Prilagodljivi model integracije upravljanja ugovaranjem usluga u bankarstvu s postojećim sustavom

Matejaš, Mladen

Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Mladen Matejaš

PRILAGODLJIVI MODEL INTEGRACIJE UPRAVLJANJA UGOVARANJEM USLUGA U BANKARSTVU S POSTOJEĆIM SUSTAVOM

DOKTORSKI RAD

Mentor:
Prof. dr. sc. Krešimir Fertalj

Zagreb, 2019.

University of Zagreb
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Mladen Matejaš

ADAPTIVE INTEGRATION MODEL FOR MANAGEMENT OF BANKING SERVICES CONTRACTS WITH A LEGACY SYSTEM

DOCTORAL THESIS

Supervisor:
Profesor Krešimir Fertalj, PhD

Zagreb, 2019.

Doktorski rad je izrađen na:

Sveučilištu u Zagrebu, Fakultet elektrotehnike i računarstva, Zavodu za primijenjeno računarstvo
i
Privrednoj banci Zagreb d. d., Sektoru za razvoj aplikacija

Mentor: Prof.dr.sc. Krešimir Fertalj

Doktorski rad ima: 207 stranica

Doktorski rad br.: _____

O mentoru:

Krešimir Fertalj je redoviti profesor na Zavodu za primijenjeno računarstvo Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu, gdje predaje nekoliko računarskih kolegija na diplomskom, poslijediplomskom specijalističkom i poslijediplomskom doktorskom studiju. Diplomirao je, magistrirao i stekao doktorat iz Računarstva na istom fakultetu. Njegov stručni i znanstveni interes je u području automatiziranog programskog inženjerstva, složenih informacijskih sustava i upravljanja projektima te sigurnosti programske podrške. Vodio je više znanstvenih i istraživačkih te nekoliko desetaka razvojnih projekata. Objavio je više od 170 znanstvenih i stručnih radova. Bio je mentor na više od 200 završnih i diplomskih radova, devet magistarskih radova te devet doktorskih disertacija i jednom specijalističkom završnom radu.

Prof. Fertalj je osnivač i voditelj Laboratorija za informacijske sustave posebne namjene. Član je profesionalne udruge IEEE i suradnik Akademije tehničkih znanosti Hrvatske (HATZ). Bio je predstojnik Zavoda za primijenjeno računarstvo, tajnik Odjela informacijskih sustava HATZ te jedan od osnivača i član upravnog odbora ogranka udruge PMI u Hrvatskoj.

About the Supervisor:

Krešimir Fertalj is a full professor at the Department of Applied Computing at the Faculty of Electrical Engineering and Computing, University of Zagreb, where he lectures a couple of computing courses on graduate, specialist and doctoral studies. He graduated and received his PhD degree in Computing at the same Faculty. Hs professional and scientific interest is in computer-aided software engineering, complex information systems, project management and in software security. He led several scientific and research and a few dozen of development projects with the industry. He has published more than 170 scientific and technical papers. He was a mentor to students for over 200 bachelor and graduate theses, nine MSc and nine PhD theses.

Prof. Fertalj is a is the founder of the Laboratory for Special Purpose Information Systems at FER. He is a senior member of IEEE and associate of Croatian Academy of Engineering (HATZ). He served as a Head of Department at FER, a Secretary of Department of Information Systems of HATZ and was one of the founders and member of management board of PMI chapter in Croatia.

Najiskrenije zahvaljujem mentorima, profesoru Damiru Kalpiću na podršci tijekom prve dvije godine studija, te profesoru Krešimiru Fertalju na iscrpnom strpljenju, trudu i savjetima datim tijekom izrade doktorskog rada te inspiraciji kroz ostatak studija. Zahvaljujem svome ocu, majci i bratu, djevojci koja je u međuvremenu postala suprugom, te ostaloj rodbini i prijateljima za svu ljubav i podršku. Također želim zahvaliti kolegama iz sektora za razvoj aplikacija Privredne Banke Zagreb koji su oduvijek bili nadahnuće i podrška.

Kratki sažetak

Različita okruženja za upravljanje poslovnim procesima danas postaju osnovni alati za kontrolu poslovnih procesa. Automatizacija izvršavanja poslovnih procesa i definiranje razumljivog procesnog pogleda na funkcioniranje tvrtke prerastaju u ključne faktore za donošenje uspješnih poslovnih odluka. Moderna okruženja za upravljanje poslovnim procesima omogućuju analizu toka poslovnih procesa, otkrivanje uskih grla i nepravilnosti u procesima, te pružaju širok spektar naprednih mogućnosti za optimizaciju pojedinih aktivnosti i cjelokupnog toka poslovnih procesa.

Ključni problem uvođenja sustava za upravljanje poslovnim procesima u postojeću okolinu organizacije je složenost i nestandardiziranost integracije sustava za upravljanje poslovnim procesima s postojećim (naslijeđenim) sustavima koji izvršavaju postojeće poslovne procese.

Niz teoretskih rasprava iz znanstvene literature te praktičnih studija iz industrije fokus integracije stavlja na redizajn naslijeđenih sustava, uz minimalno iskorištavanje postojećih funkcionalnosti i oslanjanje na postojeće strukture. Postojeće aplikacije se restrukturiraju, razdvoje i prepišu kao servisni moduli, koji se zatim koriste kod izgradnje rješenja za upravljanje poslovnim procesima. Kako proces redizajniranja složenih naslijeđenih okruženja zahtijeva velike količine vremena i resursa, to je luksuz koji si mnoge organizacije ne mogu priuštiti.

Osim složenosti integracije, dodatan problem predstavlja činjenica da je rukovanje s poslovnim podacima usko vezano uz okruženje za upravljanje poslovnim procesima, te uvelike ovisi o mogućnostima tog okruženja. Primarna zadaća okruženja za upravljanje poslovnim procesima je upravljanje tokom poslovnih procesa, pa ih je često potrebno koristiti u kombinaciji s dodatnim alatima i radnim okvirima kako bi se zadovoljile napredne poslovne potrebe (primjerice, izgradnja složenih i interaktivnih korisničkih sučelja).

Vidljiv je nedostatak detaljnih modela koji opisuju integraciju sustava za upravljanje poslovnim procesima s naslijeđenim sustavima, a samo mali broj studija bavi se analizom adekvatnih alata i implementacijskih metodologija koje bi omogućile opisanu integraciju. Posljedično, uvođenje sustava za upravljanje poslovnim procesima u organizaciju često se odradi tako da se dodatno zakompliciraju postojeći sustavi, a time i samo poslovanje organizacije.

S obzirom na navedene poteškoće ovaj doktorski rad detaljno istražuje problemsko područje te definira novi model integracije sustava za upravljanje poslovnim procesima u naslijeđene

okoline. Osnovni cilj novog modela je da se temelji na postojećim strukturama te iskorištava postojeće funkcionalnosti.

Doktorski rad opisuje razvoj te daje pregled područja upravljanja poslovnim procesima. U detalje se istražuje veza između poslovnih procesa i funkcionalnosti sustava koji ih podupiru. Analiziraju se naslijeđeni sustavi temeljeni na uslužno usmjerenoj arhitekturi te se istražuje na koji način bi se postojeće funkcionalnosti mogle koristiti u sustavu za upravljanje poslovnim procesima. Na primjerima iz prakse opsuju se izazovi i poteškoće koje nosi proces implementacije sustava za upravljanje poslovnim procesima, temeljem čega se definiraju karakteristike i ideja novog, vlastitog, modela integracije.

Nakon detaljne teorijske razrade te formalne definicije vlastitog modela integracije slijedi sveobuhvatna razrada modela s tehničkog aspekta koja uključuje identifikaciju tehnologija, alata te radnih okvira koji se mogu koristiti u vlastitom modelu. Izložen je prijedlog strukture osnovnih komponenti modela uz opise principa tehničke realizacije. Izložena je tehnička specifikacija radnog okvira za implementaciju vlastitog modela u postojeći sustav. Definirani su postupci integracije na arhitekturnoj i procesno-podatkovnoj razini. Arhitekturna razina obuhvaća tehničku specifikaciju programskih komponenti radnog okvira te odabir tehnologija i programskih alata. Prikazuje postupke izgradnje novih komponenti te način na koji se iskorištavaju postojeće komponente. Definira komunikaciju te opisuje principe povezivanja aplikacija za upravljanje poslovnim procesima (BPMA) s postojećim komponentama, resursima i aplikacijama. Procesno-podatkovna razina sastoji se od iterativnog postupka otkrivanja postojećih poslovnih procesa, modeliranja tokova procesa i njihove integracije s BPMA uz definiranje korisničke interakcije kombiniranjem jednostavnih sučelja BPMA i složenih korisničkih ekrana naslijeđenih sustava.

Temeljem predloženog modela integracije, razrađene tehničke specifikacije i postupaka radnog okvira, izgrađen je prototip rješenja koji demonstrira funkcioniranje cjelokupnog sustava na primjeru složenog poslovnog procesa iz domene bankarstva. Na poslovnom primjeru primijenjen je vlastiti postupak iterativne izgradnje te integracije procesne aplikacije s naslijeđenim sustavima. Analizirani su razni tehnički koncepti integracije, u detalje razrađeni postupci komunikacije svih uključenih dijelova te su opisani principi izrade korisničkih sučelja.

Opisan je vlastiti postupak praćenja, mjerenja i analize uspješnosti poslovnih procesa u sustavu temeljenom na predloženom modelu integracije. Postupak se temelji na povezivanju metrika definiranih u sklopu BPM rješenja s metrikama definiranim na razini naslijeđenih aplikacija.

Na širom skupu ispitanika provedeno je sveobuhvatno ispitivanje mišljenja o ideji koja stoji iza predloženog modela integracije. Dodatno, na užem uzorku stručnjaka iz interesnog područja, provedena je detaljna evaluacija scenarija uporabe predloženog vlastitog modela, koja je potvrdila korisnost modela.

Ključne riječi

Upravljanje poslovnim procesima, BPM alati, modeliranje poslovnih procesa, naslijeđeni sustavi, SOA, integracija sustava, zaprimanje zahtjeva, obrada zahtjeva

Extended abstract

ADAPTIVE INTEGRATION MODEL FOR MANAGEMENT OF BANKING SERVICES CONTRACTS WITH A LEGACY SYSTEM

By closely observing complex business environments, one can identify business processes as its core elements. A business process can be defined as a sequential flow of business activities over a certain period. The performance of an organization depends upon business processes which provide the means for achieving the organization's fundamental objectives.

In a volatile modern economy, organizations must understand and constantly improve their business processes to make successful and competitive business decisions. The relevance of information technology and different business process management (BPM) methodologies are becoming key factors in controlling vast business environments. The process-oriented view of business environment provides a better understanding of the organization and helps in comprehending the needs of the information systems that support the business process. BPM provides support for the efficient management of a business environment with the purpose of increasing its flexibility and productivity. BPM implementation is a process of building a BPM solution in an existing business environment. BPM solution consists of multiple BPM applications (BPMA) which support different business processes in the organization.

A set of modern BPM tools (suites, systems, or environments) offer the ability to model and analyze business process flows, detect defects and irregularities in processes and provide a wide range of advanced capabilities for business process optimization. BPM tools are mainly used to develop, maintain and optimize a composite process application.

The key problem with the implementation of BPM in existing environments is the complexity and lack of standardized methods for the integration of the BPM solution with the existing business processes, legacy applications and systems. A series of theoretical discussions, found in scientific literature and practical studies from the industry, put the focus of the integration on the redesign of legacy systems, with little exploitation of existing functionalities and minimal reliance on existing structures. Existing applications are restructured and written as service modules, which are then used to build BPM solutions. The redesign of complex legacy environments requires large amounts of resources and time, and therefore, is a luxury that many organizations can not afford. Also, the organization's ability to grow, respond to market demands and cope with ongoing challenges would be limited during the redesign period. Consequently, business owners rarely approve such a task.

Apart from the complexity of the integration, an additional problem is the fact that presentation and interaction with business data is closely related to BPM suite and largely depends on the capabilities of that suite. E.g., the BPM implementation effort could encounter problems if the used tool does not support the implementation of complex business rules or the construction of interactive user interfaces. As the primary task of BPM suite is to manage business process flows, they often need to be used in combination with additional tools and frameworks to achieve more advanced business needs.

In addition, the implemented BPM tool must be able to handle large volumes of business data and massive objects, which complicate the implementation process. Complete business logic required for the effective presentation of data must be included when creating complex forms. Such a cumbersome architecture can create quite a vulnerable application, which is difficult to maintain or upgrade efficiently. Furthermore, a business arranged in that manner is closely dependent on the BPM tool itself.

There is still a lack of detailed models that describe the integration of a BPM system with existing legacy environments, and there are only few studies dealing with the analysis of the appropriate tools and implementation methodologies that would enable the described integration. As a result, the introduction of BPM systems in the organization is often done in a way that further complicates the existing environment and creates additional difficulties for the organization's business.

In light of the above difficulties, this doctoral work aims to explore the interest areas in detail and develop a flexible model for integrating a BPM solution into an existing legacy environment by utilizing existing structures and exploiting existing functionalities. The usage of existing business activities, supported by the functionalities of the legacy applications, removes the need for extensive redesigns of the existing environment. By applying minimal changes to the existing legacy environment, organizations can, to the satisfaction of the business owners, minimize the time and resources needed for BPM implementation while enjoying the benefits provided by BPM. The focus of the work are legacy environments based on service-oriented architecture (SOA), i.e. environments comprising legacy applications based on an SOA architectural style, where a solid service background is present. Solid service backgrounds enable the flexibility of legacy applications and constitute a good foundation for the endeavor of BPM integration.

The work also describes the evolution and gives an overview of BPM domain. The relationship between business processes and the functionality of a system that supports the actions of a business process is explored in detail. Legacy systems based on service-oriented

architecture are analyzed with the focus on inherited functionalities, and the usage of those functionalities in conjunction with a BPM system. Challenges and difficulties, involved in the implementation and the integration of a BPM system in a legacy environment, are analyzed on a real-life example. BPM suites are discussed, their importance is outlined, and guidelines for choosing the most appropriate suite are given.

Further on, the work analyses existing models for integrating BPM systems with legacy environments, and presents a study of used architectures, tools and technologies. An overview of the existing research papers from the area of interest is presented, with a conclusion that they were written quite general and lack detailed elaboration of the integration process. Existing papers mainly emphasize the importance of controlling business processes, analyze the success factors of the BPM initiatives and try to determine the role of the BPM in the overall business of the organization. The discussed BPM implementation models are mainly focused on building a BPM solutions from scratch, or overwriting existing functionalities and redesigning existing legacy application environments. Reuse is limited to existing SOA services, while the concept of using parts of existing applications, user interfaces and complete functionalities incorporated in existing business processes, is not thoroughly studied. In the literature authors often discus the question: When, how, and to what extent can one use existing systems as part of a BPM solution? The lack of guidelines and models that would describe the structure and the implementation of such a BPM solution is also underlined.

Disadvantages of the existing models were also noted on a study of a real-life candidate for a BPM implementation, a complex banking application. Based on the discussion, the idea for a new, improved, integration model is presented in detail. It includes the use of rudimentary business process management functionalities, provided by the BPM tool, with a combination of a specific solutions and interfaces implemented in existing applications. On a process level, all activities of the business process are executed trough the legacy applications, while the created BPM application holds the basic business process data and provides a central point of interaction with the activities of the process.

Basic characteristics of the new, flexible, model for the integration of BPM systems with existing service-oriented legacy applications are also defined. The main characteristic of the new model prevents the extensive redesign and complete re-writing of existing functionalities and business processes. Therefore, the new model maximizes the utilization of existing functionalities, existing applications and architecture while requiring as little adaptation and change as possible. Consequently, it reduces the cost of BPM implementation, use of resources and the amount of work needed. Adaptability of the new model provides a loose coupling

between a BPMA and the existing systems which enables the operation of legacy applications without the BPM solution. In that manner, the business of the organization is not dependent on the BPM solution and one can still execute existing business processes through legacy applications. The central point of interaction with the activities of the process is achieved which sets the foundation for easier tracking, measurement and optimization of individual parts of the process. Every business process flow on the BPM environment reflects the state of the underlying data in the background systems in real-time. Changes in business processes must be visible in the underlying applications automatically, and the state of the data in those applications needs to be promptly displayed in the process application. A procedure for a simple integration and use of various legacy applications in the BPM solution was proposed. Due to simplicity, a BPMA manipulates with only a reduced set of business data required to achieve the desired functionality. The new model allows changes in business process flow by changing the process model in BPMA with minimal, or no, changes to existing applications. Model is also independent of different hardware, operating systems, or technologies used in the organization. The advantages of a new model over the existing integration attempts are discussed in detail.

A more detailed theoretical elaboration and formal definition of the integration model is also given. The importance and the role of the models' layers is explained, communication between the layers is elaborated. The integration model is based on three key layers: the legacy application layer, BPM communication layer (BPMCL) and a BPMA built in the BPM tool. Legacy applications contain parts of the functionalities of existing business processes and execute the actions of the existing business processes. To enable their communication with the BPMA, a Legacy Application Adapter (LAA) component is created. Integrated into the legacy application, the LAA has the task of sending instructions to the BPMA and responding to the requests of the BPMA with the help of the BPMCL. BPMCL is a component, a middleware layer, which enables the communications between the legacy systems that perform specific business actions and the BPMA. It contains the methods for manipulating process data by calling process application programming interfaces (APIs) exposed in the BPM environment. BPMCL gives flexibility to the integration model by providing means for the orchestration of business processes built in a legacy environment using a generic BPM suite. BPMA describes the existing business process with a BPD modeled in congruence with the Process Model and Notation (BPMN) standard. A BPMA is built in a BPM environment with the use of BPM tools. The process of BPMA creation requires that flow charts of business processes and sub-processes be defined. A BPMA created in this manner contains a process model of an existing

business process that is executed in a legacy environment with the help of several existing legacy applications. Further on, the role, functionality and capabilities of each layer are shown and explained in detail. In addition, the question of user roles, security and access control mechanisms is also considered.

To clarify the communication between the components in the new model the terms "process points" and "interaction points" were defined. Process points mark the locations where the LAA and the BPMA exchange information, while interaction points identify parts of the process where the user begins/ends the interaction with the process in the legacy applications.

Additionally, discussion regarding similarities and differences of the integration model with Enterprise Service Bus (ESB) architectures, Enterprise Application Integration (EAI) principles and the role of SOA is undertaken.

The work presents a detailed elaboration of the proposed integration model from the technical side, with the goal of identifying technologies, tools and frameworks that will be used in the design of the proposed model. Ideas on different technical realizations of individual layers are presented. According to the defined requirements, ease of implementation and overall efficiency, appropriate tools, technologies and implementation methods are proposed. Detailed technical description of the framework that enables the implementation of the described model and its integration with the existing legacy environment is presented. Guidelines for modeling process flows, connecting the processes with service modules and legacy applications and creating user interfaces are given. An original, iterative, approach for the development of a BPM application and the integration of the BPM application with the existing systems, according to the proposed model, is defined and elaborated. Also, some experience based conclusions and guidelines regarding the implementation of the model and its components are given.

The application of the proposed integration model in a real-life use case scenario is also shown. The process of selecting most favorable tools, frameworks and methods for the implementation effort of the integration model is outlined. A specific business use case scenario (the business process of issuing and processing loan utilization requests) is chosen to illustrate the feasibility of the integration and demonstrate the structural innovations introduced into the mentioned process. The results of the development, and the integration of the BPM application with existing systems, are presented based on the original iterative approach that was defined previously. Additionally, the introduced structural innovations and process improvements are elaborated. A brief discussion regarding the definition of the visual identity of BPM application was also undertaken.

Further on the principles of business process monitoring, analysis and the techniques for determining process measures, in the environment with the proposed integration model, are briefly described. An original model for measuring and mapping the metrics of the lower level (simple services and human actions in the legacy applications) to the metrics of activities, business processes and business goals is proposed. A direct connection between the low-level metrics and the strategic business goals of the organization is shown. Optimization recommendations are given based on detected shortcomings in the process flows. This sets the foundations for the performance optimization of business processes supported by the proposed integration model and opens a wide area for a future research.

A detailed evaluation of the proposed integration model, along with the validation of the idea on which the model is based is also provided. The evaluation was done in two separate studies. The first study involved a broader set of respondents and was aimed at the validation of the idea behind the proposed integration model. Views about the importance of BPM are collected, along with the opinions on the methods for integrating a BPM system in a legacy environment. The main goal of the first study is to answer whether, and to which dosage, one should rely on existing legacy systems during the implementation of a BPM solution. In the second study, the proposed integration model was presented to a narrower set of relevant experts from the BPM area, and they were asked to give their opinions, criticisms, suggestions and identify possible use cases for the integration model. All interviewed experts were familiar with the BPM but differed in the amount of experience with different BPM initiatives. A Delphi method based detailed evaluation of the usefulness of the proposed integration model was undertaken. Further on, comments on potential disadvantages of the proposed integration model, as concluded in the analysis with experts, were discussed. Additionally, a review of the state and the needs of business environments in which the interviewed experts operate are given.

At the end, a summary of the conducted research, critical review of the research results, guidelines and suggestions for further development are provided.

This doctoral work gives a review of the BPM research areas with an emphasis on the integration of the BPM solution with legacy environments. Special attention is given to modern BPM suites used to model business process flows and their ability to integrate with existing applications and reuse existing functionalities. A model which allows the integration of a BPM suite with the legacy environment was developed, and framework for the implementation of the described model and its integration with the existing systems is proposed.

This research explores how important it is for organizations to control their business processes. Furthermore, it shows how control of business processes can be achieved by integrating BPM into existing SOA based legacy environments.

The original contribution of this work can be seen in the definition of the BPMCL layer and the principles of integrating the BPMA with different legacy applications via the BPMCL in a way that exploits the existing functionalities and user interfaces while avoiding extensive redesigns of existing environments. Additional contributions lie in the conclusions of the presented case study, proof-of-concept implementation study and the extensive evaluation of the model by relevant experts from the BPM area.

The presented research may provide organizations with the opportunity to use the proposed integration model as a platform for achieving greater business value and thereby help them to become more flexible and more focused on their core business objectives.

Keywords

Business process management, BPM suite, business process modelling, legacy systems, SOA, system integration, request issuing, request processing

Sadržaj

1 UVOD

Posljednjih nekoliko godina raste važnost upravljanja poslovnim procesima kao izravan rezultat učestalih promjena i povećane dinamičnosti poslovnih okolina u kojima suvremene organizacije djeluju. Usprkos tome, integracija sustava za upravljanje poslovnim procesima u postojeće okoline organizacija i dalje je ogroman praktičan izazov zbog opsežnih organizacijskih i tehnoloških promjena koje je potrebno poduzeti. Nedostatak adekvatnih modela i postupaka integracije otežava implementaciju, povećava utrošak resursa i rezultira povećim promjenama aplikacijske infrastrukture organizacija. Dugotrajne i skupe implementacije, bez konkretnih rezultata, nerijetko uzrokuju propašću ili odustajanjem od inicijative uvođenja sustava za kontrolu poslovnih procesa.

1.1 Motivacija

Poslovni procesi (PP, engl. Business Processes) imaju ključnu ulogu u nastojanju privrednih subjekata (organizacija) da ostvare svoje poslovne ciljeve pa postoji potreba za detaljnim uvidom i kontrolom njihovih tokova.

Postojeći PP u pravilu uključuju više različitih sustava što stvara probleme kod njihove kontrole, koordinacije i izvršavanja aktivnosti, te smanjuje mogućnosti optimizacije procesa. Osim toga poslovni procesi su redovito složeni, isprepleteni i međuovisni što dodatno otežava njihovu analizu i praćenje.

Dosadašnji pokušaji integracije sustava za upravljanje poslovnim procesima s postojećim okolinama primarno su usredotočeni na redizajn postojećih okolina što je često problematično te zahtijeva mnogo vremena i resursa. Prijedlozi integracije ne daju adekvatan pregled ni konceptualni model integracije, te uglavnom nisu dovoljno razrađeni za provedbu integracije.

Takva situacija predstavlja motivaciju za poboljšanjem postojećeg stanja dizajniranjem i uvođenjem novog, prilagodljivog i efikasnog sustava kontrole poslovnih procesa. S obzirom na navedeno potreban je jedinstveni model integracije takvog sustava koji ima mogućnost praćenja procesa u postojećim okolinama bez potrebe značajnog preoblikovanja postojećih sustava.

Upravo takav jednostavan i funkcionalan model predložen je u ovom radu. Pažnja je usmjerena na funkcionalnosti postojećih aplikacija i postojeću arhitekturu nad kojom je potrebno napraviti samo nužne izmjene. Prilagodljivost i jednostavna međuovisnost sustava za upravljanje poslovnim procesima i postojeće arhitekture osigurava integraciju u razumnom

roku bez potrebe za cjelokupnim preoblikovanjem procesa organizacije te bez štete za poslovanje. Postojeće aplikacije i arhitektura mogu se koristiti neovisno o sustavu za upravljanje poslovnim procesima čak i u slučaju kada taj sustav nije dostupan.

1.2 Znanstveni doprinosi

Cilj ovog doktorskog rada je analizirati izvedivost integracije sustava za upravljanje poslovnim procesima (engl. Business Process Management - BPM) u postojeću okolinu aplikacija temeljenih na uslužno usmjerenoj arhitekturi (engl. Service Oriented Architecture - SOA).

Istraživanje provedeno u okviru ovog rada prikazuje prilagodljiv model koji opisuje izgradnju aplikacije za upravljanje poslovnim procesima (Business Process Management Application - BPMA) i integraciju BPMA s postojećim sustavima temeljenim na konceptima SOA. BPMA je aplikacija razvijena s ciljem da podrži određeni PP koji podržavaju postojeće aplikacije. Kombinacija više BPMA-a pruža podršku za više PP-a unutar organizacije i čini BPM rješenje.

Nadalje, istraživanje navodi preporuke radnog okvira za iterativnu integraciju opisanih sustava, te opisuje prototip rješenja na konkretnom primjeru iz bankarskog poslovanja.

U ovom radu predstavljeni su sljedeći znanstveni doprinosi:

- Definicija vlastitog prilagodljivog modela integracije sustava za upravljanje poslovnim procesima u uslužno orijentiranu arhitekturu u svrhu korištenja u kompleksnom bankarskom sustavu ugovaranja usluga poslovnih korisnika.
- Specifikacija radnog okvira koji omogućuje implementaciju opisanog modela te njegovu integraciju s postojećim sustavom, od modeliranja toka procesa, povezivanja s postojećim servisnim modulima, resursima i aplikacijama do definiranja prikaza i sučelja za korisničku interakciju.
- Programski podržana strukturna inovacija i racionalizacija PP-a zadavanja zahtjeva od strane klijenata te obrade i realizacije od strane davatelja usluga.

1.3 Struktura rada

U drugom poglavlju dan je sveobuhvatan pregled i razvoj interesnih područja. Objašnjavaju se osnovni koncepti modeliranja poslovnih procesa (MPP, engl. Business Process Modeling) standardi za modeliranje procesa i ostali važniji pojmovi. Definira se važnost korištenja BPM

alata ili okruženja za upravljanje poslovnim procesima (eng. Business Process Management Systems, Business Process Management Suites ili Business Process Management Software, skraćeno BPMS) u izgradnji BPM rješenja, te su dane smjernice za odabir najprikladnijeg okruženja. Prikazana je važnost upravljanja poslovnim procesima te su objašnjene sličnosti, razlike te međuovisnost SOA-e i BPM-a.

Treće poglavlje sadrži analizu postojećih modela integracije sustava za upravljanje poslovnim procesima u postojeća aplikacijska okruženja, prikaz korištenih arhitektura, alata i tehnologija. Analizirani su dosadašnji istraživački radovi u području te je ustanovljeno da su napisani općenito, bez detaljne razrade problematike. Uočeni su i obrazloženi nedostaci postojećih, klasičnih modela te je predočena analiza praktičnog primjera na kome ti nedostatci dolaze do izražaja. Predstavljena je ideja novog prilagodljivog modela integracije sustava za upravljanje poslovnim procesima u okolinu servisno orijentiranih naslijeđenih aplikacija (u nastavku teksta: novi ili vlastiti model integracije). Prikazana je pojednostavljena skica arhitekture sustava temeljenog na novom modelu, te su definirane karakteristike koje model treba zadovoljiti. Skiciran je tok procesa u sustavu baziranom na novom modelu temeljem konkretnog poslovnog primjera. Navedene su prednosti novog modela u odnosu na postojeće klasične modele iz literature.

Detaljnija teorijska razrada te formalna definicija modela integracije prikazana je u četvrtom poglavlju. Prezentirana je uloga i struktura pojedinih komponenti modela te opisana njihova međusobna interakcija. Posebna pažnja posvećena je prilagodljivosti, jednostavnoj međuovisnosti BPM rješenja i postojećih sustava te proširljivosti i prilagodljivosti. Osigurana je sinkronost stanja u BPM rješenju i postojećim aplikacijama te je razmotrena mogućnost rada poslovnog sustava bez BPM rješenja. Definirani su principi i točke komunikacije između BPM rješenja i naslijeđenih sustava u vlastitom modelu te način korisničke interakcije. Dodatno je razrađeno pitanje korisničkih uloga, sigurnosti i kontrole pristupa. Poglavlje sadrži i raspravu o sličnostima i razlikama modela integracije s ESB arhitekturama (engl. Enterprise Service Bus - ESB), principima integracije aplikacija poduzeća (engl. Enterprise Application Integration - EAI), te definira ulogu SOA-e u modelu.

U petom poglavlju detaljno se razrađuje predloženi model integracije s tehničke strane, a cilj je identificirati tehnologije, alate te radne okvire koje je preporučljivo koristiti u izradi predloženog modela. Iznesene su ideje različitih tehničkih realizacija pojedinih komponenti, odabrane adekvatne tehnologije i metode s obzirom na postavljene zahtjeve, lakoću implementacije i sveukupnu učinkovitost. Navedena je detaljna tehnička specifikacija postupaka radnog okvira koji omogućava implementaciju opisanog modela u postojeći sustav.

Opisan je postupak integracije na arhitekturnoj razini, izgradnjom svih potrebnih komponenti, te integracija na procesno-podatkovnoj razini. Objašnjeno je modeliranje procesnih tokova, povezivanje procesa sa servisnim modulima i naslijeđenim aplikacijama te definiranje prikaza podataka i korisničkih sučelja. Definiran je i razrađen originalni iterativni pristup razvoja BPM rješenja i integracije tog rješenja s postojećim sustavima prema predloženom modelu.

Šesto poglavlje prikazuje primjenu modela integracije na konkretnom poslovnom primjeru zadavanja i obrade zahtjeva za korištenje kredita u svrhu strukturne racionalizacije navedenog procesa. Prikazani su rezultati razvoja/integracije BPMA temeljem iteracija definiranih u prethodnom poglavlju, te su opisane ostvarene strukturne inovacije i poboljšanja procesa. Dan je i kratak osvrt na definiranje prepoznatljivog dizajna korisničkih sučelja BPM rješenja u skladu sa željama pojedine organizacije.

Sedmo poglavlje ukratko opisuje principe praćenja i analize PP-a, te načine određivanja mjera u okolini s predloženim modelom integracije. Izložen je vlastiti postupak preslikavanja metrika na najnižoj razini modela (metrike servisnih poziva te ljudskih interakcija unutar naslijeđenih aplikacija) u metrike aktivnosti i PP-a u svrhu provjere ostvarenja poslovnih ciljeva organizacije. Tako je ostvarena veza između metrika najniže razine te strateških ciljeva organizacije, a dane su i preporuke za optimizaciju PP-a u slučaju da određeni poslovni cilj nije realiziran. Ovo poglavlje postavlja temelj za optimizaciju PP-a podržanih predloženim modelom integracije i otvara široko područje za buduće istraživanje.

Osmo poglavlje donosi detaljnu evaluaciju predloženog modela integracije, te opravdava ideju na kojoj je model zasnovan. Evaluacija je odrađena u dva dijela. U prvom dijelu na širem skupu ispitanika provodi se sveobuhvatno ispitivanje mišljenja o ideji koja stoji iza predloženog modela integracije. Skupljaju se mišljenja o važnosti BPM-a, viđenja kako bi se BPM trebao implementirati u organizaciji te da li bi se, i u kojoj mjeri, trebao oslanjati na postojeće naslijeđene sustave. U drugom dijelu, predloženi model integracije predstavljen je užem skupu relevantnih stručnjaka iz BPM područja koji su dali svoja mišljenja, kritike i prijedloge. Zatim je provedena detaljna evaluacija korisnosti predloženog modela integracije. Dani su komentari potencijalnih nedostataka modela proizašlih iz analize sa stručnjacima. Dodatno je prikazan osvrt na stanje i potrebe poslovnih okruženja u kojima ispitani stručnjaci djeluju. Kroz istraživanje i ispitivanje iskustava ispitanih stručnjaka identificirani su neki od osnovnih razloga propasti BPM inicijativa, te su ukazane karakteristike predloženog modela integracije koje bi mogle pomoći u tim slučajevima.

Deveto poglavlje donosi sažetak napravljenog istraživanja, kritički osvrt na rezultate istraživanja, te postavlja smjernice i prijedlog mogućeg daljnjeg razvoja.

1.4 Izjava o zaštiti intelektualnog vlasništva i povjerljivih podataka

Model integracije sustava za upravljanje poslovnim procesima s postojećom računalnom okolinom naslijeđenih aplikacija, koji je razrađen u dijelovima poglavlja 3.2.1, 4.4.1.1 i 6 na primjeru naslijeđene okoline Privredne Banke Zagreb d.o.o. (PBZ).

U svrhu zaštite intelektualnog vlasništva PBZ-a, maske za unos podataka i ekrani naslijeđenih aplikacija, detalji njihove implementacije te njihova koncizna struktura nisu prikazani u ovom radu. Iznimku predstavljaju ekrani i forme PBZ aplikacija dostupnih javnosti, npr. internetskog bankarstva, te prezentirani rezultati analize kompleksnosti naslijeđenih aplikacija. Dodatno, povjerljivi podaci na prikazanim ekranima, slikama alata i sustava zamaskirani su u svrhu zaštite informacija o klijentima, korisnicima, zaposlenicima te informatičkoj infrastrukturi PBZ. Integracija sustava prezentirana je pomoću općenitih principa kako bi se zaštitili stvarni detalje implementacije internih sustava u PBZ. Izloženi poslovni primjeri odražavaju stanje stvarnih poslovnih procesa iz PBZ, ali su prezentirani tako da ne otkrivaju detalje implementacije i poslovno relevantne podatke procesa. Promijenjeni su stvarni nazivi aplikacija, procesa i korisničkih uloga u procesu.

2 Pregled problemskog područja

Na pitanje "Što nas zanimljivog čeka u budućnosti?" većina ljudi će odgovoriti da je to povećan pristup informacijama, obavljanje transakcija pomoću mobilnih uređaja, obrada velike količine podataka, korištenje usluga računarstva u oblacima[1] te dobra integracija sa socijalnim mrežama. Prema tome, navedene usluge osnovna su obilježja modernih poduzeća, međutim ima puno nejasnoća oko njihova utjecaja na izgradnju poslovnih modela te na razvoj i rad IT[2] rješenja [22].

U današnje vrijeme programeri u svojem radu koriste gotove komponente i besplatne alate preuzete s Interneta, a izrada mobilnih aplikacija prepušta se vanjskim partnerima koji ujedno obavljaju sve više ključnih operacija u sveukupnom poslovanju. U takvom okruženju, organizacije su izgubile centraliziranu kontrolu nad svojim razvojnim te produkcijskim okolinama (Slika 2.1).

Slika 2.1. Izazovi moderne organizacije

Kako bi organizacije povratile kontrolu nad svim aspektima poslovanja, potreban je sveobuhvatan i detaljan prikaz svakog segmenta poslovanja [22]. Postojeće poslovanje tvrtke uvelike ovisi o poslovnim procesima, to jest kolektivnoj mogućnosti tvrtke da postigne svoje ciljeve. Međutim, sposobnost tvrtke da dugoročno ostvaruje pozitivan rezultat jako ovisi o tome

[1] **Računarstvo u oblacima (engl. cloud computing)** je pružanje računalne procesne moći i prostornog kapaciteta u obliku servisa heterogenoj zajednici krajnjih korisnika. Ime mu dolazi od korištenja oblaka kao simbola, a on zapravo predstavlja apstrakciju složene infrastrukture koja se nalazi pozadini.

[2] **Informacijska tehnologija (engl. Information Technology - IT)** uključuje svaki rad i upravljanje s informatičkim sustavima, programskom opremom ili računalnim sklopovima. Predstavlja općeniti naziv za svaku tehnologiju koja pomaže u radu s informacijama.

kako se nosi s promjenama. Promjene mogu biti unutarnje poput izmjena u životnom ciklusu određenog proizvoda ili vanjske poput pojave novih konkurenata i zakonskih regulativa [23].

Odgovor na postavljene izazove leži u transparentnosti i kontroli osnovnih "žila kucavica" svake organizacije, PP-a. Zbog bilo kojeg razloga, neki od tih PP-a može biti spor, neučinkovit, nepouzdan ili suvišan. Uloga upravljanja PP-ima je u detalje pogledati svaki proces, te uz pomoć modela mjerenja i analize otkriti potrebne promjene u svrhu poboljšanja procesa.

Vrlo važan faktor kvalitetnog upravljanja poslovnim procesima je podloga na kojoj će se to upravljanje temeljiti, to jest postojeći sustavi organizacije. Postojeći sustavi sastoji se od aplikacija koje u sebi imaju integrirane PP-e. Ako aplikacije nisu dovoljno robusne i fleksibilne, često se ispostavi da je takve PP-e teško nadgledati i kontrolirati.

Kod fleksibilnih arhitektura, primjerice sustava baziranog na SOA, funkcionalnosti koje čine dijelove procesa izdvojene su u određene servisne module, a ti servisni moduli se zatim lakše kontroliraju i analiziraju.

Općenito, ugradnja sustava za upravljanje PP-ima je dugotrajan i kompleksan posao, ali na kraju rezultira višestrukim pogodnostima za tvrtku [23]. Tipične prednosti uključuju skraćivanje vremena odaziva u slučaju problema ili izmjene, manje pogrešaka prilikom izvođenja te veću fleksibilnost spram promjene strukture određenog segmenta poslovanja.

2.1 Uslužno usmjerena arhitektura

SOA je stil arhitekture aplikacija koji je donio prekretnicu u području IT osiguravajući veću produktivnost te mogućnost povezivanja raznorodnih sustava [6]. Omogućio je poslovnim sektorima i odjelu informacijskih tehnologija (OIT) da budu fleksibilniji te da smanje vrijeme potrebno da se proizvod ili usluga plasira na tržište [9].

SOA je definirana kao softverska arhitektura za izradu poslovnih aplikacija koristeći skup labavo povezanih[3] softverskih komponenti u obliku crnih kutija[4] [5]. Softverske komponente koje čine temelj SOA-e nazivaju se servisi, a predstavljaju osnovne građevne jedinice koje izvršavaju specifičnu funkciju uz pomoć točno definiranog sučelja. Servis je funkcija koju izvršava određena aplikacija; jednom implementirana funkcija po potrebi se može koristiti višestruko i na više različitih mjesta [6].

[3] Pojam **labave povezanosti** (engl. **loose coupling**) opisuje sustav u kojem svaka od komponenata sustava ne sadrži nikakve, li sadrži malu količinu, informacija o definicijama drugih zasebnih komponenti.

[4] U području računarstva pojam **crna kutija (**engl. **black-box)** odnosi se na uređaj, sustav ili objekt koji se može promatrati u smislu ulaza i izlaza, bez ikakvog znanja o unutrašnjosti. Implementacija takve komponente je "neprozirna" ili "crna".

Autor u [6] naziva servise ciglama koje izgrađuju SOA arhitekturu. Što su dijelovi veći, bolji su i implementiraju više funkcionalnosti, međutim preglomazni servisi ne pružaju dovoljnu zrnatost da bi omogućili ponovo korištenje servisa.

SOA se temelji na principu orkestracije, a to je proces koji opisuje automatsko slaganje, upravljanje i koordinaciju složenih računalnih komponenti i servisa u svrhu izgradnje informacijskih sustava (IS).

Velika prednost SOA-e leži u pretpostavci da teoretska marginalna cijena n-te aplikacije iznosi nula novčanih jedinica. Budući da sav potreban softver već postoji, sama orkestracija dovoljna je kako bi se stvorila nova aplikacija.

