capri, 1997, pp. 304-311

[83] C. Emig, C. Momm, J. Weisser, S. Abeck: “Programming in the Large based on the Business Process Modeling Notation”, Informatik 2005, GI Jahrestagung, Vol. 2, September 2005, pp. 627-631

[84] “VPL Introduction”, Microsoft Robotics Developer Center, Microsoft Corporation, 2009, World Wide Web, http://msdn.microsoft.com/en-us/library/bb483088.aspx

[85] N. Ertugrul: “Towards Virtual Laboratories: A Survey of LabVIEW-based Teaching/Learning Tools and Future Trends”, International Journal of Engineering Education, Vol. 16, No. 3, 2000, pp. 171-180

[86] “What is LabVIEW”, National Instruments, World Wide Web, http://www.ni.com/labview/whatis

[87] D. J. Power: “A Brief History of Spreadsheets”, DSSResources.COM, World Wide Web, August 2004, http://dssresources.com/history/sshistory.html

[88] D. Bricklin, B. Frankston: “VisiCalc: Information from Its Creators, Dan Bricklin and Bob Frankston”, World Wide Web, http://www.bricklin.com/visicalc.htm

[89] R. Mattessich: “Spreadsheet: Its First Computerization (1961-1964)”, DSSResources.COM, World Wide Web, http://dssresources.com/history/mattessichspreadsheet.htm

[90] “Lotus 1-2-3”, IBM Corporation, World Wide Web, http://www-01.ibm.com/software/lotus/products/123

[91] “Microsoft Office Excel”, Microsoft Office Online, Microsoft Corporation, World Wide Web, http://office.microsoft.com/en-us/excel/default.aspx

[92] “Google Spreadsheets APIs and Tools”, Google Code, Google Inc., World Wide Web, http://code.google.com/apis/spreadsheets

[93] D. Wakeling: “Spreadsheet Functional Programming”, Journal of Functional Programming, Vol. 17, No. 1, January 2007, pp. 131-143

[94] M. Burnett, J. Atwood, R. W. Djang, J. Reichwein, H. Gottfried, S. Yang: "Forms/3: A First-Order Visual Language to Explore the Boundaries of the Spreadsheet Paradigm", Journal of Functional Programming, Vol. 11, No. 2, March 2001, pp. 155-206

[95] M. P. Singh, M. N. Huhns: "Service-Oriented Computing: Semantics, Processes, Agents", John Wiley & Sons, 2005

[96] M. Huhns, M. P. Singh: "Service-Oriented Computing: Key Concepts and Principles", IEEE Internet Computing, Vol. 9, No. 1, January/February 2005, pp. 75-81

[97] M. P. Paparazoglou, D. Georgakopoulos: "Service-Oriented Computing", Communications of the ACM, Vol. 46, No. 10, October 2003, pp. 25-28

[98] M. P. Papazoglou: "Service-Oriented Computing: Concepts, Characteristics and Directions", 4th IEEE International Conference on Web Information Systems Engineering (WISE 2003), December 2003, Roma, Italy

[99] T. Margaria, B. Steffen: "Service Engineering: Linking Business and IT", IEEE Computer, Vol. 39, No. 10, October 2006, pp. 45-55

[100] M. P. Papazoglou: "Extending the Service-Oriented Architecture", Business Integration Journal, February 2005, pp. 18-21

[101] W. T. Tsai, Y. Chen, G. Bitter, D. Miron: "Introduction to Service-Oriented Computing", ASU Workshop on Service-Oriented Architecture Education, Research, and Applications, Arizona State University, Tempe, AZ, USA, May 2006

[102] L. Dimitriou: "Distributed Parallel Computing with Web Services", Web Services Journal, Vol. 5, No. 2, February 2005, pp. 28-31

[103] J.-Y. Chung, K.-J. Lin, R.G. Mathieu: "Web Services Computing: Advancing Software Interoperability", Computer, vol. 36, no. 10, October 2003, pp. 35-37

[104] A. Milanović: "Programski model zasnovan na uslugama", doktorska disertacija, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, prosinac 2005

[105] I. Benc: "Samooblikujuća arhitektura zasnovana na uslugama", doktorska disertacija, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, srpanj 2008

[106] I. Skuliber: "Sredstvima uvjetovana suradnja i natjecanje u sustavima zasnovanim na uslugama", doktorska disertacija, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, veljača 2009

[107] D. Skrobo: “Kompozicija usluga zasnovana na tabličnom programiranju”, doktorska disertacija, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, listopad 2008

[108] D. Škvorc: “Prividna mreža računalnih sustava zasnovanih na uslugama”, magistarski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, siječanj 2006

[109] M. Podravec: “Otkrivanje i postavljanje usluga u sustavima zasnovanim na uslugama”, magistarski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, siječanj 2006

[110] I. Gavran: “Korisnički jezik programskog modela zasnovanog na uslugama”, magistarski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, ožujak 2006

[111] D. Skrobo: “Raspodijeljeno usporedno interpretiranje programa u arhitekturama zasnovanim na uslugama”, magistarski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, siječanj 2006

[112] M. Popović: “Nadziranje pristupa računalnim sustavima zasnovanim na uslugama”, magistarski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, Zagreb, svibanj 2006

[113] D. Skvorc, S. Srbljic, I. Gavran: “Application-level Quality of Security Service Provisioning in Distributed Service-Oriented Middleware”, Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics, and Informatics (WMSCI’05), Vol. 2, Network Security and Security Technologies, Orlando, FL, USA, 2005