Prema OASIS (Organization for Advancement of Structured Information Systems) referentnom modelu SOA, SOA služi za pristup dijeljenim resursima organizacije uz pomoć standardiziranih servisa [46]. Analitička kuća Gartner opisuje SOA-u kao programsku arhitekturu koja počinje s definicijom sučelja te zatim gradi cijelu strukturu aplikacija kombinirajući sučelja, implementacije i pozive tih sučelja. Kompanija IBM navodi da je SOA aplikacijski radni okvir koji svakodnevne poslovne aplikacije razbija u pojedinačne poslovne funkcije, servise. Microsoft gleda na SOA-u kao na skup raspršenih i javno dostupnih računalnih usluga koje se mogu integrirati u svrhu stvaranja neke veće vrijednosti. Multinacionalna menadžerska i konzultantska tvrtka BearingPoint definira SOA-u kao metodologiju dizajna i implementacije programskih rješenja koja rezultira arhitekturom sačinjenom od skupa labavo spojenih softverskih komponenti (servisa). Ti servisi mogu se nadalje međusobno povezivati uz pomoć raznih sučelja te neovisno o platformama [43].

SOA područje definirano je slično i od strane ostalih autora u radovima [1], [3], [4], [8], [38], [40], [44].

2.1.1 Korijeni uslužno usmjerene arhitekture

Prvu ideju komponentnog razvoja softvera spominje Douglas Mcilroy u svom radu "*Mass Produced Software Components*" na NATO-voj konferenciji o programskom inženjerstvu Njemačkoj, u Garmischu 1968. godine. Glavna namjera konferencije bila je uskladiti strategiju razvoja softvera s načinom rada u elektronici gdje se kao dijelovi integriranih krugova koriste osnovne elektroničke komponente koje se po potrebi mogu zamijeniti s drugim komponentama s istim ili sličnim funkcionalnostima [32]. Pri tome različiti standardi garantiraju da su komponente kompatibilne i zamjenjive.

Razvojem modela klijent-poslužitelj, 1991. godine, javlja se Common Object Request Broker Architecture (CORBA). CORBA je razvijena od strane OMG-a[5] (Object Management Group) te je akronim za *Common ORB Architecture*, a dizajnirana je sa svrhom da omogući komunikaciju sustava koji se nalaze na različitim platformama. ORB (Object Request Broker) predstavlja objektno orijentiranu verziju udaljenog poziva procedura (Remote Procedure Call - RPC).

1993. godine Microsoft je predstavio Component Object Model (COM). Suština COM-a je neovisnost o platformi i jeziku implementacije te korištenje kreiranih komponenti bez znanja o njihovoj unutarnjoj implementaciji. Predstavlja raspodijeljeni, objektno-orijentirani sustav za stvaranje binarnih softverskih komponenti koje mogu međusobno komunicirati.

Nastavno na COM u veljači 1997. izašao je Java Development Kit (JDK) 1.1 koji je u sebi imao novo aplikacijsko programsko sučelje (API[6]) *JavaBeans*. *JavaBeans* je prenosiva, platformski neovisna, komponentna arhitektura za razvoj Java aplikacija koja omogućava da se aplikacije sastavljaju od već gotovih komponenata koje se nazivaju *beans*.

Kao glavni odgovor na kompleksnosti navedenih postojećih tehnologija, te teškoća koje su se javljale prilikom međusobne komunikacije aplikacija implementiranih u tim tehnologijama, javila se glavna ideja iza servisno usmjerene arhitekture. Umjesto da se koriste složeni prilagodnici u svrhu komunikacije i prijenosa podataka između različitih aplikacija, svaka aplikacija bi trebala "javno pružiti svoje usluge". Te usluge nazivamo servisima, a servisi su dostupni ostalim aplikacijama po potrebi.

SOA je prvi puta opisana od strane analitičke kuće Gartner 1996. godine u istraživačkim publikacijama [75] i [76].

[5] **The Object Management Group (OMG)** je neprofitno međunarodno tijelo otvorenog članstva osnovano 1989. godine, a bavi se definiranjem različitih tehnoloških standarda. U definiciji standarda sudjeluje više strana iz raznih sfera poslovnog i društvenog svijeta, proizvođači, dobavljači, krajnji korisnici, akademske institucije i vladine agencije.

[6] **Aplikacijsko programsko sučelje** (engl. **Application Programming Interface - API)** je skup definicija, protokola i alata za izradu softvera, koji sadrže jasno definirane metode komunikacije između različitih softverskih komponenti, te tako definiraju uvjete međusobnog povezivanja komponenti. U praksi API najčešće dolazi u obliku biblioteke koja sadrži specifikacije za funkcije, strukture podataka, klase objekata i varijable ili pak opisuje specifikaciju udaljenih poziva.

2.2 Upravljanje poslovnim procesima

Paul Harmon i Michael Hammer definiraju Upravljanje poslovnim procesima kao holistički[7] pristup upravljanju [12] usmjeren na usklađivanje svih aspekata organizacije sa željama i potrebama klijenata [15]. Ističu da promiče poslovnu učinkovitost, težnju za inovacijama, fleksibilnost i integraciju s tehnologijom. To je područje koje objedinjuje PP-e i informacijsku tehnologiju kroz korištenje raznih metoda i tehnika za izradu, upravljanje i analizu procesa [12].

BPM je strukturirana metoda razumijevanja, dokumentiranja, modeliranja, analize, simuliranja, izvršavanja i neprestane prilagodbe cjelokupnih PP-a te popratnih resursa u svrhu povećanja vrijednosti poslovanja određene tvrtke [24]. Pojam označava procesno usmjeren pristup u svrhu postizanja što veće poslovne učinkovitosti [25]. Problemi koji se rješavaju nisu suštinski novi, već BPM pruža inovativan način na koji se oni mogu rješavati [26]. Koriste se suvremene tehnologije kako bi se organizacijama pružile mogućnosti da otkriju, modeliraju i preprave svoje PP-e. Zatim, omogućuje da se ti procesi izlože kao zasebne aplikacije integrirane s postojećim sustavima, te pruža menadžerima mogućnost praćenja, analiziranja, kontroliranja te unapređivanja tih procesa tijekom izvršavanja u stvarnom vremenu.

BPM možemo gledati na četiri različita načina, u vidu PP-a kako ih obavljaju ljudi, spram toka PP-a kroz IS-ove, temeljem automatizacije poslovnih pravila ili s ciljem praćenja poslovnih aktivnosti (engl. Business Activity Monitoring - BAM) [11].

Gartner naglašava da je BPM ključ za usklađivanje IT s operativnim djelovanjem u svrhu ostvarivanja zacrtanih poslovnih strategija. BPM definiraju kao disciplinu koja koristi razne metode za otkrivanje, modeliranje, analizu, mjerenje, poboljšanje te optimiranje PP-a. BPM pokušava kontinuirano poboljšati PP-e, pa se slikovito može opisati kao "proces optimiranja procesa". PP koordinira ponašanje ljudi, sustava, informacija i svih strana koje zajedno stvaraju poslovne rezultate u svrhu ostvarenja poslovnih strategija. Prava vrijednost BPM-a proizlazi iz činjenice da pruža transparentnost i kontrolu nad poslovnim procesima [9]. Isto tako, omogućava sinkronizaciju i optimizaciju interakcije između unutarnjih te vanjskih dijelova poslovanja neke organizacije [44].

Slično navedenim definicijama, BPM područje opisuje i većina ostalih autora [3], [6], [7], [14], [23], [26], [37], [39].

[7] **Holizam** (grčki 'όλα' = cijeli) je pogled na svijet i tendencija u filozofiji koja kaže "istina je uvijek samo u cjelini", istinitost pojedinačnih iskaza proizlazi iz cjeline, ali cjelina se ne može svesti na sastavne dijelove.

2.2.1 Početci

Početci upravljanja poslovnim procesima leže još davnih 1980-tih u sustavima protoka poslova (engl. workflow management system). Sustav protoka poslova ili sustav s dijagramom toka je računalni sustav koji definira niz zadataka i njima upravlja u svrhu proizvodnje te ostvarenja poslovnih ciljeva organizacije. Sustavi protoka poslova omogućuju korisniku da definira različite tokove rada za različite vrste poslova ili procese.

Paralelno sa sustavima protoka poslova pojavio se i EAI okvir za međusobnu integraciju različitih sustava. EAI je integracijski okvir sastavljen od skupa tehnologija i usluga čineći tako jednu vrstu međuprograma (eng. middleware) koji omogućava integraciju različitih sustava u svim razinama tvrtke [67], [68]. Ključni proizvod temeljen na tom principu bio je IBM MQSeries, sustav za razmjenu poruka koji je s vremenom postao standard u mnogim organizacijama [34].

Iako se sada smatraju kao dvije implementacije iste ideje, u ono doba sustavi protoka poslova i EAI sustavi živjeli su odvojenim životima. Sustavi protoka poslova bili su namijenjeni su poslovnim jedinicama, dok je EAI arhitektura bila projektirana za OIT [34].

S vremenom u EAI arhitekture počeli su se ugrađivati osnovni elementi praćenja toka zadataka te sučelja za interakciju s korisnicima, dok su sustavi protoka poslova bili proširivani elementima EAI arhitekture.

Potreba otkrivanja nepravilnosti, nedostataka i uskih grla u sustavima protoka poslova dovodi do razvoja alata za praćenje i izvještavanje, koji se počinju ugrađivati u sustave protoka poslova te tako nastaje BAM. Analize podataka i praćenje poslovnih aktivnosti davale su klijentu važne informacije o uspješnosti tvrtke, postojećim nedostatcima, te su isto tako rezultirale potrebom da se uočeni nedostatci otklone. U svrhu otklanjanja otkrivenih nedostataka sustavi protoka poslova dodatno su nadograđivani s različitim alatima za simulaciju i optimiranje (Slika 2.2).

Razvoj EAI proizvoda krenuo je u smjeru integracije poslovanja (engl. Business To Business integration - B2Bi)[8] više organizacija koje su svojim aktivnostima gradile određeni proizvod ili uslugu. Navedena B2Bi integracija bila je jedna od važnijih poboljšanja u smjeru BPM-a kakav postoji danas.

[8] **Integracija poslovanja (B2Bi)** se definira kao nastojanje poduzeća da proširi svoje poslovne procese na partnere, kupce, dobavljače, distributere, te ostala poduzeća s kojima posluje. B2B integracija uključuje integraciju različitih tehnologija, a istovremeno predstavlja jedinstven pogled na razmjenu poslovnih informacija. Primjer arhitekture pogodne za B2B integraciju bila bi SOA.

Danjim razvojem tržišta sustavi protoka poslova te EAI proizvodi počinju sličiti što je zbunilo klijente koji nisu vidjeli granicu između ta dva proizvoda, te se nisu mogli odlučiti kada kupiti jedan, a kada drugi proizvod [34]. Dodatna zbrka za klijente nastala je kada je prostor djelovanja EAI proizvoda te sustava protoka poslova preimenovan u područje BPM-a.

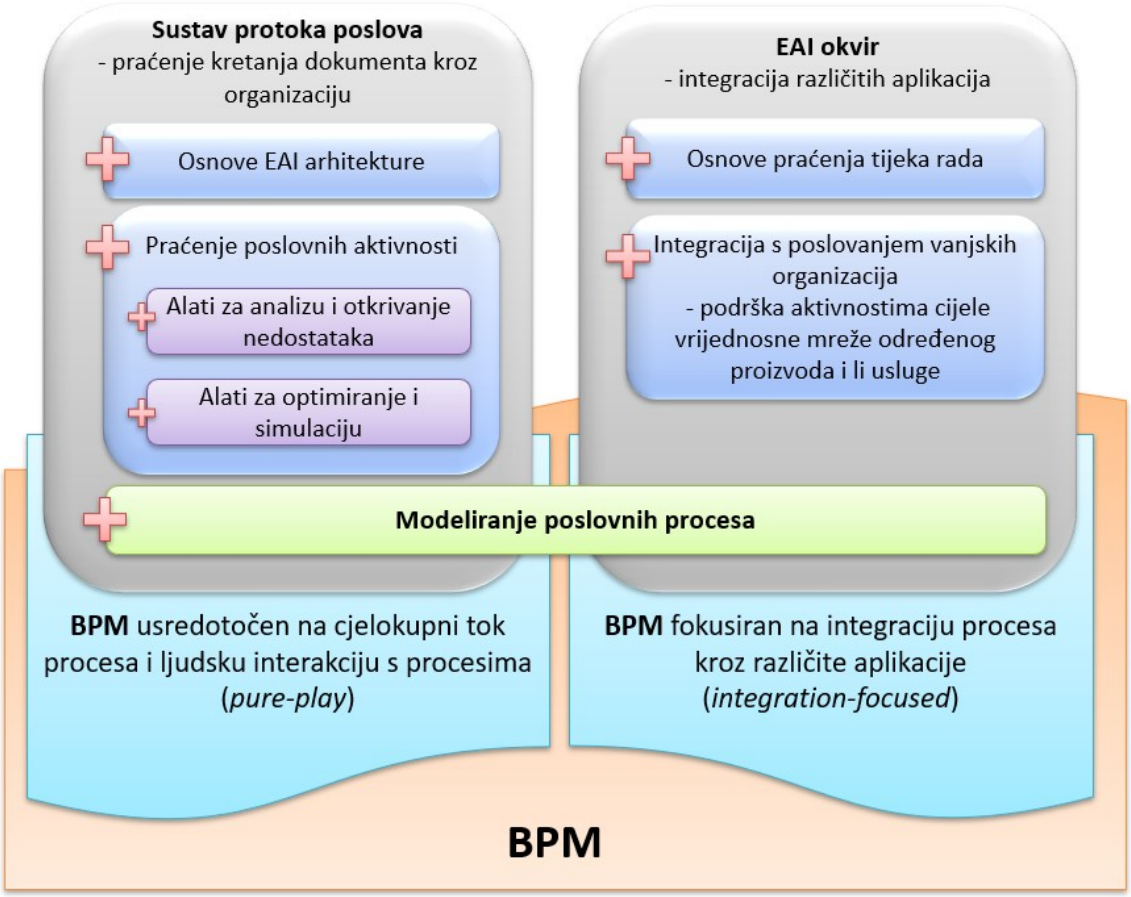
Kao rezultat stvaraju se podjele unutar BPM prostora prema funkcionalnostima (zasebnim BPM programskim paketima) koje su pokrivale specifične potrebe. Organizacije su tada često bile primorane kupovati više paketa od različitih dobavljača kako bi zadovoljile sve svoje potrebe.

Većina proizvoda na tržištu i dalje se mogla svrstati u dvije osnovne kategorije ili grane BPM programskih paketa, koje su zapravo odgovarale staroj podjeli na sustave protoka poslova i EAI proizvode:

- *pure-play BPM* - usredotočena na tok PP-a i ljudsku interakciju s tim procesima, proizašla iz tržišta sustava protoka poslova. Upravo je ova grana rezultirala jako dobrim rezultatima za organizacije te je time uvelike povećala interes za BPM.
- *integration-focused BPM* - proizašla iz filozofije EAI proizvoda te se fokusirala na kontrolu međusistemskih procesa i integraciju tih procesa kroz različite aplikacije.

Početkom novog tisućljeća velik broj novih dobavljača BPM-a počine s razvojem od nule, dakle bez upotrebe postojećih tehnologija i ograničenja tih tehnologija. Nove ideje i funkcionalnosti uvelike mijenjaju percepciju BPM tržišta, te formuliraju BPM kakav postoji danas. Dolazi do razvoja BAM-a ili barem nekog donekle prihvatljivog procesa za praćenje i analitiku, razvoja alata za simulaciju i optimiranje, izgrađuje se integracija sa stranim alatima za modeliranje te sustav za integraciju poslovnih pravila [34].

Slika 2.2 prikazuje opisani razvoj BPM područja kroz evoluciju EAI proizvoda te sustava protoka poslova.



Slika 2.2. Uloga EAI proizvoda i sustava protoka poslova u formiranju BPM područja

Današnji BPM savršen je primjer sustava gdje je cjelina veća od zbroja sastavnih dijelova. Ne predstavlja samo skup sustava protoka poslova, EAI proizvoda, B2B integracije poslovanja, raznih poslovnih pravila itd., već čini gotovo besprijekornu integraciju svih tih alata u cjelovit sustav koji organizaciji osigurava uvećanu sposobnost djelovanja.

BPM stvara jedinstvenu definiciju cjelokupnog PP-a iz koje se mogu generirati različiti pogledi na procese. Omogućava različitim osobama, s različitim vještinama (menadžerima, poslovođama, analitičarima, programerima) da pregledavaju i manipuliraju istim procesima u njima razumljivom prikazu, koji je generiran iz istog izvora.

2.2.2 Osnove modeliranja poslovnih procesa

Modeliranje poslovnih procesa (MPP, engl. Business Process Modelling) predstavlja aktivnost prikaza PP-a određene tvrtke u svrhu automatizacije, analize i optimiranja. Obično se provodi od strane analitičara i menadžera koji žele poboljšati kvalitetu i učinkovitost procesa. Identificiraju se nedostatci u izvršavanju PP-a, modeliraju se poboljšani PP-i bez otkrivenih nedostataka te se izvodi simulacija izvršavanja novih, poboljšanih, PP-a.

MPP zahtjeva poznavanje određenih općenitih koncepata i pojmova korištenih u svrhu lakše izgradnje te razumijevanja PP-a [33]. MPP se obično radi u alatu za modeliranje koji je dio BPMS-a, uključuje neki od standarda za MPP, a rezultira dijagramom poslovnog procesa (DPP, engl. Business Process Diagram). U nastavku je dan općeniti pregled osnovnih koncepata koji su bitni za razumijevanje MPP-a:

- **Aktivnosti, sudionici, uloge i koraci**

  Određena aktivnost je dio PP-a, predstavlja jedinicu rada s početkom i krajem, a odrađuje se od strane zaduženih pojedinaca, grupe pojedinaca ili sustava.

  Zadatak možemo promatrati kao aktivnost koja je dodijeljena na izvršavanje određenom pojedincu, grupi pojedinaca ili sustavu. Aktivnosti mogu biti jednostavne, poput slanja e-mail obavijesti ili složenije kao što je odobravanje i provjera ispravnosti zahtjeva za određenu uslugu.

  Zadužene pojedince koji obavljaju aktivnosti nazivamo sudionicima, a ako sudionici obavljaju slične ili srodne aktivnosti, mogu se grupirati prema ulogama. Stoga, uloga je skup zadaća dodijeljenih jednoj ili nekoliko osoba koje su povezane s određenom aktivnošću, to jest korakom u poslovnom procesu [33].

- **Tok PP-a**

  Tok PP-a predstavlja redoslijed izvršavanja aktivnosti (zadataka) PP-a. Osim prve i posljednje aktivnosti u procesu sve ostale aktivnosti mogu imati više ulaznih i izlaznih ruta.

  Uvjeti grananja definirani su unutar procesa, a evaluiraju se u trenutku završetka aktivnosti, te određuju sljedeću aktivnost. Dodatno, sudionici mogu odabrati željenu rutu na temelju vlastitih znanja i zapažanja [33].

  Postoje tri različita načina grananja: jednostavno, uvjetno i višestruko. Kod jednostavnog grananja aktivnosti se mogu odvijati samo slijedno, postoji direktna veza između dvije susjedne aktivnosti, te u bilo kojem trenutku samo jedna aktivna aktivnost. Uvjetno grananje uključuje specifikaciju uvjeta (logički izraz ili zaključak sudionika), temeljem kojeg se odabire ruta prema sljedećoj aktivnosti. Višestruko grananje omogućuje razdvajanje i paralelno izvršavanje aktivnosti. Po završetku svih aktivnosti višestrukog grananja izlazne rute se spoje uz pomoć tzv. "koraka sakupljača“ (engl. collector step), te se agregiraju podaci prikupljeni u pojedinim koracima.

- **Obavijesti**

  Obavijesti su točke unutar PP-a u sklopu kojih sudionici dobivaju poruku o zadatku ili skupini zadataka koje moraju izvršiti [33]. Poruka može sadržavati tekst s uputama ili određenim poslovnim podacima važnim za proces.

- **Rokovi (Deadlines)**

  Rokovi predstavljaju vremenska ograničenja koja se mogu primijeniti na aktivnosti unutar PP-a, a osiguravaju da sudionik završi određeni zadatak unutar određenog vremenskog okvira. Po isteku roka moguće je pokrenuti i eskalaciju zadatka, dodijeliti ga na izvršavanje drugom sudioniku ili nadležnoj osobi [33].

- **Preraspodjela zadataka/aktivnosti**

  Omogućava preraspodjelu zadataka ili aktivnosti na ostale sudionike. Postoje dvije vrste raspodjela zadataka. *Delegiranje* omogućuje prenošenje aktivnosti na drugog sudionika, te nakon što on obavi traženi posao, vraćanje aktivnosti na prvobitnu zaduženu osobu u svrhu revizije napravljenog posla. Drugi način preraspodjele je *prepuštanje* gdje se aktivnost u potpunosti dodjeljuje drugom sudioniku, te nakon što on taj zadatak obavi, proces kreće na sljedeću aktivnost u procesnim tokom.

- **Procesno glasovanje**

  Koristi se kada je potrebna povratnu informaciju od više različitih sudionika PP-a temeljem čega se određuje sljedeća aktivnost procesa. Omogućuje tvrtkama da implementiraju kolaborativno donošenje odluka unutar strukturiranog procesnog sustava [33].

Navedeni principi i koncepti MPP-a, realizirani su i preciznije opisani elementima standarda Business Process Model and Notation (BPMN) koji služe kao osnova za MPP.

2.2.2.1 Kratak pregled BPMN standarda

BPMN je globalni standard za MPP, omogućuje kreiranje definicije PP-a (engl. Business Process Definition) izgradnjom DPP-a, te je ujedno jedna od najvažnijih komponenti u usklađivanju i povezivanju poslovne strane i OIT-a [62].

Vrlo je jednostavan za korištenje, omogućuje relativno brzo MPP, te je lako razumljiv ljudima koji nemaju puno tehničkog znanja, kao što su menadžment i uprava. BPMN pruža mogućnost modeliranja složenih procesa te mapiranje između grafičke reprezentacije i izvršnih jezika za specificiranje akcija, primjerice BPEL-a (Business Process Execution Language) [62]. Podržava koncepte modeliranja primjenjive samo na PP-e dok ostale vrste modeliranja, poput izrade organizacijskih struktura ili modela podataka nisu podržane.

Modelirani proces može se sastojati od jednog ili više potprocesa (engl. subprocess). Potprocesi se definiraju zasebnim dijagramima koji su spojeni na glavni proces (veza u obliku

znaka "+" na procesnom simbolu), a BPMN standard podržava čak jedanaest različitih tipova potprocesa.

Zaduženja vezana uz određeni proces najčešće obavljaju osobe iz različitih odjela organizacije, pa je dobra praksa razvrstati proces kroz polja (engl. *pools*) te staze (engl. *lanes*), pri čemu se polja sastoje od staza. Iako nema strogo definiranog pravila, u poslovnom smislu, polje predstavlja skupinu ljudi, organizaciju ili dio organizacije kojem pripada određeni dio PP-a, dok staza označava specijalizirane funkcije ili pojedince koji obavljaju određene akcije [63]. Često se polja i staze koriste za grupiranje dijelova procesa koji pripadaju različitim sektorima u organizaciji ili prema sustavima koji su uključeni u izvršavanje procesa, što će se iskoristiti kod modeliranja procesa u ovom radu (poglavlje 6.3.2).

BPMN ima vrlo važan koncept koji se naziva token. Token nije vidljiv, već je samo teoretski koncept koji omogućava da bolje razumijemo kako se proces izvršava. Token nastaje kod pokretanja procesa, putuje DPP-om, te kako proces teče redom aktivira različite dijelove procesa.

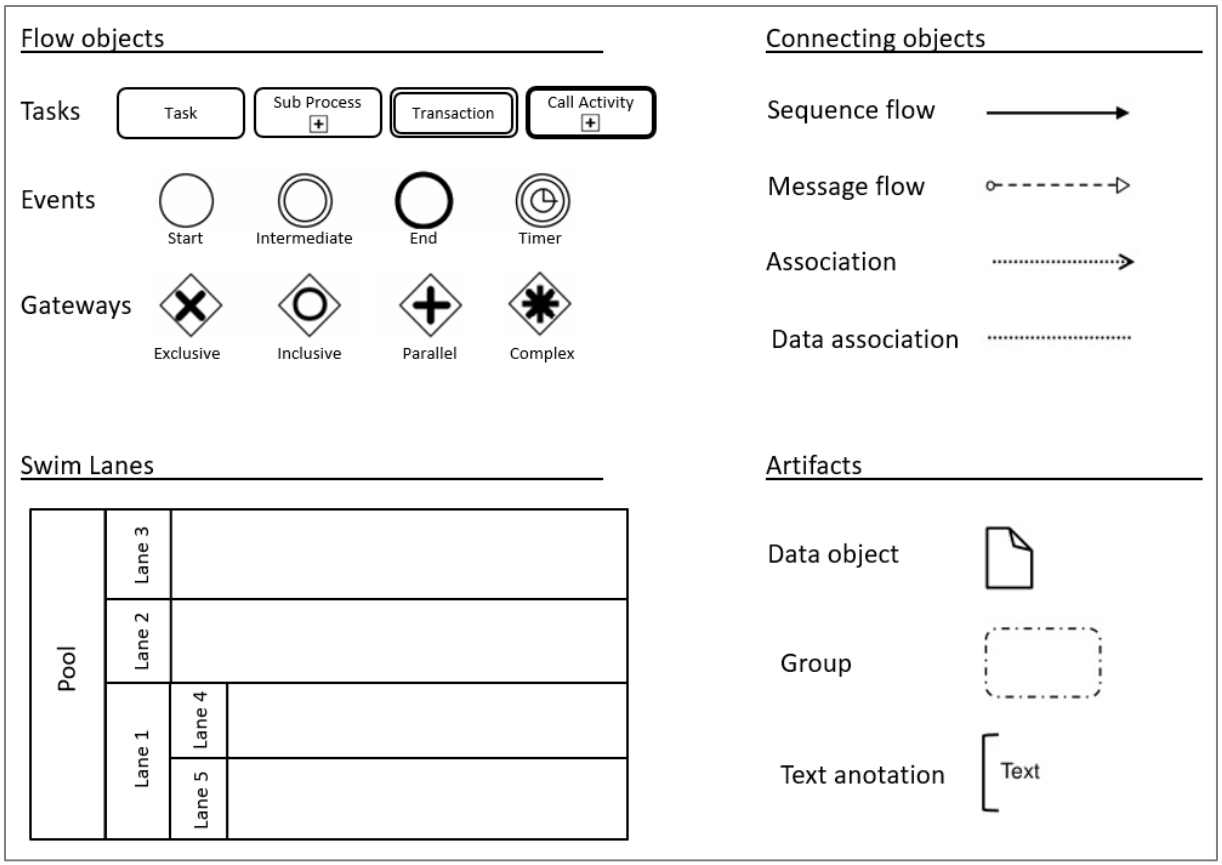
BPMN modeli sastoje se od jednostavnih dijagrama izrađenih od ograničenog broja grafičkih elemenata. U nastavku je dan sažet pregled osnovnih kategorija BPMN elemenata te njihov grafički prikaz (Slika 2.3 proizašla iz [59], [61], [62], [63]), a sve detaljnije informacije, objašnjenja te primjeri njihovog korištenja mogu se naći u literaturi [62] i [63]. Osnovne kategorije BPMN elemenata su:

- **Objekti protoka (*Flow objects*)** – glavni elementi dijagrama te mogu biti:
  - ***Događaji (Events)*** - prilikom modeliranja PP-a, modeliraju se događaji (*events*) koji se odvijaju u poslovanju te njihov utjecaj na ostale procese u organizaciji. Događaj započinje tok procesa (*start event*), javlja se tijekom izvršavanja procesa (*intermediate event*), završava proces (*end event*) ili počinje nakon neke odgode (*timer event*). BPMN koristi posebnu notaciju za sve vrste tih događaja, a unutar oznake događaja može imati biti dodatan simbol koji predstavlja vrstu pokretača događaja.
  - ***Aktivnosti (Activities)*** - aktivnosti predstavljaju izvršne elemente u BPMN-u koji odrađuju jednostavnije ili složenije poslove. U aktivnostima imamo zadatke (*task*) koji predstavljaju atomarne jedinice posla unutar procesa, a mogu biti različitih tipova (*manual task, user task, system task,* itd.). Osim zadataka u aktivnosti spadaju i potprocesi (*sub-proces*), *transakcije* (transaction), aktivnosti poziva (*call activity*) te događaj potprocesa (*event subprocess*). Aktivnost poziva je zapravo veza prema globalno dostupnom zadataku ili potprocesu, a događaj potprocesa omogućava da se prilikom pojave određenog događaja pokrene željeni potproces.

- ***Skretnice (Gateways)*** - važno je napomenuti da objekt skretnice ne uključuje aktivnost odlučivanja o grananju, koja se odrađuje prije ulaska u skretnicu, već samo barata rezultatima odluke. Imena skretnica obično su obliku pitanja (npr. Zahtjev odobren?), a BPMN podržava više različitih skretnica: *exclusive*, *event-based*, *inclusive*, *parallel* i *complex gateway*. Različite puteve toka procesa koji izlaze iz skretnica nazivamo grane toka procesa.

- **Objekti za spajanje (*Connecting objects*)**, kao što im samo ime govori, služe za povezivanje elemenata dijagrama.
  - ***Asocijacija (Association)*** - služi za povezivanje artefakta ili teksta s objektom protoka.
  - ***Slijedni tok (Sequence flow)*** - prikazuje tijek redoslijeda procesa, sastoji se od čvrste linije i strelice, a može na sebi imati i simbole koji označavaju vrstu toka iz određene aktivnosti, primjerice uvjetni tok (*conditional flow*) ili zadani tok (*default flow*). Slijedni tok može se koristiti samo unutar jednog polja, a da bi prenijeli informaciju van polja, možemo koristiti *tok poruka*.
  - ***Tok poruka (Message flow)*** - prikazuje kako poruke teku između različitih organizacijskih dijelova i/ili organizacija, to jest između različitih *polja*.
- **Trake aktivnosti (*Swim Lanes*)** su vizualni mehanizam organiziranja i kategorizacije aktivnosti, a sastoji se od dvije vrste:
  - ***Polja (Pool)***
  - ***Staze (Lanes)***
- **Artefakti (*Artifacts*)** omogućavaju programerima da unesu neke dodatne informacije u DPP, koji tako postaje čitljiviji. Postoje tri unaprijed definirana artefakta:
  - ***Podatkovni objekt (Data object)*** - služi za vizualni prikaz podatkovnih struktura s kojima baratamo unutar procesa. Primjerice, ako proces opisuje realizaciju zahtjeva za korištenje kredita, podatkovni objekt drži sve elemente tog zahtjeva.
  - ***Grupa (Group)*** - pruža mehanizam grupiranja objekata po različitim kategorijama.
  - ***Tekstualna anotacija (Text annotation)*** - ne utječe na tok procesa, a koristi se za prikaz dodatnih informacija o pojedinim objektima toka u DPP-u.

Prikazane kategorije omogućuju stvaranje raznih dijagrama poslovnih procesa, a BPMN također dopušta stvaranje nove vrste objekta protoka te artefakata kako bi dijagram bio još razumljiviji.



Slika 2.3 Grafički prikaz osnovnih BPMN elemenata

Gradnjom procesa kroz polja i staze određujemo tko što radi, događajima specificiramo gdje i kada želimo da se desi određena akcija, dok skretnice podržavaju donošenje i provođenje odluka [63]. Modeliranje uz pomoć BPMN-a bitno je za razumijevanje poslovnih procesa u poduzeću te omogućava kompanijama da razumiju i oblikuju svoju poslovnu arhitekturu kako bi brže i pouzdanije proveli željene promjene.

2.2.2.2 Ostali važniji standardi i pojmovi

Osim BPMN-a postoji više od desetak standarda za MPP, koji su u pravilu bazirani na XML-u, npr.: Business Process Definition Metamodel (BPDM), Business Process Modeling Language (BPML), Web Services Business Process Execution Language (WS-BPEL, kraće BPEL), Event-Driven Process Chains Markup Language (EPML), Business Process Specification Schema (BPSS), UML[9] Activity Diagram, itd. Pogledajmo ulogu nekih od navedenih standarda koji su obilježili BPM područje.

[9] **Unified Modeling Language** (**UML**) je standardizirani jezik za modeliranje u području programskog inženjerstva koji ima za cilj pružiti standardni način vizualizacije dizajna sustava prema definiciji OMG-a (Object Management Group).

U kontekstu BPM-a, često se spominje BPML. To je jezik temeljen na XML-u, razvijen od strane neprofitne organizacije za standardizaciju Business Process Management Initiative (BPMI), kao sredstvo za MPP. Zamišljen kao formalno potpuni jezik[10] s mogućnošću modeliranja bilo kakvog procesa koji se zatim može koristiti kao gotova softverska komponenta. Međutim BPMI odustaje od standarda BPML 2008. godine te se od tada više ne koristi.

Zatim, poželjno je razumjeti i ulogu standarda BPEL. BPEL je jezik temeljen na XML-u koji omogućuje razmjenu zadataka u raspodijeljenom računalnom okruženju [57]. BPEL je standardiziran od strane OASIS-a (Organization for the Advancement of Structured Information Standards), a naglasak mu je na automatizaciji. Dizajniran je s idejom orkestracije interakcije različitih sustava te standardizacije tokova poslovnih procesa uz pomoć web servisa. BPEL polazi od pretpostavke da je svaki uključeni proces implementiran u obliku web servisa.

Inicijalno, BPMN i BPEL su se često koristili zajedno. BPMN je korišten za perspektivu usmjerenu na poslovne korisnike, a BPEL za tehničku specifikaciju. Osnovna razlika između standarda BPMN i BPEL je da se BPMN koristi kod dizajniranja poslovnih procesa dok BPEL služi za implementaciju procesa [3]. BPMN dijagram je namijenjen za prikaz osnovne strukture, tijeka aktivnosti i podataka unutar PP-a, dok je za potrebe kreiranja BPEL izvršne datoteke procesa, potrebno sakupiti i mapirati dodatne podatke ključne za rad procesa [13]. Primjer tih podataka mogu biti razni parametri koji će proces koristiti, lokacije pomoćnih datoteka poput WSDL-a[11] ili neke varijable specifične za okolinu u kojoj će se proces izvršavati.

U starijim verzijama standarda BPMN ključno je bilo mapiranje BPMN u BPEL. Međutim, s verzijom 2.0 standardu BPMN dodan je vlastiti XML format (koji je zapravo prošireni BPEL) te je time BPEL u navedenom kontekstu postao nevažan.

2.2.3 Okruženja za upravljanje poslovnim procesima

BPMS je integrirano rješenje za osmišljavanje, provedbu i poboljšanje aktivnosti i PP-a kojima se postiže određeni cilj organizacije. Dizajnirana su s ciljem da bi se mogla identificirati

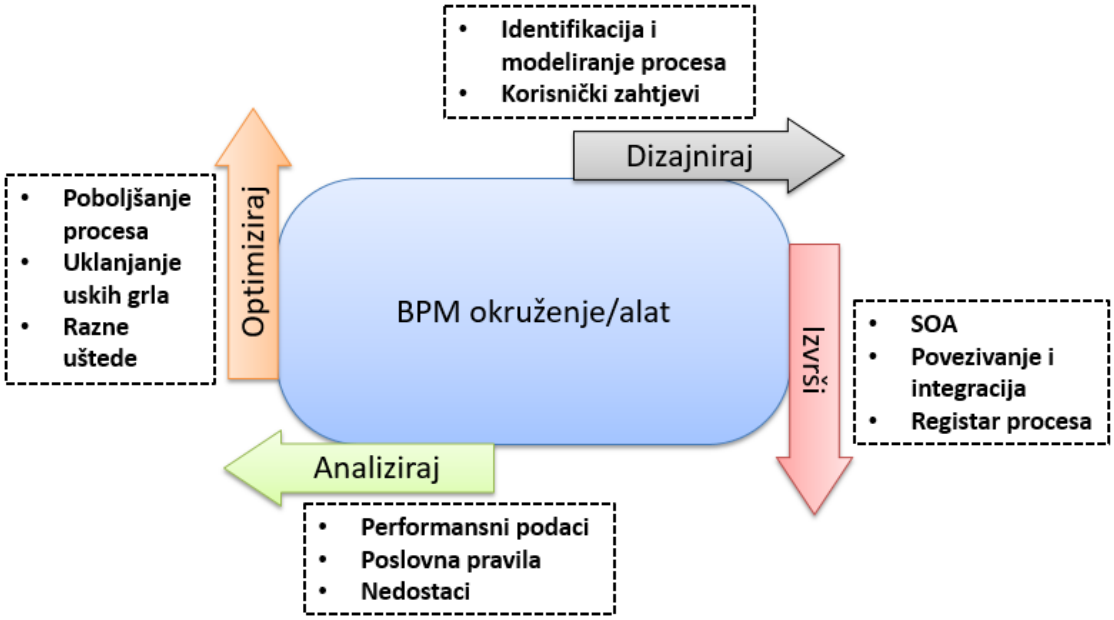
[10] **Formalno potpun** ili **Turing-potpun** jezik (engl. Turing complete programming language) je jezik u kojem se može simulirati rad Turingovog stroja, konačnog automata povezanog s vanjskim medijem za pohranu ili memoriranje podataka. Općenito, kako bi jezik bio Turing-potpun, treba imati neki oblik uvjetnog ponavljanja ili uvjetnog skoka (npr., *while* naredbu ili pak *if* u kombinaciji s *goto* naredbom) i način za čitanje i pisanje podataka s nekog oblika uređaja za čuvanje podataka (npr. papirnate ili magnetske trake, tvrdog diska, itd.).

[11] **Web Services Description Language** (WSDL) je sučelje bazirano na XML-u, a koristi se za opisivanje funkcionalnosti koju nudi određeni internetski servis. Sadrži podatke o lokaciji na kojoj je servis izložen (engl. endpoint), svim operacijama koje servis pruža te definira strukturu XML zahtjeva (engl. request) i odgovora (engl. response).

i izmjeriti brzina odziva organizacije na potrebe klijenta, te da bi se stvorila cjelokupna slika organizacije [15], [16].

Često susrećemo i pojam BPM okruženja (eng. BPM environment) koji je širi od BPMS-a, te se odnosi na BPMS i BPM rješenje koje je u njemu izgrađeno.

Važno je uočiti razliku između pojmova BPM i BPMS. BPM je profesionalna disciplina koju obavljaju ljudi, dok je BPMS paket alata osmišljen kako bi pomogao stručnjacima da osmisle, provedu i poboljšaju aktivnosti i PP-e organizacije.



Slika 2.4. Životni ciklus PP-a unutar BPMS-a

BPM se također ne smije miješati s BPMA koja je razvijena kao podrška određenom procesu [57]. BPMA se sastoji od modela PP-a (procesni model) izgrađenog u BPMS-u (skupu BPM alata), a koristi naslijeđene aplikacije ili servisne module za obavljanje potrebnih funkcionalnosti (aktivnosti) modeliranog procesa.

BPMS pomaže u automatizaciji i upravljanju poslovnim pravilima, procesima, povećava ukupnu poslovnu vrijednost, omogućava učinkovito iskorištavanje informatičkih resursa te učinkovito rješava integracijske probleme tijekom cijelog životnog ciklusa BPM rješenja (Slika 2.4 napravljena u skladu s [3], [7], [9]) .

Dobar BPMS treba sadržavati mogućnost grafičkog modeliranja PP-a koje je dovoljno jednostavno da bude razumljivo poslovnim korisnicima, ali ujedno i dovoljno složeno da dozvoljava modeliranje složenih sustava. Potrebna je mogućnost kreiranja korisničkih sučelja, simuliranja toka PP-a, bilježenja i prikaza podataka o performansama sustava, integracije s vanjskim sustavima, mogućnost eskalacija, slanja poruka, itd.

Dijagrami procesa unutar BPMS-a najčešće se modeliraju temeljem standarda BPMN koji je opisan u poglavlju 2.2.2.1.

U pogledu osoblja i novčanih troškova, resursi potrebni za implementaciju BPMS okruženja su ogromni, međutim, potencijalne koristi i prednosti za organizacije također mogu biti velike.

2.2.3.1 Zašto koristiti BPM alate?

Svrha BPM-a je povezivanje različitih dijelova sustava, bilo povezivanje različitih aplikacija, zaposlenika i aplikacija, ili zaposlenika i klijenata. Povezivanjem nastaju poslovne aktivnosti, PP-i, a njih je potrebno održavati, pratiti, analizirati te trajno optimizirati.

Potražimo najprije odgovor na pitanje; Može li se BPM provoditi bez korištenja složenih BPM alata?! Ako pogledamo poslovnu stranu, vlasnike i kreatore poslovnih procesa, razumljivo je da oni mogu relativno jednostavnim korekcijama u definiciji toka i načinu izvršavanja poslovnih procesa promijeniti i optimizirati PP-e. Dakle, odgovor na pitanje bio bi pozitivan. Međutim, promjene koje možemo napraviti ograničene su na jednostavne i očigledne korekcijske akcije, čiju korisnost bez naprednih analitičkih alata nije moguće sa sigurnošću potvrditi. Dodatno, takve promjene su jednokratne te ne mogu učinkovito pratiti razvoj poslovanja organizacije i promjene u poslovnim procesima. Sam proces provođenja promjena je mukotrpan, dugotrajan, a dobiveni rezultati nisu potpuno jasni i objektivni.

Dakle, u procesu uvođenja BPM-a u organizaciju, početni napori mogu se odraditi bez korištenja BPM alata. Međutim, kako BPM inicijativa sazrijeva ulazimo u područje veće složenosti, ozbiljnijih analiza, a kratkoročna poboljšanja želimo zamijeniti dugoročnom optimizacijama i fleksibilnošću poslovnih procesa. U tom slučaju korištenje BPM alata postaje ključan faktor.

BPM alati omogućuju automatizaciju upravljanja poslovnim aktivnostima, pomažu osigurati pravilnu integraciju, praćenje i prikupljanje informacija. MPP omogućeno je na intuitivan način uz pomoć standardiziranih notacija što pruža transparentnost i fleksibilnost, te jasan pregled resursa i aktivnosti koje su kritične za izvršavanje procesa. Otkrivanje poboljšanja u toku procesa lako je ostvarivo korištenjem BPM alata. Implementacija PP-a unutar BPM alata omogućava brži razvoj, nadzor i optimizaciju PP-a. Moguće je otkrivanje uskih grla čime se izbjegavaju problemi u daljnjem radu. Kontrolom koju pruža BPM alat, proces donošenja odluka postaje fleksibilniji, a produktivnost veća. Nakon izvršenja procesa, BPM alati mogu se koristiti za analizu odluka donesenih u procesu, a na temelju dostupnih informacija mogu se poduzeti korektivne mjere i optimizirati proces.

BPM alati pružaju organizacijama jedan jednostavan način za stvaranje podrške vlastitoj procesnoj arhitekturi. Kako tvrtke polagano prihvaćaju ideju da ne trebaju od nule raditi vlastitu arhitekturu praćenja procesa, dolazi do mnogih pozitivnih promjena i napredaka u svim aspektima rada organizacije.

Neakademske istraživačke institucije redovito objavljuju istraživanja provedena o raznim BPM alatima što pokazuje važnost i relevantnost ove teme. Primjerice, Gartner u istraživačkim radovima [48], [49], [50] uz pomoć svojeg magičnog kvadranta[12] analizira tržište BPM i iBPM[13] alata. Slično, američka istraživačka tvrtka Forester[14] u svojim istraživanjima konstantno evaluira i rangira vodeće distributere BPM alata temeljem velikog broja kriterija koji odražavaju potrebe stručnjaka iz BPM područja; primjerice [51], [52]. Istodobno postoji mnogo analiza, kritika i usporedba postojećih BPM alata od strane nezavisnih stručnjaka iz različitih mrežnih izvora kao što su [53], [54], [55], [56].

Svi navedeni izvori obično procjenjuju alate prema širokom rasponu funkcionalnih i nefunkcionalnih kriterija te stoga pružaju dobar pregled dostupnih opcija na tržištu, međutim ne uzimaju u obzir specifične potrebe i mogućnosti organizacija koje se upuštaju u BPM

[12] **Gartnerov magični kvadrant** je skup istraživanja provedenih na određenom tržištu koji daje široki pogled na relativne pozicije konkurenata na tom tržištu. Grafički prikazuje mogućnosti organizacija kroz 4 kategorije ovisno o mogućnosti izvedbe i količini vizije (inovativnosti) koju imaju. ***Lideri*** koji dobro izvršavaju trenutne vizije i spremni su na izazove u budućnosti; ***Vizionari*** shvaćaju promjene na tržištu ili imaju ideje za promjenu tržišnih pravila, ali isto tako dobro izvršavaju trenutne vizije; ***Igrači niše*** se pak uspješno fokusiraju na mali specijalizirani segment, ali su nekreativni i nisu sposobni se nadigravati s ostalim konkurentima; ***Izazivači*** koji dobro izvršavaju trenutne potrebe i mogu dominirati velikim segmentima tržišta, ali razumiju smjer u kojem se tržište kreće. Kvadrant se koristi kao prvi korak za razumijevanje tehnologija koje se uzimaju u obzir kod odabira mogućih ulaganja, a bitno je imati na umu da fokusiranje na lidera u kvadrantu nije uvijek najbolji način djelovanja.

[13] **Inteligentni BPM alati** (engl. Intelligent Business Process Management ili iBPM) su zapravo konvencionalni alati za upravljanje poslovnim procesima obogaćeni dodatnom „inteligencijom“. Dodatna inteligencija uključuje značajke kao što su računarstvo u oblaku, obrada događaja i donošenje odluka u stvarnom vremenu, poboljšanu podršku za ljudsku suradnju putem integracije s društvenim medijima, podršku za mobilne uređaje i slično [50]. Iako iBPM zvuči kao novi koncept, već postoji unutar organizacija koje koriste sustave za upravljanje poslovnim procesima (BPMS) u kombinaciji s drugim pametnim alatima u svrhu optimizacije procesa.

[14] **Forrester** je američka tvrtka za istraživanje tržišta koja pruža savjete o trenutnim i potencijalnim utjecajima tehnologije svojim klijentima te široj javnosti. Nudi niz usluga, uključujući troškovno isplativa istraživanja tehnoloških područja interesantnih za poslovanje određenih organizacija, kvantitativno istraživanje tržišta o usvajanju potrošačkih tehnologija, analizu potrošnje resursa unutar OIT organizacije, istraživačke konzultacije i savjetodavne usluge, razne radionice, telekonferencije, itd.

implementaciju. Kako bi odabrali najadekvatniji BPMS za potrebe organizaciju potrebno je razmotriti nekoliko faktora koje ćemo razraditi u sljedećem poglavlju (2.2.3.2).

2.2.3.2 Kako odabrati odgovarajući BPMS?

Da bi se uspješno odgovorilo na postavljeno pitanje iz naslova, potrebno se usredotočiti na poslovne potrebe koje postoje u organizaciji. PP-i čine temelj svake organizacije. Utječu na poslovne ciljeve, financijsko stanje, produktivnost, konkurentnost na tržištu, te zapravo određuju uspjeh ili neuspjeh organizacije. Pravi BPMS omogućit će da se uz pomoć jednog cjelovitog proizvoda preoblikuje kompletni životni ciklus organizacije, dakle procese usredotočene na čovjeka (oni koji traže interakciju s pojedincima u organizaciji) te one usredotočene na druge sustave (integracija s ERP ili CRM[15] sustavima) [4]. Odabir jednog sveobuhvatnog BPMS-a, umjesto više različitih aplikacija, značajno će smanjiti troškove, povećati fleksibilnost i spretnost u poboljšanju određenih procesa [18].

Da bi najučinkovitije iskoristili sve što nude najnovija BPM rješenja na tržištu moramo najprije proučiti sve opcije koje su nam dostupne, te prema tome odabrati pravo rješenje za svoju organizaciju [44]. Kvalitetan alat može olakšati težak posao, međutim pronaći takav alat nije uvijek lagana zadaća, a i samo korištenje alata zahtijeva određeni stupanj vještine [19].

Neki od faktora koji utječu na odabir BPMS-a su:

- **Specifični zahtjevi projekta**

  Najvažniji faktor kod odabira BPMS-a je razmotriti što se želi postići, to jest koji problem unutar organizacije se planira riješiti. Treba imati na umu da se zapravo traži način na koji će se najjednostavnije zadovoljiti potrebe organizacije, a ne sam BPMS. Ako BPMS ne podržava sve potrebe organizacije, specifični problemi rješavaju se uz pomoć drugih alata, te se onda uključuju u rješenje za BPM. Takve vanjske komponente nisu u potpunosti pod kontrolom BPM sustava te mogu uzrokovati nepredvidivo ponašanje, primjerice nepotrebna usporenja pri izvršavanju procesa.

[15] **Customer Relationship Management (CRM)** je model za upravljanje interakcijom tvrtke sa sadašnjim i budućim klijentima. Uključuje korištenje tehnologije za organiziranje, automatiziranje i usklađivanje prodaje, marketinga, raznih usluga te tehničke podrške

- **Kompleksnost**

  Važnu ulogu igra podržana razina kompleksnosti koju BPMS nudi. Primjerice, ako su procesi unutar organizacije niske zrelosti prema CMM[16] modelu, fokus organizacije leži na analizi, otkrivanju i definiranju vlastitih poslovnih procesa. Takva situacija zahtijeva jednostavan BPMS čija primarna uloga je modelirati proces te ga učiniti razumljivim i pristupačnijim za zaposlenike [17]. S druge strane, organizacija s visokim stupnjem zrelosti traži praćenje performansi u stvarnom vremenu u svrhu detaljnog izvješćivanja ili otkrivanja uskih grla. Isto tako, ovisno o stanju izmjerenih vrijednosti, potrebno je definirati pokretanje različitih događaja i uzbuna u stvarnom vremenu, kako bi organizacija mogla pravovremeno i ispravno reagirati u slučaju problema [17]. Razina kompleksnosti BPMS-a koji sadrži te mogućnosti daleko je veća od alata u prvom slučaju.

- **Opcije integracije s postojećim sustavima**

  Važna karakteristika BPMS-a, osobito u kontekstu problematike koja se obrađuje u ovom radu, je mogućnost integracije s vanjskim sustavima. Integracija podrazumijeva postojanje dobro definiranih API-a koja omogućuju pristup te izlažu metode za manipulaciju modeliranim poslovnim procesima i procesnim artefaktima. Bitan je i broj različitih načina na koje se integracija može odvijati, npr. uz pomoću aplikacijskih sučelja, servisa, razmjenom datoteka, automatskim obradama itd. Međutim, ove mogućnosti zahtijevaju i razvojne inženjere sa stručnim znanjima koji će navedena sučelja znati iskoristiti.

  Dobra integracija fokusirana je na smanjuje ljudske intervencije i pruža potpuno automatizirano rješenje. Integrirano BPM rješenje kreirano kvalitetnim BPMS-om omogućit će podršku u svim dijelovima životnog ciklusa PP-a [18].

- **Skalabilnost**

  Skalabilni i robusni BPMS omogućava proširenja arhitekture te tako podržava nepredvidiv rast poslovanja neke tvrtke. Često, dostupni BPMS-ovi na tržištu nisu projektirani da podrže opsežan razvoj poslovanja u slučaju da se opseg poslovanja tvrtke mijenja.

- **Vrijeme i način implementacije**

  Važno je u slučaju postojanja rokova kod izrade rješenja, a određeno je načinom, to jest metodologijom razvoja. Primjerice u slučaju korištenja agilnog iterativnog pristupa, projekt se podijeli na niz funkcionalnosti koje se zatim parcijalno isporučuju u primjenu. Tako

[16] **Capability Maturity Model (CMM)** je model ocjene zrelosti organizacijske strukture tvrtki nastao nakon proučavanja podataka prikupljenih od organizacija koje su radile za američko ministarstvo obrane.

korisnik ranije može koristiti dijelove funkcionalnosti, dati kvalitetne povratne informacije te odobriti daljnje financiranje projekta.

Način implementacije utječe na razvojni proces te je poželjno da BPMS podržava iterativni razvojni pristup simulacijama parcijalno dovršenih funkcionalnosti. Simulacije su interaktivne radionice gdje poslovna strana i stručnjaci iz OIT-a zajedno analiziraju trenutno stanje BPM projekta te donose konstruktivne smjernice za budući razvoj.

- **Praćenje i optimiranje**

  Razina praćenja i bilježenja povijesnih podataka izvršavanja procesa omogućava kvalitetne i detaljne analize tokova procesa, otkrivanje nedostataka te daje smjernice za optimizaciju.

- **Osigurana podrška i lakoća korištenja**

  BPMS kojeg organizacija odabere bit će okosnica IT infrastrukture kroz sljedećih nekoliko godina, stoga je jedan od važnijih faktora kvaliteta i vrsta podrške od strane proizvođača. Tvrtka koja istovremeno pruža softver, usluge implementacije, prilagodbe i kontinuirane podrške ključ je za uspjeh BPM inicijative. Podrška može biti tijekom implementacije ili kod održavanja gotovog rješenja, a dugotrajno partnerstvo partnerstva s dobavljačem BPMS-a, često je glavni pokretač uspjeha osobito u kritičnim trenutcima koji prijete poslovanju organizacije.

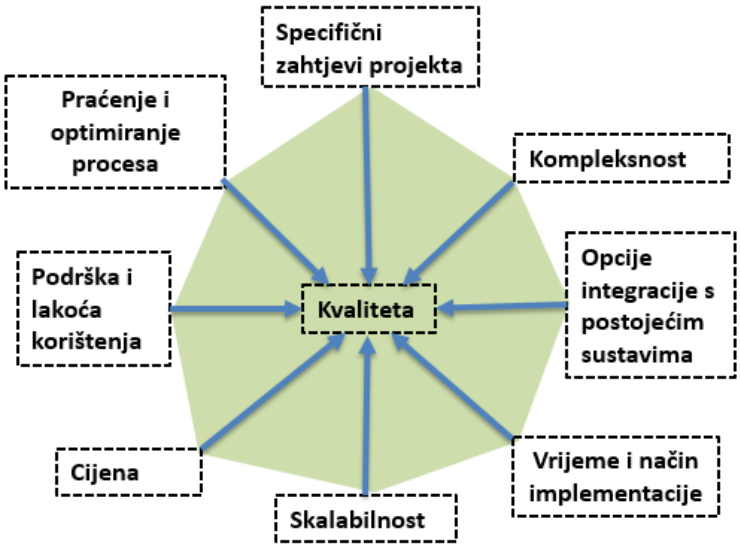
  Kako bi se osigurala kvalitetna implementacija, osim dobre podrške važna je i lakoća korištenja odabranog alata. Odgovori na pitanja poput: "Koliko traje minimalni period treninga?", "Podržava li alat istovremeni rad više korisnika?", "Postoji li pretraživanje po specifičnim funkcionalnostima?", "Postoji li centralni repozitorij procesa?", "Na koji način se radi administracija korisnika? ", "Koja je razina sigurnosti?" mogu nam dati dobru predodžbu o izazovima s kojima će se susretati ljudi zaduženi za rad s BPMS-om [21]. Izgrađeno BPM rješenje također mora pružiti sveukupni razumljiviji pogled na poslovnu strukturu te tokove poslovnih procesa organizacije i to za poslovne korisnike, analitičare i razvojne stručnjake.

- **Cijena**

  Često, odabir besplatnog BPMS-a može biti najskuplja opcija [20]. Cijena naravno mora biti usklađena s financijskim mogućnostima organizacije, međutim važnija je usklađenost specifičnih potreba s mogućnostima alata. Ukupna cijena za organizaciju uključuje više faktora poput adekvatnog hardvera na klijentskim i serverskim stranama, BPM alate i licence potrebne za rad s tim alatima, poduku osoblja te osiguranu podršku u slučaju problema [17].

Uštede se mogu realizirati kroz druge aspekte, primjerice besplatnu podršku ili besplatne edukacije i treninge za zaposlenike organizacije koja kupuje alat.

Kombinacija navedenih faktora, prilagođenih uvjetima i potrebama organizacije, rezultirat će kvalitetnim BPM rješenjem (Slika 2.5):



Slika 2.5. Faktori utjecaja na odabir BPMS-a

U idealnoj situaciji organizacije trebaju biti usredotočene na odabir alata koji zadovoljava njihove potrebe, jednostavan je za korištenje, ima dobru podršku te prihvatljivu cijenu. Na kraju, BPMS ne definiraju njegove bezbrojne mogućnosti ili cijena, već sposobnost da obavi konkretan zadatak koji je potreban određenoj organizaciji te da bude prihvaćen od strane tijela za donošenje odluka i krajnjih korisnika unutar organizacije [17]. Svaki od navedenih kriterija treba odvagnuti prema potrebama specifičnog projekta. Vrijeme je važno u slučaju ako je potrebno brzo završiti projekt, lakoća korištenja i brzo otklanjanje problema usmjeravat će prema alatu s kvalitetnom i dugotrajnom podrškom, dok na kraju veliku ulogu igraju i financijske mogućnosti organizacije.

2.2.3.3 Važnost iterativnog pristupa izgradnje BPM rješenja

Izgradnja procesne aplikacije te MPP zahtijeva dobru viziju cjelokupnog niza poslovnih događaja koje želimo opisati, pa je često teško odrediti s koje strane te kako pristupiti izgradnji.

Usvojena praksa kod izgradnje procesnih aplikacija (koja je u pravilu podržana i od strane BPMS-ova) je definirati razvojni ciklus kroz nekoliko iteracija. Za svaku od iteracija definiraju se ciljevi, a po završetku implementacije određenog skupa ciljeva slijedi rekapitulacija i validacija napravljenog provođenjem simulacije (engl. playback).

Cilj simulacija je provjera je li model u skladu sa stvarnim poslovnim procesom, te sakuplja li korisne povratne informacije koje će pomoći u daljnjem razvoju procesa. Simulacije se mogu obaviti u bilo kojoj iteraciji ugradnje projekta [31], a može se provesti više simulacija u različitim točkama pojedine iteracije. Simulacije mogu služiti i kao demonstracije izvođenja parcijalno dovršene BPMA poslovnim korisnicima, menadžerima te voditeljima OIT-a, a poželjno je i prisustvo osoba koje sudjeluju u samom procesu te ga u potpunosti razumiju [24]. Dodatno, simulacije pružaju validaciju da razvoj procesa ide u pravom smjeru i opravdava sredstva za projekt pa ih je stoga poželjno imati više u različitim iteracijama projekta [31], [60], [64].

Nepisano pravilo preporučuje minimalno četiri iteracije tijekom izgradnje BPMA koje su popraćene simulacijama, a svaka od iteracija mora imati jasno zadane ciljeve, očekivanja te ograničenja. Simulacije osiguraju kvalitetnu validaciju i analizu modeliranih elemenata unutar trenutne iteracije, a rezultiraju pitanjima, sugestijama i rješenjima koja se koriste u razvoju tijekom sljedeće iteracije.

Po kraju svake od iteracija, dobra praksa je napraviti snimku trenutnog stanja BPMA. Napravljene snimke mogu se koristiti za usporedbu napretka u razvoju između dvije verzije, a služi i kao sredstvo verzioniranja te omogućuje povratak na prijašnju verziju u slučaju određenih problema.

2.2.4 Upravljanje poslovnim procesima kroz nezavisni procesni sloj

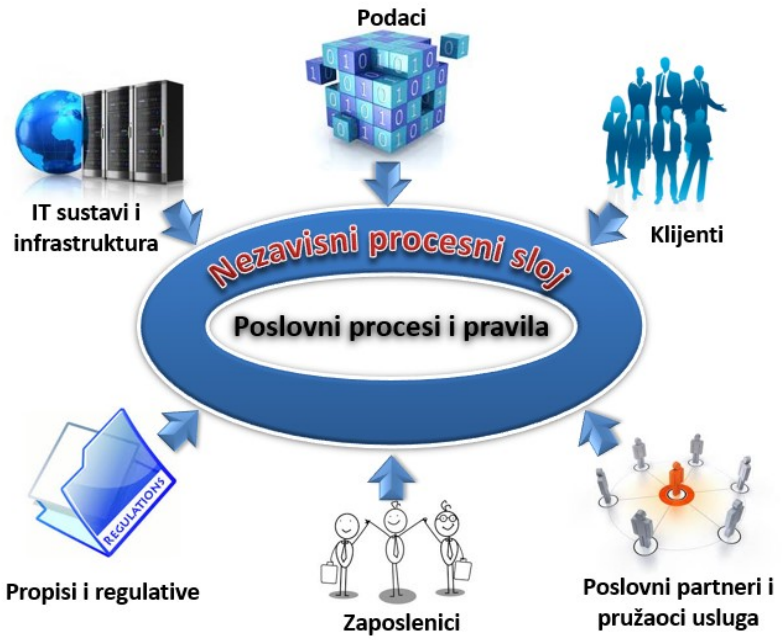
BPM zagovara procesno orijentiran pogled na IT, gdje se cjelokupno upravljanje procesima, od njihova početka do kraja, gleda odvojeno od aplikacija kroz koje se procesi provode, međusobnih veza tih aplikacija te podataka kojima one rukuju. Takav pogled zahtijeva stvaranje nezavisnog "procesnog sloja" koji sadrži informacije o svim aktivnostima od kojih se sastoji neki PP, bez obzira uključuju li te aktivnosti različite aplikacije, ljude ili njihovu kombinaciju.

Definiranje tako opisanog procesnog sloja nadilazi dvije ključne prepreke koje imaju tendenciju stopiti PP-e s infrastrukturom te tako uvelike povećati krutost IS-a:

- **postojeće aplikacije** koje su često međusobno usko povezane, čineći tako zapetljanu ne-robusnu arhitekturu koju je nakon ugradnje najčešće teško dodatno promijeniti i konfigurirati.
- **tokovi postojećih PP-a** koji se protežu kroz više postojećih aplikacija te često zahtijevaju ljudske korekcije i intervencije [28].

Upravljanje procesima kroz odvojeni procesni sloj omogućuje da se, relativno brzo, poveća stupanj automatizacije procesa kroz međusobno povezivanje postojećih sustava. Također pomaže u popunjavanju praznina između sustava koje je teško automatizirati, najčešće zbog potrebe ljudske intervencije. Uz to, pruža disciplinirani pristup upravljanju procesima tamo gdje su procesi jasno definirani te se od strane procesnog sloja mogu izvršavati, aktivno kontrolirati i mjeriti na svakom koraku izvršavanja (Slika 2.6 izgrađena prema [26]). Konačno, možda i najvažnija prednost odvojenog procesnog sloja je da poslovni korisnici mogu preuzeti vlasništvo nad svojim procesima te ih lako modificirati prema željama i potrebama.

Kako bi uspješno funkcionirao kroz cijelu organizaciju i obuhvatio sve ključne procese, nezavisni procesni sloj mora moći nositi se s velikim brojem različitih procesa različite kompleksnosti te omogućiti brzo kreiranje i uvođenje novih procesa. Menadžerima i poslovnim korisnicima nezavisni procesni sloj mora omogućiti da u što kraćem vremenskom roku definiraju i provedu promjenu u procesu [26].



Slika 2.6. Upravljanje procesima kroz nezavisni procesni

Koncept nezavisnog procesnog sloja karakterističan je i za BPMS-e koji kroz centralno mjesto, izgrađenu BPMA, omogućuju pristup svim resursima, funkcionalnostima i podacima koji su potrebni za obavljanje poslovnog procesa.

2.3 Razlike i sličnosti pristupa BPM i SOA

Kako bi što bolje razumjeli razlike između BPM i SOA pristupa, pogledajmo najprije što im je zajedničko. Oba pristupa imaju zajednički cilj, postići uspjeh u poslovnom okruženju.

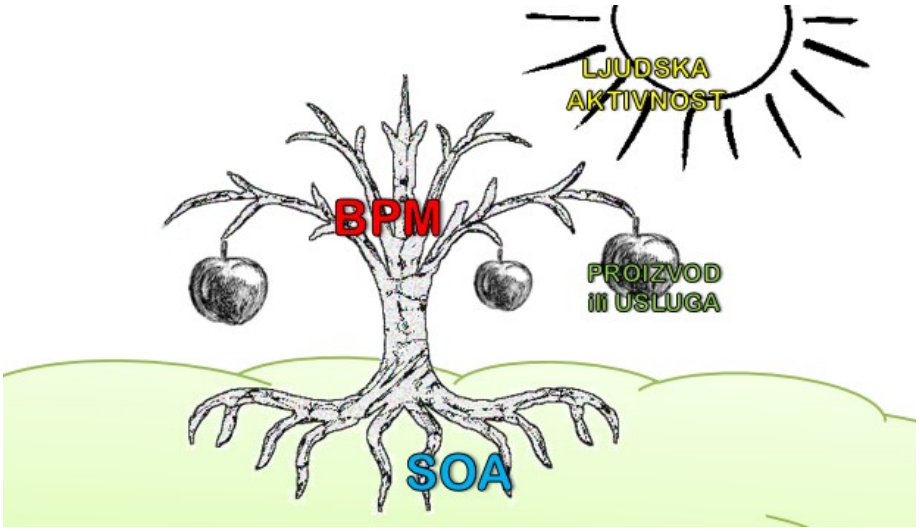
Općenito govoreći BPM se bavi izvršavanjem dobro definiranih procesa u određenom redu dok SOA za zadaću ima izlaganje funkcionalnosti određenog sustava pomoću dobro definiranih servisa [9]. BPM i SOA koncentriraju se na upravljanje i usklađivanje bilo procesa ili servisa, organiziraju ih u određene repozitorije te iz zatim prate i nadziru [3]. Kod oba pristupa potiče se praksa višestrukog korištenja istih elemenata (servisa ili procesa) te se podržavaju načela dinamičkih izmjena pojedinih elemenata. Svaki element dizajniran je s namjerom da bude što neovisniji od ostalih elemenata i sustava [4]. Po prirodi, BPM i SOA su iterativni procesi, gdje se kroz određene faze vrše predefinirani koraci kojima se izgradi, testira i analizira željena funkcionalnost, servis ili proces [7]. Nastoje integrirati PP-e u organizaciji, osigurati dostupnost ključnih podataka i tako isporučiti brzu i točnu uslugu za krajnjeg kupca.

Iako se SOA i BPM nadopunjuju, nalaze se na različitim razinama. BPM kao skup načela upravljanja nalazi se na višoj razini, dok je SOA na nižoj jer predstavlja skup tehničkih načela. BPM se više usmjerava na dekompoziciju procesa na potprocese, aktivnosti, općenito razgradnju složenih na jednostavnije (engl. top-down pristup) uz uvažavanje poslovnih i tehnoloških potreba. BPM osigurava da su PP-i međusobno povezani, automatizirani, optimizirani, praćeni i dokumentirani kako bi pružili učinkovitu podršku procesu donošenja odluka [37], [65].

S druge strane, SOA ima zadaću iz jednostavnijih servisa kompozicijom izgraditi složenije, koji pružaju više funkcionalnosti (engl. bottom-up pristup), uvažavajući samo postavljena tehnološka ograničenja [3], [7]. SOA ujedno daje i dozu fleksibilnosti arhitekturi IS-a [38], [65]. Organizacija bazirana na SOA principima je zapravo organizacija strukturirana kao kompozicija servisa [45].

2.3.1 Analogija iz prirode

Da bi jasnije razumjeli načela BPM-SOA integracije, pogledajmo primjer iz prirode. Stabilna poslovna organizacija, sa svojim SOA i BPM poslovnim modelom, je poput velikog stabla (Slika 2.7).



Slika 2.7. Prikaz međuovisnosti SOA-BPM

Korijeni stabla probijaju zemlju u potrazi za hranjivim tvarima, vodom i mineralima, koje zatim upijaju te dalje prenose tamo gdje su potrebni. SOA koristi isti princip, dohvaća sirove podatke, grupira ih u smislene poslovne cjeline i kroz servise omogućuje jednostavan pristup do tih podataka. Kao takva, ona služi poput mreže za popunjavanje praznine između infrastrukture, podataka i sustava za upravljanje poslovnim procesima [8]. SOA pruža tehnološke resurse koji osiguravaju sredstva za provedbu ciljeva definiranih s BPM-om.

S druge strane, deblo i grane stabla prenose hranjive tvari do dijelova na kojima su potrebne, kombiniraju ih uz pomoć sunčeve energije i tako grade sočan i zdrav plod. Grane i deblo ekvivalent su tehničke implementacije BPM međusustava koji grupira, sortira i prikazuje dohvaćene podatke na intuitivan i razumljiv način. Prikazani podaci nakon toga su podložni raznim korisničkim aktivnostima (energija sunca) uz pomoć kojih se formira kvalitetna usluga ili proizvod za krajnjeg korisnika (plod na stablu). Dakle, BPM predstavlja upravljanje resursima, disciplinu koja objedinjuje procese temeljene na ljudskoj interakciji s korištenjem tehnologije u svrhu stvaranja željene usluge ili proizvoda.

SOA predstavlja važan dio u pravilnom funkcioniranju BPM-a [7]. Primjerice, ako je stablo posađeno na stjenovitom tlu, korijeni stabla neće prodrijeti dovoljno duboko te neće pronaći dovoljno kvalitetnih hranjivih tvari. Stablo će biti pothranjeno, neće se pravilno razviti, te će proizvoditi male i gorke plodove.

S druge strane, ako se stablo posadi u hladu, korijenje će prikupiti kvalitetne hranjive tvari, ali budući da je stablo bez sunca i ima nerazvijene grane neće moći kvalitetno upravljati prikupljenim resursima. Kao rezultat toga izostat će kvalitetni i zdravi plodovi. Zaključno, ako imamo kvalitetnu SOA implementaciju s neučinkovitim BPM-om, ili samo lošu vezu prema BPM-u, možemo očekivati nisku kvalitetu konačnih proizvoda i usluga.

2.4 SOA kao podrška BPM-u

Čest problem tradicionalnih IS-a predstavlja velika složenost uz trajnu potrebu nadograđivanja postojećih funkcionalnosti, razvoja novih uz poštivanje rokova te kontinuirano održavanje [10].

Ako se unutar tvrtke radi samo na temelju načela SOA (bez BPM-a), mogu se kreirati pouzdani servisi dostupni za višestruko korištenje što stvara prilagodljiv sustav koji podržava poslovni model, ali servisi neće imati mogućnost kontinuiranog poboljšanja i optimiranja. S druge strane okruženje koje se zasniva samo na BPM-u može izgraditi kvalitetne aplikacije za poslovanje poduzeća, ali te aplikacije će biti toliko međuovisne i povezane da poduzeće neće biti dovoljno prilagodljivo u slučaju potrebe proširenja poslovnog modela [10].

Korištenjem alata kao što su web servisi SOA pruža projektantima procesa resurse koje oni mogu povezivati, višestruko koristiti, te tako stvoriti PP-e neovisne o tehnologijama koje se nalaze u pozadini [4], [7]. Sukladno navedenom, modeliranje novih procesa u rješenju za BPM puno je brže i lakše jer SOA odvaja sam proces od implementacije pojedinih akcija tog procesa, to jest aplikacija koje te akcije obavljaju, a koje mogu biti napravljene u različitim tehnologijama [1].

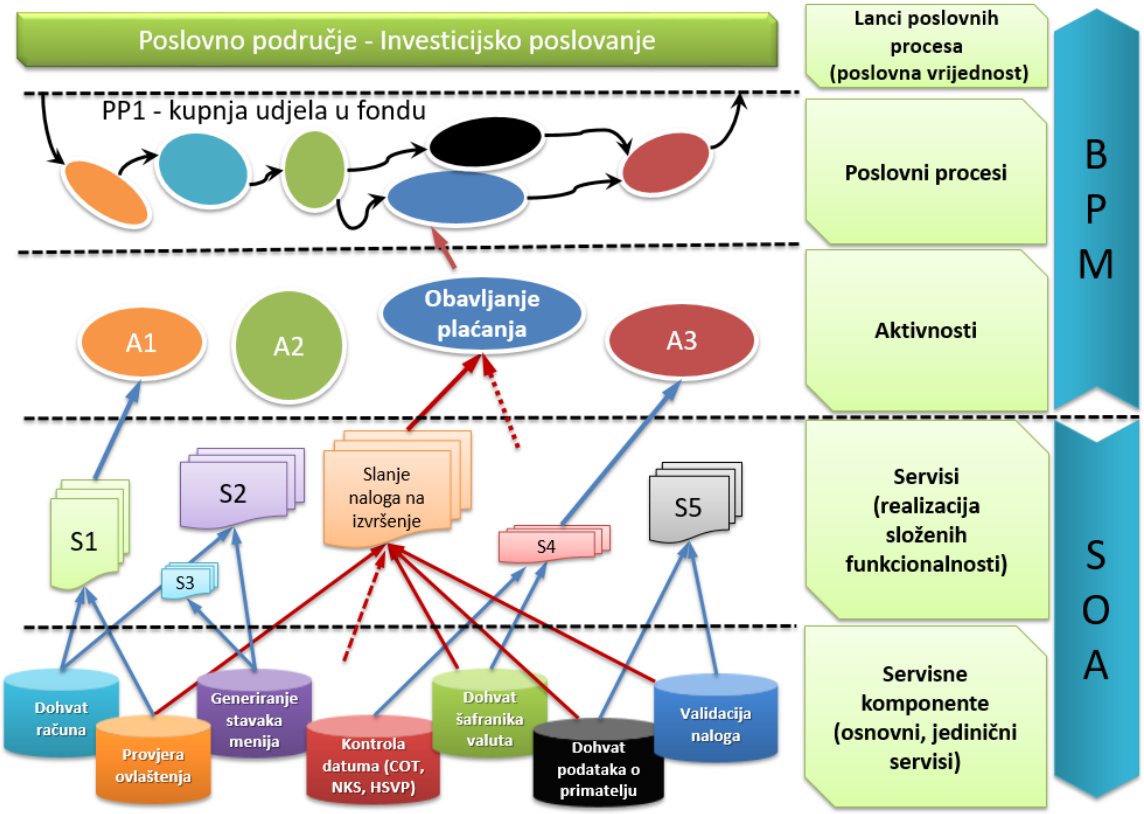
Ako u organizaciji postoji implementacija BPM-a i SOA-e, SOA servisi postaju podrška poslovnim procesima, mogu se uvijek iznova koristiti od strane procesa kojima je to potrebno, te tako ubrzavaju i olakšavaju cjelokupnu ugradnju rješenja za BPM [9]. SOA odvaja tok PP-a od implementacije tih procesa kroz servise i aplikacije koje obavljaju potrebne aktivnosti [1].

Konceptualni princip interakcije SOA-e i BPM-a možemo opisati kroz nekoliko slojeva koje po uzoru na primjere iz [3], [4], [7] i [8] prikazuje Slika 2.8. Svi slojevi navedeni su u nastavku te objašnjeni primjerima iz domene bankarstva:

- **Sloj servisnih komponenti (osnovnih jediničnih servisa)** – čine servisne komponente koje obavljaju elementarne funkcionalnosti. Servisna komponenta može se definirati kao samostalna jedinica funkcionalnosti [47], akcija koja nema određeno poslovno značenje. Primjer takve akcije može biti dohvat računa korisnika, ili validacija pojedinog polja na platnom nalogu.
- **Sloj servisa** – nastaje kombinacijom servisnih komponenti i tvori složenije operacije koje odrađuju neku smisleniju cjelinu; primjerice popunjavanje, validacija i slanje platnog naloga u izvršni sustav.
- **Sloj poslovnih aktivnosti** – objedinjava više servisa u smislenu cjelinu. Ta smislena cjelina naziva se poslovna aktivnost ako proizvodi neku vrijednost gledano s poslovne

strane. Aktivnost je glavna jedinica toka poslovnih procesa. Kao primjer aktivnosti možemo navesti provođenje platne transakcije uz obavljanje svih popratnih akcija: prijenos sredstva, ažuriranje stanja računa platitelja, primatelja te obračunavanje naknade.

- **Sloj poslovnih procesa –** uključuje niz aktivnosti koje rezultiraju ostvarenjem određenog poslovnog cilja u organizaciji. Primjer poslovnog procesa bila bi kupnja udjela u investicijskom fondu.
- **Sloj lanaca poslovni procesa** - više poslovnih procesa, s obzirom na sličnost ostvarenih ciljeva, čini određeno poslovno područje. Primjerice područje investicijskog poslovanja sadrži procese kupnje, prodaje i zamjene udjela u fondu.



Slika 2.8 Skica ovisnosti BPM-a i SOA-e na primjeru investicijskog poslovanja

Sloj osnovnih i složenih servisa čini SOA implementaciju koja je dio postojećih IS-a mnogih organizacija. SOA implementacija pruža podršku PP-ima organizacije te omogućava ostvarenje poslovnih ciljeva organizacije.

Unatoč dostupnosti raznih BPM i SOA softverskih rješenja na tržištu, puna integracija navedenih pristupa je i dalje ogroman izazov u praksi, većinom zbog opsežnih organizacijskih i tehnoloških promjena koje organizacije trebaju poduzeti [2]. Međutim, iskustvo mnogih organizacija je pokazalo da je BPM i SOA partnerstvo postalo učinkovito i nezamjenjivo

rješenje koje daje fleksibilnost PP-ima te omogućava organizacijama da se lakše nose s izazovima suvremenog poslovanja [3].

2.5 Osnovni pojmovi bankarstva poslovnih korisnika

Kako bi osigurali razumijevanje primjera i slučajeva prezentiranih u ovom radu, a koji se primarno odnose na bankarsko poslovanje s pravnim subjektima, ukratko ćemo objasniti relevantne pojmove.

Privredni, poslovni ili pravni subjekt, to jest organizacija koja ima otvoren jedan ili više poslovnih računa u banci, ugovoreno internetsko bankarstvo ili koristi neke bankarske usluge naziva se klijentom banke. Bankarsku uslugu možemo definirati kao skup financijskih proizvoda koji su namijenjeni rješavanju različitih financijskih problema bankarskih klijenata. Zaposlenici organizacija, koji u ime organizacije rade s bankom, koriste bankarske usluge, te imaju dodijeljene određene ovlasti kod korištenja tih usluga nazivaju se korisnicima ili službenicima.

Svaki klijent banke pripada određenom organizacijskom dijelu te na temelju toga ima dodijeljenu zaduženu osobu, menadžera za odnose s klijentima (engl. Relationship Manager ili RM) koji izravno brine o potrebama klijenta.

3 Postojeći postupci i ideja novog modela integracije BPM rješenja

Ovo poglavlje daje pregled postojećih istraživanja koja se bave ulogom i važnošću BPM-a u svakodnevnom poslovanju. Prikazani su pokušaji integracije BPM-a u postojeće sustave, te su naglašeni nedostaci navedenih pokušaja. Iznesena je ideja novog modela integracije, definirane su njegove osnovne karakteristike, te je opisana arhitektura i tok procesa u sustavu temeljenom na novom modelu.

3.1 Pregled i kritika postojećih istraživanja

Izgradnja kvalitetnog BPM rješenja te integracija tog sustava u postojeće okoline bila je tema mnogih projekata i istraživanja. Autori se slažu da BPM sustav donosi poboljšanje za strukturu i poslovanje organizacije, međutim postoje razilaženja u načinu kako taj sustav integrirati te kako postići poboljšanja. Svaki autor pristupio je problemu na svoj način. Poznate metode razvoja i integracije nadograđivale su se i prilagođavale kako bi zadovoljile specifične potrebe. Istraživači su gradili model rješenja koji je najbolje odgovarao potrebama okoline u kojoj su se nalazili. U nastavku dan je kratak pregled te kritički osvrt na postojeće radove u području relevantnom za temu ove disertacije.

3.1.1 Neki pristupi uvođenju upravljanja poslovnim procesima

Važnost procesno usmjerenog pogleda na poslovanje organizacije te potreba da se organizacije upoznaju sa svojim poslovnim procesima naglašena je u radu [35]. Autori ističu da organizacije trebaju automatizirati i kontrolirati svoje procese, kako bi se prepoznale potrebne izmjene, nadogradnje i buduće potrebe cjelokupnog poslovnog sustava. Dan je niz smjernica za pomoć pri predviđanju potencijalnih potreba specifičnog poslovnog sustava na temelju analize postojećih PP-a. Predlaže se i rudimentarna metoda za analizu i izgradnju poslovnih IS-a tako da se unaprijed sustavno identificiraju i kategoriziraju cjelokupne poslovne potrebe, što bi uvelike olakšalo daljnji razvoj sustava. Međutim, članak nedovoljno pozornosti posvećuje rezultatima predložene metode, te načinu kako te rezultate interpretirati i iskoristiti. Kako podržati identificirane poslovne potrebe, treba li iskoristiti postojeću arhitekturu informacijskog sustava, razviti nove modele te ih integrirati u postojeće tokove PP-a ili krenuti u kupovinu novog proizvoda? Ostaje upitno koje razvojne postupke treba koristiti da bi se osigurala kvaliteta novokreiranih procesa te kako te procese kontrolirati i održavati. Izostala je validacija metode na nekoliko različitih primjera.

U [23] autori istražuju i rangiraju po važnosti glavne faktore koji otežavaju provedbu BPM-a u malim i srednjim poduzećima. Prikupljanje podataka vrši se razgovorom s relevantnim stručnjacima i provođenjem opsežnog terenskog istraživanja u iranskom poslovnom sektoru. Članak pruža zanimljive uvide u izazove kod implementacije BPM rješenja u stvarnom životu. Međutim, napisan je isključivo iz perspektive upravljanja poslovanjem, fokusiran na osnovne BPM koncepte, te ne uzima u obzir velika i kompleksna poslovna okruženja.