[114] D. Skvorc, S. Srbljic, M. Podravec: “Virtual Network for Development and Execution of Service-Oriented Applications”, Proceedings of International Conference on Networking and Services (ICNS’06), Santa Clara, CA, USA, 2006, pp. 96-101

[115] A. Milanovic, S. Srbljic, D. Skrobo, D. Capalija, S. Reskovic: “Coopetition Mechanisms for Service-Oriented Distributed Systems”, Proceedings of 3rd International Conference on Computing, Communications and Control Technologies, Cybernetics and Informatics (CCCT’05), Austin, TX, USA, July 2005, pp. 118-123

[116] D. Skrobo, A. Milanovic, S. Srbljic: “Distributed Program Interpretation in Service-Oriented Distributed Systems”, Proceedings of the WMSCI’05, Orlando, FL, USA, July 2005, pp. 193-197

[117] I. Gavran, A. Milanovic, S. Srbljic: "End-User Programming Language for Service-Oriented Integration", Proceedings of the 7th Workshop on Distributed Data and Structures, Santa Clara, CA, USA, 2006, pp. 1-8

[118] M. Podravec, I. Skuliber, S. Srbljic: "Service Discovery and Deployment in Service-Oriented Computing Environment", Proceedings of the WMSCI'05, Vol. 3, Orlando, FL, USA, July 2005, pp. 5-10

[119] D. Skrobo, K. Vladimir, S. Srbljic: "Usage Tracking Components for Service-Oriented Middleware Systems", Proceedings of the 2007 Middleware for Web Services Workshop (MWS 2007), Annapolis, MD, USA, October 2007, pp. 1-8

[120] D. Skrobo, A. Milanovic, S. Srbljic: "Performance Evaluation of Program Translation in Service-Oriented Architectures", Proceedings of the International Conference on Networking and Services (ICNS'06), Santa Clara, CA, USA, July 2006, pp. 14-19

[121] M. Popovic, I. Benc, S. Srbljic: "Access Control in Hierarchies of Protected Cells", Proceedings of 9th World Multi-Conference on Systemics, Cybernetics and Informatics, Orlando, FL, USA, July 2005, pp. 356-361

[122] N. B. Lakhal, T. Kobayashi, H. Yokota: "THROWS: An Architecture for Highly Available Distributed Execution of Web Services Compositions", Proceedings of the 14th International Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications (RIDE'04), Boston, MA, USA, March 2004, pp.103-110

[123] R. S. Silva Filho, J. Wainer, E. R. M. Madeira: "A Fully Distributed Architecture for Large-scale Workflow Enactment", International Journal of Cooperative Information Systems (IJCIS), Vol. 12, No. 4, December 2003, pp. 411-440

[124] M. Tatsubori, K. Takashi: "Decomposition and Abstraction of Web Applications for Web Service Extraction and Composition", Proceedings of the IEEE International Conference on Web Services (ICWS'06), 2006, pp. 859-868

[125] S. Vinoski: "REST Eye for the SOA Guy", IEEE Internet Computing, Vol. 11, No. 1, January/February 2007, pp. 82-84

[126] "The application/rss+xml Media Type", Internet Draft, Network Working Group, May 2006, World Wide Web, http://www.rssboard.org/rss-mime-type-application.txt

[127] B. Brauer: "Next Evolution of Data Integration into Microsoft Excel", Technical Report, StrikeIron Inc., 2006

[128] P. Hudak, et. al.: “Report on the Programming Language Haskell: A Non-Strict, Purely Functional Language”, ACM SIGPLAN Notices, 1992

[129] J. Sajaniemi: “A New Interface to Spreadsheet Programming: A Truly Seamless Fusion of Spreadsheet and Word Processing Paradigms”, Proceedings of the IEEE Symposia on Human Centric Computing Languages and Environments, 2002, pp. 40-42

[130] M. Burnett, S. Yang, J. Summet: “A Scalable Method for Deductive Generalization in the Spreadsheet Paradigm”, ACM Transactions on Computer-Human Interaction, Vol. 9, No. 4, December 2002, pp. 253-284

[131] T. Chusho, K. Fujiwara: “FACL: A Form-Based Agent Communication Language for Enduser-Initiative Agent-Based Application Development”, Proceedings of the 24th Annual International Computer Software and Applications Conference (COMPSAC 2000), 2000, pp. 139-148

[132] T. Chusho, K. Fujiwara, H. Ishigure, K. Shimada: “A Form-Based Approach for Web Services by End-User-Initiative Application Development”, Proceedings of the 2002 Symposium on Applications and the Internet (SAINT), 2002, pp. 196-203

[133] D. Halbert: “Programming by Example”, Ph.D. Thesis, University of California, Berkeley, CA, USA, November 1984

[134] A. Billard, S. Calinon, R. Dillmann, S. Schaal: “Robot Programming by Demonstration”, Handbook of Robotics, Chapter 59, MIT Press, 2007

[135] A. Safonov, J. A. Konstan, J. V. Carlis: “End-user Web Automation: Challenges, Experiences, Recommendations”, Proceedings of the World Conference on the WWW and Internet (WebNet 2001), Orlando, Florida, October 2001, pp. 1077-1085

[136] V. Anupam, J. Freire, B. Kumar, D. Lieuwen: “Automating Web Navigation with the WebVCR”, Proceedings of the 9th International World Wide Web Conference, Amsterdam, May 2000, pp. 503-517

[137] “Introducing iMacros”, iMacros.net, World Wide Web, http://wiki.imacros.net/Introducing_iMacros