Autor u [37] ukazuje na činjenicu da je preduvjet za kvalitetnu BPM integraciju odgovarajuće poslovno okruženje i adekvatan informacijski sustav koji će omogućiti kontinuirano poboljšanje PP-a. Na temelju iskustva u bankarskom sektoru, razvijen je teoretski model identificiranja čimbenika uspjeha BPM projekta. Pokušava se otkriti veza između BPM sustava i jačine te konkurentnosti organizacija na tržištu, tako da se istraže kompleksne veze između raznih aspekata organizacije, npr. PP-a, informatičke infrastrukture i reakcija na novonastale situacije. Naglašava se da je BPM jedan od ključnih faktora u poticanju organizacijskih promjena koje se trebaju desiti na putu do ispunjenja poslovnih ciljeva. Autor iznosi kvalitetne preporuke kako prihvatiti glavnu ideju BPM-a u poslovnim okruženjima, ali zanemaruje analizu tehnoloških sustava koji BPM čine mogućim. Osim toga, ne naglašava se veza između optimizacije poslovanja prilagodbom PP-a te tehnološke arhitekture koja tu prilagodbu omogućava.

U članku [14] može se vidjeti zanimljiv način utvrđivanja osnovnih elemenata koji podržavaju agilnost u upravljanju poslovnim procesima organizacije. Dana je preporuka integriranog pristupa projektiranju i preoblikovanju procesa, uz pomoć CMM modela za ocjenu zrelosti organizacije, kako bi se uspješno primijenili elementi agilnosti u cijelom poslovanju. Povučena je usporedba između stanja zrelosti sustava za upravljanje poslovnim procesima te agilnosti organizacije. Više stanje zrelosti PP-a rezultira visokom agilnošću te tako omogućuje organizacijama da brže reagiraju na promjene. Navedeni zaključak autor uglavnom pokazuje kroz analizu PP-a specifičnih za organizacije koje djeluju na "mladim" tržištima u razvoju. Ipak, dobiveni rezultati mogu se još detaljnije razraditi u svrhu stvaranja skupa smjernica koje će pomoći prvenstveno u adaptaciji, a onda i u razvoju sustava za BPM.

3.1.2 Utjecaj postojećih sustava na integraciju SOA-BPM i modeli integracije

Detaljna objašnjenja BPM i SOA arhitektura dana su u članku [9]. Autori pokušavaju utvrditi uloge pojedine arhitekture u sustavu koji je baziran na kombinaciji oba pristupa. Raspravlja se o važnosti BPMS te su objašnjene neke osnovne karakteristike koje bi ta okruženja trebala imati. Međutim, fokus članka leži na opisu poslovnog okruženja te definiciji uloga razvojnih

timova, poslovnih korisnika i voditelja projekta integracije SOA-BPM. Uglavnom se analizira ljudski faktor u svrhu uspostave uspješne radne skupine za provedbu procesa integracije.

U članku [7] autori detaljno opisuju sličnosti i razlike BPM i SOA pristupa. Radi se analiza i usporedba struktura organizacija koje su bazirane samo na BPM ili samo na SOA principima, te se ukazuje na nedostatke takvih struktura. Sličnosti SOA i BPM istražuju se u svrhu pronalaska uspješnog načina integracije ta dva pristupa u svrhu poboljšanja poslovnih performansi organizacije. Prikazana je jednostavna skica arhitekture sa servisnim međuslojem koji grupira servise u odgovarajuće PP-e na temelju definiranih poslovnih pravila. Tako definirani PP-i čine procesni sloj nad kojim se nalazi skup korisničkih sučelja. Unatoč dobroj ideji, servisni međusloj izgrađen je iznad sloja postojećih sustava, te ih tako koristi kao resurs umjesto da enkapsulira sučelja i funkcionalnosti tih sustava direktno u tok PP-a. Ako postojeći sustavi imaju korisnička sučelja, ta sučelja se ne koriste već se funkcionalnosti pojedinih aplikacija cijepaju u servise koji se zatim zovu s viših slojeva. Ovaj pristup je prihvatljiv za sustave projektirane tako da se većina poslovne logike nalazi u bazi podataka (npr. unutar procedura i funkcija), ali ne bi bio prikladan u slučaju kompleksnih aplikacija s puno poslovne logike na prezentacijskom sloju. Skica arhitekture prikazana je isključivo na konceptualnoj razini te je potrebna dodatna razrada za uspješnu implementaciju.

Istraživanje [3] otkriva kako fleksibilnost SOA pristupa utječe na glavne PP-e organizacije. Opisuje SOA-BPM integracijski model koji bi omogućio stvaranje fleksibilnog pogleda na strukturu PP-a organizacije. Model je kombinacija različitih postojećih tehnologija i provjerenih rješenja testiranih u produkcijskim okruženjima, a sastoji se od šest slojeva. Autori prezentiraju detaljni pogled na svaki od šest slojeva integracije iz tehničke perspektive. Međutim, model zahtijeva izdvajanje dijelova PP-a, poslovnih pravila te specifičnih poslovnih rutina u odvojene komponente koje će se onda koristiti za izgradnju složenog, ali učinkovitog BPM rješenja. Isto tako autori ne definiraju koju ulogu imaju postojeće aplikacije i postojeći PP-i u konačnom BPM rješenju. Tako se stvara dojam potrebe redizajna cijele informacijske arhitekture, umjesto da se nadograde i iskoriste postojeći sustavi. Ne razmatra se korištenje standardiziranog BPMS-a za izgradnju SOA-BPM integriranog rješenja.

Promatranja iznesena u [41] imaju naglasak na prevladavanju jaza između poslovnih odjela te OIT uz pomoć BPM okoline bazirane na SOA arhitekturi. Utvrđivanjem prednosti i slabosti glavnih postojećih pristupa za usklađivanje odnosa poslovne i IT strane (metodološki, transformacijski i pristup pojačanja), autori daju ideje za novo rješenje koje je bazirano na najboljim elementima svakog pristupa. Opća ideja istraživanja je poboljšati čitljivost PP-a te olakšati komunikaciju između poslovnih odjela i OIT-a, što je važan aspekt u razumijevanju

poslovnih zahtjeva i razvoju učinkovitih BPM rješenja. U radu je naglašeno da, kako bi BPM ispunio svoju zadaću, mora imati okolinu u koju se može integrirati tako da ga se poveže s cjelokupnim tokom postojećih procesa koji su realizirani postojećim aplikacijama, međutim ta tvrdnja nije detaljnije razrađena.

Članak [8] pruža pregled razvoja BPM rješenja koje je specifično za takozvane 3PL[17] organizacije. Opisana arhitektura BPM rješenja bazira se na razvojnoj okolini Apache Tuscany[18], koja pruža infrastrukturu za servisno usmjerenu pozadinu, te alatu OSWorkflow[19] koji ima ulogu upravljanja tokovima PP-a. Rješenje je osmišljeno tako da se stvori sučelje koje će integrirati PP-e unutar organizacije s procesima poslovnih partnera i krajnjih klijenata te tako osigurati učinkovitu komunikaciju svih sudionika procesa. Zanimljiva ideja koju autori spominju je modifikacija postojećih ili kreiranje novih programskih komponenti koje će pristupati postojećim sustavima te ih koristiti kao osnovne građevne jedinice BPM orijentirane okoline. Iako autori spominju ponovnu uporabu naslijeđenih aplikacija, razmatrano je samo korištenje postojećih servisnih modula. Koncept korištenja aplikacija se dalje ne razrađuje niti proučava. Stoga je potrebno dodatno istraživanje i razrada osnovnih koncepata da bi prezentirana ideja našla primjenu u stvarnom životu.

Općenita rasprava o SOA i BPM načelima te prednostima koje svaki pristup donosi poslovnom okruženju vodi se u [6]. Naglašava se važnost tehnologije u ulozi potpore poslovnim odlukama, za razliku od tradicionalnih sustava gdje su poslovne odluke uvelike bile ograničene tehnologijom. Autor se usredotočuje na opisivanje specifične, zasebno razvijene te o platformi neovisne servisno usmjerene arhitekture, definira opće obrasce razmjene podataka te načine komunikacije unutar opisane arhitekture. Optimizacija PP-a, koji su izgrađeni na opisanoj SOA arhitekturi, radi se orkestracijom postojećih SOA servisa. Sve postojeće aplikacije unutar organizacije trebaju se redizajnirati kao skup modula ili servisa koji odgovaraju definiranoj

[17] **3PL (Third Party Logistics) organizacije** su specijaliziranje tvrtke koje pružaju svojim klijentima širok spektar logističkih usluga te tako upravljaju dijelom ili cijelim opskrbnim lancem klijentske organizacije.

[18] **Apache Tuscany razvojna okolina** pojednostavljuje izradu SOA rješenja pružajući sveobuhvatnu infrastrukturu za razvoj i upravljanje SOA komponentama, a temelji se na standardu uslužne komponente arhitekture (engl. Service Component Architecture ili SCA). Pruža model za kreiranje složenih aplikacija definirajući pojedine servise, koji mogu biti implementirani u različitim tehnologijama, te njihove međusobne odnose.

[19] **OSWorkflow** je besplatan alat za upravljanje poslovnim procesima baziran na Java programskom jeziku, a jezgra alata čini konačni automat kod kojeg obavljanje određene akcije uzrokuje prijelaz iz jednog stanja u drugo. Omogućava korisnicima usredotočivanje na poslovnu logiku i pravila umjesto da pišu kompleksni programski kod. Integracija OSWorkflow-a obavlja se vrlo lako i brzo, osobito u postojeće Java projekte. Podržava sve načine rada koje možemo vidjeti u stvarnom životu, procese, uvjete grananja, petlje, uloge, itd. Opremljen je i grafičkim sučeljem koje omogućava praćenje tokova procesa kroz jednostavnije i složenije situacije.

SOA arhitekturi, te se mogu uvijek iznova koristiti kako bi podržali PP-e. Predložen pristup redizajniranja prihvatljiv je za manje i jednostavnije sustave, međutim ako postoje opsežne i kompleksne aplikacija taj proces može biti vrlo dugotrajan i težak, a ponekad i nemoguć zbog ograničenih resursa. Analiza SOA i BPM pristupa, te njihove međusobne koegzistencije, poduzeta je isključivo iz perspektive poslovnih i logističkih prednosti koje oba pristupa pružaju organizaciji. Autor ne uzima u obzir tehnička pitanja vezana uz integraciju SOA-BPM.

Članak [38] opisuje kako se SOA može koristiti kao alat za izradu fleksibilnog sloja za upravljanje poslovnim procesima. Definira se *servisno usmjeren poslovni proces* kao PP koji, osim poslovnih pravila, enkapsulira i servise, a ti servisi implementiraju pojedine funkcionalnosti procesa. To je pokušaj da se SOA ugraditi u BPM, te da se životnim ciklusom PP-a upravlja mijenjajući te optimirajući enkapsulirane SOA servise. Međutim, autor se primarno fokusira na funkcioniranje dovršenog sustava, optimizacije, analize učinkovitosti i utjecaja gotovog rješenja na ukupnu poslovnu fleksibilnost. Nedostaju opće smjernice o načinu razvoja opisanog rješenja, pogotovo u postojećem servisno usmjerenom okruženju.

Meta model prezentiran u [40] specificira interakciju između aktivnosti PP-a te funkcionalnosti koje pružaju programski servisi. Primijenjena je metoda analize PP-a odozgo prema dolje uz evaluaciju servisa s dna prema gore. Iako autori ne proučavaju u detalje vezu između modeliranja procesa i modeliranja servisa, predložena metodologija je korisna u definiranju iterativnih koraka za razvoj BPM rješenja. Nadalje, uz dodatni razvoj, metodologija bi se mogla primijeniti za otkrivanje izdvojenih dijelova poslovnih aktivnosti unutar postojećih sustava. Otkrivene aktivnosti se zatim mogu povezati u cjelovite PP-e te poput servisa uvijek iznova koristiti.

Glavna tema u članku [39] je istražiti kako se kombinacija SOA-e i BPM-a može iskoristiti za izgradnju učinkovitog sustava za određivanje cijena određene količine roba (engl. Apparel Quotation System). Dobivene informacije se zatim koriste u međunarodnoj trgovini roba i usluga. Prezentirana je troslojna arhitektura koja se sastoji od sloja PP-a, servisnog sloja te sloja pristupa resursima, a bazira se na centralnom servisnom repozitoriju. Servisni sloj služi kao poveznica između sloja pristupa resursima i procesnog sloja, a cijela arhitektura je specijalno prilagođena za proces određivanja cijena (engl. quotation process). Autori isto tako ističu da postoji potreba za daljnjim istraživanjima kako bi se odgovorilo na pitanje koji je najbolji način za integraciju BPM-a u postojeće sustave. Treba li graditi novi servisni sloj koji bi podržao potrebne funkcionalnosti upravljanja procesima ili se postojeći sustavi sa svim postojećim funkcionalnostima, uz ograničene modifikacije i transformacije, mogu ponovo iskoristiti kao standardna sučelja kod BPM integracije.

Istraživanje opisano u radu [2] predstavlja model PP-a koji se temelji na skupu raspodijeljenih servisa koje nude pružatelji servisa u sklopu poslovnog modela raspodijeljenih pružatelja usluga (engl. Software-as-a-Service ili SaaS business model). SaaS model uključuje davatelja usluge, koji u obliku raznih računalnih servisa pruža određenu funkcionalnost klijentu, a klijent zatim koristi te usluge da bi postigao svoje poslovne ciljeve. Postavljena je hipoteza koja pretpostavlja da raspodijeljena SOA okruženja osiguravaju veću fleksibilnost za tvrtke, optimalan skup servisa za podršku uključenim poslovnim procesima, te stoga daju više agilnosti u procesu razvoja ključnih aplikacija. Autori predlažu konceptualni model sustava za otkrivanje servisa (engl. service discovery model) koji bi bio koristan u raspodijeljenim okolinama, te koji bi mogao, dinamički, u stvarnom vremenu pronaći najprikladniji servis između mnogo potencijalnih opcija koje nude različiti pružatelji. Rješenje je korisno za tvrtke koje koriste računarstvo u oblacima ili imaju nastojanje delegirati svoje osnovno poslovanje vanjskim partnerima, te nije nužno da imaju potpunu kontrolu nad cijelim procesom stvaranja vrijednosti i usluge. Međutim, potrebe organizacije koja koristi podršku SaaS modela moraju biti dovoljno općenite kako bi raspodijeljeni pružatelj imao uslugu koja odgovara potrebama organizacije. Isto tako, zbog kompatibilnosti i lakšeg povezivanja, organizacije moraju poštivati općeprihvaćene norme i standarde branše u kojoj djeluju. U slučaju složenih ili visoko specijaliziranih poslovnih grana, gdje postoje jedinstvene poslovne aktivnosti, model raspodijeljenih pružatelja usluga neće omogućiti odgovarajuću podršku. Drugi potencijalni problem je osiguranje tajnosti poslovnih podataka, koji su u slučaju SaaS modela, dostupni i pružateljima usluga.

U [1] autor naglašava da je kombinacija SOA-e i BPM-a moderni nasljednik tradicionalnih ERP[20] sustava. Prikazan je pristup gdje se prema načelima objektno orijentiranog programiranja, SOA pokušava učahuriti u BPM u svrhu poboljšanja IS-a. Autor je uglavnom usredotočen na modularnost SOA-e i BPM-a te na temelju toga gradi vezu između ta dva pristupa. U predloženom integracijskom okviru, procesi su implementirani kao skup servisa te se optimizacija procesa vrši optimizacijom servisa. Ideja je stvoriti BPM rješenje u SOA okruženju ispočetka, bez korištenja postojećih aplikacija. Razvoj BPM rješenja uključuje stvaranje sučelja, MPP, kodiranje poslovnih pravila i slično, a sve to bez upotrebe BPM alata.

[20] **Enterprise Resource Planning** (ERP) je paket integriranih softverskih rješenja koje tvrtka može koristiti za prikupljanje, pohranu, upravljanje i tumačenje podataka iz mnogih poslovnih aktivnosti. ERP pruža integrirani pogled na temeljne poslovne procese, često u stvarnom vremenu, koristeći u pozadini zajedničke baze podataka te zajednički sustav za upravljanje tim bazama podataka.

U [42] je naglašena važnost ugovora o razini usluge (SLA[21]) kod izlaganja PP-a kao usluga određenom potrošaču tih usluga. Autor daje smjernice kako definirati uvjete izvođenja PP-a ovisno o konkretnim ciljevima koje zadaje poslovna strana. Nadalje, naznačena je važnost praćenja ključnih pokazatelja uspješnosti[22] (KPI), kroz cijeli životni ciklus procesa (faza modeliranja, konfiguracije te izvršavanja). Tako bilo kakva odstupanja od SLA mogu biti otkrivena na vrijeme. Međutim, autor se usredotočuje na mjerenje KPI-ova samo na razini PP-a bez da ih razgradi na indikatore nižih razina, razina koje služe kao podrška poslovnim procesima. Daljnja razgradnja KPI-jeva omogućavala bi preciznu identifikaciju nedostatka i uskih grla u pozadini PP-a, te bi se mogla napraviti konkretna poboljšanja i nadogradnje informacijske infrastrukture.

Pristup iz [66] pokazuje integraciju BPM sustava u okruženjima iz domene zdravstva. Fokus leži na dinamičkoj dodjeli dugotrajnih zadataka trenutačno dostupnim ili najprikladnijim timovima liječnika i zdravstvenih djelatnika. Autori definiraju zasebni međusloj koji povezuje semantički sloj (ontologiju i pravila koja se koriste za definiranje koncepata zdravstvene zaštite i načela upravljanja interdisciplinarnim timova zdravstvene zaštite) i izvršni sloj (PP-ovi definirani u BPM alatu zajedno s postojećim naslijeđenim bolničkim informacijskim sustavima). Kako je zdravstvena djelatnost grana u kojoj se većina posla obavlja u praksi, uloga postojećih bolničkih IS-a nije jasno definirana. Postojeći bolnički IS-i, uglavnom imaju ulogu pokretanja BPM procesa po dolasku pacijenta. Uz pomoć semantičkog sloja BPM se primarno koristi kao alat za dodjeljivanje zadataka i za upravljanje zadacima i procesima. Koncept ponovnog korištenja postojećih funkcionalnosti i aplikacijskih sučelja nije proučavan u detalje. Ne analiziraju se procesi koji se protežu kroz više naslijeđenih bolničkim IS-a. Dodatno, prezentirana studija slučaja je specifična, temeljena na jednom komercijalnom BPMS-u, bez da prvo definira generički model koji je primjenjiv na više naslijeđenih okruženja i BPMS-ova.

Autori u [65] bave se potrebama poslovne agilnosti u domeni osiguravajućih društava stvaranjem SOA arhitekture koja se sastoji od sustava za upravljanje tokovima procesa (engl.

[21] **Service Level Agreement** (SLA) ili ugovor o razini usluge, u strogo poslovnom smislu, je dio ugovora o pružanju usluga u kojima se formalno definira razina usluge te parametri odnosa dobavljača i primatelja. U tehničkom smislu mjerenja performansi SLA indikator označava agregaciju različitih KPI-a za određenu instancu procesa u određene smislene cjeline koje odgovaraju stavkama SLA ugovora.

[22] **Ključni pokazatelji uspješnosti** (engl. **Key Performance Indicator -** KPI) je vrsta indikatora kojim se obavljaju performansna mjerenja. Određena tvrtka može koristiti KPI-e da bi izmjerila uspjeh svog ukupnog poslovanja, ili uspješnost obavljanja određene aktivnosti. Uspjeh je ponekad definiran u smislu postizanja određenog strateškog cilja, ali često je to kontinuirana operacijska učinkovitost, npr. proizvodnja bez defekata, potpuno zadovoljstvo klijenata. Sukladno tome, odabir dobrih KPI-jeva leži na dobrom razumijevanju onoga što je važno za tvrtke, što dalje ovisi o vrsti tvrtke.

Workflow Management Systems - WfMS), ESB-a, sustava za praćenje poslovnih aktivnosti (BAM) i sustava za upravljanje poslovnim pravilima (engl. Business Rules Management - BRM). Arhitektura koristi ESB kako bi se putem Remote Method Invocation (RMI) metode prozivali različiti servisi, no ne uzima se u obzir pojam korištenja aplikacija, sučelja i dijelova funkcionalnosti tih aplikacija umjesto izloženih servisa.

3.1.3 Zaključak

Postojeća istraživanja većinom se bave važnošću procesnog pogleda na poslovanje [35], analiziraju i rangiraju organizacijske faktore koji utječu na provedbu BPM rješenja [23], [37], ili razmatraju utjecaj BPM-a na poslovanje [9], [14].

Prikazani modeli implementacije BPM-a u organizacije su pretežno usredotočeni na izgradnju BPM rješenja ispočetka [1], bez da definiraju jasnu vezu s postojećim sustavima. U slučajevima kada se razmatraju postojeći sustavi, modeli su primarno fokusirani na prepisivanje postojećih funkcionalnosti i sučelja [7], preoblikovanje aplikacija, te izdvajanje dijelova PP-a, poslovnih pravila i specifičnih poslovnih rutina u odvojene komponente [3], [6].

Ponovo iskorištavanje ograničeno je na postojeće na postojeće servisne module naslijeđene uslužno usmjerene arhitekture, dok koncept ponovnog korištenja gotovih postojećih funkcionalnosti i aplikacijskih sučelja nije proučavan u detalje [8], [65], [66]. Dodatno, većina postojećih BPM rješenja, integriranih u postojeća SOA okruženja, temelji se na činjenici da je rukovanje poslovnim podacima usko vezano uz korišteni BPMS [1]. Čest slučaj je izgradnja dodatnog sloja nad postojećim sustavima što dodatno komplicira postojeću arhitekturu sustave umjesto da doprinosi fleksibilnosti i agilnosti cjelokupnog poslovanja tvrtke.

Iako se proučava interakcija između aktivnosti PP-a te funkcionalnosti koje pružaju programski servisi [40] naglašava se da nedostaju opće smjernice o načinu razvoja BPM rješenja, pogotovo u postojećim servisno usmjerenim okolinama [38].

Postavlja se pitanje treba li graditi novi servisni sloj koji bi podržao potrebne funkcionalnosti upravljanja procesima ili se postojeći sustavi sa svim postojećim funkcionalnostima mogu ponovo iskoristiti kao dijelovi BPM rješenja [39].

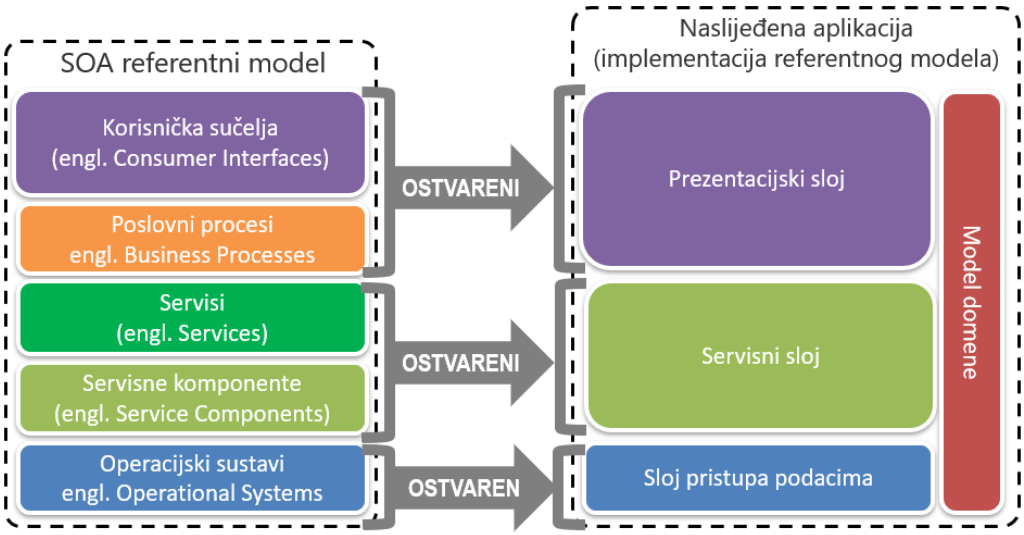
Naglašava se važnost povezivanja BPM-a s cjelokupnim tokom postojećih procesa koji su realizirani postojećim aplikacijama [41], međutim nedostaju modeli koji navedenu vezu opisuju.

Potreba za daljnjim istraživanjima proizlazi iz činjenice da još uvijek ne postoji dovoljno modela integracije BPM-a u postojeće servisno usmjerene okoline. Samo nekoliko studija se bave analizom alata i implementacijskih metodologija za ostvarivanje opisane integracije.

Radovi koji opisuju načine integracije BPM-a u postojeće okoline uglavnom ne daju detaljan i sveobuhvatan pregled. Ne razmatra se konceptualni model te uloga komponenti modela temeljem kojeg je integracija postignuta. Nedostaje sveobuhvatni opis korištenih principa i postupaka za provođenje integracije. Tehnologije, alati, radni okviri i smjernice za provedbu integracije nisu razrađeni dovoljno detaljno. Često nedostaje primjer provedbe na konkretnom poslovnom slučaju, pregled situacije iz stvarnog života temeljem koje je model nastao ili evaluacija predloženog modela.

3.2 Klasični pristup integraciji BPM-a u naslijeđeni sustav

Nastavno na zaključke iz poglavlja 3.1.3 ilustrirat ćemo osnovni klasični pristup implementaciji BPM-a temeljem navoda iz literature. U svrhu ilustracije prikazat ćemo naslijeđenu aplikaciju izgrađenu prema OASIS referentnom modelu SOA (Slika 3.1).



Slika 3.1 Struktura naslijeđene aplikacije temeljena na SOA referentnom modelu

Sloj **operacijskih sustava** referentnog modela SOA pruža pristup resursima poduzeća, a najčešće uključuje programe i podatke operativnih sustava poduzeća kao što su transakcijski sustavi, spremišta i baze podataka. Predočen je **slojem za pristup podacima** naslijeđene aplikacije koji primarno služi za dohvat te manipulaciju s podacima iz različitih baza podataka u organizaciji.

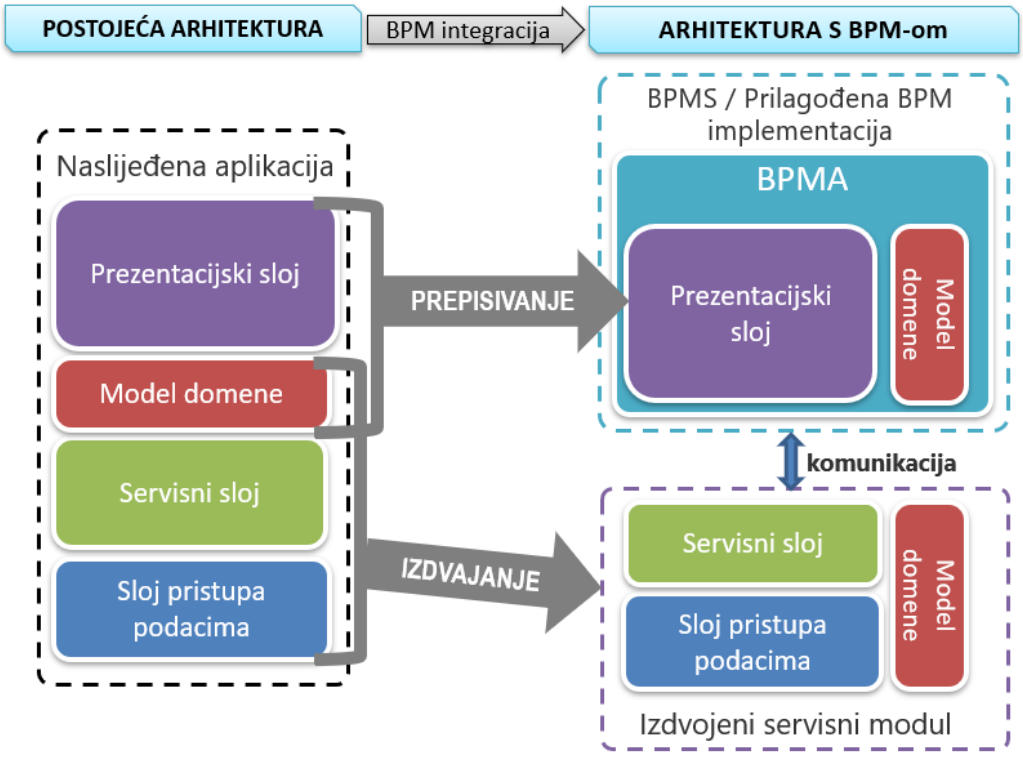
Sloj **servisa** i **servisnih komponenti** referentnog modela SOA predstavljen je **servisnim slojem** naslijeđene arhitekture. **Servisne komponente** su samostalne jedinice funkcionalnosti

koje uz pomoć jednostavnih poslovnih pravila grupiraju dohvaćene podatke. Sloj **servisa** koristi više **servisnih komponenti** te dodatno implementira pozive prema drugim sustavima organizacije i vanjskim sustavima poslovnih partnera.

Sloj **poslovnih procesa** referentnog modela SOA sadrži poslovnu logiku koja povezuje servise u svrhu kreiranja procesnog toka, dok sloj **korisničkih sučelja** sadrži resurse, pravila i korisničke ekrane potrebne za kontroliranu i centraliziranu interakciju s poslovnim procesom. Navedeni sojevi referentnog modela SOA predstavljeni su **prezentacijskim slojem** u arhitekturi naslijeđene aplikacije.

U arhitekturi naslijeđene aplikacije prikazan je još i **model domene** ili domenski model koji uz pomoć skupa objekta (artefakata iz stvarnog svijeta) opisuje poslovno područje organizacije. **Model domene** rezultat je izgradnje aplikacije temeljem DDD pristupa (engl. domain-driven design) i ključan je element SOA-e jer pomaže u enkapsuliranju poslovne logike i pravila u objektima domene [74]. Objekti modela domene koriste se u svim dijelovima aplikacije.

Temeljem prikazane naslijeđene aplikacije, možemo demonstrirati integraciju BPM-a u naslijeđeni sustav klasičnim pristupom. Slika 3.2 prikazuje arhitekturu sustava prije i poslije klasične BPM integracije.



Slika 3.2 Arhitektura sustava bez i s BPM-om, klasični pristup

Osnovni nedostaci ovog pristupa:

- velik stupanj redizajna postojeće okoline i postojećih aplikacija što zahtijeva velik utrošak resursa i vremena kao što je prikazano na konkretnom primjeru u poglavlju 3.2.1.
- ovisnost poslovanja o BPM rješenju što:
  - otežava prodaju programskih rješenja organizacijama koje nemaju implementiran BPM (potreba opsežnijih dorada programskih rješenja prije prodaje)
  - poteškoće kod promjene BPM alata ili odluke o napuštanju BPM alata te izrade vlastite podrške za BPM
  - potencijalno uzrokuje smetnje u poslovanju u slučaju problema prilikom kompleksnih nadogradnji BPMS sustava
- ograničenja korištenih BPM alata koji su primarno fokusirani na praćenje toka procesa
  - izrada složenih, interaktivnih sučelja (korisnička sučelja BPM alata su često ograničena i uglavnom usredotočena na podržavanje toka procesa)
  - komplicirana poslovna logika potrebna za učinkovit prikaz podataka i korisničku interakciju
  - baratanje velikim i složenim modelom podataka u BPM alatima što otežava implementaciju
- glomazna arhitektura BPM rješenja koja stvara dosta ranjivu, nefleksibilnu, okolinu koju je teško održavati i učinkovito nadograđivati
- dodatne poteškoće kod procesa koji se protežu kroz više različitih naslijeđenih aplikacija.

Osim navedenih nedostataka opisanog pristupa, osnova uspješnog izvršavanja toka PP-a je komunikacija između različitih programskih rješenja te ljudi koji donose odluke. Velik dio izmjena i/ili preoblikovanja procesa u postojećim sustavima ovisi o tome koliko je teško promijeniti komunikaciju između pojedinih komponenti. Kad nema standardiziranog načina, komunikacija je zasebno sadržana u odvojenim monolitnim softverskim rješenjima što rezultira tzv. "špageti integracijom" gdje svi komuniciraju sa svima, a i komunikacija je strogo ovisna o sučeljima i podatkovnom modelu pojedine aplikacije. Tako izgrađeni procesi striktno su vezani za tehnologiju korištenu u pozadini te su stoga vrlo inertni na promjene koje postaju teške i skupe za provedbu [4].

3.2.1 Primjer iz bankarskog okruženja

Kako bi pobliže ilustrirali opseg promjena postojećih sustava prilikom implementacije BPM-a klasičnim pristupom, koji je opisan u prethodnom poglavlju, pogledajmo primjer bankarske naslijeđene aplikacije.

Aplikacija služi za kontrolu poslovanja pravnih subjekata, a od prvog dana primjene stalno je razvijana i nadograđivana kako su poslovne potrebe rasle i kako su se mijenjali regulatorni zahtjevi. Tijekom prošlih 10 godina na aplikaciji je konstantno radio tim od 8 ljudi, koji je uz aplikaciju nadograđivao pripadne baze podataka.

Osnovni podaci potrebni za rad aplikacije pohranjeni su u dvije baze podataka od kojih svaka sadrži model podataka sačinjen od oko 400 podatkovnih tablica. Aplikacija je povezana s 18 različitih sustava i drugih aplikacija u banci koje pružaju različite usluge, kao što su: upravljanje kreditnim rejtingom te izračun kreditnog rejtinga prema želji, usluge i funkcije kataloga proizvoda, izračun sveukupne izloženosti klijenta i povezanih grupa klijenata, upravljanje depozitima i rizičnim proizvodima, pristup raznim centralnim evidencijama i šifrarnicima, upravljanje likvidnošću, podršku za digitalna potpisivanja, opsežne funkcije izvješćivanja, pristup skladištima podataka, mogućnost verzioniranja i upravljanja s dokumentima, itd. Osim toga, postoji i veza prema vanjskim pružateljima usluga u slučaju konzultacija glede pitanja kao što su FATCA (Foreign Account Tax Compliance) ili verifikacija crnih lista klijenata (npr. Hrvatski registar obaveza po kreditima, skraćeno HROK).

Aplikacija je temeljena na radnom okviru Java Spring[23] (obrascu programske arhitekture MVC[24] koji je uparen s dinamičkim stranicama JSP[25] te jezikom JavaScript[26]) i uslužno

[23] **Spring framework** je skup biblioteka otvorenog koda koji pruža sveobuhvatan model konfiguracije i programiranja modernih poslovnih aplikacija baziranih na programskom jeziku Java. Osnovna namjena mu je pojednostavljenje razvoja kroz brojne module za integraciju s raznovrsnim tehnologijama te pružanje cjelokupne infrastrukturalne podrške. Tako Spring barata infrastrukturom dok se razvojni stručnjaci mogu baviti razvojem poslovnih aplikacija.

[24] **Model–view–controller (MVC)** je softverska tehnika za izradu korisničkih sučelja na temelju tri usko povezane komponente čiji je zadatak odvojiti prezentaciju podataka od samih podataka. Središnja komponenta, *Model*, sadrži potrebne poslovne podatke i poslovna pravila. Zatim, *View* služi za prikaz *Modela* korisnicima na različite načine, ovisno o potrebi. Treća komponenta, *Controller*, ima ulogu zaprimanja akcija od korisnika, koje zatim pretvara u instrukcije za *Model* i/ili *View*.

[25] **JavaServer Pages (JSP)** je tehnologija za razvoj softvera koja omogućava stvaranje dinamički generiranih internetskih stranica baziranih na HTML, XHTML i XML elementima te ugrađenim akcijama. Tako uz pomoć servleta, malih programčića koji se izvršavaju na web serveru, a specificirani su na web stranici, moguće je modificirati sadržaj web stranica prije nego što se stranica pošalje korisniku koji ju je zahtijevao.

[26] **JavaScript** je najpopularniji skriptni programski jezik koji se izvršava na strani korisnika, dakle u web pregledniku korisničkog računala. Skriptni jezici su programski jezici manjih mogućnosti, koji se sastoje od izvršnog računalnog koda, obično ugrađenog u HTML stranice. Cilj JavaScript jezika je dodati interaktivnost HTML stranicama.

usmjerenoj arhitekturi. Zbog različitih, često loših praksi te kompleksnosti aplikacije, prezentacijski sloj naslijeđene aplikacije sadržava velik dio poslovne logike čineći nešto što se naziva debela web arhitektura (engl. fat web). U kontekstu opisanih tehnologija (radnog okvira Spring) prezentacijski sloj sačinjavaju komponente pogled (eng. *View)* i kontroler (eng. *Controller)* arhitekture MVC, a debeli prezentacijski sloj znači veliku količinu poslovne logike ugrađene u kontrolere.

Kako bi se jasno vidjeli izazovi integracije opisane bankarske aplikacije s BPM-om, mora se dobiti slika o veličini aplikacije, a time i količina rada uloženog u njen razvoj. Još važnije, potrebno je usporediti veličine pojedinih slojeva aplikacije.

Sukladno prethodno prikazanoj općenitoj arhitekturi (Slika 3.1 i Slika 3.2) strukturu opisane naslijeđene aplikacije možemo analizirati kroz četiri osnovna sloja: sloj pristupa podacima, servisni sloj, prezentacijski sloj i sloj modela domene. Za izvođenje analize korišten je alat SonarQube[27] i razvojno okruženja Eclipse[28]

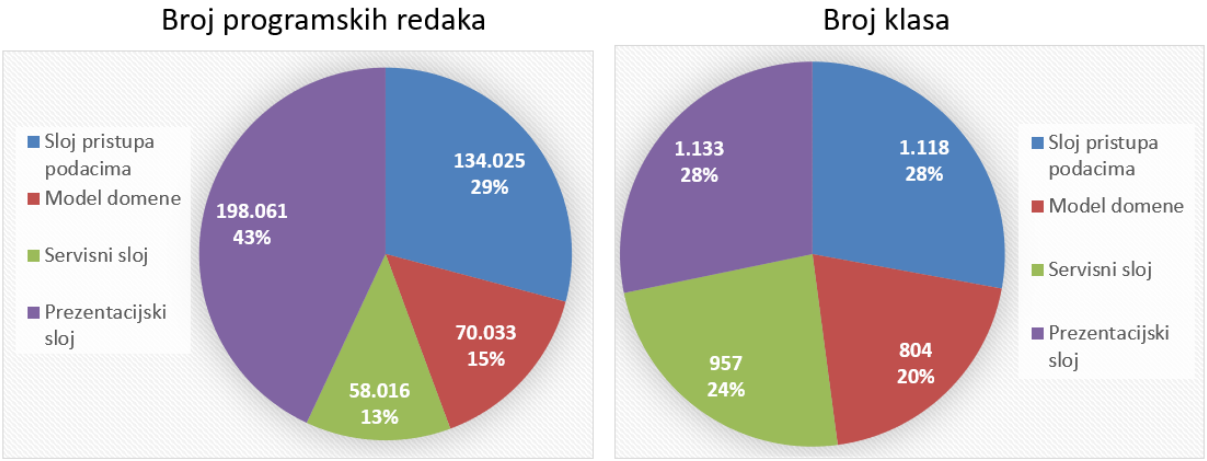
Analiza je provedena za različita aspekta:

1) Analizu na temelju broja programskih redaka (engl. Lines of Code - LOC), isključivo Java programskog koda. Razni obrasci za izvješća, predlošci, JavaScript, CSS (Cascading Style Sheets) i JSP komponente te XML konfiguracijske datoteke nisu uključene.
2) Slojeve aplikacije spram broja Java klasa
3) Broj konfiguracijskih i programskih datoteka po određenim slojevima aplikacije.

Slika 3.3 prikazuje analizu u prva dva slučaja. Vidljivo je da se velik dio programskog koda aplikacije (43% od ukupno 484.475 programskih redaka) nalazi na prezentacijskom sloju, a do sličnog zaključka dolazimo i ako usporedimo slojeve temeljem broja Java klasa od kojih se sastoje.

[27] **SonarQube** je platforma otvorenog koda za kontinuiranu provjeru kvalitete koda. Podržava različite programske jezike, nudi niz izvješća, pruža potpuno automatiziranu analizu, lako se integrira s vanjskim alatima (primjerice sustav za verzioniranje programskog koda, te se može proširiti uz pomoć raznih dodataka (eng. plugin).

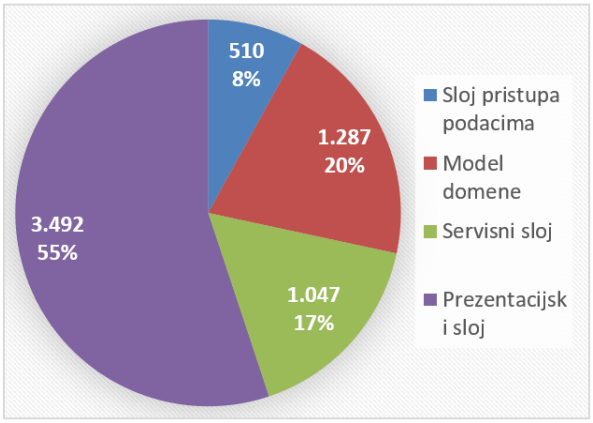
[28] **Eclipse** je programska razvojna okolina (IDE) pisana u Javi, a može se koristiti za razvoj aplikacija u raznim programskim jezicima kao što su Java, Ada, C, C++, COBOL, Perl, PHP, Python, R, Ruby, Scala, Clojure i Scheme.



Slika 3.3 Veličina slojeva bankarske aplikacije mjerena brojem programskih redaka i klasa

Analiza broja programskih redaka i broja klasa daje kvalitetan odnos u veličini pojedinih slojeva aplikacije, međutim uzima se u obzir samo Java komponenta aplikacije. Budući da aplikacija sadrži i druge (JavaScript, CSS i JSP te XML) komponente, napravljena je analiza prema broju različitih tipova datoteka (Slika 3.4).

Tip datoteke	Sloj pristupa podacima	Domenski model	Servisni sloj	Web sloj
.xml	20	32	57	63
.properties	5	16	49	317
.jrxml	0	0	59	554
.jsp	0	0	3	1.176
.tag	0	0	0	99
.css	0	0	0	31
.js	0	0	0	122
.java	485	1.239	864	1.129
.xls	0	0	15	1
UKUPNO	510	1.287	1.047	3.492



Slika 3.4. Raspodjela broja datoteka po slojevima naslijeđene aplikacije

Vidljivo je da se više od polovice datoteka nalazi u prezentacijskom sloju naslijeđene aplikacije (Slika 3.4).

Važno je naglasiti da aplikacija sadrži integraciju s fizičkim uređajima (čitači kartica, potpisne pločice te ostala PKI infrastruktura[29]), servisima za verifikaciju digitalnih potpisa i

[29] **Infrastruktura javnog ključa** (engl. **Public key infrastructure** ili **PKI**) skup je uloga, pravila i postupaka potrebnih za stvaranje, upravljanje, distribuciju, korištenje, pohranu i opozivanje digitalnih certifikata. Svrha PKI-a je olakšati siguran elektronički prijenos informacija kombinacijom tehnologije enkripcije i servisa koji organizacijama omogućavaju sigurnu međusobnu komunikaciju i poslovne transakcije. Koristi se kod niza mrežnih aktivnosti kao što su e-trgovina, internet bankarstvo i slični.

autorizaciju putem tokena, te pružateljima financijskih usluga kao što je FINA[30] (npr, izdavanje elektroničkih certifikata, napredno elektroničko potpisivanje dokumenata prema AdES[31] uredbi uz korištenje vremenskih žigova, itd.).

Ako se analizirana naslijeđena aplikacija želi integrirati s BPM rješenjem klasičnim pristupom (poglavlje 3.2, Slika 3.2) postupak bi bio odvojiti servisni sloj i sloj pristupa podacima te ga izložiti kao skup nezavisnih servisa. Te servise koristila bi nova BPM aplikacija. Međutim, da bi razvili BPM aplikaciju cijeli prezentacijski sloj trebalo bi prepisati koristeći dostupne BPM alate ili ga nanovo izgraditi u obliku vlastitog prilagođenog BPM rješenja.

Rezultati analiza pokazuju da prezentacijski sloj čini gotovo polovicu naslijeđene aplikacije koju je tim od 8 ljudi razvijao kroz period od 10 godina. Prepisivanje tolikog volumena aplikacije, nanovo, u vlastitom prilagođenom BPM rješenju ili koristeći BPM alate bio bi izrazito kompleksan, zahtjevan i dugotrajan zadatak koje si mnoge organizacije ne mogu priuštiti. Zato se u ovom radu predlaže alternativni način ugradnje sustava za BPM u postojeće uslužno usmjerene poslovne okoline s ciljem očuvanja postojećih sustava.

3.3 Ideja vlastitog modela integracije

Temeljem pregleda navedenog u poglavljima 2.2.3.1 i 2.2.3.2 važnost BPMS-a prilikom izrade BPM rješenja je neupitna. Ostaje pitanje na koji način koristi taj skup alata da bi iz njih izvukli maksimum u specifičnoj situaciji, uz resurse koji su nam na raspolaganju? Kao primjer odgovora može poslužiti izjava jednog upravitelja projekta koji je sudjelovao u istraživanju provedenom za ovaj rad:

„Često je jedini način korištenja određenog alata koristiti samo najbolje što taj alat nudi“.

Kako su BPMS-ovi izgrađeni s ciljem modeliranja i praćenja PP-a, osnovni dio svakog BPMS-a, tj. "najbolje što BPMS nudi", je procesni modul ili komponenta za modeliranje i upravljanje poslovnim procesima. Budući da je izrečena u kontekstu BPMS-a, navedena izjava

[30] **Financijska agencija** (FINA) vodeća je hrvatska tvrtka na području financijskog posredovanja. Nacionalna pokrivenost, informatički sustav iskušan na najzahtjevnijim poslovima od nacionalne važnosti te profesionalna stručnost timova omogućuje pripremu i provedbu različitih projekata, od jednostavnih financijskih transakcija do najsofisticiranijih poslova u elektroničkom poslovanju.

[31] **Advanced Electronic Signature (AdES)** je napredni elektronički potpis koji ispunjava uvjete iz uredbe Europske Unije broj br. 910/2014 (eIDAS-regulacija) o elektroničkim identifikacijama i povjerenju elektroničkih transakcija na unutarnjem tržištu EU.

upravitelja projekta ukazuje da komponenta za modeliranje i upravljanje poslovnim procesima podržava najbitnije funkcionalnosti BPMS-a, te je često preporučljivo koristiti samo tu komponentu.

Osim procesnog modula, BPMS ima predefinirane komponente koje omogućuju interakciju s tokovima podataka u modeliranim procesima. Te komponente često uključuju standardna sučelja, unaprijed općenito definirane akcije ili resurse, koji nisu uvijek dovoljni za specifične potrebe organizacija.

Kako su specifične potrebe organizacija već razrađene u postojećim sustavima i informacijskoj strukturi organizacija, ideja vlastitog modela integracije je kombinirati ključne funkcionalnosti BPMS-a i specifična rješenja implementirana u postojećim aplikacijama.

3.3.1 Osnovne karakteristike vlastitog modela integracije

Temeljem nedostataka klasičnih pristupa BPM integraciji identificiranih u poglavlju 3.2, analize praktičnog primjera iz poglavlja 3.2.1 te ideje prikazane u uvodu poglavlja 3.3 definirajmo karakteristike koje novi model integracije treba zadovoljiti:

- **Iskoristiti postojeće sustave, izbjeći opsežan redizajn**

  Osnovna karakteristika novog modela je očuvati postojeću informacijsku arhitekturu organizacije, to jest izbjeći opsežan redizajn i potpuno prepisivanje postojećih funkcionalnosti i postojećih PP-a uz pomoć BPM alata, novih programskih radnih okvira[32] i/ili izgradnjom novih aplikacija i servisnih modula.

  Maksimalno iskorištavanje postojeće arhitekture osigurava minimalnu potrošnju resursa, a time i nisku cijenu BPM implementacije. Također, smanjuje se vrijeme implementacije zbog smanjenog obujma posla.

- **Prilagodljivost**

  Označava labavu povezanost između BPMA i postojeće informacijske arhitekture. Labava povezanost postojećih aplikacija s BPM okruženjem ne stvara ovisnost poslovanja organizacije o BPM rješenju, olakšava prodaju programskih rješenja organizacijama bez BPM-a, ne stvara poteškoće kod odluke o promjeni ili napuštanju BPMS-a te ne uzrokuje potencijalne smetnje za poslovanje u slučaju problema prilikom kompleksnih nadogradnji

[32] **Programski radni okvir** (engl. software framework) u području računarstva je skup apstraktnih, općenitih, programskih komponenti, biblioteka programskog koda, skupova alata, programskih sučelja i predloženih metoda koje imaju ulogu pružanja neke generičke funkcionalnosti. Generičke funkcionalnosti moguće je selektivno mijenjati dodatnim korisničkim kodom, čime se omogućava razvoj specifičnog softvera za različite organizacije. Kao softversko okruženje, radni okvir olakšava razvoj softverskih aplikacija, proizvoda i rješenja.

BPMS sustava. Također, potencijalna ograničenja BPMS-a (smanjene mogućnosti korisničke interakcije, nemogućnost rješavanja specifičnih problema, glomazna arhitektura, problemi integracije, itd.) nemaju direktnog utjecaja na kvalitetu poslovanja organizacije.

Postojeće aplikacije moraju biti funkcionalne bez nadodanih BPMA što omogućava izvršavanje postojećih poslovnih procesa kroz postojeće aplikacije bez korištenja BPM rješenja.

Predloženi model mora moći raditi s različitim BPM alatima i to samo uz prilagodbu implementacije API-ja preko kojih se vrši interakcija s novim alatom. Tako se olakšava poslovna odluka o zamjeni BPMS-a; organizacija se lakše može prebaciti na drugi, bolji alat drugog proizvođača. U ekstremnim slučajevima, organizacija može odlučiti prestati koristiti BPMS zbog smanjenja troškova ili inicijative za razvoj tzv. in-house[33] BPM rješenja koje bolje odgovara njihovim potrebama.

- **Prikaz smanjenog skupa poslovno relevantnih podataka na BPMA**

  Za razliku od naslijeđenih aplikacija, BPMA mora raspolagati sa smanjenim skupom poslovnih podataka koji su neophodni za postizanje željene funkcionalnosti.

  Smanjeni skup poslovnih podataka sadrži osnovne informacije o procesu potrebne za inicijalnu korisničku interakciju s procesom, a definiraju ga poslovni korisnici koji navedene podatke odabiru temeljem vlastitog iskustva s procesom.

  Tako se izbjegava izgradnja složenog modela domene naslijeđenih aplikacija u BPMA. Nema potrebe za interaktivnim korisničkim sučeljima niti kompliciranom poslovnom logikom koja je nužna za rukovanje s podacima složenog modela domene. Također olakšavamo i ubrzavamo proces implementacije BPMA.

- **Mogućnost manjih, jednostavnijih izmjena tokova procesa kroz BPM alate**

  Treba omogućiti manje promjene u toku PP-a mijenjanjem modela procesa sadržanog u BPMA. To mora biti učinjeno uz minimalne promjene postojećih aplikacija, ili idealno, bez izmjena.

[33] **In-house** softver je softver koji određena organizacija proizvodi u svrhu korištenja unutar organizacije. Potreba za razvojem takvog softvera može nastati ovisno o različitim okolnostima, npr. nedostupnost softvera na tržištu, mogućnost ili sposobnost korporacije da razvije takav softver ili razvoj softvera zbog specifičnih potreba organizacije.

- **Univerzalnost**

  Idealno rješenje mora biti neovisno o hardveru, operacijskim sustavima ili tehnologijama koje se koriste u organizaciji. Integracija je uspješnija kada je komunikacija neovisna o arhitekturi i tehnologiji postojećih sustava.

- **Sinkronost**

  Tok svakog PP-a mora u stvarnom vremenu odražavati stanje podataka u postojećim sustavima. Promjene napravljene u procesnoj aplikaciji moraju biti automatski vidljive u postojećim aplikacijama, a stanje podataka u postojećim aplikacijama mora biti odmah dostupno u procesnoj aplikaciji.

- **Proširljivost**

  U skladu s ***iskorištavanjem postojećih sustava i izbjegavanjem opsežnog redizajna*** te nastavno na ***prilagodljivost*** važan aspekt modela integracije je ***proširljivost***. Ta karakteristika omogućava integraciju i korištenje različitih naslijeđenih aplikacija, razvijenih u različitim tehnologijama i iz različitih dijelova organizacije, u BPM rješenju. Proširljivost je ostvarena definicijom univerzalnih postupaka koji omogućavaju uključivanje različitih postojećih aplikacija u BPM rješenje, neovisno o tehnologiji postojeće aplikacije.

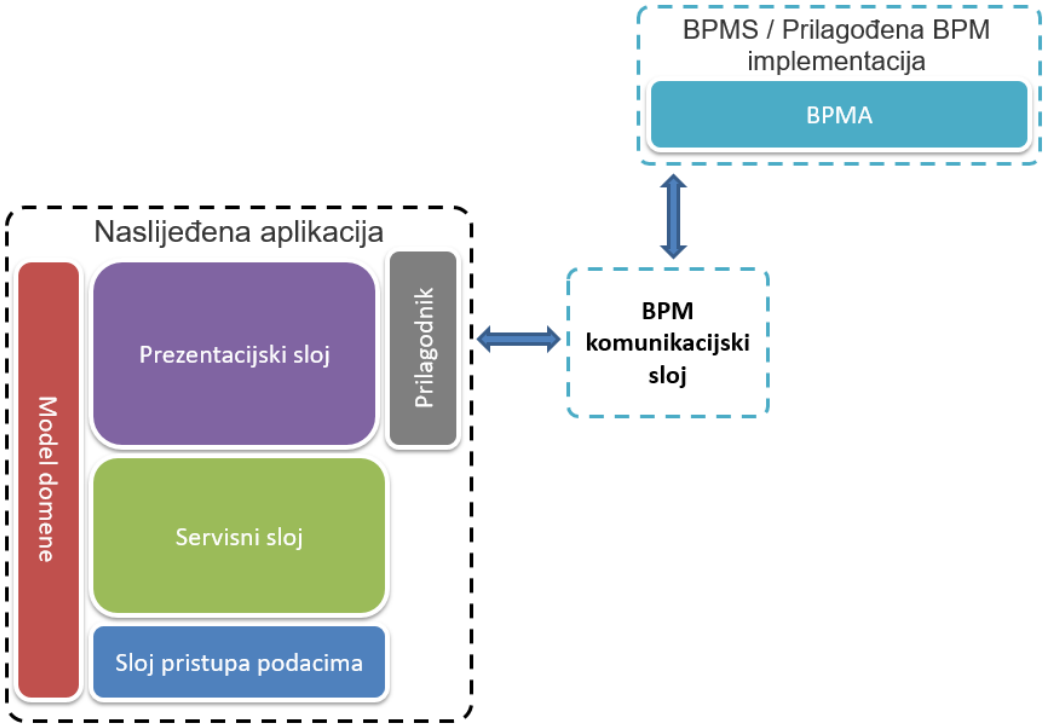
- **Centralizacija**

  Model mora omogućavati centralizirani i sistematiziran pristup aktivnostima i procesima koji se obavljaju kroz više različitih naslijeđenih aplikacija. Cilj je pružiti centraliziranu interakciju i automatizaciju PP-a, postaviti temelje za lakše praćenje, mjerenje te optimiziranje pojedinih dijelova procesa.

3.3.2 Pojednostavljena arhitektura sustava temeljenog na vlastitom modelu integracije

Suprotno klasičnim pristupima (poglavlje 3.2, Slika 3.2) te sukladno osnovnoj karakteristici vlastitog modela integracije (poglavlje 3.3.1) vlastiti model čuva arhitekturu i integritet naslijeđenih aplikacija.

Na uvelike pojednostavljenoj arhitekturi sustava s implementiranim vlastitim modelom (Slika 3.5) vidljivo je da se iskorištavaju svi dijelovi naslijeđenih aplikacija. U naslijeđene aplikacije uvodi se komponenta prilagodnik, koja zajedno s komponentom BPM komunikacijski sloj (BPMKS) ima ulogu "posredovanja" između naslijeđenih sustava te BPMA. Sve prikazane komponente detaljnije su opisane i razrađene u poglavlju 4, a ovdje su prikazane samo u svrhu ilustracije arhitekture novog sustava.



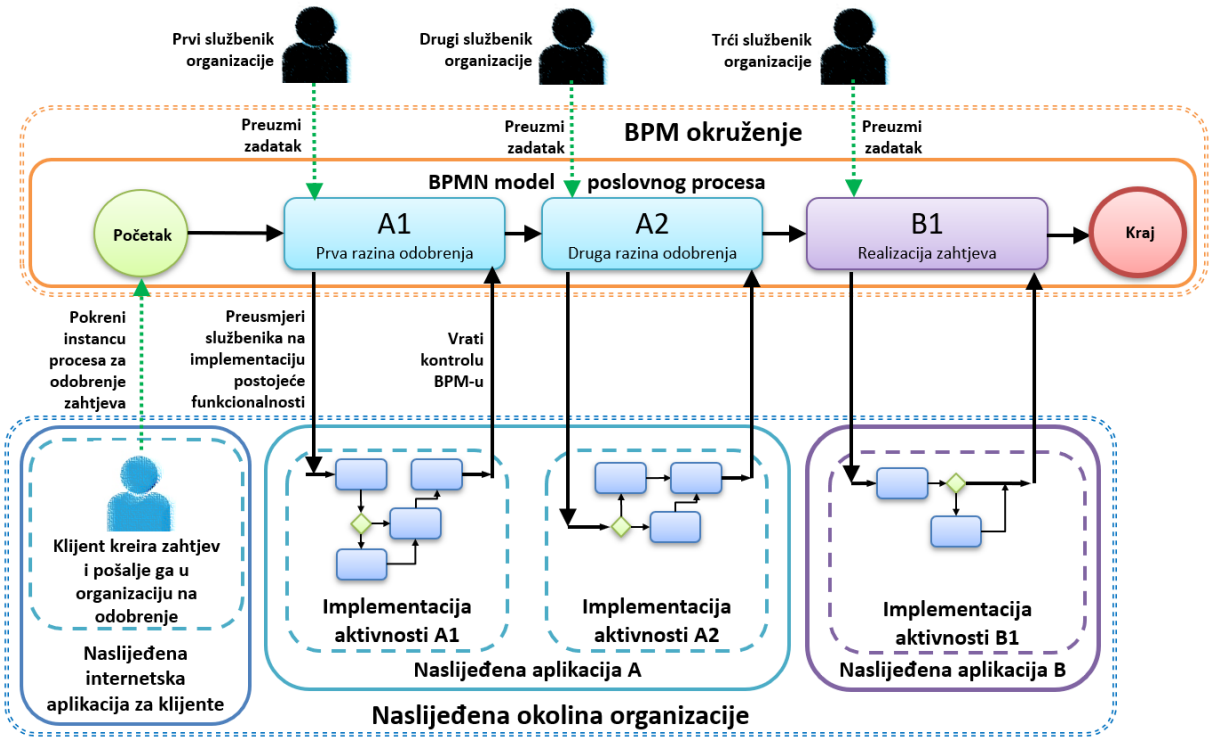
Slika 3.5 Pojednostavljena arhitektura sustava temeljenog na vlastitom modelu

Prikazana arhitektura sustava (Slika 3.5) oslanja se samo na jednu naslijeđenu aplikaciju. Model može uključivati više naslijeđenih sustava, što se jasnije vidi u primjeru iz sljedećeg poglavlja.

3.3.3 Tok procesa u sustavu temeljenom na vlastitom modelu integracije

BPMA izgrađena u BPM okruženju predstavlja centralnu točku interakcije službenika i PP-a. Kroz BPMA službenici preuzimaju dodijeljene zadatke te bivaju preusmjereni na obavljanje tih zadataka u naslijeđenim aplikacijama. Po izvršavanju zadataka u naslijeđenim aplikacijama, službenici bivaju vraćeni na BPMA gdje na isti način preuzimaju i odrađuju sljedeći zadatak.

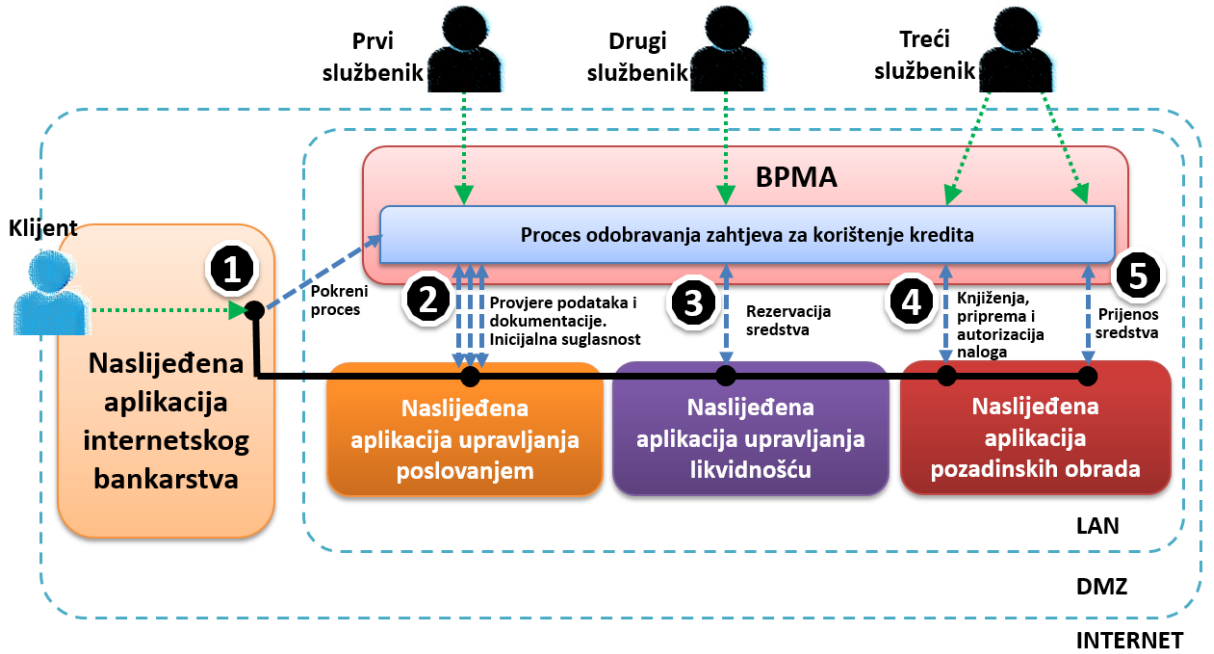
Slika 3.6 prikazuje generalizirani primjer opisanog postupka dok je detaljna razrada opisana u poglavlju 3.3.3.1 na konkretnom poslovnom primjeru.



Slika 3.6 Tok procesa u sustavu temeljenom na vlastitom modelu integracije

3.3.3.1 Pojednostavljeni primjer PP-a iz bankarstva

U ovom odlomku, ideja opisanog modela integracije dodatno je objašnjena na pojednostavljenom postupku odobravanja zahtjeva za korištenje kredita (ZKK) poslovnih korisnika (Slika 3.7).



Slika 3.7 Primjer procesa u sustavu temeljenom na vlastitom modelu integracije

Po odobrenju kredita, klijenti putem ZKK-a definiraju instrukcije isplate sredstva odobrenog kredita. Detaljan PP zadavanja i obrade ZKK-a objašnjen je u 6. poglavlju, a skica u ovom poglavlju prikazuje okvirni tok procesa, principe rada, te korisničku interakciju u sustavu temeljenom na vlastitom modelu integracije.

Pretpostavimo da je klijentu odobren kredit u banci. Kako bi dobio sredstva na račune svoje organizacije, klijent mora zadati ZKK i poslati ga u banku na odobrenje. Također, pretpostavimo da banka ima implementiran BPM temeljem predloženog vlastitog modela integracije. Proces zadavanja i odobrenja teče prema sljedećim točkama:

- **Točka 1:** Koristeći aplikaciju internetskog bankarstva klijent zadaje ZKK, popunjava potrebne podatke, formira instrukcije za isplatu sredstva i po potrebi prilaže dokumentaciju. Klijent zatim zahtjev potpisuje i šalje u banku na niz odobrenja. U ovom trenutku, preko integriranog prilagodnika (o čemu više u poglavlju 4.1.2.2 i 5.4), aplikacija internetskog bankarstva obavještava BPMA koja stvara novu instancu PP-a odobravanja ZKK-a, te pokreće prvu aktivnost procesa.
- **Točka 2:** Bankarski službenik (službenica) zadužen za klijenta koji je na odobrenje poslao ZKK (Slika 3.7 - *prvi službenik*), primjećuje da mu je dodijeljen novi zadatak na BPM rješenju, aktivnost davanja inicijalne suglasnosti (provjera podataka i dokumentacije) u sklopu odobravanja novog ZKK-a. Službenik preuzima zadatak te ga krene rješavati. Temeljem statusa i vrste zahtjeva određuje se koja postojeća aplikacija u banci je odgovorna za izvršavanje aktivnosti davanja inicijalne suglasnosti. BPMA zatim preusmjerava službenika na naslijeđenu aplikaciju, na točno određeni ekran u aplikaciji gdje se može izvršiti tražena akcija nad zadanim ZKK-om. Prilikom preusmjeravanja predaju se podaci potrebni za obavljanje aktivnosti, primjerice, vrsta aktivnosti, informacije o službeniku koji aktivnost izvršava, podaci o autentičnosti za inicijalnu prijavu u naslijeđenu aplikaciju, itd. Na ekranu za odobrenje u naslijeđenoj aplikaciji, službenik odabire željenu akciju s popisa dopuštenih akcija. Naslijeđena aplikacija izvršava odabranu akciju, provjerava ZKK i ažurira status zahtjeva u postojećem sustavu. U ovom trenutku naslijeđena aplikacija također obavještava BPMA o promjenama u stanju podataka i završava akciju inicijalnog odobrenja zahtjeva.
- **Točke 3, 4** i **5:** Aktivnosti iz ovih točaka slijedno se, ovisno o ovlaštenjima, dodjeljuju istom ili drugom bankarskom službeniku. Postupak izvršavanja tih aktivnosti analogan je postupku opisanom u **Točci 2**. Rezultat navedenih točaka je konačni status ZKK-a, te uspješni završetak procesa ZKK. Na kraju, o svemu se obavještava klijent koji je zahtjev zadao.

Temeljem prikazanog PP-a vidljivo je da se izbjegava izvršavanje akcija direktno iz BPMA-e. Koriste se funkcionalnosti naslijeđenih aplikacija pa nije potrebno razvijati kompleksna korisnička sučelja u BPM alatu ni graditi servise za izvršavanje iz BPMA. Također ne treba rekreirati cijeli postojeći model domene (masivni objekti s mnogobrojnim svojstvima) u BPMA, jer BPMA može funkcionirati sa smanjenim skupom podataka.

3.3.4 Osnovne prednosti vlastitog modela integracije

U odnosu na klasični pristup integraciji BPM-a (3.2) vlastiti model integracije ima sljedeće osnovne prednosti:

- Očuvanje postojeće arhitekture i postojećih aplikacija
  - Izbjegava se opsežno restrukturiranje i redizajn postojećih sustava što štedi vrijeme i resurse.
  - Mogućnost daljnjeg unapređenja i daljnjeg razvoja postojećih sustava
  - Specifične potrebe organizacije zadovoljene su postojećim specifičnim rješenjima koja se nalaze u postojećim aplikacijama
  - Izvršavanje aktivnosti PP-a korištenjem postojećih sustava
- Neovisnost poslovanja o BPMS-u i BPM rješenju
  - Mogućnost korištenja postojećih aplikacija bez BPM rješenja (obrazloženo u poglavlju 4.1.2.2)
  - Službenici većinu posla rade u postojećim sustavima te koriste postojeće znanje i vještine za rad u postojećim sustavima, što olakšava privikavanje na novi sustav
- Centralizirani pristup poslovnom procesu
  - Mogućnost praćenja toka procesa kroz različite aplikacije
  - Cjelokupna slika procesa i svih sudionika u procesu
- Mogućnost mjerenja u svrhu optimizacije
  - Otkrivanje nedostataka u postojećim sustavima
- Prikaz smanjenog skupa procesno najvažnijih podataka na BPMA
- Agregacija procesno najvažnijih podataka iz različitih naslijeđenih sustava na BPMA.

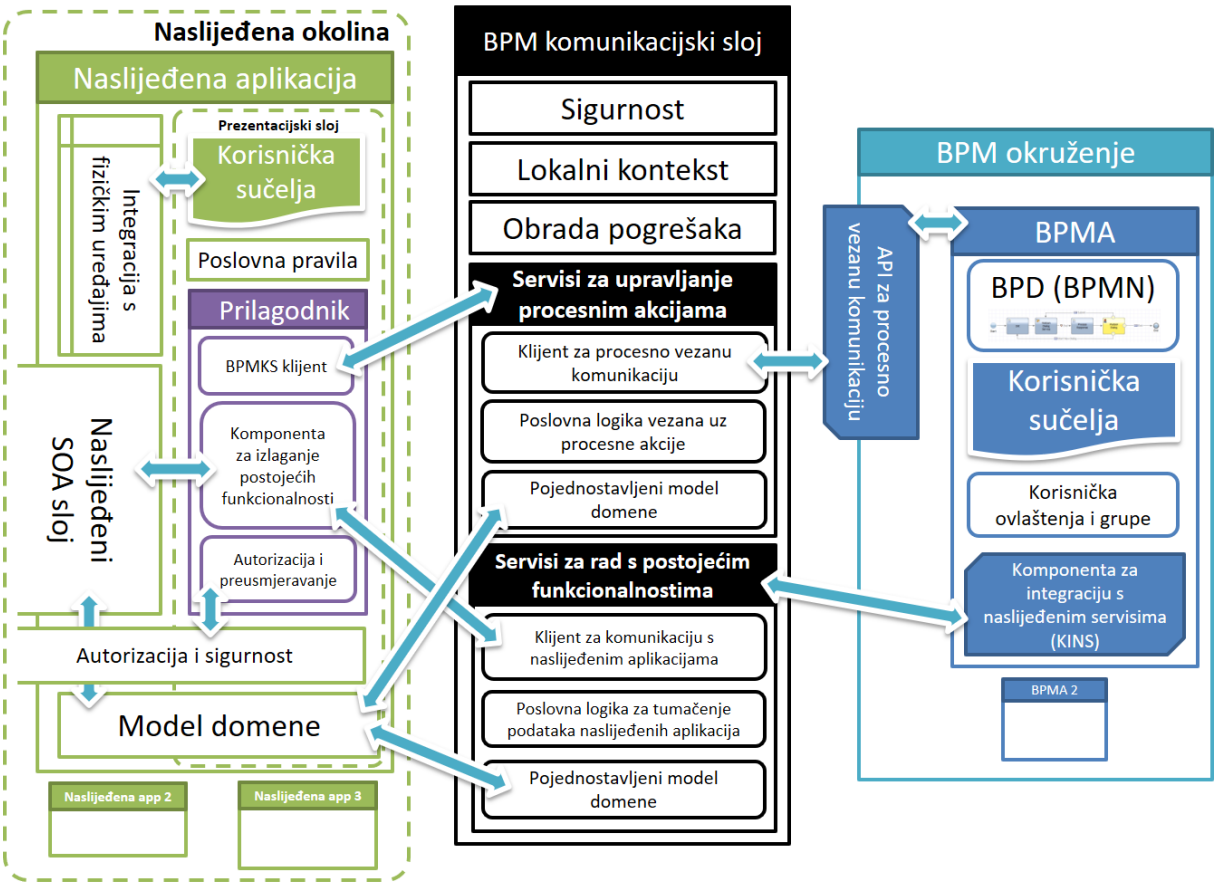
Detaljan prikaz prednosti, strukturnih inovacija i racionalizacija PP-a, korištenjem vlastitog modela integracije, izložen je u 6. poglavlju (potpoglavlje 6.4).

4 Vlastiti model integracije BPM rješenja u postojeću okolinu SOA

Pošto smo razjasnili osnovne koncepte, principe, ideju, te ustanovili prednosti modela definirajmo vlastiti model integracije. Kako bi ostvarili karakteristike opisane u poglavlju 3.3.1, postigli fluidan tok procesa, omogućili pristup podacima te jasno odredili poslovno okruženje, vlastiti prilagodljivi model integracije definiran je u tri ključna dijela:

- naslijeđene aplikacije (sustavi) koji sadrže dijelove funkcionalnosti postojećih PP-a
- BPMA koja BPMN dijagramima opisuje naslijeđene PP-e, izgrađena u BPMS-u
- BPM komunikacijski sloj (BPMKS) koji omogućuje univerzalan način komunikacije navedenih strana.

Slika 4.1 prikazuje konceptualni dijagram vlastitog modela integracije sa svim važnijim dijelovima. Svaki od prikazanih dijelova, zajedno s komponentama od kojih se sastoji, detaljno je objašnjen kroz poglavlja u nastavku, gdje su također opisani i svi korišteni mehanizmi.



Slika 4.1 Konceptualni dijagram vlastitog modela integracije

4.1 Naslijeđena okolina

Naslijeđena okolina sastoji se od jedne ili više naslijeđenih aplikacija i čini okosnicu postojećeg IS-a organizacije. Naslijeđeni SOA sloj predstavlja postojeću SOA implementaciju u naslijeđenoj okolini organizacije.

Naslijeđene aplikacije koriste funkcionalnosti naslijeđenog SOA sloja i tako odrađuju poslovne aktivnosti i podržavaju postojeće PP-e organizacije. Naslijeđene aplikacije također su integrirane s raznim fizičkim uređajima (čitači kartica, potpisne pločice, itd.) sadrže korisnička sučelja, poslovna pravila, model podataka, te ostale komponente koje omogućavaju obavljanje svakodnevnih zadaća poslovanja.

Kao dio predloženog vlastitog modela integracije, u naslijeđene aplikacije ugrađuje se nova komponenta, prilagodnik. Prilagodnik je centralno mjesto preko kojeg naslijeđena aplikacija komunicira s BPM rješenjem. Dodatno, to je komponenta koja BPM rješenju izlaže sve funkcionalnosti dostupne u naslijeđenoj aplikaciji.

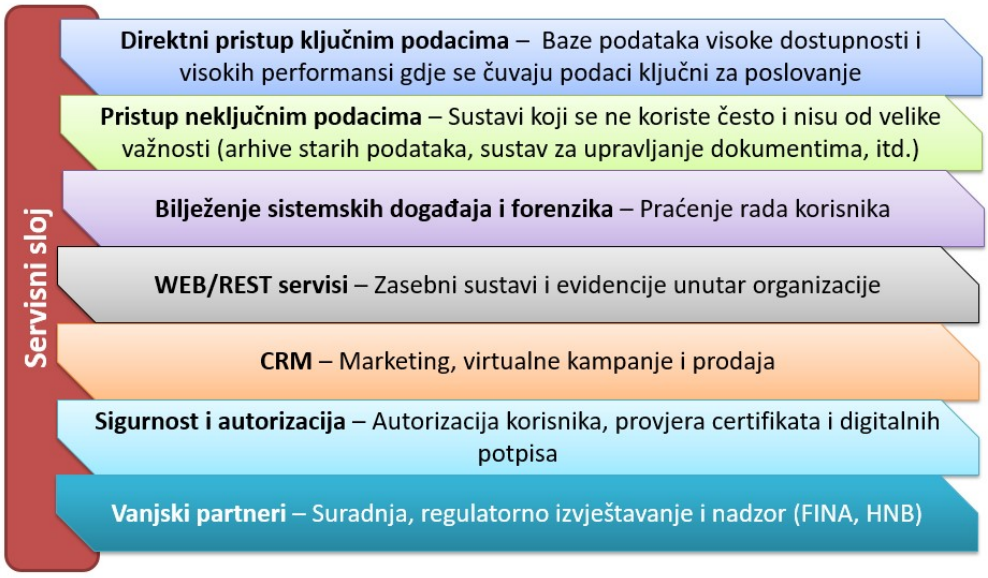
4.1.1 Naslijeđeni sloj SOA

Naslijeđeni sloj SOA predstavlja SOA implementaciju u organizaciji te tako omogućava korištenje postojećih funkcionalnosti organizacije. BPMA (poglavlje 4.3) uz pomoć BPMKS-a (poglavlje 4.2) te prilagodnika integriranog u naslijeđene aplikacije (poglavlje 4.1.2.2), može pozivati servise iz SOA naslijeđenog sloja te tako dohvaćati potrebne podatke ili izvoditi potrebne aktivnosti. Naslijeđeni SOA sloj karakterističan je za pojedine organizacije, a u pravilu je izgrađen je od sloja pristupa podacima i servisnog sloja.

Sloj pristupa podacima sastoji se od raznih poslužitelja, medija za pohranu svih vrsta podataka, različitih baza podataka na različitim strojevima i u različitim tehnologijama. Općenito svi resursi potrebni za funkcioniranje sustava inicijalno se dohvaćaju preko ovog sloja. Sloj uobičajeno sadrži i osnovnu logiku potrebnu za različite dohvate.

Servisni sloj sastoji se od različitih vrsta servisa koji unutar organizacije imaju određenu namjenu. Da bi lakše razumjeli skup dostupnih funkcionalnosti, grupiramo servise prema njihovim namjenama. Podjela servisa u grupe podiže performanse, omogućava lakše razumijevanje te održavanje čitavog sustava [4].

Općeniti prijedlog grupa servisa prikazan je na sljedećoj slici (Slika 4.2). U konkretnim organizacijama grupiranje može biti provedeno i drugačije ovisno o specifičnim potrebama.



Slika 4.2 Predložene generičke grupe servisa servisnog sloja

Predložene su sljedeće generičke grupe servisa:

- **Izravan pristup ključnim bazama podataka.** Ključne baze podataka barataju podacima koji su neophodni za osnovne funkcionalnosti poslovnog sustava organizacije. Primjerice u bankarskom sustavu internetskog plaćanja, to su baze podataka koje čuvaju informacije o platnim nalozima, računima klijenata, itd. Osnovna karakteristika ove kategorije servisa je brzina te trajna dostupnost baza podataka u pozadini.
- **Pristup resursima ili bazama podataka koje nemaju ključnu ulogu u funkcioniranju cjelokupnog sustava.** Tipični primjer bile bi arhivske baze podataka ili resursi s podacima koji nisu toliko važni i rijetko se koriste. Primjerice, arhivska baza podataka može sadržavati izvršene naloge klijenta koji su stariji od dvije godine. U slučaju da klijent traži naloge starije od navedenog perioda, njegov upit se preusmjerava na arhivsku bazu podataka. Pretraga po arhivskim bazama obično traje duže zbog velike količine podataka te performansi strojeva koji često budu slabijih karakteristika. Primjer neključnog resursa može biti sustav za upravljanje dokumentima (engl. Document Management System - DMS). DMS je skup programskih alata koji služe za organiziranje, praćenje i pohranu digitalnih dokumenata. Može se koristiti za spremanje raznih ugovora i potvrda koji su dostupni klijentu, ali nisu ključni za obavljanje primarnih funkcija kao što je primjerice platni promet u sustavu internetskog bankarstva.
- **Bilježenje sistemskih događaja, praćenje rada korisnika i forenzika**. Obuhvaća servise za bilježenje ključnih akcija kojima se prati rad i navigacija korisnika u aplikacijama. Za

potrebe ovih akcija dobro je u pozadini koristiti NoSQL[34] baze podataka koje imaju mogućnost bržeg spremanja veće količine podataka. Budući da ti podaci nisu od ključne važnosti u slučaju gubitka određenog zapisa nije počinjena velika šteta. Bilježenje sistemskih događaja se koristi u slučaju pojave raznih pogrešaka u aplikacijama te za analizu podataka ako dođe do pronevjere ili sigurnosnih problema.

- **Pozivi prema izloženim servisima zasebne aplikacije ili sustava organizacije.** Ovi servisi, u širokom smislu, predstavljaju komunikaciju između dvije aplikacije preko interneta (ili LAN mreže organizacije) te ih se primjerice može realizirati kao REST[35] ili SOAP[36] servise. Primjerice, kod aplikacije internetskog bankarstva pravnih osoba, skup podataka koji se dohvaća iz zasebnih sustava čine informacije o klijentima iz matične evidencije ili ipak dohvat podataka o računima fizičkih osoba.
- **Veza prema CRM sustavu**. CRM sustav, kao dio marketinga i prodaje, služi za informiranje klijenta o trenutnim pogodnostima, dodatnim ponudama te novim uslugama. CRM može biti nezavisni sustav unutar tvrtke gdje se nalaze sve kampanje, prilagođene pojedinom klijentu, te određenom kanalu distribucije. Osim toga, CRM može biti implementiran unutar pojedine aplikacije gdje se koristiti kao sustav za razmjenu poruka. Tako se klijent obavještava o promjenama u aplikaciji, razdobljima nadogradnje ili nekim novim uslugama i proizvodima.
- **Autorizacijske i sigurnosne provjere.** Ova grupa servisa služi za provjere autentičnosti kod prijave korisnika (npr. korištenje Validation Authority - VA poslužitelja za provjeru certifikata korisnika ili Single Sign On - SSO metode unutar organizacije), pruža način

[34] **NoSQL** baze podataka koriste mehanizme za pohranu i dohvaćanje podataka koji su različiti od mehanizama u konvencionalnim relacijskim bazama podataka (RDBMS). Motivacije ovog pristupa su jednostavnost dizajna, horizontalna skaliranja i bolja kontrola dostupnosti. Struktura podataka razlikuje se od RDBMS i zato su neke akcije dosta brže u NoSQL a bazama. NoSQL baza podataka pronalaze značajno mjesto u sve više rastućoj industriji internet aplikacija koje rade u stvarnom vremenu, primjerice Facebook, Twitter itd.