[138] “Newbie – Automating the Web Browsing Experience”, World Wide Web, http://www.newbielabs.com

[139] “IRobotSoft – Visual Web Scraping and Web Automation Tool”, World Wide Web, http://www.irobotsoft.com/

[140] A. Cypher: “EAGER: Programming Repetitive Tasks by Example”, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 1991, pp. 33-39

[141] C. D. Hundhausen, S. Farley, J. L. Brown: “Can Direct Manipulation Lower the Barriers to Programming and Promote Positive Transfer to Textual Programming? An Experimental Study”, Proceedings of the Visual Languages and Human-Centric Computing (VL/HCC’06), Brighton, UK, September 2006, pp. 157-164

[142] B. Schneiderman: “Direct Manipulation: A Step Beyond Programming Languages”, IEEE Computer, Vol. 16, No. 8, August 1983, pp. 57-69

[143] J. Golbeck: “Direct Manipulation”, Theories in Computer Human Interaction, University of Maryland, College Park, MD, USA, October 2002

[144] A. Khella: “Objects-Actions Interface Model”, Theories in Computer Human Interaction, University of Maryland, College Park, MD, USA, October 2002

[145] H. Zhao: “Fitt’s Law: Modeling Movement Time in HCI”, Theories in Computer Human Interaction, University of Maryland, College Park, MD, USA, October 2002

[146] L. Hochstein: “GOMS”, Theories in Computer Human Interaction, University of Maryland, College Park, MD, USA, October 2002

[147] R. Mihalcea, H. Liu, H. Lieberman: “NLP (Natural Language Processing) for NLP (Natural Language Programming)”, Lecture Notes in Computer Science, Vol. 3878, 2006, pp. 319-330

[148] J. Allen, Q. Duong, C. Thompson: “Natural Language Service for Controlling Robots and Other Agents”, IEEE International Conference on Knowledge Intensive Multiagent Systems (KIMAS’05), Waltham, MA, USA, April 2005, pp. 592-595

[149] G. Nelson: “Inform 7: A Design System for Interactive Fiction Based on Natural Language”, inform-fiction.org, World Wide Web, http://inform-fiction.org/I7/Welcome.html

[150] E. Lecolinet: “A Brick Construction Game Model for Creating Graphical User Interfaces: The Ubit Toolkit”, Proceedings of the 7th IFIP Conference on Human-Computer Interaction (INTERACT’99), Edinburgh, Scotland, August-September 1999, pp. 510-518

[151] K. Ebcioglu, V. Saraswat, V. Sarkar: “X10: An Experimental Language for High Productivity Programming of Scalable Systems”, Proceedings of Workshop on

Productivity and Performance in High-End Computing (PPHEC-05), San Francisco, CA, USA, February 2005, pp. 45-52

[152] L. Beckwith, M. Burnett, V. Grigoreanu, S. Wiedenbeck: “Gender HCI: What About the Software”, IEEE Computer, Vol. 39, No. 11, November 2006, pp. 97-101

[153] D. A. Grier: “The Benefits of Being Different”, IEEE Computer, Vol. 39, No. 11, November 2006, pp. 6-8

[154] S. J. Vaughan-Nichols: “Researchers Make Web Searches More Intelligent”, IEEE Computer, Vol. 39, No. 12, December 2006, pp. 16-18

[155] N. Kobayashi: “Useless-Code Elimination and Program Slicing for the Pi-Calculus”, Lecture Notes in Computer Science, Vol. 2895, November 2003, pp. 55-72

[156] W. L. Yeung: “Automated Translation of JSD into CSP: A Case Study in Methods Integration”, Journal of Systems and Software, Vol. 55, No. 2, December 2000, pp. 193-202

[157] J. A. Bergstra, J. W. Klop, J. V. Tucker: “Algebraic Tools for System Construction”, Lecture Notes in Computer Science, Vol. 164, 1984, pp. 34-44

[158] C. A. R. Hoare: “An Axiomatic Basis for Computer Programming”, Communications of the ACM, Vol. 12, No. 10, October 1969, pp. 576-580

[159] C. O’Hanlon: “A Conversation with John Hennessy and David Patterson”, ACM Queue, Vol. 4, No. 10, December/January 2006/2007, pp. 14-22

[160] A.-R. Adl-Tabatabai, C. Kozyrakis, B. Saha: “Unlocking Concurrency”, ACM Queue, Vol. 4, No. 10, December/January 2006/2007, pp. 24-33

[161] C. O’Hanlon: “Forward Thinking”, ACM Queue, Vol. 4, No. 10, December/January 2006/2007, p. 8

[162] S. Crosby, D. Brown: “The Virtualization Reality”, ACM Queue, Vol. 4, No. 10, December/January 2006/2007, pp. 34-41

[163] C. A. R. Hoare: “Communicating Sequential Processes”, Communications of the ACM, Vol. 21, No. 8, August 1978, pp. 666-677

[164] S. D. Brookes, C. A. R. Hoare, A. W. Roscoe: “A Theory of Communicating Sequential Processes”, Journal of the ACM, Vol. 31, No. 3, July 1984, pp. 560-599

[165] B. Padmanabhan, Y. Yang: “Clickprints on the Web: Are There Signatures in Web Browsing Data?”, Knowledge@Wharton Network, University of Pennsylvania,

September 2006, World Wide Web, http://knowledge.wharton.upenn.edu/papers/1323.pdf

[166] T. Kistler, H. Marais: "WebL – A Programming Language for the Web", Computer Networks and ISDN Systems, Vol. 30, No. 1-7, April 1998, pp. 259-270