[35] **Representational State Transfer** ili REST opisuje skup arhitektonskih principa pomoću kojih se podaci prenose preko standardiziranih sučelja kao što je HTTP protokol i ne sadrži dodatni sloj za definiranje, to jest kreiranje poruka. Klijent može doći do određenih podataka ili resursa preko specifične URL (Uniform Resource Locator) adrese koja služi kao lokator podataka, dok su GET, PUT, DELETE, POST i HEAD standardne HTTP operacije koje se izvršavaju na tim podacima.

[36] **Simple Object Access Protocol** ili SOAP definira standardni komunikacijski protokol baziran na razmjeni XML poruka. Koristi različite transportne protokole kao što su HTTP, JMS i SMTP. Standardni HTTP protokol olakšava SOAP modelu da prolazi kroz vatrozide i proxy poslužitelje bez dodatnih modifikacija. SOAP klijent, u skladu sa SOAP specifikacijom, kreira XML dokument koji sadrži odgovarajući zahtjev. Taj dokument dolazi do SOAP poslužitelja, poslužitelj obrađuje pristigle zahtjeve te po završetku korištenjem SOAP protokola vraća poruku odgovora SOAP klijentu.

autorizacije transakcija (npr. digitalnim potpisima), ili potpisivanje dokumenata kvalificiranim certifikatima (AdES uredba).

- **Komunikaciju s vanjskim tvrtkama, poslovnim partnerima ili regulatornim tijelima**. U ovu grupu servisa spadala bi primjerice razmjena podataka s FINA-om ili slanje obaveznih izvještaja u HNB[37].

4.1.2 Komponente naslijeđenih aplikacija

Naslijeđene aplikacije, uz ostale, sadrže dvije vrste komponenti koje se koriste prema vlastitom modelu integracije: postojeće komponente te novu komponentu prilagodnik koji se ugrađuje u svaku od naslijeđenih aplikacija.

4.1.2.1 Postojeće komponente

Postojeće komponente, važne za vlastiti model integracije:

- **Prezentacijski sloj, to jest postojeća korisnička sučelja.** Prezentacijski sloj služi za prezentaciju i interakciju s podacima dohvaćenim od strane servisnog sloja. Omogućuje korištenje kompletnih poslovnih funkcionalnosti i sveobuhvatan prikaz svih poslovno relevantnih podataka koji su dostupni naslijeđenim aplikacijama. Prezentacijski sloj uključuje i postojeću poslovnu logiku za kontrolu prikaza i manipulaciju sučeljima.
- **Poslovna pravila i poslovni procesi.** Poslovnih pravila povezuju i grupiraju osnovne funkcionalnosti izložene servisima naslijeđenih sustava u aktivnosti PP-a.
- **Metode autorizacije i sigurnosti.** Postojeći mehanizmi za autentifikaciju, autorizaciju i drugo, koji se oslanjaju na servise *Autorizacijskih i sigurnosnih provjera* iz naslijeđenog sloja SOA (poglavlje 4.1.1).
- **Naslijeđeni model domene.** Opisuje poslovno područje kojim se organizacija bavi, a sadrži skup objekata (klasa) koji predstavljaju poslovne entitete. Predstavlja temelj na kojem će se izgraditi pojednostavljeni model domene za BPM rješenje.
- **Integracija s fizičkim uređajima.** Uključuje pristup fizičkim uređajima koji se koriste u obavljanju poslovnih aktivnosti. To mogu biti čitači kartica, potpisne pločice, i slično.

[37] **Hrvatska narodna banka** (HNB) je središnja banka Republike Hrvatske kao što je određeno Ustavom Republike Hrvatske.

4.1.2.2 Prilagodnik ugrađen u naslijeđene aplikacije

Važan dio vlastitog modela integracije je prilagodnik koji se ugrađuje u postojeće aplikacije, čime se omogućuje njihova komunikacija s BPMA. Osnovne zadaće prilagodnika obavljaju tri komponente:

- **BPMKS klijent -** ima ulogu slanja poruka prema BPMA i obrađivanje odgovora od BPMA i to sve uz pomoć BPMKS-a, koji je opisan u sljedećem poglavlju (4.2). Komunikacija prema BPMA se ostvaruje u određenim točkama poslovnog procesa (detaljnije opisanim u poglavlju 4.4.1) a uključuje obavještavanje BPMA o promjenama stanja PP-a. Komunikacija s BPMA-om može uključivati stvaranje instance procesa, ažuriranja stanja procesa, obavijesti o različitim događajima, dohvaćanje podataka iz mehanizama odlučivanja (engl. decision engine) ili tablica odlučivanja (engl. decision table) itd.
- **Komponenta za izlaganje postojećih funkcionalnosti** – izlaže funkcionalnosti naslijeđene aplikacije za korištenje od strane BPMA. Prema potrebi sadrži dodatnu poslovnu logiku koja grupira, manipulira li prilagođava postojeće funkcionalnosti. Može sadržavati prilagođene metode koje su potrebne BPMA, a vezane su uz naslijeđene sustave. Primjerice, metode za generiranje poveznica na naslijeđenu aplikaciju. Poveznice korisnika usmjeravaju iz BPMA na određeni ekran naslijeđene aplikacije u kojem se obavlja aktivnost procesa.
- **Komponenta za autorizaciju i preusmjeravanje (KAP)** – autorizira korisnika koji dolazi s BPMA u naslijeđenu aplikaciju te može vršiti preusmjeravanja po naslijeđenoj aplikaciji temeljem informacija koje dobije od BPMA. Preko poveznica opisanih u prethodnom odlomku, korisnik s BPMA biva preusmjeren na aktivnost u naslijeđenoj aplikaciji. Temeljem podataka iz pristupnih poveznica, ***KAP*** prijavi korisnika u aplikaciju koristeći postojeći mehanizam naslijeđene aplikacije, pripremi potrebne podatke te preusmjeri korisnika na obavljanje aktivnosti. Po završetku aktivnosti prilagodnik koristeći komponentu ***BPMKS klijent*** ažurira stanje procesa u BPMA, te pomoću ***KAP*** vraća kontrolu BPM okruženju odjavom klijenta iz naslijeđene aplikacije.

U slučaju MVC aplikacija, prilagodnika može biti implementiran posebnim skupom kontrolera. Ako imamo jednostraničnu aplikaciju (SPA[38]) prilagodnik može biti implementiran kao posebna komponenta s pratećim modulom za preusmjeravanje.

Općenito, prilagodnik se sastoji od dva osnovna dijela, aplikacijski specifičnog i općeg dijela.

Aplikacijski specifičan dio sadrži ***komponentu za izlaganje postojećih funkcionalnosti*** i ***komponentu za autorizaciju i preusmjeravanje***, informacije o naslijeđenoj aplikaciji, te sredstva temeljem kojih korisnik dolazi do funkcionalnosti procesa implementiranih u naslijeđenoj aplikaciji. Specifični dio prilagodnika je različit za svaku od naslijeđenih aplikacija zbog posebnosti njihove implementacije.

Opći dio prilagodnika sadrži komponentu ***BPMKS klijent*** koja zna kako koristiti BPM komunikacijski sloj (opisan u poglavlju 4.2) za interakciju s BPMA. Opći dio prilagodnika jednak je za različite naslijeđene aplikacije jer koristi metode koje BPMKS.

Prilagodnik također ima zadatak odvojiti naslijeđene aplikacije od BPMA, a po potrebi može se upotrijebiti za potpuno isključivanje BPMA iz naslijeđene aplikacije. Ovime se djelomično ostvaruje karakteristika ***prilagodljivosti*** vlastitog modela integracije (poglavlje 3.3.1). Prilagodnik može dopustiti interakciju samo za određene korisničke uloge sukladno njihovim ovlaštenjima. Korisnici naslijeđenih aplikacija, koji nemaju ovlaštenja za pristup BPM rješenju, mogu bez smetnji nastaviti izvršavati svoje zadatke u naslijeđenim aplikacijama.

4.2 BPM komunikacijski sloj

Kao što mu ime govori, sadrži metode za komunikaciju između rješenja za BPM i naslijeđenih aplikacija. BPMKS ima ulogu "naprednog" posrednika (engl. proxy), koji uz preusmjeravanje poziva već sadrži i poslovnu logiku (znanje o naslijeđenim aplikacijama i o BPM rješenju,) uz pomoć koje radi sigurnosne provjere, preslikavanje podataka, konstruiranje

[38] **Single Page Aplikacija** ili **SPA** je web aplikacija ili web stranica s ciljem pružanja korisničkog iskustva sličnog desktop aplikacijama. Proces renderiranja stranice SPA se uglavnom događa klijentskoj strani, unutar preglednika i ne zahtijeva ponovno učitavanje stranice tijekom upotrebe. Sav potreban programski kod SPA (HTML, JavaScript i CSS) dohvaća se s jednim učitavanjem stranice. Nakon toga se dodatni resursi dinamički učitavaju i dodaju na stranicu po potrebi, obično kao odgovor na korisničke radnje.

poziva prema BPMA, itd. BPMKS sadrži informacije u kojim aplikacijama se nalaze traženi podaci, koje servise treba pozvati, te kako dohvaćene podatke prilagoditi za BPMA.

U BPMKS-u definirano je nekoliko osnovnih komponenti:

- **Servisi za upravljanje procesnim akcijama** – Predstavljaju komponentu BPMKS-a putem koje se odvija komunikacija naslijeđenih sustava prema BPMA. Komunikacija se odvija neovisno o toku PP-a, a sastoji se od posebno definiranih servisa specijaliziranih za obavljanje općenitih akcija. Akcija može biti pokretanje instance procesa, promjena statusa i podataka u procesu, završavanja određene aktivnosti procesa, itd.. Sadrži:
  - **Klijenta za procesno vezanu komunikaciju** - komponenta za manipulaciju procesima, aktivnostima i procesnim podacima na BPMA. Poziva procesne API-je koji su izloženi u BPM okruženju.
  - **Poslovnu logiku vezanu uz procesne akcije** – služi za grupiranje i ulančavanje poziva procesnih API-a na BPM okruženju, te tumačenje podataka iz naslijeđenih sustava.

  Servisi za upravljanje procesnim akcijama pozivaju se od strane komponente ***BPMKS klijent*** koja je dio prilagodnika ugrađenog u naslijeđene aplikacije (poglavlje 4.1.2.2).
- **Servisi za rad s postojećim funkcionalnostima –** Komponenta putem koje BPMA ostvaruje komunikaciju s naslijeđenim sustavima te po potrebi radi provjere ili dohvat podataka. Sadrži:
  - **Klijenta za komunikaciju s naslijeđenim aplikacijama** – sastoji se od metoda za dohvat podataka iz naslijeđenih sustava, to jest poziva servisa koji su izloženi prilagodnikom naslijeđenih aplikacija (***komponentom za izlaganje postojećih funkcionalnosti***, poglavlje 4.1.2.2)
  - **Poslovnu logiku za tumačenje podataka naslijeđenih aplikacija** – služi za tumačenje, grupiranje i preslikavanje podataka naslijeđenih aplikacija prema potrebama BPMA.
- **Pojednostavljeni model domene** – sadrži pojednostavljene poslovne objekte domene naslijeđenih aplikacija koji opisuju poslovno područje se kojim se organizacija bavi. Koristi se od strane ***servisa za rad s postojećim funkcionalnostima*** i ***servisa za upravljanje procesnim akcijama*.** Taj model, bitno pojednostavljen, nastaje preslikavanjem modela domene klijentskih aplikacija tako da sadrži samo osnovni skup informacija koje su potrebne rješenju za BPM. Dodatno, ovdje se nalaze i entiteti koji predstavljaju artefakte iz BPM okruženja (poglavlje 5.3.5), a koriste se prilikom komunikacije s BPM rješenjem, primjerice entiteti procesa, aktivnosti, korisnika, itd. Tehnička razrada mogućnosti modela

domene detaljnije je opisana u poglavlju 5.3.5.1 dok je razina pojednostavljenja s obzirom na model u naslijeđenim aplikacijama prikazana temeljem konkretnog primjera u poglavlju 6.3.4.

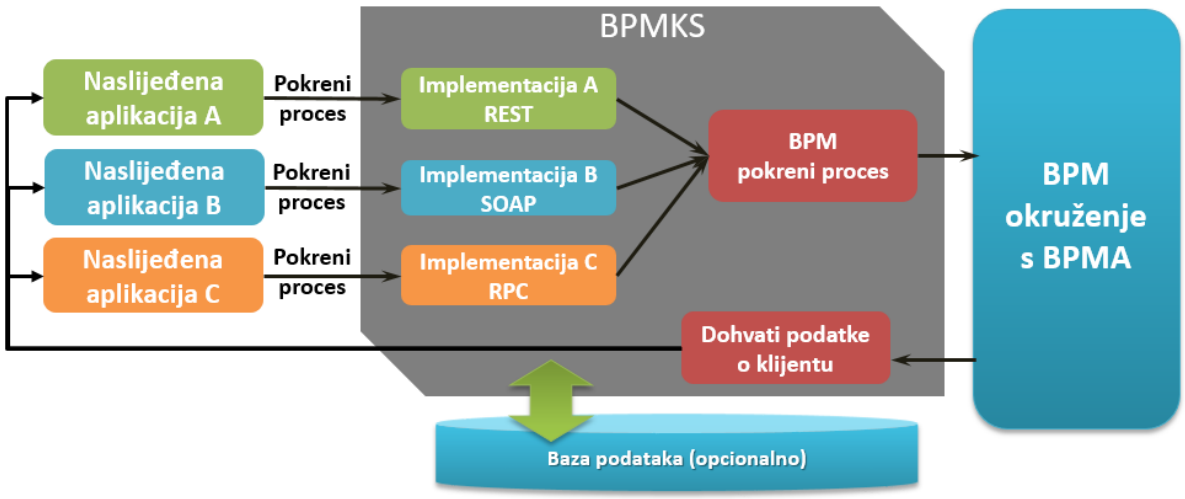
- **Ostale pomoćne komponente za**:
  - **Sigurnost** – osigurava odabrani način komunikacije (REST, SOAP, …), pruža kontrolu pristupa temeljem prilagođenih HTTP zaglavlja i podataka iz lokalnog konteksta poziva
  - **Obradu pogrešaka** – omogućuje obradu pogrešaka na razini BPMKS-a, definira prilagođene pogreške za pojedine situacije, tumači i preslikava pogreške s BPMA / naslijeđenih sustava, definira jedinstvenu struktura odgovora u slučaju pogreške
  - Čuvanje podataka **lokanog konteksta** – lokalni kontekst predstavlja skup obaveznih generičkih informacija vezanih za poziv servisa na BPMKS-u. Primjerice svaki poziv iz naslijeđenih aplikacija prema BPMKS-u mora sadržavati identifikator naslijeđene aplikacije koja je uputila poziv, identifikator korisnika koji radi u naslijeđenoj aplikaciji, vrstu procesa/usluge na kojoj korisnik radi, itd. Uloga i rukovanje podacima lokalnog konteksta detaljnije je razrađeno kroz tehničku specifikaciju u poglavlju 5.3.6.

4.2.1 Uloga komponente BPM komunikacijskog sloja

Zašto je BPMKS potreban? Zašto svaka od naslijeđenih aplikacija ne bi direktno komunicirala s BPMA?

Kako bi što bolje odgovorili na postavljena pitanja, pogledajmo širu sliku. BPMKS je zamišljen kao "poveznica između dva raznorodna sustava", BPM okruženja i naslijeđenih aplikacija. Omogućava komunikaciju između ta dva sustava s ciljem da svaki od sudionika komunikacije (BPMA i naslijeđena aplikacija) sadrže što manje informacija jedan o drugome. Tako se realizira "čista" implementacija komunikacije u oba sudionika.

Naslijeđene aplikacije ne moraju sadržavati podatke specifične za BPMA i BPM okruženje, a BPMA ne treba prilagođavati svakoj od naslijeđenih aplikacija. Naslijeđene aplikacije mogu koristiti vlastiti model domene te metode koje već imaju na raspolaganju za ostvarenje komunikacije prema BPMKS-u, a BPMKS dalje prilagođava poziv za BPMA (Slika 4.3). Tako više različitih naslijeđenih aplikacija, pisanih u različitim tehnologijama i različitih arhitektura, može komunicirati s BPMA. BPM okruženje i pojedina BPMA zapravo nisu svjesni specifičnih implementacijskih karakteristika postojećih funkcionalnosti koje koriste, te su stoga neovisni o naslijeđenoj tehnologiji i arhitekturi.



Slika 4.3 Pojednostavljeni prikaz komunikacije različitih naslijeđenih aplikacija s BPMA

Bez BPMKS-a, u BPM okruženju bi trebalo izložiti više različitih metoda koje obavljaju iste funkcionalnosti da bi se omogućilo izvršavanje akcija iz različitih sustava. Svaki naslijeđeni sustav trebao bi sadržavati podatke o BPM okruženju i BPMA, te se prilagođavati specifičnostima vezanim uz BPM okruženje. Kod komunikacije s BPMA naslijeđene aplikacije trebale bi pripremati i formatirati skup podataka prema očekivanjima BPMA. Kao posljedica, svaka od naslijeđenih aplikacija sadržavala bi poslovnu logiku za podršku opisanom načinu komunikacije, a koja nije direktno vezana za funkcionalnosti naslijeđene aplikacije. Ista poslovna logika bila bi implementirana u svakoj naslijeđenoj aplikaciji što bi dovelo do umnažanja programskog koda.

Dodatno, BPMKS može imati vlastitu bazu podataka koja se ovisno o potrebi može koristiti za čuvanje podataka koji opisuju PP-e u sustavu temeljenom na vlastitom modelu integracije, bilježenje podataka o odazivu sustava i događajima koji se u sustavu odvijaju. U vlastitom modelu integracije razrađenom u okviru ovog rada ne postoji potreba za bazom podataka, pa se ona neće koristiti (razlozi odluke objašnjeni u poglavlju 5.7).

BPMKS gradi labavu povezanost BPM okruženja i naslijeđenih sustava, omogućuje zamjene korištenih alata za BPM ili olakšava prodaju naslijeđenih aplikacija vanjskim kompanijama koje ne koriste BPM alate. Kao takav, BPMKS osigurava karakteristiku ***prilagodljivosti*** vlastitog modela integracije (3.3.1). Također, omogućava podjelu sustava u ograničene dijelove gdje pojedinosti implementacije dijelova nisu izložene prema vanjskom svijetu, ali su dostupne preko jasno definiranih sučelja.

4.3 BPM okruženje i BPMA

BPMA je izgrađena u BPM okruženju uz pomoć alata iz BPMS-a, a sadrži model postojećeg PP-a koji se u naslijeđenoj okolini odvija putem više postojećih naslijeđenih aplikacija. U BPMA se kreiraju tokovi PP-a i potprocesa, izrađuju se metode za komunikaciju s BPMKS-om, konstruiraju korisnička sučelja te se dodaje potrebna poslovna logika.

Upravljanje poslovnim procesom (pokretanje procesa i aktivnosti, ažuriranje podataka, dodjeljivanje zadataka, itd.) odvija se putem sučelja za procesno vezanu komunikaciju (komponenta **API za procesno vezanu komunikaciju**) koja su izložena u BPM okruženju. Ta sučelja dio su BPMS-a koji organizacija koristi te ovise o implementaciji proizvođača. Općenito, omogućuju pristup artefaktima, korisničkim zadacima, poslovnim procesima, i podacima PP-a u BPM okruženju.

Dodatno, BPMA sadrži i **komponentu za integraciju s naslijeđenim servisima (KINS)** koja preko BPMKS-a poziva servise nad naslijeđenim aplikacijama, te tako dohvaća poslovno značajne podatke potrebne u određenom koracima PP-a. Na taj način ostvaruje se izravna komunikacija sa SOA naslijeđenim slojem organizacije, a uz pomoć poslovne logike na BPMKS-u, po potrebi se može raditi manipulacija nad traženim poslovnim podacima.

Na razini BPM okruženja definirana su korisnička ovlaštenja i grupe, koja se koriste unutar BPMA prilikom definicije dijelova DPP-a

BPMA putem svojih sučelja pruža centralnu točku korisničke interakcije te zatim preusmjerava korisnike na naslijeđene aplikacije koje obavljaju aktivnosti PP-a. Podaci procesa prikazani na BPMA sučeljima samo su za čitanje, dok se ažuriranje i promjena podataka procesa vrši isključivo kroz sučelja naslijeđenih aplikacija.

Cilj vlastitog modela je odrađivati manipulaciju s podacima PP-a kroz postojeća sučelja naslijeđenih aplikacija što je i izraženo kroz treću karakteristiku vlastitog modela (poglavlje 3.3.1). Iz navedenog razloga složeni tipovi podataka korišteni u naslijeđenim aplikacijama, dodatno se pojednostavljuju u BPMKS-u, te se preslikavaju u jednostavne tipove koji su neophodni za funkcioniranje BPMA. Posljedično, količina podataka koja se prikazuje na sučeljima BPMA uvelike je pojednostavljena s obzirom na obujam prikazanih podataka u naslijeđenim aplikacijama (primjer u poglavlju 6.3.4. na konkretnom slučaju). Sadrži osnovni skup informacija o procesu, a definira se u suradnji s poslovnim korisnicima koji odabiru poslovno najbitnije podatke o procesu.

4.3.1 Odabir BPMS-a za korištenje u vlastitom modelu

Nastavno na poglavlje 2.2.3.2, ovdje su razrađene osnovne karakteristike koje BPMS mora imati kako bi bio kandidat za korištenje u predloženom modelu integracije. Karakteristike možemo podijeliti na obavezne i poželjne. Obavezne karakteristike su preduvjet da bi određeni alat bilo moguće koristiti, dok poželjne olakšavaju proces implementacije i integracije.

Obavezne karakteristike:

- Integracija – mogućnost manipulacije s artefaktima procesa na BPM rješenju putem API-ja izloženih na BPM okruženju. API mora omogućiti:
  - pokretanje i završavanje instanci procesa i aktivnosti
  - mogućnost ažuriranja podataka unutar instance i aktivnosti procesa
  - manipulaciju toka procesa
  - dostupnost informacija o korisnicima i korisničkim ulogama
  - preraspodjelu aktivnosti.
- Korisnička interakcija - mogućnost izrade jednostavnih korisničkih sučelja u sklopu BPMS-a koja prikazuju podatke PP-a.
- Iterativni razvoj - podrška iterativnom razvojnom procesu kako bi se omogućio predloženi iterativni razvoj i integracija BPMA (poglavlje 5.5.2.3).

Poželjne karakteristike:

- Mogućnost parcijalnih simulacija procesa tokom razvoja BPMA
- Lakoća korištenja i implementacije
- Niska cijena.

4.4 Integracija s procesima u naslijeđenim sustavima temeljem vlastitog modela

Prema primjeru navedenom u 3.3.3 vidljivo je da predloženi model integracije čuva tok procesa kroz različite naslijeđene aplikacije, a istovremeno omogućava centralizirani pogled i kontrolu nad poslovnim procesom.

Detalji modelom postignutih strukturnih inovacija i racionalizacija PP-a tema su 6. poglavlja. Ovo poglavlje opisuje kako, kada i pomoću kojih komponenti vlastitog modela vršimo interakciju s postojećim poslovnim procesom u naslijeđenim sustavima.

4.4.1 Procesne točke i točke interakcije

Kako bi se točno odredila integracija BPMA s procesom u naslijeđenim aplikacijama, potrebno je odrediti točke u kojima će se odvijati pozadinska komunikacija te korisnička interakcija s procesom. U tu svrhu definirat će se pojmovi procesnih te interakcijskih točaka.

Procesne točke vezane su uz komunikaciju naslijeđenih aplikacija i BPMA, služe za pripremu i usklađivanje podataka, a mogu biti standarde i pozadinske. Procesne točke osiguravaju karakteristiku ***sinkronosti*** vlastitog modela integracije (3.3.1).

Standardne procesne točke su događaji u životnom toku procesa gdje dolazi do promjena u stanju procesa ili podataka koji su vezani uz stanje procesa. Standardne procesne točke zahtijevaju jednosmjernu komunikaciju prema BPMA, prilikom čega se nastala promjena u procesu ažurira u BPMA. Ažuriranja mogu uključivati manje promjene u podacima instance procesa, ali se mogu sastojati i od većih promjena u stanju procesa; npr. označavati kraj korisničke aktivnosti ili točku gdje se tok procesa kreće iz jedne naslijeđene aplikacije u drugu. Primjerice, klijent u internetskoj bankarskoj aplikaciji zadaje zahtjev za uslugu mobilnog internetskog bankarstva, a na BPM okuženju postoji pokrenut proces koji odražava stanje zahtjeva. Prije slanja zahtjeva u banku, klijent odrađuje akciju digitalnog potpisivanja zahtjeva, što predstavlja tipičnu standardnu procesnu točku. U trenutku potpisivanja, naslijeđena aplikacija šalje informaciju o akciji potpisivanja s informacijama o potpisniku, vremenu potpisa te promjeni stanja zahtjeva prema BPMA gdje se ti podaci ažuriraju.

Pozadinske procesne točke važne su za BPMA i označavaju poziv BPMA prema prilagodniku naslijeđene aplikacije uz pomoć ***KINS***. BPMA tako dohvaća podatke potrebne za funkcioniranje procesa iz naslijeđenih sustava, primjerice poveznice do mjesta u naslijeđenim aplikacijama gdje će korisnik moći odraditi dodijeljenu aktivnost. Dodatni primjer pozadinske procesne točke može biti dohvat povijesti akcija koje su rađene na zahtjevu u naslijeđenim aplikacijama.

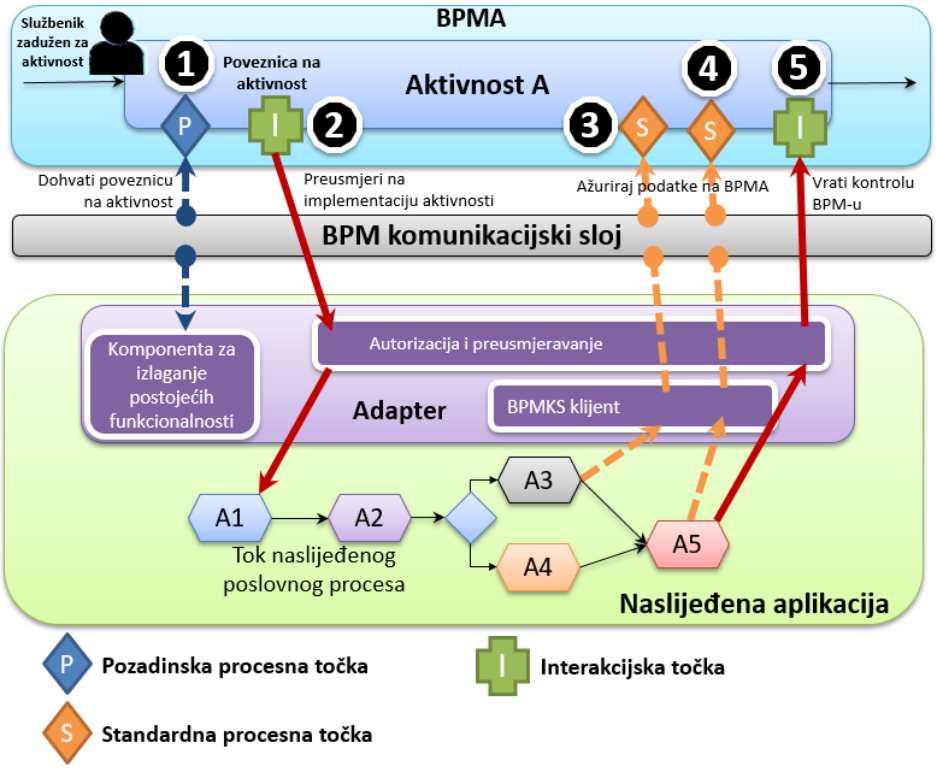
Kombinacijom standardnih i pozadinskih procesnih točaka ostvaruje se interakcija naslijeđenih sustava i BPMA kroz tok PP-a. Procesne točke pokazale su se važnim konceptima tijekom otkrivanja procesa u naslijeđenim aplikacijama i njihove integracije s BPMA.

Točke interakcije opisuju interakciju korisnika s naslijeđenim aplikacijama, to jest dijelove procesa gdje korisnik započinje/završava interakciju s procesom u naslijeđenoj aplikaciji. Točke interakcije u pravilu slijede iza pozadinskih procesnih točaka, a nakon interakcijskih točaka u pravilu nalazimo standardne procesne točke. Dijelimo ih na ulazne i izlazne.

U ulaznim interakcijskim točkama BPMA pomoću veze na naslijeđenu aplikaciju (generiranu od strane naslijeđene aplikacije u pozadinskoj procesnoj točci) upućuje službenika na ekran naslijeđene aplikacije gdje može obaviti traženu akciju. Nakon što službenik odradi akciju procesa u naslijeđenoj aplikaciji, u izlaznim točkama interakcije odjavljuje se iz naslijeđene aplikacije, a kontrola se vraća BPM-u.

4.4.1.1 Primjer procesnih i interakcijskih točaka

Pretpostavimo da službenik radi u sustavu izgrađenom prema opisanom vlastitom modelu integracije, te da mu je na BPM okruženju dodijeljena aktivnost **A** koju mora izvršiti. Zatim, dolazi do niza događaja koje prikazuje Slika 4.4.



Slika 4.4 Uloga prilagodnika u toku procesa kroz sustav temeljen na vlastitom modelu integracije

Detaljna objašnjena prikazanih događaja dana su u nastavku:

Točka 1: Službenik preuzima zadatak – budući da je zadatak preuzet, BPMA treba informaciju gdje i kako se aktivnost **A** može izvršiti, to jest treba poveznicu na naslijeđenu aplikaciju koja aktivnost obavlja. BPMA stoga (putem ***KINS*** komponente), preko BPMKS-a, upućuje poziv prema prilagodniku naslijeđene aplikacije. U prilagodniku, ***komponenta za izlaganje postojećih funkcionalnosti*** zaprima poziv, kreira potrebnu poveznicu koju vraća na

BPMA. BPMA zatim korisniku prikazuje ekran s osnovnim podacima o aktivnosti te dohvaćenom poveznicom. Ova akcija dohvata poveznice čini pozadinsku procesnu točku.

Točka 2: Službenik "klikne" na poveznicu na sučeljima BPMA – Poveznica vodi službenika na odgovarajuću naslijeđenu aplikaciju, na ***komponentu za autorizaciju i preusmjeravanje*** ugrađenog prilagodnika. Komponenta obavlja prijavu korisnika u aplikaciju, te ga preusmjerava na ekran aplikacije gdje može početi raditi na aktivnosti. Ova točka predstavlja ulaznu interakcijsku točku.

Točka 3 i točka 4: Službenik kreće u obavljanje aktivnosti A – u naslijeđenoj aplikaciji, službenik kreće obavljati skup podaktivnosti (A1, A2 A3/A4 i A5). Kako službenik obavlja podaktivnosti, mijenja podatke u naslijeđenim sustavima. Pretpostavimo da u podaktivnostima A3 i A5 dolazi do ključnih promjena u podacima, koje treba ažurirati na BPMA. Stoga, u navedenim aktivnostima, naslijeđena aplikacija, putem komponente prilagodnika ***BPMKS klijent***, preko BPMKS-a, ažurira navedene podatke na BPMA. Ove točke predstavljaju standardne procesne točke. Vidljivo je da je standardna procesna točka vezana uz aktivnost A3 opcionalna (postoji ako tok PP-a prolazi granom kroz podaktivnosti A3) dok točka vezana uz A5 postoji uvijek. Standardna procesna točka vezana uz podaktivnost A5 ujedno je zadnji zadatak unutar aktivnosti A, te ima ulogu završavanja aktivnosti A u BPMA.

Točka 5: Službenik završava obavljanje aktivnosti – kada službenik odradi sve podaktivnosti u naslijeđenoj aplikaciji, ***komponenta za autorizaciju i preusmjeravanja*** ugrađenog prilagodnika, odjavljuje službenika iz naslijeđene aplikacije te ga vraća u BPMA. Ova točka predstavlja izlaznu interakcijsku točku.

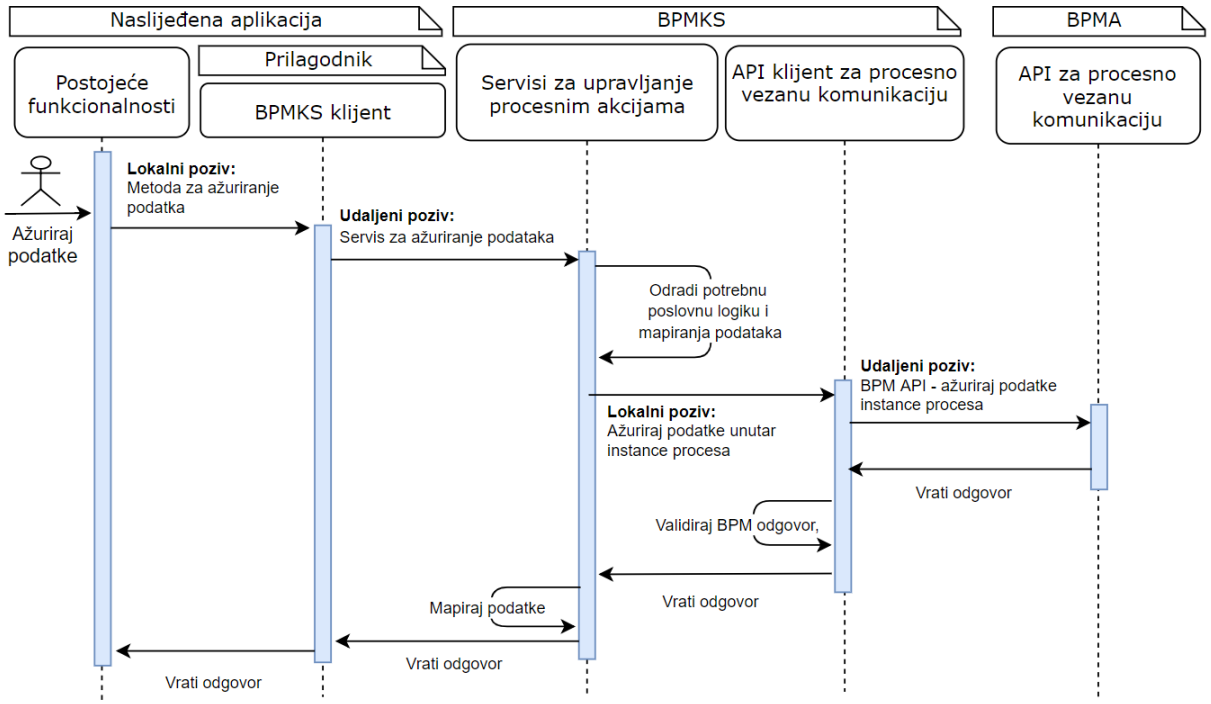
Važno je napomenuti da se komunikacija u procesnim točkama odvija preko BPMKS-a, dok se preusmjeravanje korisnika u interakcijskim točkama odvija putem poveznica, koje vode direktno na naslijeđene aplikacije.

4.4.1.2 Dijagram komunikacije u procesnim točkama

Pogledajmo kako teče komunikacija između komponenti u procesnim točkama u sustavu temeljenom na vlastitom modelu integracije.

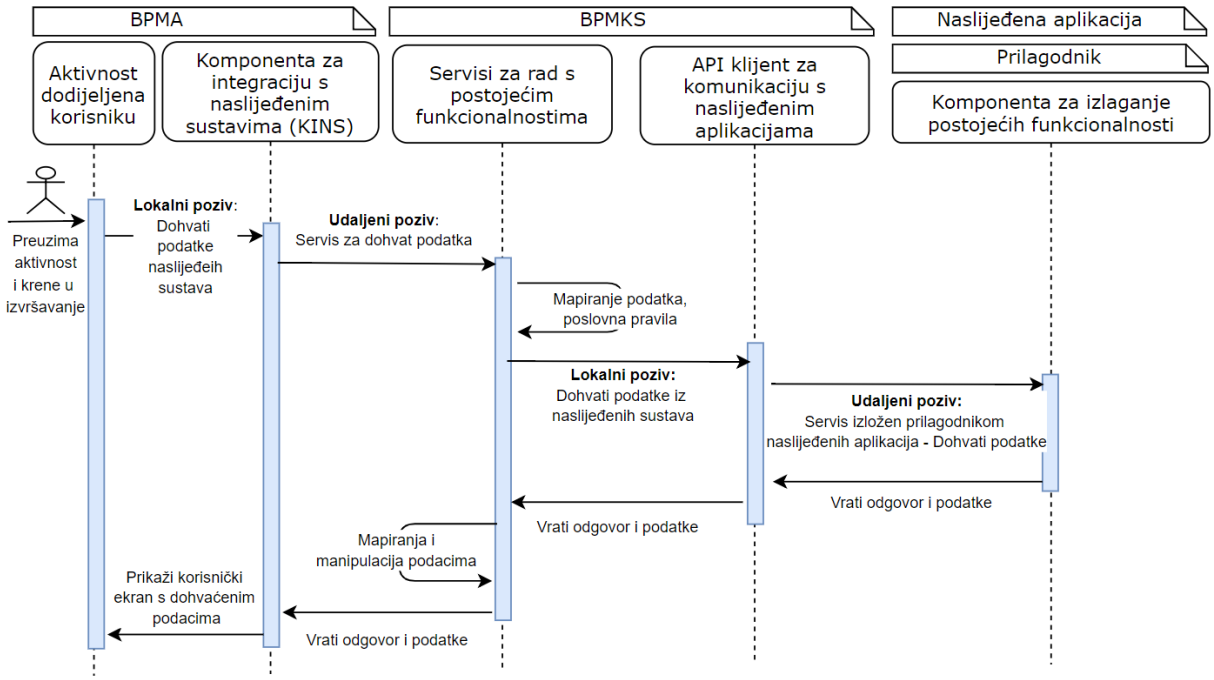
Slika 4.5 prikazuje komunikaciju u slučaju standardne procesne točke koja se odvija na sljedeći način. Nakon što se u naslijeđenoj aplikaciji promijeni podatak važan za PP, naslijeđena

aplikacija uz pomoć ugrađenog adaptera poziva servis na BPMKS-u za ažuriranje navedenog podatka u BPMA. BPMKS zaprima poziv, odradi sigurnosnu provjeru (utvrđujući tako da je poziv došao iz pouzdanog izvora) mapira podatke te primijeni potrebnu logiku nad podacima te tako pripremi podatke za BPMA. Zatim BPMKS napravi novi poziv prema BPM okruženju, poziva API sučelje za ažuriranje podataka procesa izloženom na BPMS-u, koje ažurira željeni podatak. Potom, preko BPMKS-a, BPMA naslijeđenoj aplikaciji vraća se odgovor. Cijeli proces još detaljnije je opisan na konkretnom primjeru u poglavlju 6.3.5.



Slika 4.5 Dijagram komunikacije u standardnoj procesnoj točci

Slika 4.6. prikazuje komunikaciju u pozadinskoj procesnoj točci, a odvija se u suprotnom smjeru nego kod standardnih procesnih točaka, dakle od BPMA prema naslijeđenim sustavima. Komunikacija započinje kada se javi potreba da BPMA dohvati određenu informaciju iz naslijeđenih sustava. U tom trenutku BPMA poziva servise za rad s postojećim funkcionalnostima na BPMKS-u. Navedeni servisi , zatim pozivaju funkcionalnosti izložene prilagodnikom naslijeđenih aplikacija, po potrebi grupiraju i mapiraju podatke, te odgovor vraćaju na BPMA. BPMA tada prikazuje dohvaćene podatke korisniku. Cijeli proces također je detaljnije opisan na konkretnom primjeru u poglavlju 6.3.5.



Slika 4.6 Dijagram komunikacije u pozadinskoj procesnoj točci

4.4.2 Mehanizmi i kontrola pristupa

Pri razmatranju interakcije korisnika s BPMA i naslijeđenim sustavima, te izvršavanju dodijeljenih aktivnosti pojavljuje se pitanje mehanizama kontrole pristupa. Kako bi se osiguralo da korisnici izvrše aktivnosti PP-a za koje su ovlašteni potrebno je primijeniti odgovarajuće mehanizme autorizacije [72]. Budući da u predloženom modelu naslijeđene aplikacije izvršavaju radnje PP-a korisnički pristup uvelike je definiran postojećim mehanizmima kontrole pristupa tih naslijeđenih aplikacija. Naslijeđene aplikacije razmatrane u sklopu ovog rada, te PP-i koji se u njima izvršavaju, temeljene su na standardnom modelu Role-based Access Control (RBAC) uz kombinaciju dinamičkih i statičkih koncepata razdvajanja dužnosti (Separation of Duty - SoD).

U RBAC modelu upravljanja pristupom pristup resursima i zadacima sustava temelji se na ulozi korisnika u sustavu. Osnovni RBAC model obuhvaća sljedeće entitete: korisnike, uloge, dozvole i korisničke sesije, gdje se dozvole sastoje od operacija primijenjenih na objekte [73]. U RBAC modelu dozvole su povezane s ulogama, a korisnici su članovi uloga. Korisnička interakcija sa sustavom modelirana je korisničkom sesijom, unutar koje korisnik aktivira podskup uloga kojima je dodijeljen [73]. Važna karakteristika RBAC modela je koncept SoD obavljanja zadatka na više pojedinaca unutar organizacije [72]. Cilj ovog koncepta je povećati sigurnost te riješiti slučaj nedostupnosti nekog od izvršitelja zadataka. Statički SoD definira se tokom faze administracije sustava, dok se dinamički provodi uživo, kada korisnici aktivno koriste sustav [72]. Korisničke uloge definirane u naslijeđenim aplikacijama sadrže ovlasti za

izvršavanje zadataka te pristup resursima. Definirane su na razini aplikacije, to jest na razini PP-a i ne ovise o zadatku ili instanci procesa koja se izvršava, uključenim resursima (dokumenti, podaci unutar procesa) ili elementima izvan standardnih entiteta kontrole pristupa.

Mehanizmi kontrole pristupa u predloženom modelu integracije definirani su dvoslojnim RBAC modelom.

Prvi sloj uključuje izgrađene mehanizme kontrole u sklopu definicije BPMA. Implementiran je u BPMA gdje se tijekom definicije BPMN modela DPP-a određuju timovi zaduženi za aktivnosti koje pripadaju određenoj stazi (poglavlje 5.5.2.3). Drugi sloj iskorištava postojeće RBAC mehanizme naslijeđenih aplikacija, to jest ovlaštenja korisnika definirana u naslijeđenim aplikacijama.

U slučaju potrebe budućih unapređenja mehanizama kontrole pristupa, ugrubo su razmatrani napredniji modeli kontrole pristupa koji su proširenja osnovnog RBAC modela s dodatnim elementima. Proširenja mogu biti specifične naravi, da zadovolje samo određenu potrebu poslovnog scenarija, primjerice ovlaštenja na razini korisničkih aktivnosti za što se onda mogu koristiti razne varijacije modela TRBAC (Task-Role-Based Access Control). U TRBAC modelu ovlaštenja su dodijeljena aktivnostima, a te aktivnosti pridružene korisničkim ulogama. Osim zadovoljavanja specifičnih situacija, proširenja RBAC modela mogu uključivati niz ograničenja koja agregacijom šireg raspona informacija donose odluku o kontroli pristupa. Primjerice, korisnik može imati različite uloge za različite instance procesa, može imati različite uloge u različitim vremenskim intervalima ili različite dozvole tijekom izvršavanja procesa. [73]. U navedenom slučaju ulazimo u domenu PP-a koji zahtijevaju kontekstno svjesne mehanizme kontrole pristupa, mehanizme koji temeljem stanja PP-a i niza trenutno dostupnih informacija određuju prava pristupa. Dobar primjer ovakvog načina kontrole pristupa je COBAC (Context-sensitive model for Business processes Access Control) model koji je detaljno opisan u [73] te dodatno unaprijeđen u [72]. COBAC model omogućuje definiranje politike kontrole pristupa na razini instance i konteksta procesa, dodjeljivanje aktivnosti ulogama i definiranje dopuštenja za pristup resursima na razini aktivnosti [72], [73].

Za implementaciju razmatranih naprednih mehanizama kontrole pristupa u predloženom modelu integracije, potrebno je osigurati da mogućnosti BPM alata podržavaju opisane mehanizme i/ili zasebno implementirati i integrirati opisane mehanizme. Korištenje modela kontrole pristupa COBAC u vlastitom modelu nadmašuje opseg ovog rada i neće se dalje razmatrati.

4.5 EAI, ESB, SOA i predloženi model integracije

Razmotrimo sličnosti i razlike predloženog modela integracije sa standardnim ESB arhitekturama, EAI principima i pogledajmo koju ulogu ima SOA.

ESB je skup načela, arhitektura, koji omogućuje komunikaciju više različitih aplikacija temeljem principa koje nalazimo u računalnim sabirnicama. U osnovi, ESB arhitektura je "komunikacijski agent", koji prima poruku, prevodi ju u odgovarajući format, te je šalje na mjesto gdje je poruka potrebna. Predloženi model integracije služi za kontrolu PP-a i preraspodijele posla koji određene aplikacije moraju, korištenjem postojećih funkcionalnosti, obavljati unutar PP-a. Osim što omogućava komunikaciju različitih sustava te iskorištava postojeća korisnička sučelja i poslovnu logiku, predloženi model integracije dodatno koordinira ljudsku interakciju u izvršavanju PP-a. ESB provodi orkestraciju servisa [65], dok predloženi model integracije orkestrira procese prepoznavanjem, izdvajanjem i integracijom aktivnosti iz različitih naslijeđenih aplikacija s ciljem postizanja centralizirane točke interakcije za uspješno izvršenje PP-a.

Integracija podataka između aplikacija s ciljem pojednostavljivanja PP-a koji se protežu kroz povezane aplikacije stavlja predloženi model integracije bliže EAI domeni. Preciznije rečeno, predloženi model integracije odrađuje EAI na prezentacijskoj razini (engl. presentation level EAI) koja uključuje elemente EAI na aplikacijskoj razini (engl. application level EAI [67], [68]) i elemente EAI na razini korisničkih sučelja (engl. user interface level EAI [67], [68]). Aplikacijska razina EAI integracije vidljiva je u pristupu poslovnim procesima i podacima naslijeđenih aplikacija kroz sučelja naslijeđenih aplikacija. Omogućuje korištenje postojeće poslovne logike, transparentna je za integrirane aplikacije i čuva integritet podataka integriranih aplikacija. Elementi EAI na razini korisničkih sučelja mogu se prepoznati u ponovnoj upotrebi korisničkih sučelja naslijeđenih sustava (zajedno s ugrađenom naslijeđenom poslovnom logikom) u kombinaciji s korisničkim sučeljima BPMA, kako bi se stvorila središnja točka za korisničku interakciju.

BPMKS možemo opisati kao specifično proširenje ESB integracije, posebno prilagođeno principima i potrebama predloženog modela integracije. BPMKS je međusloj koji ima ulogu pružiti sredstva za kontrolu PP-a i naslijeđenih aplikacija. Dok tipični ESB prenosi i prevodi podatke, BPMKS također ima ulogu pojednostavljenja poslovnih podataka uz pomoć vlastitog modela domene koji je detaljnije razrađen u poglavlju 5.3.5. Načela poput automatskog generiranja klijenta za komunikaciju s BPMKS servisima, omogućava kompatibilnost BPMKS-a s naslijeđenim aplikacijama izgrađenim u različitim tehnologijama, o čemu više u poglavlju

5.4.2. Kompatibilnost ESB implementacije s postojećim sustavima smanjuje napore integracije s postojećom arhitekturom [65]. BPMKS ne samo da omogućuje interakciju između različitih sustava nego uključuje poslovnu logiku za izoliranje dijelova postojećih aplikacija, s cjelokupnim funkcionalnostima i korisničkim sučeljima, za korištenje od strane BPMA. Dodatno, omogućuje upravljanje modelima PP-a na BPM okruženju, te pomaže u orkestriranju korisničke interakcije. Prilagođeno baratanje pogreškama, dodatna sigurnost te ostali principi BPMKS implementacije, koji su detaljnije opisani u poglavlju 5.1.1, dodatno proširuju standardnu ESB arhitekturu. U slučaju postojeće ESB implementacije u organizaciji, uloga BPMKS-a može se implementirati uz pomoć alata i funkcionalnosti dostupnih u ESB paketu. Međutim, ovisno o mogućnostima i vrsti implementiranog ESB rješenja, moguća je potreba prilagodbe opisanog modela integracije, prilagodnika ugrađenog u naslijeđene aplikacije, BPMKS-a te predloženih načela integracije koja su detaljnije razrađena u poglavlju 4.4.1. Da bi se dao odgovor na pitanje opsega i dubine izmjena predloženog modela integracije, potrebno je daljnje razmatranje ove teme, zajedno s konkretnom evaluacijom odabranog ESB rješenja koji se koriste u organizaciji.

Koja je zapravo uloga SOA-e u modelu? Moglo bi se reći da je ESB jedan od načina implementacije SOA-e gdje se jedan središnji servis koristi za ostvarenje međusobne komunikacije između ostalih servisa. Dodatno, SOA također može pružiti sredstva za postizanje EAI, olakšavajući protok informacija između aplikacija. Može se reći da SOA čini građevne blokove za ESB i EAI, a upravo je modularnost SOA-e, kao što je opisano u prva tri odlomka poglavlja 4.6, ono što omogućuje rad predloženog modela integracije.

4.6 Naslijeđene okoline koje nisu temeljene na SOA

Korištenje naslijeđenih aplikacija koje nisu temeljene na principima SOA u modelu integracije, primjerice monolitnih programskih rješenja, također je razmatrano u okviru tehničke razrade radnog okvira. Zbog mnogobrojnih međusobnih zavisnosti i isprepletenih funkcionalnosti, monolitni sustavi teški su za razumijevanje što uzrokuje probleme u prepoznavanju i izoliranju dijelova postojećih procesa te poteškoće u određivanju *procesnih točaka* i *točaka interakcije*.

Ponovna iskoristivost dijelova postojećih funkcionalnosti osnovni je koncept predloženog modela integracije, a za uspješnu ponovnu iskoristivost potrebna je neka vrsta modularnosti postojećih sustava. U monolitnim aplikacijama, identifikacija te ponovna uporaba zasebnih, prepoznatljivih komponenti s dijelovima poslovnih funkcionalnosti je upitna. Tijekom

razvojnog ciklusa monolitnih aplikacija, servisni moduli te arhitekture imaju tendenciju postati međusobno čvrsto povezani i zapleteni, što otežava njihovu izolaciju te ponovo korištenje u nove svrhe. Za pokušaj uspješne integracije s monolitnim aplikacijama bile bi potrebne dodatne modifikacije u strukturi komponente prilagodnika. To bi rezultiralo čvršćom povezanošću prilagodnika i monolitne aplikacije, te tako dodatno zakompliciralo naslijeđenu aplikaciju.

Zaključno, potrebne su dodatne studije i prilagodbe modela integracije za uporabu u okruženjima koja nisu bazirana na SOA principima, te praktična implementacija u navedenim okruženjima za dokazivanje izvedivosti i korisnosti prilagođenog modela.

5 Specifikacija radnog okvira predloženog modela

U prethodnom poglavlju izložen je predloženi model integracije s teorijske razine, opisane su pojedine komponente, njihova uloga te međusobne veze. Ustanovljene su prednosti modela i prikazano je funkcioniranje integriranog sustava.

Ovo poglavlje specificira radni okvir koji omogućava integraciju naslijeđenih sustava s BPM-om prema prethodno teorijski razrađenom vlastitom modelu integracije. Opisan je postupak primjene vlastitog modela, prikazuje se uloga postojećih sustava te se definira tehnička specifikacija novih komponenti. Razvijene su metode za realizaciju prikazanih koncepata te su opisani principi odabra programskih alata. Prijedlozi realizacije pojedinih funkcionalnosti poduprti su konkretnim primjerima i objašnjenjima

Zbog kompleksnosti, tehnička specifikacija radnog okvira za implementaciju vlastitog modela te njegovu integraciju s postojećim sustavom prikazana je na dvije razine:

- **Arhitekturnoj razini** (poglavlja 5.1 do 5.6) koja predstavlja preduvjet za implementaciju. Obuhvaća tehničku specifikaciju programskih komponenti radnog okvira te odabir tehnologija i programskih alata. Prikazuje postupke izgradnje novih komponenti te način na koji se iskorištavaju postojeće. Definira komunikaciju, te opisuje principe povezivanja BPMA s postojećim komponentama, resursima i aplikacijama
- **Procesno-podatkovnoj razini** (poglavlja 5.7 i 5.8) koja obuhvaća otkrivanje postojećih PP-a, modeliranje tokova tih procesa te njihove integracije s BPMA. Dodatno, na ovoj razini razmatra se i rukovanje s podacima PP-a kroz kombinaciju jednostavnih sučelja na BPMA te kompleksnih korisničkih ekrana naslijeđenih sustava.

Jedan od bitnijih zahtjeva je brzina i lakoća implementacije vlastitog modela, pa su tehnologije, alati i metode odabrane da bi ispunile navedene zahtjeve.

Radi ograničenog obujma ovog rada, gotova programska rješenja, alati, općepoznati standardi te korištene tehnologije objašnjeni su samo na konceptualnoj razini, dok su model integracije, postupci, uloga i realizacija svake od komponenata modela razrađeni u detalje.

5.1 Način komunikacije, alati i tehnologije

Temeljem definiranih karakteristika (poglavlje 3.3.1) te konceptualne strukture predloženog modela integracije (poglavlje 4) razmotrit će se opcije, te dati preporuke principa komunikacije, alata te tehnologija za korištenje u radnom okviru.

5.1.1 Realizacija komunikacije između komponenti

Ugrubo, komponente modela čine izdvojene aplikacije koje se mogu nalaziti na različitim poslužiteljima unutar organizacije. U nastavku su razmotrene tri tehnologije komunikacije, te je odabrana najprikladnija.

- **RMI/RPC**

Remote Procedure Call (RPC) je tehnika za izradu raspodijeljenih aplikacija temeljenih na klijentsko-poslužiteljskoj arhitekturi. Bazira se na proširenju pojma standardnog lokalnog poziva procedura, međutim pozivatelj procedure i pozvana procedura mogu se nalaziti na različitim sustavima unutar mreže.

Iako je RPC jednostavan za implementaciju, u duhu opisanog modela postoji nekoliko nedostataka zbog kojih nije adekvatan odabir.

- Specifična implementacija RPC-a ovisna je o jeziku implementacije. U naslijeđenoj okolini postoji više vrsta aplikacija pisanih u različitim tehnologijama, pa implementacija komunikacije ovom metodom nije jednostavna.
- Realizacija komunikacije prema BPM alatu/okolini nije izvediva jer u pravilu BPM alati ne podržavaju programabilna sučelja temeljena na RPC implementacijama.
- Mogući problemi kod komunikacije između aplikacija odvojenih vatrozidovima zbog dinamičke alokacije priključnica (portova) koju radi RPC.
- Dodatan nedostatak je relativna sporost, sigurnosni problemi i slaba tolerancija na pogreške. Izvršavanje udaljenih metoda je sinkrono, što može dovesti do "smrzavanja" aplikacija u slučaju problema na mreži, ili pogreške na jednom od sudionika komunikacije. RPC je neprikladan kada aplikacije trebaju prekinuti komunikaciju u slučaju zastoja ili drugih problema.

- **SOAP web servisi**

Pojam "web servis" opisuje standardizirani način integracije web-aplikacija korištenjem otvorenih standarda XML, SOAP, WSDL i UDDI (engl. Universal Description Discovery and

Integration). XML pruža format podataka za prijenos sadržaja te definiciju strukture poruke (tzv. SOAP omotnica), SOAP je protokol koji omogućava prijenos podataka, WSDL opisuje funkcionalnosti koje određeni web servis nudi, dok UDDI predstavlja centralni registar dostupnih web servisa. SOAP protokol omogućuje komunikaciju između aplikacija koje rade na različitim operacijskim sustavima i različitim tehnologijama.

SOAP nije najbolji izbor za realizaciju komunikacije zbog sljedećih razloga:

- Pojačana sigurnost koju pruža SOAP nije potrebna za slučaj koji se obrađuje u ovom radu. Cijela arhitektura predloženog modela integracije se nalazi unutar infrastrukture organizacije, gdje se koriste postojeći sigurnosni protokoli i procedure.
- Postoji kruti "odnos" između SOAP klijenta i poslužitelja. Kada se bilo koja strana promijeni, "odnos" se prekida i komunikacija prestaje raditi. Potrebna su stalna ažuriranja obje strane nakon bilo kakve izmjene, što komplicira proces razvoja i ne doprinosi fleksibilnosti komunikacije.
- XML, koji se koristi za izradu SOAP poruka, može postati vrlo kompleksan i težak za održavanje te tako podložan pogreškama i zahtjevan za parsiranje. Veći dio podataka SOAP XML poruke gradi strukturu SOAP omotnice poruke te predstavlja višak (engl. overhead) koji je potrebno slati sa svakom porukom. Cilj predloženog modela integracije je jednostavnost komunikacije i prijenos male količine ključnih podataka prema BPM rješenju.
- Podržan je samo XML, dok formati koji su čitljiviji ljudima (običnи tekst, JSON[39], HTML) nisu podržani, što otežava testiranje. Dodatno, XML nije kompatibilan s binarnim podacima čime je otežan prijenos priloga kao što su dokumenti, slike, itd.
- SOAP servis zahtijeva održavanje otvorene konekcije prema klijentu, a za testiranje je potreban gotov SOAP klijent ili zasebni programski alat.

Općenito, SOAP web servisi su dobra opcija u slučaju da je potrebno implementirati složeniju funkcionalnost te realizirati kompleksnu interakciju između više sustava, što nije slučaj u predloženom modelu integracije.

[39] **JSON** (JavaScript Object Notation) je standard za razmjenu podataka. Ljudima je razumljiv i lagan za čitanje, a računalima za parsiranje i generiranje. JSON je tekstualni format koji je potpuno neovisan o jeziku, ali koristi konvencije slične onima u jezicima kao što su C, C + +, C #, Java, JavaScript, Perl, Python, i mnogi drugi. Navedene osobine čine JSON idealni jezik za razmjenu podataka.

- **REST**

Moglo bi se reći da je REST sučelje između sustava koji koriste HTTP protokol, a ključna zadaća REST-a je maksimalno iskoristiti mogućnosti koje nudi HTTP protokol. Pomoću HTTP protokola sustavi dohvaćaju podatke u određenim formatima (primjerice XML ili JSON), te generiraju operacije nad tim podacima. REST API (ili RESTful API) je API koji je napravljen prema REST principima, dok je REST web servis (ili RESTful web servis) primjer REST API-a. Osnovne karakteristike REST arhitekture koje se mogu koristiti u realizaciji predloženog modela integracije, navedene su u nastavku.

  - Ako koristimo REST, sve što je potrebno je mrežna veza između dva sustava. Jedinstveno sučelje omogućuje klijentu da razgovara s poslužiteljem na jednom jeziku, neovisno o arhitekturi u pozadini.
  - Za razliku od SOAP servisa, REST servisi su lakši za korištenje i implementaciju jer se temelje na jednostavnim HTTP operacijama, koje odrađuju čitanje, pisanje i brisanje podataka.
  - REST API je brz i jednostavan za integraciju te je lako primjenjiv u naslijeđenim sustavima pisanim u različitim tehnologijama. Relativno lagano je napraviti REST poziv iz pohranjene procedure u bazi podataka.
  - REST ne pamti stanje, što znači da pozivi mogu biti međusobno neovisni, a svaki poziv sadrži sve podatke potrebne za uspješno izvršavanje. Kao rezultat toga, testiranje je jednostavnije, a kao alat za test dovoljan je samo internetski preglednik.
  - Jedna od ključnih prednosti REST API-ja jest velika fleksibilnost što je važno za predloženi model integracije. Podaci nisu vezani za resurse i metode, radi se s reprezentacijom stvarnih resursa koja se po potrebi može dodatno pojednostaviti.
  - Unatoč raznolikosti podržanih formata poruka (običan tekst, JSON, XML, itd.), većina REST API-ja koristi JSON kao zadani format poruka. JSON je čitljiv, jednostavan i fleksibilan format poruka koji povećava brzinu komunikacije. Ujedno, HTTP odgovor REST poziva vraća sirove podatke koje nije potrebno izvlačiti iz neke strukture ili procesirati, već ih možemo direktno koristiti.
  - Omogućava jednostavno osiguravanje komunikacije korištenjem HTTP zaglavlja. HTTP zaglavlja su sastavni dijelovi poruka HTTP protokola, a definiraju dodatne parametre HTTP transakcije.

REST ostvaruje labavu i fleksibilnu komunikaciju koja se može jednostavno prilagođavati i nadograđivati. Usprkos jednostavnosti REST je vrlo moćan, te praktički ne postoji ništa u području web servisa što se ne može ostvariti RESTful arhitekturom.

Zbog razloga opisanih u provedenoj analizi REST je odabran kao primarni način komunikacije u predloženom modelu integracije.

5.1.2 Tehničke preporuke implementacije BPMKS-a

Odabir tehnologija, alata i programskih okvira za tehničku specifikaciju i izgradnju BPMKS-a vođen je navedenim karakteristikama modela (poglavlje 3.3.1) uz fokus na jednostavnost i brzinu implementacije.

Neke od poželjnih karakteristika skupa alata koji bi bio dobar za korištenje kod izgradnje BPMKS-a su:

- Osiguranje nefunkcionalnih zahtjeva BPMKS-a (engl. non-functional requirements) kao što su sigurnost, upravljanje transakcijama, uspostavljanje konekcija prema bazama podataka, itd.
- Dostupnost skupa gotovih programskih biblioteka koje se mogu uključiti u aplikaciju, a pružaju različite gotove funkcionalnosti potrebne za funkcioniranje BPMKS-a. Primjerice, mogućnost uspostave komunikacije (gotovi alati za uspostavu REST, SOAP poziva), mapiranje podataka iz poziva (XML/JSON serijalizacija/deserijalizacija, itd.).
- Mogućnost jednostavne konfiguracije, te ponovog korištenja programskih komponenti (gotovih i razvijenih) u različitim dijelovima aplikacije.
- Sustav za organizaciju svih ovisnih komponenti (engl. Dependency Management System) kako bi svi dijelovi aplikacije bili uredno povezani i posloženi.
- Uključeni poslužitelj (servlet kontejner) na kojem se aplikacija može izvršavati, što značajno olakšava konfiguraciju poslužitelja te razvoj aplikacije na poslužitelju.
- Jednostavan način pisanja testova, mogućnost testiranja pojedinih funkcionalnosti i dijelova aplikacija što uvelike olakšava razvoj.
- Mali početni napor kreiranja aplikacije uz fokus na razvoj poslovnih funkcionalnosti.
- Dostupnost materijala, edukativnih sadržaja, resursa i informacija o navedenom alatu/okviru (primjerice, raširena internetska zajednica korisnika koji surađuju i pomažu si u rješavanju specifičnih problema).

Obrazložimo i navedimo primjere alata odabrane temeljem opisanih poželjnih karakteristika za implementaciju BPMKS-a. Navedeni alati pretežno će se koristiti za izradu programske podrške i primjera u ovom radu. Odabrani su:

- **Spring Boot radno okruženje** kao osnovni radni okvir za izradu BPMKS-a. Bazirano je na radnom okviru Spring koji omogućuje izgradnju labavo povezanih aplikacija te znatno olakšava razvoj poslovnih aplikacija pružajući opsežnu nefunkcionalnu podršku. Spring je prisutan na tržištu već petnaestak godina te se još uvijek intenzivno koristi, a ima i širok skup dostupnih materijala i resursa. Spring Boot zadržava mogućnosti i pozitivne strane Springa (čuva koncepte i principe kao što su *Controller*, *Bean*[40], *Services*, *JPA*[41]*, injekcija ovisnosti i inverzija kontrole*[42]), te pokušava poboljšati nedostatke Springa (primjerice opsežnu XML Spring konfiguraciju u potpunosti zamjenjuje jednostavnim anotacijama[43]). Dolazi s ugrađenim poslužiteljem (npr. Apache Tomcat) te omogućava brzu izgradnju aplikacija uz pomoć tzv. *starter projekta* (skupova potrebnih funkcionalnosti za određene tipove aplikacija koje želimo razvijati, primjerice SOAP ili REST servise, web aplikacije, itd.).
- sustav **Apache Maven** za organizaciju ovisnosti (engl. Dependency Management System) kako bi razvojni stručnjaci kod izgradnje BPMKS-a mogli uredno koristiti, povezati i organizirati razne gotove komade programskog koda, biblioteke programskih komponenti te različite API-je.

[40] U kontekstu Spring aplikacije, ***bean*** predstavlja osnovni element aplikacije, instancirani objekt kojim upravlja Spring IoC kontejner. Spring IoC kontejner možemo gledati kao "vreću objekata" kojom upravlja Spring, te koji se mogu višestruko koristiti u raznim dijelovima aplikacije. *Beanovi* su stvoreni temeljem metapodataka koju se serviraju Springu (primjerice u obliku XML definicija).

[41] **Java Persistence API** (**JPA**) je specifikacija Java aplikacijskog programskog sučelja koja opisuje pristup, održavanje i upravljanje podacima između Java objekata (Java klasa) i relacijskih baza podataka. JPA je postao standardni pristup za ORM (Object to Relational Mapping) u Java programskom svijetu. ORM je programska tehnika za pretvaranje podataka između nekompatibilnih sustava pomoću objektno orijentiranih programskih jezika. ORM zapravo stvara "bazu virtualnih objekata" koja se može koristiti iz određenog programskog jezika.

[42] **Injekcija ovisnosti** (engl. **Dependency injection**, **DI**) je uzorak dizajna korišten u razvoju programskih rješenja koji implementira Inverziju kontrole (engl. **Inversion of Control**, **IoC**) za rješavanje ovisnosti. Najčešće korišteni ovisni objekti su servisi, a prosljeđuju se klijentskom objektu koji te ovisne objekte koristi. Klijentski objekt također može vanjskom programu dati odgovornost pružanja njegovih ovisnosti, što se naziva principom obrnute ovisnosti i zapravo je jedan od ključnih koncepata Spring radnog okruženja.

[43] **Anotacije** (engl. **annotations**) su uvedene još u Java 5 programskom paketu i nisu specifične samo za Spring. Općenito, anotacije omogućuju dodavanje metapodataka u klase, na metode ili varijable. Anotaciju može interpretirati kompajler (primjerice @Override anotacija) ili radni okvir poput Springa (npr. @Component).

- alat **Swagger** u svrhu preglednosti, lakšeg strukturiranja te kreiranja dokumentacije REST servisa. Swagger je specifikacija i kompletni radni okvir za opisivanje, dokumentiranje, korištenje te vizualizaciju RESTful API-ja. Omogućava kreiranje dokumentacije funkcionalnosti BPMKS-a, u skladu s OpenAPI specifikacijom (OAS), što će se pokazati korisnim kod realizacije karakteristike ***proširljivosti*** vlastitog modela (poglavlje 5.4.2). U BPMKS-u koristi se *SpringFox* implementacija Swagger specifikacije.
- Preglednik **Hypertext Application Language (HAL)** je alat za nadzor aplikacije koji prati status aplikacije i stanje pojedinačnih servisa. Bilježi metrike o broju poziva pojedinih servisa, informacije o pogreškama te omogućava analizu detalja obrađenih zahtjeva.
- alat **SoapUI** korišten za funkcionalno testiranje izgrađenih REST i SOAP servisa. Omogućuje jednostavno i brzo kreiranje i izvršavanje automatiziranih funkcionalnih i regresivnih testova te konkretne simulacije stvarnih poslovnih slučajeva.

5.1.3 Odabrani BPMS, terminologija i koncepti

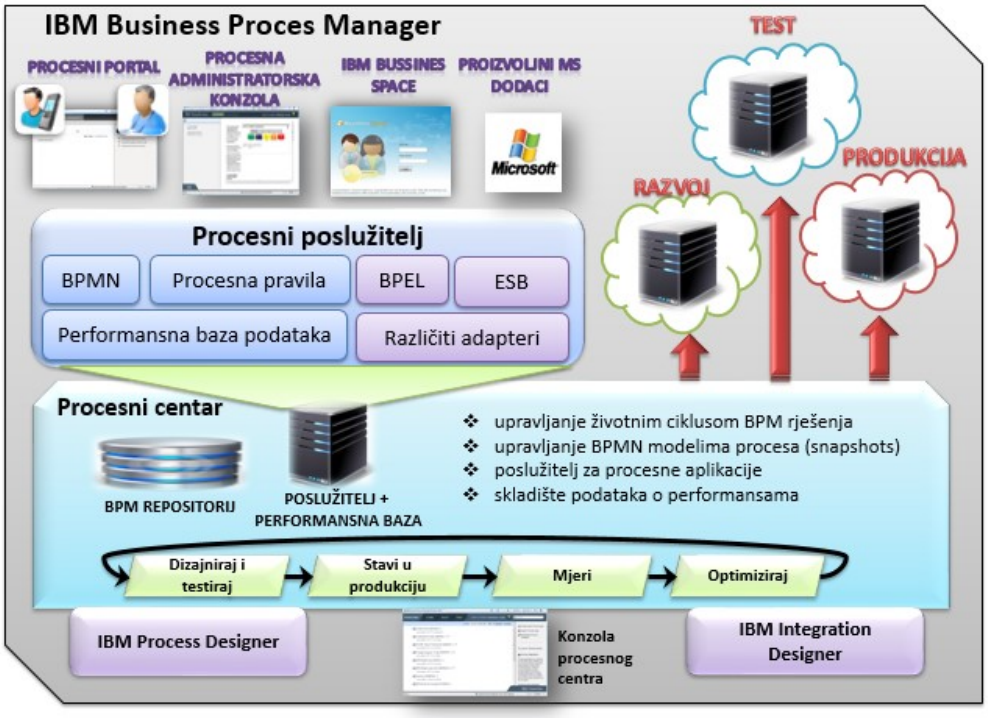
Pretpostavka korištenja BPMS-a u vlastitom modelu integracije su obavezne karakteristike izložene u poglavlju 4.3.1, a veći broj današnjih BPMS-ova, što otvorenog koda (npr. Alfresco Activiti, Camunda BPM, jBPM, itd.), što komercijalnih alata (TIBCO ActiveMatrix BPM, Oracle BPM, IBM BPM, itd.), zadovoljava navedene karakteristike.

U vrijeme pisanja ovog rada, u sklopu organizacije u kojoj je rad pisan intenzivno se koristio BPMS kompanije IBM (IBM BPM ili IBPM), Budući da taj skup alata ima sve potrebne karakteristike za korištenje u vlastitom modelu, odabran je za izradu primjera, demonstraciju koncepata i prikaz strukturne inovacije PP-a u predloženom vlastitom modelu integracije. Dodatna tvrdnja koja podupire ovu odluku je činjenica da je, trenutno, IBPM jedan od vodećih komercijalnih alata u BPM području [66].

Zbog razumijevanja primjera prikazanih u radu, ukratko su objašnjene osnovne karakteristike i koncepti IBPM BPMS-a te prikazani načini programabilne interakcije s PP-ima.

5.1.3.1 Struktura, osnovne komponente i pojmovi

IBPM se sastoji od nekoliko usko povezanih komponenti od kojih svaka obavlja specifičnu ulogu te se koristi u određenom stupnju razvoja BPM rješenja. Slika 5.1 (izrađena prema [58], [59]) prikazuje nekoliko osnovnih komponenti koje se mogu izdvojiti u slučaju IBPM-a.



Slika 5.1 Struktura i osnovne komponente IBM BPMS-a

- **Procesni poslužitelj** (engl. Process server) komponenta je na kojoj se izvršavaju definicije procesa opisanih dijagramima procesa [58]. Važno je razlikovati dijagram procesa od instance PP-a. Dijagram PP-a vizualizira model procesa koji je predložak procesa, a instanca predstavlja proces koji je kreiran i pokrenut na temelju predloška.
  - ***Procesna administratorska konzola*** (engl. Process Admin Console) važan je dodatak na procesni poslužitelj, omogućuje upravljanje procesnim poslužiteljima u testnim i produkcijskim okruženjima te poslužiteljem procesnog centra tijekom razvoja.
  - ***Procesni portal*** (engl. Process Portal) je komponenta koja služi kao sučelje prema krajnjim korisnicima, sudionicima procesa, te im omogućuje pokretanje instanci različitih procesa i obavljanje zadataka za koje su zaduženi.
- **Procesni centar** (engl. Process Center) je "spremište" za sve definicije PP-a, servise te ostale komponente neophodne za izvršavanje procesa. Procesni centar realizira tzv. koncept "zajedničkog modela", to jest postojanje jednog "spremišta" gdje su pohranjeni svi elementi procesnog rješenja [58]. Tako se sprečava postojanje dvije različite faze istog rješenja koje nije moguće međusobno uskladiti [60]. Osim repozitorija procesni centar sadrži jednu ili više instanci procesnog poslužitelja. Tu je i performansno skladište podataka (engl. Performance Data Warehouse ili PDW) koje služi za prikupljanje, sortiranje i analizu podataka generiranih od strane procesnih poslužitelja.

- ***Konzola procesnog centra*** (engl. Process Center Console) je internetsko sučelje za upravljanje artefaktima koji se nalaze u procesnom centru, a služi za instalaciju procesnih aplikacija na određene procesne poslužitelje.

- **Procesni dizajner** (engl. IBPM Proces Designer ili IBPM PD), je razvojni alat za modeliranje, dizajniranje te izgradnju procesa, to jest izgradnju definicije PP-a kreiranjem DPP-a. IBPM PD se povezuje na jednu instancu procesnog centra i nema mogućnosti promjene te instance [58], a također sadrži integriranu konzolu procesnog centra. Omogućuje MPP u BPMN notaciji.
- **Integracijski dizajner** (engl. Integration Designer) je alat za razvoj samostalnih modula koji se mogu integrirati s procesnim aplikacijama što uvelike pomaže u odvajanju poslovne logike od implementacijskih detalja [59].
- **IBM poslovni prostor** (engl. Business Spaces) je zapravo korisničko sučelje koje ima ulogu agregirati relevantan sadržaj te ga prikazati putem internet preglednika na razumljiv način [58], [59], [61]. Pomoću njega poslovni korisnici mogu pregledavati portfelj korisnih podataka koje pruža IBM Business Process Management alat.

5.1.3.2 Izvršavanje akcija, sučelja i korisnička interakcija

Za obavljanje aktivnosti, osnovnih jedinica funkcionalnosti procesa, IBPM koristi koncept servisa, a razlikujemo nekoliko tipova servisa koji se koriste u različitim slučajevima.

Tzv. *ljudski servisi* (engl. *Human Services*) najvažniji su koncept IBPM-a te se koriste kroz primjere u ovom radu. Pokretanjem *ljudskog servisa* kreira se zadatak za čije rješavanje je potrebna ljudska interakcija, a tok procesa suspendira se sve dok se kreirani zadatak ne odradi. *Ljudski servisi* obično obuhvaćaju određene zaslonske maske, sučelja, gdje korisnik obavi potrebnu akciju.

Internetska sučelja *ljudskih servisa* koja prikazuju procesne podatke korisnicima te sakupljaju unosne podatke od korisnika, a nazivaju se *coach*-evi. Svaki *coach* odgovara jednom ekranu ili formi koju korisnik vidi, a sastoji se od jedne ili više *coach view* (*CV*) komponenti. CV komponente predstavljaju elementarna sučelja prema određenom tipu podataka (primjerice polja za unos, labele, itd.), a uz pomoć CSS-a može im se prilagoditi izgled ili JavaScript-om promijeniti ponašanje. *Coach* može imati višestruke akcije kojima se može realizirati grananje toka procesa [58].

U kontekstu ovog rada, sljedeći važan pojam IBPM-a je integracijski servis (engl. Integration service) koji omogućava procesu da obavi poziv vanjskog servisa. Upravo su integracijski servisi jedna od komponenti koja pruža integraciju procesne aplikacije s

funkcionalnostima van BPM okruženja, uključujući i integraciju s naslijeđenim sustavima (KINS komponenta).

5.1.3.3 Programabilna interakcija s procesnim artefaktima BPM rješenja

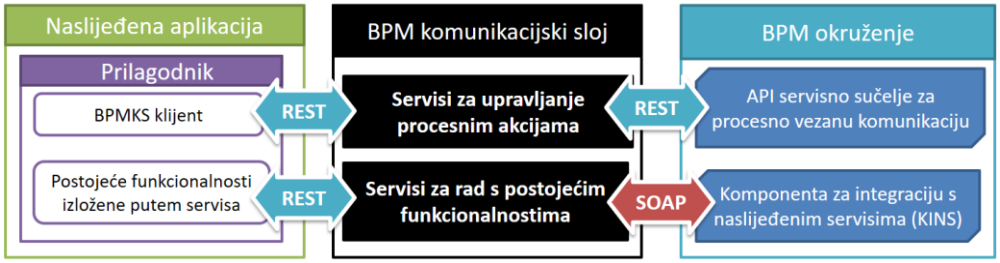
Jedan od glavnih argumenta kod odabira IBPM alata za korištenje u vlastitom modelu integracije je detaljna te dobro razrađena interakcija s procesnim artefaktima izgrađenog BPM rješenja. IBPM onogućuje interakciju temeljem IBPM REST API-ja, skupa preddefiniranih REST servisa, izloženih na IBPM alatu, koji omogućuju interakciju s procesnom aplikacijom. Više o mogućnostima IBPM REST API-ja u poglavlju 5.5.1.

5.2 Shema komunikacije između komponenti

Iako je REST odabran kao primarni način komunikacije (poglavlje 5.1.1), jedan dio komunikacije između komponenti implementiran je korištenjem SOAP servisa kako bi pokazali dodatnu fleksibilnost predloženog modela integracije spram tehnologija komunikacije.

Dodatan razlog leži u verziji BPMS-a koji se koristi za demonstraciju u ovom radu, IBPM standardni paket alata verzije 8.5.7 (poglavlje 5.1.3.). Jedan od glavnih nedostataka korištene verzije je implementacija poziva vanjskih REST servisa iz BPM okruženja, koja nije najbolje realizirana. Za implementaciju poziva potrebne su dodatne programske biblioteke i dodatni IBM alati kao što je IBM Integration Designer, koji nisu bili dostupni u standardnom paketu IBPM alata. Od novije verzije IBPM alata (verzije 8.5.7) pozivanje vanjskih REST servisa bitno je pojednostavljeno, temelji se ne otkrivanju OpenAPI specifikacije REST servisa, te je dostupno u standardnom paketu IBPM alata.

Zbog navedenog, dio komunikacije koji uključuje pozive BPMA prema BPMKS-u za dohvaćanje podataka iz naslijeđenih sustava realiziran je SOAP web servisima. Slika 5.2 prikazuje konceptualnu shemu komunikacije između komponenti modela integracije, s obzirom na korištene tehnologije komunikacije.



Slika 5.2 Korištene tehnologije komunikacije između komponenti

- **REST dio komunikacije**

  Strana koja inicira REST poziv naziva se REST klijent, a REST servis koji se poziva poznat je pod imenom REST *endpoint*. Kako bi se podaci prenijeli od REST klijenta do *endpointa*, potrebno je pretvoriti (serijalizirati[44]) podatke, objekte iz programskog jezika u kojem je napisan klijent, u oblik koji može putovati mrežom, primjerice JSON, XML ili *byte array* format. Standardni oblik korišten uz REST pozive kroz ovaj rad je JSON.

  REST komunikacija vezana je uz HTTP protokol te se za baratanje podacima koriste standardne HTTP metode, a u BPMKS-u primarno će se koristiti GET (dohvat podataka), POST (spremanje ili kreiranje podataka) te PUT (promjena podataka).

  Većina komunikacije odvija se od naslijeđenih sustava prema BPM rješenju, a kao podršku tom kanalu BPMKS sadrži REST *endpointove* za naslijeđene aplikacije, te REST klijenta kojim se pozivaju REST *endpointovi* izloženi na BPM okruženju. Mali dio komunikacije odvija se od BPM rješenja prema naslijeđenim sustavima, a u svrhu podrške toj komunikaciji BPMKS sadrži REST pozive prema naslijeđenim aplikacijama.