[167] B. Pierce: "Foundational Calculi for Programming Languages", CRC Handbook of Computer Science and Engineering, CRC Press, 1996

[168] P. Van Roy, S. Haridi: "Concepts, Techniques, and Models of Computer Programming", MIT Press, 2004

[169] I. Sommerville: "Software Engineering", 6th Edition, Addison-Wesley, 2001

[170] R. S. Pressman: "Software Engineering: A Practitioner's Approach", 5th Edition, McGraw-Hill, 2001

[171] S. Schach: "Object-Oriented and Classical Software Engineering", Seventh Edition, McGraw-Hill, 2006

[172] I. Crnkovic, S. Larsson, M. Chaudron: "Component-based Development Process and Component Lifecycle", Journal of Computing and Information Technology (CIT), Vol. 13, No. 4, 2005, pp. 321-327

[173] V. Safonov: "Aspect.NET: Aspect-Oriented Programming for Microsoft.NET in Practice", .NET Developer's Journal, July 2005

[174] F. Berman, G. C. Fox, A. J. G. Hey (Editors): "Grid Computing: Making the Global Infrastructure a Reality", John Wiley & Sons, 2003

[175] O. Ezenwoye, S. M. Sadjadi, A. Cary, M. Robinson: "Grid Service Composition in BPEL for Scientific Applications", Lecture Notes in Computer Science, Vol. 4804, 2009, pp. 1304-1312

[176] B. A. Nardi: "A Small Matter of Programming: Perspectives on End User Computing", MIT Press

[177] J. Yu, B. Benatallah, F. Casati, F. Daniel: "Understanding Mashup Development", IEEE Internet Computing, Vol. 12, No. 5, September/October 2008, pp. 44-52

[178] L. Grammel, M.-A. Storey: "An End User Perspective on Mashup Makers", University of Victoria Technical Report DCS-324-IR, University of Victoria, Victoria, BC, Canada, September 2008

[179] A. Koschmider, V. Torres, V. Pelechano: “Elucidating the Mashup Hype: Definition, Challenges, Methodical Guide and Tools for Mashups”, Proceedings of the 2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web, Madrid, Spain, 2009

[180] L. L. Morgan: “To Define What A Mashup Is”, SD Times, April 2007, No. 172, pp. 33-34, World Wide Web, http://www.sdtimes.com/content/article.aspx?ArticleID=30483

[181] D. Merrill: “Mashups: The New Breed of Web App”, IBM, August 2006, World Wide Web, http://www.ibm.com/developerworks/xml/library/x-mashups.html

[182] J. Montgomery: “Microsoft Popfly: Building Games Without a CS Degree”, Microsoft Expression Newsletter, September 2008, World Wide Web, http://expression.microsoft.com/en-us/cc963994.aspx

[183] M. Muchmore: “Microsoft Popfly”, PC Magazine, October 2007, World Wide Web, http://www.pcmag.com/article2/0,2817,2210348,00.asp

[184] “Pipes: Rewire the Web”, Yahoo! Inc., World Wide Web, http://pipes.yahoo.com/pipes

[185] B. Dyszel: “Create No-Code Mashups with Yahoo! Pipes”, PC Magazine, October 2007, World Wide Web, http://www.pcmag.com/article2/0,2817,2193042,00.asp

[186] P. Wainewright: “Yahoo! Pipes Tutorial: Build an RSS Mashup”, ZDNet, World Wide Web, http://blogs.zdnet.com/SAAS/?p=280&tag=rbxccnbzd1

[187] R. Ennals, M. Garofalakis: “MashMaker: Mashups for the Masses”, Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, Beijing, China, 2007, pp. 1116-1118

[188] “Google Mashup Editor”, Google Inc., World Wide Web, http://code.google.com/gme

[189] “iGoogle Gadget Maker”, Google Inc., April 2007, World Wide Web, http://googlesystem.blogspot.com/2007/04/google-gadget-maker.html

[190] “QEDwiki”, IBM Corporation, World Wide Web, http://services.alphaworks.ibm.com/graduated/qedwiki.html

[191] C. Evans: “QEDWiki, IBM, and Situational Applications”, Zend Developer Zone, July 2006, World Wide Web, http://devzone.zend.com/article/678

[192] “OpenKapow”, Kapow Technologies, World Wide Web, http://www.openkapow.com

[193] “What is Proto”, Proto Software, World Wide Web, http://www.protosw.com

[194] “Dapper: FAQs”, Dapper.net, World Wide Web, http://www.dapper.net/faqs.php

[195] J. Wong, J. I. Hong: “Making Mashups with Marmite: Towards End-User Programming for the Web”, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, 2007, pp. 1435-1444

[196] “PRESTO: An Enterprise-Ready Mashup Solution”, JackBe Corporation, World Wide Web, http://www.jackbe.com/products

[197] J. Fujima, A. Lunzer, K. Hornbaek, Y. Tanaka: “Clip, Connect, Clone: Combining Application Elements to Build Custom Interfaces for Information Access”, Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology, Santa Fe, NM, USA, 2004, pp. 175-184

[198] M. Bolin, M. Webber, P. Rha, T. Wilson, R. C. Miller: “Automation and Customization of Rendered Web Pages”, Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology, Seattle, WA, USA, 2005, pp. 163-172

[199] M. Bolin, R. C. Miller: “Naming Page Elements in End-User Web Automation”, ACM SIGSOFT Software Engineering Notes, Vol. 3, No. 2, Proceedings of the First Workshop on End-User Software Engineering (WEUSE), March 2005, pp. 1-5

[200] J. Howell, C. Jackson, H. J. Wang, X. Fan: “MashupOS: Operating System Abstractions for Client Mashups”, Proceedings of the 11th USENIX Workshop on Hot Topics in Operating Systems, San Diego, CA, USA, May 2007

[201] H. J. Wang, X. Fan, J. Howell, C. Jackson: “Protection and Communication Abstractions for Web Browsers in MashupOS”, Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles, Stevenson, WA, USA, October 2007, pp. 1-16

[202] A. Repenning: “The Pragmatic Web: Customizable Web Applications”, Workshop on End User Development, Fort Lauderdale, FL, USA, April 2003, pp. 84-87

[203] D. Ayers: “The Shortest Path to the Future Web”, IEEE Internet Computing, Vol. 10, No. 6, November/December 2006, pp. 76-79

[204] M. P. Singh: “The Pragmatic Web”, IEEE Internet Computing, Vol. 6, No. 3, May/June 2002, pp. 4-5

[205] M. Bolin: “End-User Programming for the Web”, M.Sc. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, May 2005

[206] B. Patil, K. Maetzel, E. J. Neuhold: “Native End-User Languages: A Design Framework”, Proceedings of the 13th Workshop of the Psychology of Programming Interest Group (PPIG 2001), Bournemouth, UK, April 2001, pp. 113-126

[207] R. Smith: “SOA World Expo: Enterprise Mashup Services”, SOA World Magazine, October 2008

[208] B. Hayes: “Cloud Computing”, Communications of the ACM, Vol. 51, No. 7, July 2008, pp. 9-11

[209] J. Hendler, N. Shadbolt, W. Hall, T. Berners-Lee, D. Weitzner: “Web Science: An Interdisciplinary Approach to Understanding the Web”, Communications of the ACM, Vol. 51, No. 7, July 2008, pp. 60-69

[210] S. Srbljic, D. Skvorc, D. Skrobo: “Widget Oriented Consumer Programming”, Automatika, Vol. 50, No. 3-4, December 2009, pp. 252-264

[211] “Gadgets API”, Google Inc., 2009, World Wide Web, http://code.google.com/apis/gadgets

[212] “Konfabulator Reference Manual”, Yahoo! Inc., 2009, World Wide Web, http://manual.widgets.yahoo.com

[213] “Windows Live Gadget Developer’s Guide”, Microsoft Corporation, 2006, World Wide Web, http://dev.live.com/gadgets/sdk/docs/default.htm

[214] “Netvibes Developers Network”, Netvibes Ltd., 2008, World Wide Web, http://dev.netvibes.com/doc/start

[215] “IBM WebSphere Portal Is First to Make Google Gadgets Available to Millions of Corporate Portal Users”, IBM Press Room, World Wide Web, http://www-03.ibm.com/press/us/en/pressrelease/21158.wss

[216] “Enhancing Your Portal Using Google Gadgets with WebSphere Portal V6.0”, IBM developerWorks, World Wide Web, http://www.ibm.com/developerworks/websphere/library/techarticles/0705_schaeck/0705_schaeck.html

[217] “HTML 5 – A Vocabulary and Associated APIs for HTML and XHTML”, World Wide Web Consortium (W3C), October 2008, World Wide Web, http://dev.w3.org/html5/spec/spec.html

[218] "Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification", World Wide Web Consortium (W3C), September 2009, World Wide Web, http://www.w3.org/TR/CSS21/cover.html

[219] D. Flanagan: "JavaScript: The Definitive Guide", Fifth Edition, O'Reilly, 2006

[220] D. Crockford: "JavaScript: The Good Parts", O'Reilly, 2008

[221] T. Negrino, D. Smith: "JavaScript and Ajax for the Web", Sixth Edition, Peachpit Press, Berkeley, CA, USA, 2007

[222] "Apache Shindig", The Apache Software Foundation, World Wide Web, http://incubator.apache.org/shindig/index.html

[223] W. Maes, T. Heyman, L. Desmet, W. Joosen: "Browser Protection Against Cross-Site Request Forgery", Proceedings of the First ACM Workshop on Secure Execution of Untrusted Code, Chicago, IL, USA, 2009, pp. 3-10

[224] C. Karlof, U. Shankar, J. D. Tygar, D. Wagner: "Dynamic Pharming Attacks and Locked Same-Origin Policies for Web Browsers", Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 2007, pp. 58-71

[225] J. M. Wilson: "Gantt Charts: A Centenary Appreciation", European Journal of Operational Research, Vol. 149, No. 2, September 2003, pp. 430-437

[226] R. Rubinhoff: "How to Quantify the User Experience", Sitepoint, April 2004, World Wide Web, http://articles.sitepoint.com/article/quantify-user-experience

[227] P. H. Bloch, F. F. Brunel, T. J. Arnold: "Individual Differences in the Centrality of Visual Product Aesthetics: Concept and Measurement", Journal of Consumer Research, Vol. 29, No. 4, March 2003, pp. 551-565

[228] D. Ribbink, A. C. R. van Riel, V. Liljander, S. Streukens: "Comfort Your Online Customer: Quality, Trust and Loyalty on the Internet", Managing Service Quality, Vol. 14, No. 6, 2004, pp. 446-456

[229] S.-S. Liaw: "Understanding User Perceptions of World-Wide Web Environments", Journal of Computer Assisted Learning, Vol. 18, No. 2, June 2002, pp. 137-148

[230] H. Chen, R. Wigand, M. Nilan: "Optimal Experience of Web Activities", Computers in Human Behavior, Vol. 15, No. 5, September 1999, pp. 585-608

[231] I. Foster, M. Fidler, A. Roy, V. Sander, L. Winkler: “End-to-End Quality of Service for High-End Applications”, Computer Communications, Vol. 27, No. 14, September 2004, pp. 1375-1388

[232] R. Chakravorty, S. Kar, P. Farjami: “End-to-End Internet Quality of Service (QoS): An Overview of Issues, Architectures and Frameworks”, Proceedings of the ICIT 2000, December 2000

[233] A. Campbell, C. Aurrecoechea, L. Hauw: “A Review of QoS Architectures”, ACM Multimedia Systems Journal, Vol. 6, No. 3, May 1998, pp. 138-151

[234] T. Strandvall: “Eye Tracking in Human-Computer Interaction and Usability Research”, Lecture Notes in Computer Science, Vol. 5727, August 2009, pp. 936-937

[235] M. Nakayama, M. Katsukura: “Assessing Usability with Eye-Movement Frequency Analysis”, Proceedings of the 2008 Symposium on Eye Tracking Research & Applications, 2008, p. 77

[236] Y. Lin, W. J. Zhang: “Evaluating Interface Usability Based on Eye Movement and Hand Movement Behavioral Parameters”, The 47th Human Factors and Ergonomics Society Annual Meeting, Denver, CO, USA, October 2003, pp. 653-657

[237] F. P. Brooks, Jr: “The Mythical Man-Month: Essays on Software Engineering”, Anniversary Edition, Addison-Wesley, 1995

[238] D. Cleary: “Web-Based Development and Functional Size Measurement”, Proceedings of the IFPUG Annual Conference, San Diego, USA, 2000

[239] W. T. Tsai, Y. Chen, G. Bitter, D. Miron: “Methods for Software Sizing: How to Decide Which Method to Use”, Total Metrics, Camberwell, Victoria, Australia, August 2007

[240] G. C. Low, D. R. Jeffery: “Function Points in the Estimation and Evaluation of the Software Process”, IEEE Transactions on Software Engineering, Vol. 16, No. 1, January 1990, pp. 64-71

[241] R. Rask, P. Laamanen, K. Lyyttinen: “Simulation and Comparison of Albrecht’s Function Point and DeMarco’s Function Bang Metrics in a CASE Environment”, IEEE Transactions on Software Engineering, Vol. 19, No. 7, July 1993, pp. 661-671

[242] L. H. Putnam, W. Myers: “Measures for Excellence: Reliable Software on Time, Within Budget”, Prentice Hall, 1991

[243] B. W. Boehm, C. Abts, A. W. Brown, S. Chulani, B. K. Clark, E. Horowitz, R. Madachy, D. J. Reifer, B. Steece: “Software Cost Estimation with COCOMO II”, Prentice Hall, NJ, USA, 2000

[244] “Intel 64 and IA-32 Architectures Software Developer’s Manual Volume 2A & 2B: Instruction Set Reference”, Intel Corporation, March 2010

[245] “AMD64 Architecture Programmer’s Manual Volume 3: General-Purpose and System Instructions”, Advanced Micro Devices, November 2009

[246] “Java Platform Standard Edition 6 API Specification”, Sun Developer Network, World Wide Web, http://java.sun.com/javase/6/docs/api/overview-summary.html

[247] B. Abrams: “Number of Types in the .NET Framework”, MSDN Blog, World Wide Web, http://blogs.msdn.com/brada/archive/2008/03/17/number-of-types-in-the-net-framework.aspx

[248] A. van Deursen, P. Klint, J. Visser: “Domain-Specific Languages: An Annotated Bibliography”, ACM SIGPLAN Notices, Vol. 35, No. 6, June 2000, pp. 26-36

[249] A. van Deursen: “Domain-Specific Languages versus Object-Oriented Frameworks: A Financial Engineering Case Study”, Smalltalk and Java in Industry and Academia (STJA’97), 1997, pp. 35-39

[250] A. van Deursen, P. Klint: “Little Languages: Little Maintenance?”, Journal of Software Maintenance: Research and Practice, Vol. 10, No. 2, March-April 1998, pp. 75-92

[251] “Google Gadgets For Your Webpage”, Google Inc., World Wide Web, http://www.google.com/ig/directory?synd=open

[252] “iGoogle”, Google Inc., World Wide Web, http://www.google.com/ig

[253] “OpenSocial API Reference”, Google Inc., 2009, World Wide Web, http://code.google.com/apis/opensocial/docs/0.8/reference

[254] M. Rasalan: “Global Developer Population and Demographic Study 2009 v1”, Evans Data Report, April 2009

[255] F. Labelle: “Programming Language Usage Graph”, UC Berkeley, World Wide Web, http://www.eecs.berkeley.edu/~flab/languages.html

[256] “Programming Language Popularity”, DedaSys LLC Open Source Consulting, February 2009, World Wide Web, http://langpop.com

[257] "Internet Usage Statistics", Internet World Stats, December 2009, World Wide Web, http://www.internetworldstats.com/stats.htm

[258] J. Hendler, J. Golbeck: "Metcalfe's Law, Web 2.0, and the Semantic Web", Journal of Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 6, No. 1, February 2008, pp. 14-20

[259] S. Mukhopadhyay, S. Samaddar, S. Nargundkar: "Predicting Electronic Commerce Growth: An Integration of Diffusion and Neural Network Models", Journal of Electronic Commerce Research, Vol. 9, No. 4, November 2008, pp. 280-295

[260] A. Rai, T. Ravichandran, S. Samaddar: "How to Anticipate the Internet's Global Diffusion", Communications of the ACM, Vol. 41, No. 10, October 1998, pp. 97-106

[261] V. G. Cerf, M. P. Singh, et. al: "Internet Predictions", IEEE Internet Computing, Vol. 14, No. 1, January/February 2010, pp. 10-42

[262] J. Nielsen: "Participation Inequality: Encouraging More Users to Contribute", Useit.com Alertbox, World Wide Web, http://www.useit.com/alertbox/participation_inequality.html

[263] "The 90-9-1 Principle: How Users Participate in Social Communities", World Wide Web, http://www.90-9-1.com

[264] S. Srbljić: "Uvod u teoriju računarstva", Element, Zagreb, 2007

[265] K. C. Louden: "Programming Languages: Principles and Practice", Second Edition, Cengage Learning, 2003

[266] M. Felleisen: "On the Expressive Power of Programming Languages", Science of Computer Programming, Vol. 17, No. 1-3, December 1991, pp. 35-75

[267] R. E. Moe: "Expressive Modes and Species of Language", Proceedings of the 23rd World Academy of Science, Engineering and Technology, Vol. 23, November 2006, pp. 138-143

[268] S. McDirmid: "Turing Completeness Considered Harmful: Component Programming with a Simple Language", 2006, World Wide Web, http://lamp.epfl.ch/~mcdirmid/papers/mcdirmid06turing.pdf

16 Sažetak

Suvremeni oblici internetskih primjenskih programa, kao što su mrežni dnevnici, raspravna središta i društvene mreže, oslanjaju se na sklonost potrošača prema samostalnom stvaranju digitalnog sadržaja. Međutim, razvoj primjenskih programa, kao najkreativniji oblik primjene računala i informacijskih tehnologija, još uvijek je ograničen na razmjerno mali broj školovanih programera. U okviru ove doktorske disertacije, predložena je programska paradigma koja širokom krugu korisnika računala omogućava samostalno uključivanje u razvoj primjenskih programa. Istraženi su kognitivno i iskustveno bliski oblici predodžbe i postupci izgradnje primjenskih programa. Definirana je metodologija za ocjenu bliskosti razvojnog postupka umnom režimu potrošača te su izabrani elementi, tehnike i pravila za oblikovanje potrošaču prilagođene programske paradigme. Predložena je programska paradigma zasnovana na usložnjavanju udomljenika u kojoj se primjenski program gradi povezivanjem programskih komponenti izloženih putem grafičkog korisničkog sučelja prikazanog u web pregledniku. Izgradnja primjenskog programa sastoji se od izbora potrošaču prikladnih udomljenika za pribavljanje, obradu i prikaz informacija te njihovog povezivanja u radni tijek primjenom programirljivog udomljenika. Programski jezik za povezivanje komponenata u radni tijek prema zamislima potrošača zasnovan je na radnjama nad elementima grafičkog korisničkog sučelja udomljenika. Pokazano je da zbog kratkog vremena potrebnog za razvoj programa i velikog broja korisnika kojima je omogućeno sudjelovanje u njihovu razvoju, stvaralačke mogućnosti programske paradigme za usložnjavanje udomljenika mnogostruko premašuju stvaralačke mogućnosti uobičajenih oblika programskih paradigmi.

17 Summary

Consumer Programming

Contemporary Internet and Web applications, such as blogs, forums, and social networks, rely on consumers' aptitude to create and publish digital content by themselves. However, development of new applications, which is one of the most inventive forms of using computers and information technology, is still limited to relatively small community of professional programmers. In this dissertation, we propose a consumer programming methodology that enables wide population of computer consumers to actively participate in application development. We analyze cognitive and experience-based factors that impact human mental performance during the application development process. A methodology for evaluation of suitability of application design and development process to consumer knowledge and skills is defined. Based on this methodology, we selected elements and defined rules and techniques for consumer-oriented application development. We propose a programming paradigm based on widget composition, where an application is built out of widgets, i.e. small and handy web applications displayed in a web browser. To define an application, consumer selects a set of widgets for information retrieval, processing, and presentation, and composes them into a workflow through a consumer-programmable widget. Programming language used to compose widgets into consumer-defined workflows is based on GUI actions over widget GUI elements. Due to a short application development time and large number of users who can participate in software development, the community is empowered with enormous application development potential, which outperforms traditional programming paradigms for several orders of magnitude.

18 Ključne riječi Keywords

Ključne riječi

Programiranje prilagođeno potrošaču, kakvoća korisničkog doživljaja, poosobljavanje, programiranje radnog tijeka, usložnjavanje udomljenika, stvaralačke mogućnosti programske paradigme

Keywords

Consumer programming, quality of experience, personalization, workflow programming, widget composition, application development potential

19 Životopis

Dejan Škvorc rođen je 1979. godine u Čakovcu. Osnovnu školu završio je u Gornjem Mihaljevcu, a srednju elektrotehničku u Čakovcu. Diplomirao je 2003. godine na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu pod vodstvom prof.dr.sc. Siniše Srbljića upisom studija s naglaskom na znanstveno-istraživačkom radu i diplomskim radom "Sigurni prijenos podataka u mrežama s posredničkim sustavima". Na kraju studija primio je priznanje "Josip Lončar" kao jedan od najboljih studenata. Tijekom dviju završnih godina studija obnašao je funkciju demonstratora na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave. Magistrirao je 2006. godine na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu pod vodstvom prof.dr.sc. Siniše Srbljića magistarskim radom "Prividna mreža računalnih sustava zasnovanih na uslugama".

Od 2004. godine zaposlen je na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu u svojstvu zavodskog suradnika, a od 2005. godine u svojstvu znanstvenog novaka. Sudjelovao je na više istraživačkih projekata pod pokroviteljstvom Ministarstva znanosti, obrazovanja i športa Republike Hrvatske te domaće i strane industrije: na tehnologijskom projektu "*Okrilje posrednika javnog informacijskog sustava*" pod vodstvom prof.dr.sc. Siniše Srbljića u sklopu poliprojekta "*CROGrid*", znanstvenim projektima "*Raspodijeljeni ugrađeni računalni sustavi*" pod vodstvom akademika Lea Budina i "*Računalne okoline za sveprisutne raspodijeljene sustave*" pod vodstvom prof.dr.sc. Siniše Srbljića, projektu "*Middleware Architecture in New Generation Networks*" u suradnji s tvrtkom Ericsson Nikola Tesla d.d. Zagreb te projektima "*End-User Tool for Gadget Composition*" i "*Unified Translation*

Memory" u suradnji s tvrtkom Google Inc, Mountain View, Kalifornija, SAD i stručnim vodstvom prof.dr.sc. Siniše Srbljića. 2004. godine izabran je u naslovno zvanje mlađeg asistenta za područje tehničkih znanosti, polje Računarstvo, grupa predmeta Računarska tehnika. U okviru nastavnih aktivnosti na Zavodu, sudjeluje u organizaciji i provedbi seminara te auditornih i laboratorijskih vježbi iz više predmeta na preddiplomskom i diplomskom studiju računarstva. Istovremeno se uključuje u organizaciju i provedbu ljetnih radionica u tvrtci Ericsson Nikola Tesla d.d. 2004. i 2005. godine.

Obavio je više stručnih posjeta Sjedinjenim Američkim Državama i Kanadi gdje je u prestižnim sveučilišnim i industrijskim središtima University of California Berkeley, University of California Irvine, Santa Clara University, University of Southern California Los Angeles, University of Toronto, Google Inc. i Intel Corporation Inc. održao niz predavanja vezanih uz programirljive internetske okoline i potrošaču prilagođeno programiranje. Tijekom 2007. godine boravio je na četveromjesečnom znanstvenom usavršavanju u istraživačkom odjelu tvrtke Google Inc., Mountain View, Kalifonija, SAD u grupi *Google Gadgets* na projektu "*Inter-Gadget Communication*" pod vodstvom Adama Saha. Rezultate istraživačkog rada objavio je u nizu radova na međunarodnim konferencijama i u časopisima. Član je strukovne udruge *IEEE* i programskog odbora međunarodne konferencije *World Congress on Computer Science and Information Engineering*.

20 Biography

Dejan Škvorc was born in 1979 in Čakovec, Croatia. He finished the elementary school in Gornji Mihaljevec and technical high school in Čakovec. He received his B.Sc. degree in 2003 from School of Electrical Engineering and Computing, University of Zagreb, under the supervision of Professor Siniša Srbljić with diploma thesis "Secure Data Transfer in Mediator-Based Networks". During his studies, Dejan was enrolled in advanced study program with emphasis on research work and was awarded with "Josip Lončar" best student of the year certificate. During the last two years of his studies, he worked with the Department of Electronics, Microelectronics, Computer, and Intelligent Systems as a teaching demonstrator. He received his M.Sc. degree from School of Electrical Engineering and Computing, University of Zagreb, under the supervision of Professor Siniša Srbljić with master thesis "Virtual Network of Service-Oriented Computing Systems".

In 2004, Dejan joined the Department of Electronics, Microelectronics, Computer, and Intelligent Systems at the School of Electrical Engineering and Computing, University of Zagreb as a research assistant. During his research, he participated in several projects sponsored by Croatian Ministry of Science, Education and Sports, as well as domestic and foreign industry: technological project "*Wrapper of the Public Information System Mediator*" led by Professor Siniša Srbljić as part of the national "*CROGrid*" polyproject, scientific projects "*Distributed Embedded Computer Systems*" led by Professor Leo Budin and "*Computing Environments for Ubiquitous Distributed Systems*" led by Professor Siniša Srbljić, project "*Middleware Architecture in New Generation Networks*" in cooperation with Ericsson Nikola Tesla d.d. Zagreb led by Professor Siniša Srbljić and projects "*End-User*

Tool for Gadget Composition" and "*Unified Translation Memory*" in cooperation with Google Inc, Mountain View, California, USA led by Professor Siniša Srbljić. Since his employment at the Department of Electronics, Microelectronics, Computer, and Intelligent Systems in 2004, Dejan holds a teaching assistant position. His teaching activities include organization of exercises, seminars, and laboratories for several undergraduate courses on computer science study program. At the same time, he participated in organization of Ericsson Nikola Tesla Summer Camp in 2004 and 2005.

He made several visits to the USA and Canada, where he gave a series of technical talks related to programmable Internet environments and consumer programming at many eminent universities and companies: University of California Berkeley, University of California Irvine, Santa Clara University, University of Southern California Los Angeles, University of Toronto, Google Inc. and Intel Corporation Inc. In 2007, he spent four months in Google Inc., Mountain View, California, USA as a software engineering intern in *Google Gadgets* group, where he worked on "*Inter-Gadget Communication*" project under the supervision of Adam Sah. His research results are published in several conference and journal papers. He is a member of *IEEE* and program committee of the *World Congress on Computer Science and Information Engineering*.