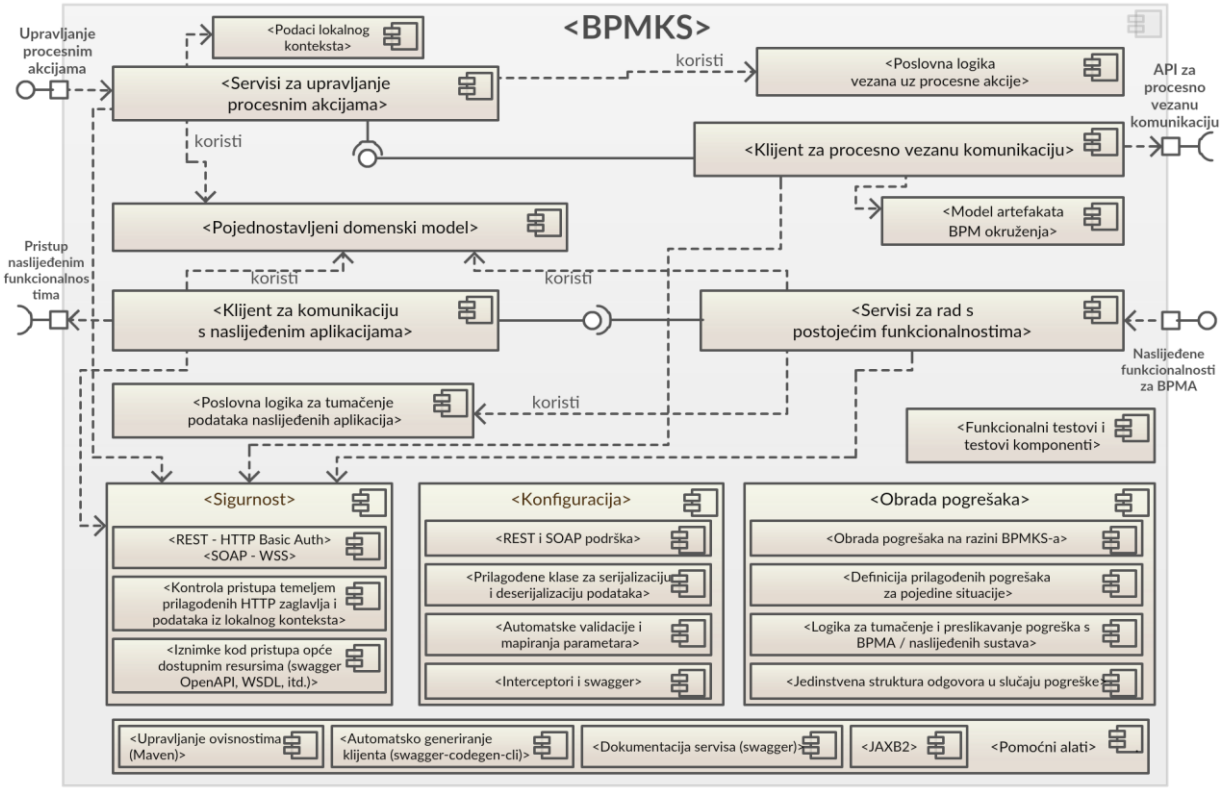
- **SOAP dio komunikacije**

  Jedan dio komunikacije od BPM rješenja prema naslijeđenim sustavima odvija se putem SOAP-a. U svrhu podrške navedenom dijelu komunikacije BPMKS ima SOAP *endpointove* izložene prema BPM alatu (BPMKS komponenta ***servisi za rad s postojećim funkcionalnostima***). Izloženi SOAP *endpointovi* koriste REST klijenta unutar BPMKS ***komponente za rad s postojećim funkcionalnostima*** kako bi generirali pozive prema naslijeđenim aplikacijama.

5.3 Struktura, tehnička specifikacija i izgradnja BPMKS-a

Kao što je već opisano u poglavlju 4.2 BPMKS je modul koji zaprima i obrađuje pozive servisa (REST i SOAP) preslikava dobivene podatke te dalje inicira pozive servisa sukladno s implementiranom poslovnom logikom. Slika 5.3 prikazuje konceptualni UML dijagram komponenti BPMKS-a, a zbog preglednosti i čitljivosti dijagrama, prikazane su samo osnovne međuovisnosti komponenata.

[44] **Serijalizacija** je proces prevođenja podatkovnih struktura ili stanja objekta u format koji se može pohraniti ili prenijeti preko mreže te naknadno rekonstruirati u izvorni oblik u drugom različitom računalnom okruženju. Proces rekonstrukcije još je poznat i pod nazivom **deserijalizacija**.



Slika 5.3 Dijagram komponenti BPMKS-a

Sukladno shemi komunikacije iz poglavlja 5.2, BPMKS se može podijeliti na dva osnovna modula, modul odgovoran za REST komunikaciju i modul zadužen za obradu SOAP zahtjeva.

BPMKS je temeljen na radnom okviru Spring s kontrolerima koji imaju ulogu baratanja dolaznim zahtjevima. Važna komponenta za razumjeti u arhitekturi MVC je središnji servlet, *DispatcherServlet* u REST modulu, te *MessageDispatcherServlet* u SOAP modulu. Središnji servlet zaprima sve zahtjeve upućene prema aplikaciji (BPMKS-u) te ih zatim preusmjerava komponentama koje obavljaju potreban posao. U obrascu arhitekture MVC komponente koje obavljaju potreban posao predstavljene su kontrolerima, to jest klasama označenim anotacijom *@RestController* za REST te anotacijom *@Endpoint* za SOAP modul.

BPMKS aplikacija sastoji se od dvije glavne komponente definirane u zasebnim paketima. Komponenta ***servisi za upravljanje procesnim akcijama*** služi za komunikaciju prema BPM alatu (npr. paket *hr.pbz.bpm*), dok komponenta ***servisi za rad s postojećim funkcionalnostima*** komunicira s naslijeđenim aplikacijama (npr. paket *hr.pbz.legacy*).

Preporuke za izgradnju modela domene navedene su u poglavlju 5.3.5 s opisima uloga prilagođenih HTTP zaglavlja i principa lokalnog konteksta te pripadnom konfiguracijom. Poglavlje 5.3.7 prikazuje korištene principe sigurnosti i obrade pogrešaka na razini BPMKS-a.

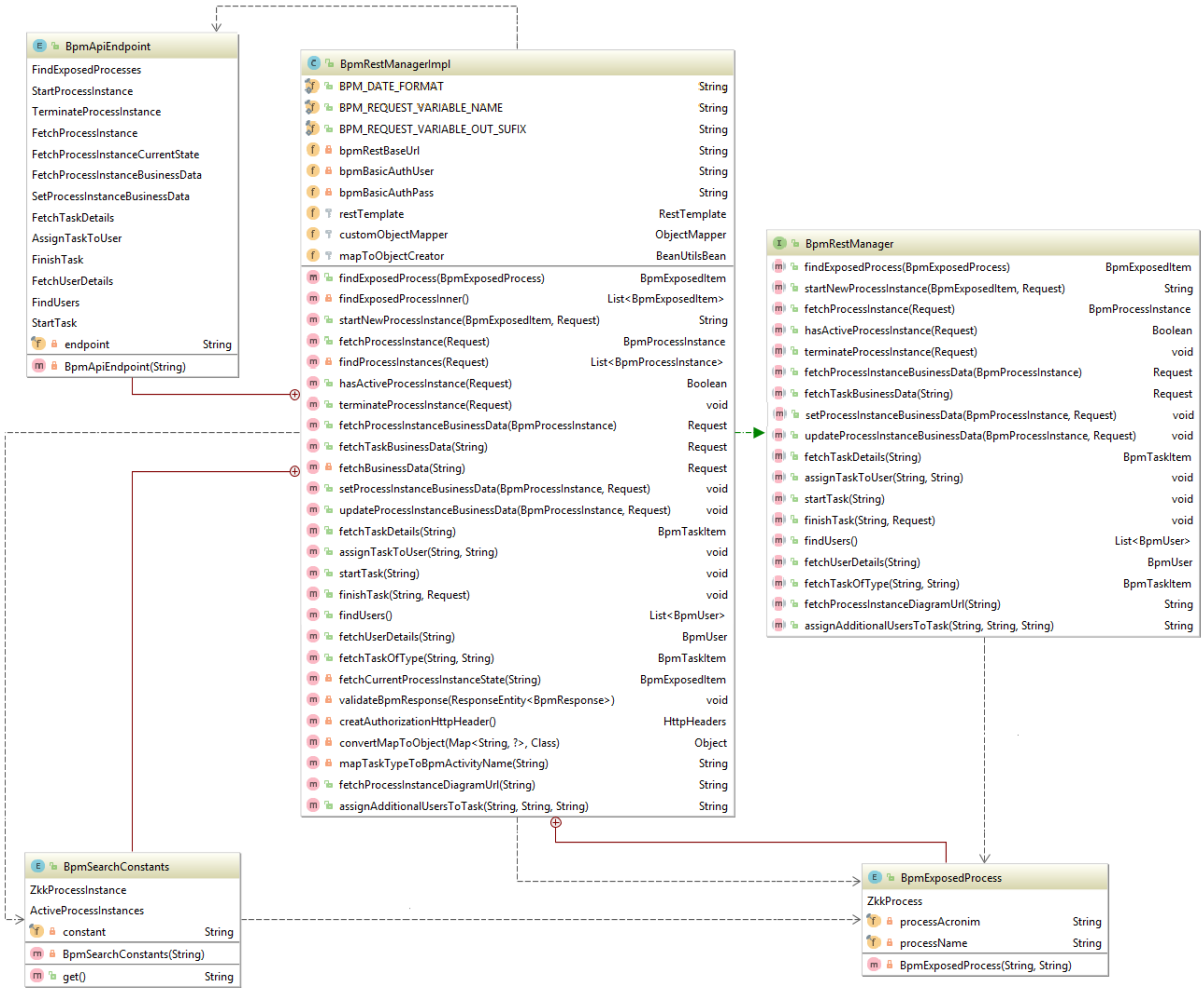
U svrhu postizanja karakteristike ***proširljivosti*** vlastitog modela integracije (poglavlje 3.3.1) predloženi su principi automatskog kreiranja REST klijenta za ugradnju u prilagodnike naslijeđenih aplikacija (poglavlje 5.4.2).

Kroz poglavlja u nastavku, nizom dijagrama klasa pojedinih komponenti, objašnjeni su detalji strukture BPMKS-a.

5.3.1 Klijent za procesno vezanu komunikaciju

Kao što je već navedeno u poglavlju 4.2 ova komponenta predstavlja implementaciju poziva API-ja za procesno vezanu komunikaciju, izloženih od strane BPM okruženja. Navedeni API-ji manipuliraju procesnim artefaktima na BPMA, te obavljaju određenu elementarnu akciju. Elementarna akcija može biti pokretanje instance procesa, mijenjanje podatka instance procesa, dodjeljivanje aktivnosti korisniku, itd.

Slika 5.4 prikazuje dijagram klasa i strukturu komponente ***klijenta za procesno vezanu komunikaciju***.



Slika 5.4 Dijagram klasa klijenta za procesno vezanu komunikaciju

Metode ***klijenta za procesno vezanu komunikaciju***, koje obavljaju elementarne akcije, nalaze se implementirane u klasi *BpmRestManagerImpl*, a koriste funkcionalnosti *RestTemplate beana*. Klasa *RestTemplate* predstavlja programsku podršku za uspostavu REST poziva, a dio je radnog okvira Spring. Sadrži metode za pretvorbu Java objekata u JSON format, ima mogućnost čitanja podataka iz JSON formata, te može obaviti HTTP pozive. Metoda *creatAuthorizationHttpHeader()* klase *BpmRestManagerImpl* postavlja autorizacijske podatke za poziv IBPM REST API sučelja, dok metoda *validateBpmResponse(...)* provjerava ispravnost odgovora.

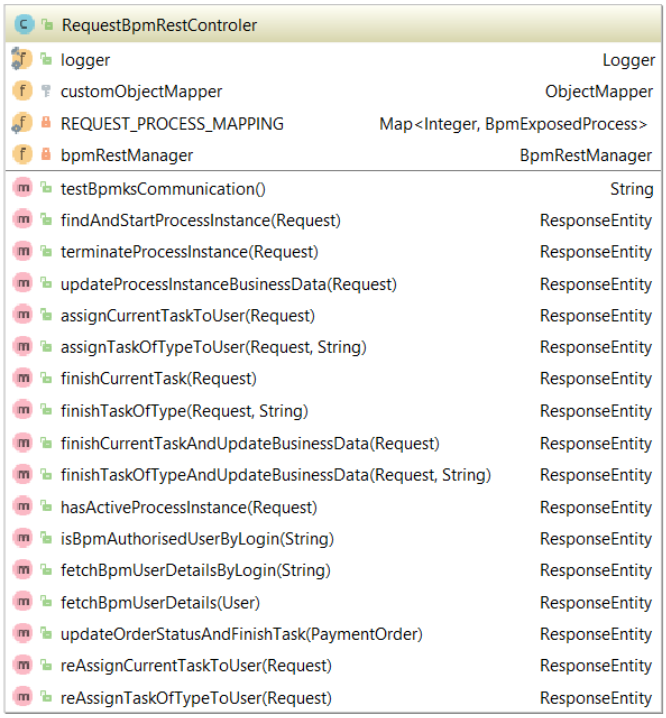
Klasa pobrojanog tipa (engl. enum) *BpmApiUrl* sadrži URL-ove (engl. Uniform Resource Locator) *endpointova* izloženih na IBPM REST API sučelju. Pobrojani tip *BpmSearchConstants* sadrži sve uvjete za pretraživanje instanci procesa na BPM okruženju o čemu više u poglavlju 5.7 i 5.7.1. Pobrojani tip *BpmExposedProcess* sadrži popis procesa podržanih od strane BPMA.

5.3.2 Servisi za upravljanje procesnim akcijama

Ova komponenta sadrži skup REST *endpointa* koji implementiraju niz operacija za manipulaciju procesnim artefaktima BPM rješenja. *Endpointovi* se pozivaju od strane naslijeđenih aplikacija, uz pomoć ugrađenog prilagodnika (poglavlje 5.4) i to u standardnim procesnim točkama PP-a.

Svaki *endpoint* komponente ***servisi za upravljanje procesnim akcijama*** prima podatke od naslijeđenih aplikacija temeljem kojih obavlja jednu ili više elementarnih akcija s procesnim artefaktima na BPMA, koristeći metode komponente ***klijent za procesno vezanu komunikaciju***.

Slika 5.5 prikazuje realizaciju komponente klasom *RequestBpmRestControler*, a pojedini *endpoint* (metoda unutar klase s odgovarajućim anotacijama) zadužen je za obavljanje određene operacije nad instancom procesa u BPMA.



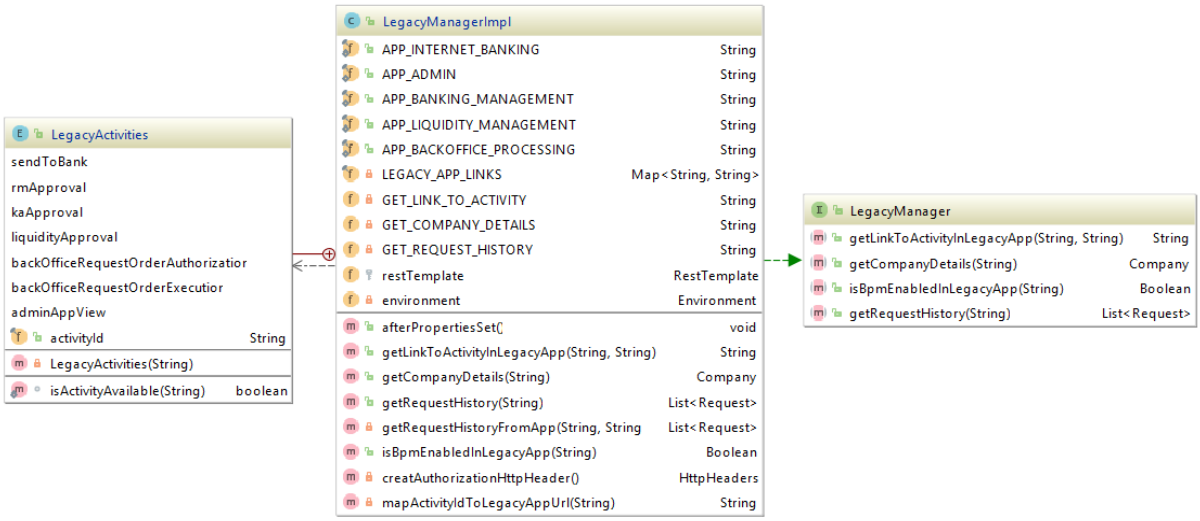
Slika 5.5 Klasa endpointa servisa za upravljanje procesnim akcijama

Svaki od izloženih REST *endpointa* odgovara na određenu HTTP akciju (GET, POST, PUT…), mapiran je na određen URL te po potrebi prima određene parametre. Klasa *ResponseEntity* predstavlja HTTP odgovor, a sadrži standardne metode pomoću kojih možemo kontrolirati i definirati svaki dio HTTP odgovora (zaglavlje, tijelo i status).

5.3.3 Klijent za komunikaciju s naslijeđenim aplikacijama

Ova komponenta implementirana je klasom *LegacyManagerImpl* (Slika 5.6), a sadrži:

- metode za dohvat podataka iz naslijeđenih aplikacija: *getLinkToActivityInLegacyApp() getCompanyDetails(), getRequestHistory(), itd..* Navedene metode pomoću REST-a pozivaju odgovarajuće servise ***komponente za izlaganje postojećih funkcionalnosti*** iz prilagodnika ugrađenog u naslijeđene aplikacije.
- identifikatore svih naslijeđenih aplikacija s njihovim osnovnim adresama (*Map<String,String> LEGACY_APP_LINKS*). Osnovne adrese su dinamički učitane iz konfiguracije BPMKS-a.
- popis dozvoljenih aktivnosti (pobrojani tip *LegacyActivities*) koje službenici izvršavaju u sklopu toka PP-a
- mapiranje aktivnosti na naslijeđene aplikacije koje te aktivnosti izvršavaju (metoda *mapActivityIdToLegacyAppUrl)*
- ostale pomoćne metode i komponente korištene u komunikaciji (npr. *restTemplate bean*)



Slika 5.6 Dijagram klasa REST klijenta za pozive prema naslijeđenim aplikacijama

5.3.4 Servisi za rad s postojećim funkcionalnostima

Ova komponenta koristi se od strane BPMA u pozadinskim procesnim točkama kada BPMA dohvaća podatke iz naslijeđenih sustava. Sadrži *endpointe* koji primaju instrukcije od BPMA te zatim vrše jedan ili više poziva prema naslijeđenim aplikacijama koristeći metode komponente ***klijent za procesno vezanu komunikaciju***.

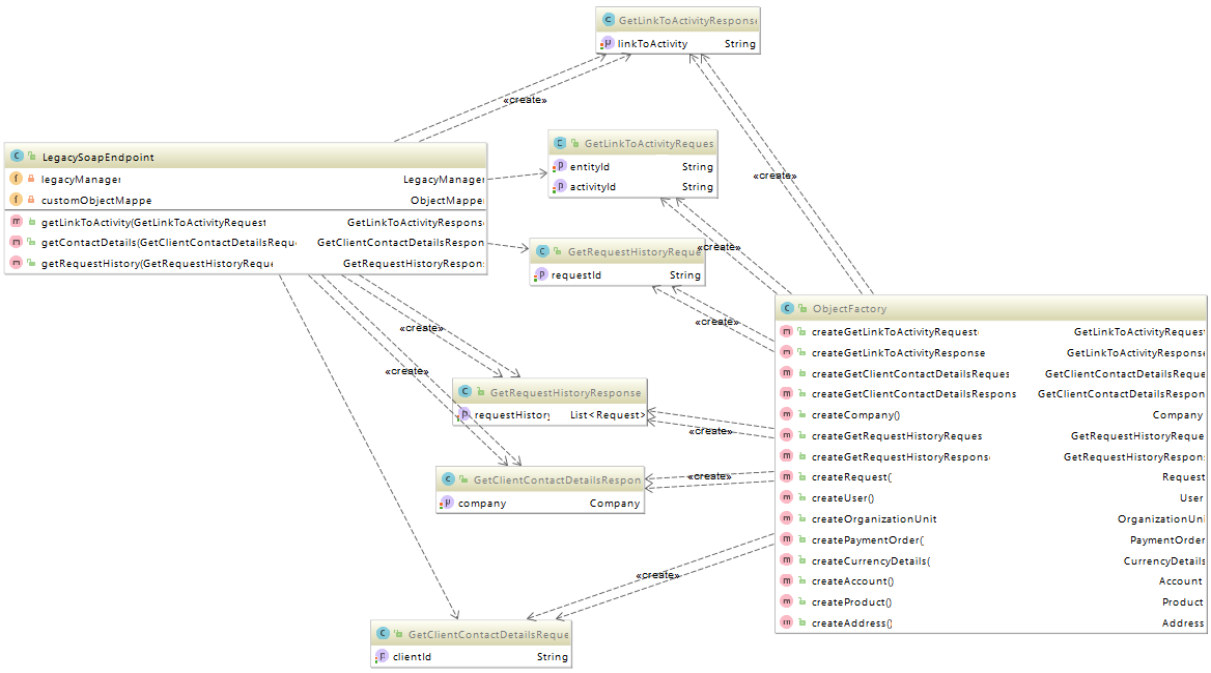
Slika 5.7 prikazuje dijagram klasa komponente ***servisi za rad s postojećim funkcionalnostima*** generiranih iz XML SOAP definicije web servisa. Zbog preglednosti, Slika 5.7 ne prikazuje auto-generirane klase modela domene korištene u SOAP pozivima, koje su strukturom identične klasama modela domene BPMKS-a. Model domene objašnjen je i razrađen u poglavlju 5.3.5.

Kreiranje XML definicije SOAP servisa (prikazanog na Slika 5.7) uključuje:

- kreiranje *legacy-ws.xml* datoteke gdje je opisana struktura SOAP XML zahtjeva i odgovora svakog SOAP web servisa.
- izgradnja *legacy-ws.xsd* datoteke koja opisuje korištene elemente, tipove podataka te pravila u XML SOAP porukama to jest definira XSD[45] dokument SOAP web servisa. XSD sadrži i definiciju tipova podataka, koji odgovaraju klasama modela domene na BPMKS-u. Da bi olakšali kreiranje XSD datoteke, preporučljivo je iskoristiti mogućnost razvojnih IDE-a (npr. IntelliJ IDEA) da automatski generiraju XSD opis temeljem Java klase.

[45] **XML Shema Definition (XSD)** omogućava primjenu pravila u XML dokumentu, definiranje strukture elemenata, određivanje vrijednosti koje određena polja mogu sadržavati, vezu između elemenata itd.

- kreiranje *legacy-ws.xjb* datoteke koja određuje naziv odredišnog paketa generiranih Java klasa (*hr.pbz.legacy.soap*) za XML shemu definiranu u *legacy-ws.xsd* datoteci.



Slika 5.7 Dijagram klasa BPMKS SOAP servisa generiranih iz XML definicije

Nakon definicije servisa, uz pomoć implementacije JAXB[46] standarda programskog sučelja zahtjevi i odgovori SOAP servisa mapiraju se na Java objekte. Korištenjem dodatka JAXB maven, iz prethodno kreiranih definicija stvaraju se Java objekti. Primjerice, tako nastaju klase oblika *Get…Request* i *Get…Response* (Slika 5.7) koje predstavljaju tijelo SOAP XML zahtjeva i odgovora u Javi.

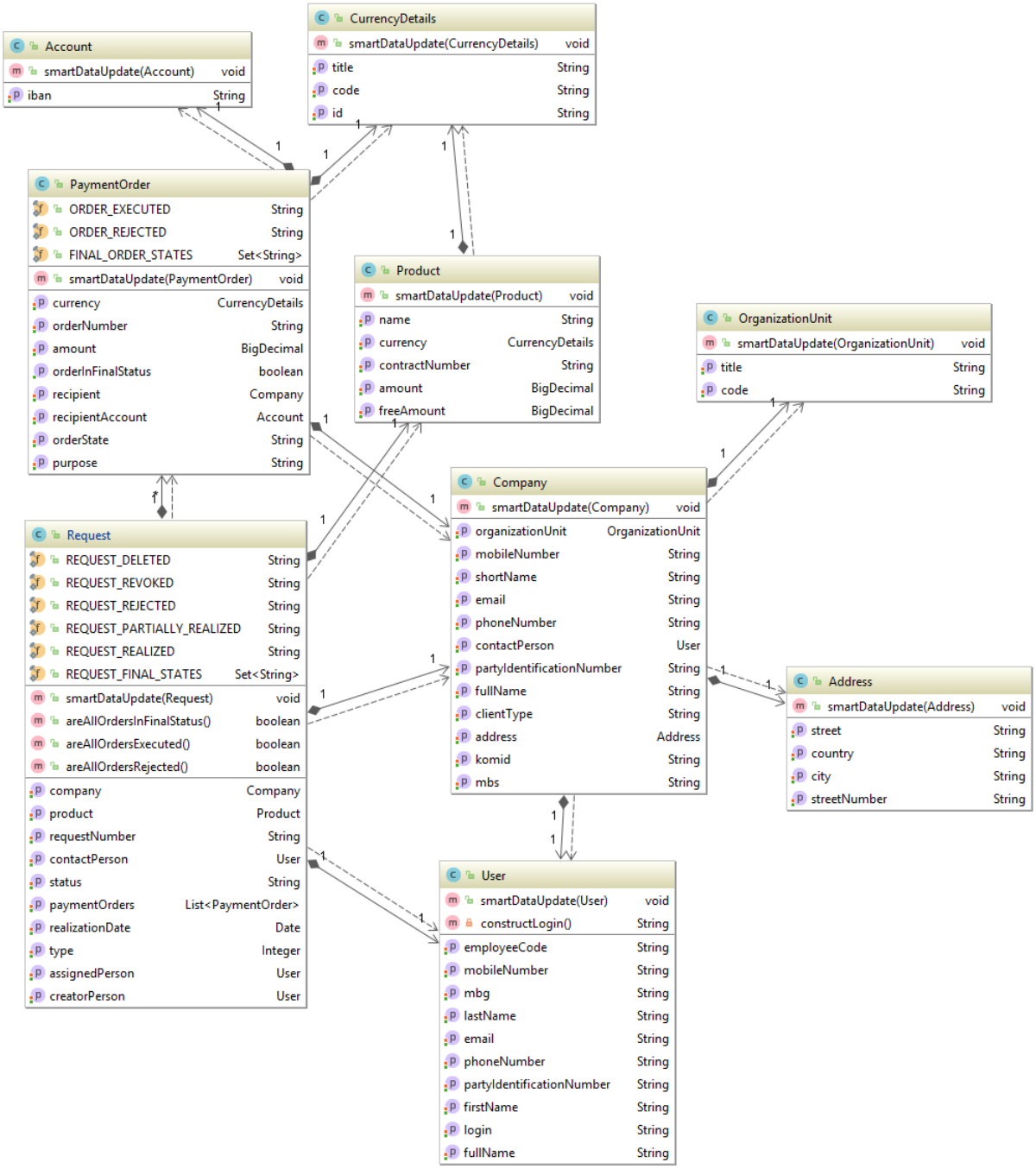
Zadnji korak uključuje kreiranje implementacije SOAP web servisa (engl. *SOAP endpoint*), to jest klase *LegacySoapEndpoint* (označene anotacijom *@Endpoint*) u kojoj se nalaze implementacije metoda SOAP web servisa.

Kao što je već navedeno, *endpoint* SOAP servisa radi pozive REST servisa izloženih prilagodnikom u naslijeđenim aplikacijama, te prilikom toga mora raditi preslikavanje podataka iz modela domene BPMKS-a u Java generirane objekte SOAP servisa. Budući da su ti objekti istih struktura, da bi izbjegli ručno kopiranje svakog polja zasebno, koristimo proces serijalizacije izvornog objekta u JSON, te deserijalizacije u objekt SOAP servisa.

[46] **Java Architecture for XML Binding** (**JAXB**) je Java standard koji definira kako se Java objekti pretvaraju u XML i obrnuto. JAXB definira i API za čitanje i pisanje Java objekata u i iz XML dokumenata.

5.3.5 Model domene BPMKS-a

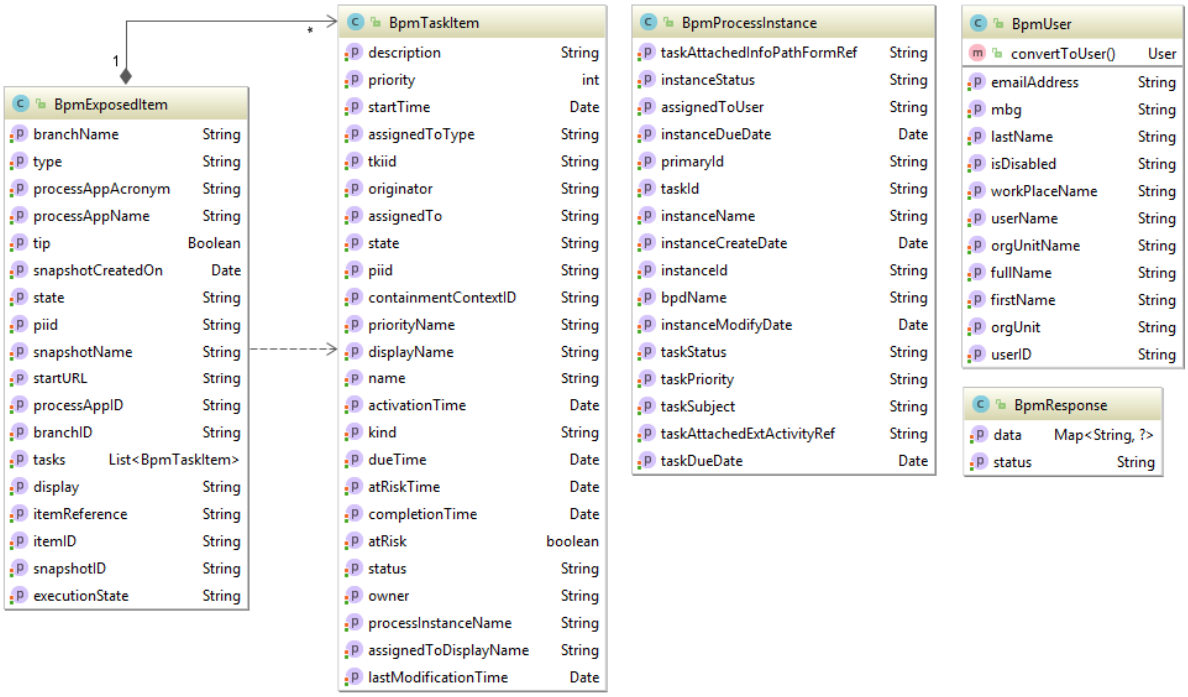
BPMKS ima vlastiti model domene koji se sastoji od pojednostavljenog modela domene naslijeđenih aplikacija. Sadrži klase osnovnih entiteta koji su potrebni za prikaz podataka PP-a. Razina pojednostavljenja modela domene BPMKS-a s obzirom na model domene naslijeđenih sustava demonstrirana je u poglavlju 6.3.4 - Tablica 6.5. Model domene BPMKS-a odgovara poslovnim objektima kreiranim na BPMA (poglavlje 6.3.4). Slika 5.8 prikazuje klase modela domene BPMKS-a te njihove međusobne odnose.



Slika 5.8 Dijagram klasa poslovnog modela domene BPMKS-a

Svaka od klasa poslovnog modela domene BPMKS-a (Slika 5.8) sadrži metodu *smartDataUpdate(...)*. Dotična metoda koristi se u postupku ažuriranja poslovnih podataka na BPMA. Kada se poslovni podaci mijenjaju u naslijeđenim aplikacijama metoda *smartDataUpdate(...)* definira koji podaci dotičnog objekta modela domene se mogu ažurirati i na koji način. Cijeli postupak dodatno je objašnjen u sklopu poglavlja 5.6.

Osim modela domene koji čuva poslovne podatke procesa, BPMKS sadrži i klase koje predstavljaju artefakte iz BPM okruženja, a koriste se prilikom komunikacije s BPM okruženjem putem komponente ***klijent za procesno vezanu komunikaciju***. Slika 5.9 prikazuje dijagram klasa navedenih artefakata iz BPM okruženja.



Slika 5.9 Klase korištene u komunikaciji s BPMA preko IBM BPM REST API-a

5.3.5.1 Kako kreirati model domene BPMKS-a?

Kod konstrukcije modela domene BPMKS-a, nameće se nekoliko pristupa koji su izloženi i objašnjeni u nastavku:

1. BPMKS može sadržavati domenske klase naslijeđene aplikacije i koristiti ih u komunikaciji s naslijeđenom aplikacijom. Te klase mogu se prilikom automatiziranog procesa izgradnje BPMKS-a preseliti iz naslijeđenih aplikacija u BPMKS putem

adekvatnog *build* sustava (alata za kontinuiranu integraciju i izgradnju programskih rješenja, npr. AnthillPro[47], Jenkins[48] i slični).

2. U domenskom modelu BPMKS-a možemo napraviti identične klase kao i u naslijeđenim aplikacijama, sa smanjenim skupom polja klasa, ostavljajući samo polja koja su potrebna u BPMA. Prilikom serijalizacije objekata u naslijeđenoj aplikaciji, generirani JSON sadrži sva polja, međutim deserijalizacija na BPMKS-u napunit će samo istoimena polja koja postoje u odgovarajućoj BPMKS klasi. Kako bi deserijalizacija uspjela potrebno je imati ista imena polja klasa u naslijeđenim aplikacijama i BPMKS-u, koristiti različite anotacije *@Json* za prilagođenu deserijalizaciju ili pak konstruirati vlastite prilagođene komponente za serijalizaciju/deserijalizaciju koje omogućavaju složenije modifikacije podataka.

3. Definirati zasebni model domene unutar BPKMS-u i prilagoditi ga potrebama BPMA. Ovaj pristup adekvatan je u slučaju postupka automatskog generiranja REST klijenta za integraciju u naslijeđene aplikacije koji je opisan u sljedećem poglavlju. Tada se model domene BPKMS-a zapakira s prilagodnikom te se uključuje u naslijeđenu aplikaciju. Dodatno, prilagodnik u naslijeđenim aplikacijama ima ulogu preslikavanja modela podataka iz modela naslijeđene aplikacije u BPMKS model podataka koji se zatim koristi u komunikaciji s BPMKS-om.

5.3.5.2 Zaključci

Ovisno o složenosti i tipu naslijeđene aplikacije koja koristi BPM rješenje, može se primijeniti bilo koji od predloženih načina definicije modela domene. Ako neki proces zahtjeva prijenos velikog broja različitih informacija prema BPMA, može se koristiti opcija 1.

Za slučaj kada model domene naslijeđene aplikacije sadrži velik broj članova koji nisu potrebni za BPMS, preporučljivo je koristiti serijalizaciju u i deserijalizaciju iz JSON-a kao svojevrsni filter za micanje nepotrebnih podataka naslijeđene aplikacije (opcija 2).

Moguća je i situacija gdje svaka od naslijeđenih aplikacija koristi svoj način kreiranja i prijenosa podataka modela domene ili kombinaciju navedenih načina.

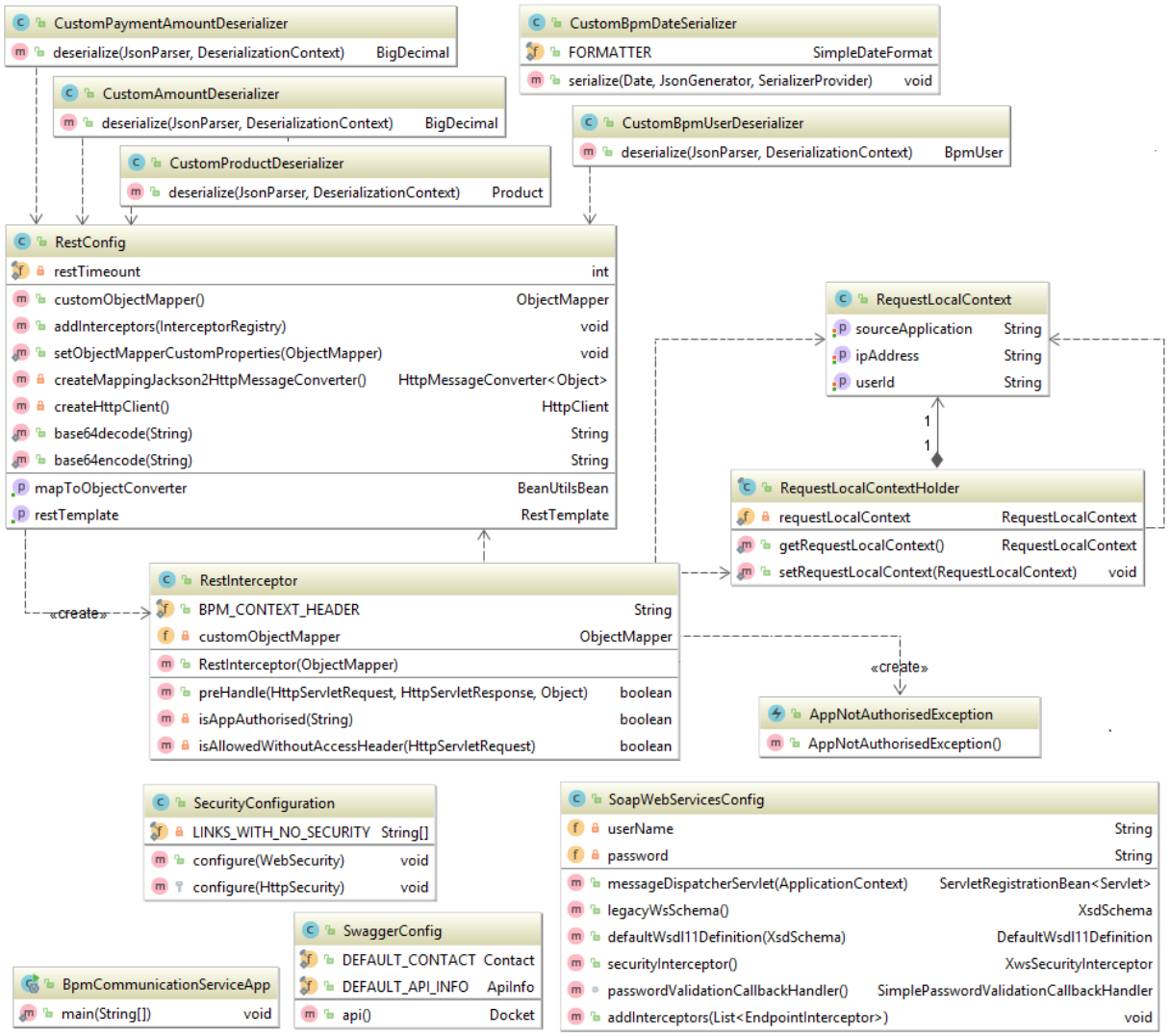
Kroz primjere opisane u ovom radu pretežno je korištena kombinacija opcija 2 i 3.

[47] **AnthillPro** je softverski alat koji automatizira proces izgradnje programskog koda u softverske projekte. Razvojni stručnjaci mogu pratiti napredak izgradnje programskog koda kroz pojedinačne faze procesa izgradnje (engl. build life).

[48] **Jenkins** je Java alat otvorenog koda za automatizaciju ne-ljudskog dijela procesa razvoja softvera, omogućava kontinuiranu integraciju i olakšava tehničke aspekte kontinuirane isporuke programskih rješenja.

5.3.6 Lokalni kontekst i konfiguracija

Komponenta ***lokalni kontekst*** vezana je uz komponentu ***konfiguracije*** BPMKS-a koja automatski puni podatke u komponentu ***lokalni kontekst***. Slika 5.10 prikazuje dijagrame klasa navedenih komponenata, a u nastavku je objašnjena uloga komponenti te pojedinih klasa kojima su realizirane.



Slika 5.10 Dijagram klasa osnovne konfiguracije BPMKS-a

Komponenta ***Lokalni kontekst*** sadrži informacije specifične za poziv koji dolazi na BPMKS. Te informacije nisu vezane uz parametre poziva (npr. URL, tijelo i parametre REST resursa), već predstavljaju metapodatke o zahtjevu i definiraju uvjete u kojima se poziv odvija.

U vidu implementacija BPMKS-a u osnovne karakteristike komponente ***lokalni kontekst*** su:

- karakteristična za REST pozive koji dolaze s naslijeđenih aplikacija
- sadrži podatke kao što je identifikator naslijeđene aplikacije koja je uputila poziv, identifikator korisnika koji radi u naslijeđenoj aplikaciji, itd.

- realizirana je klasama *RequestLocalContext* (definira strukturu opisanih podataka**)** i *RequestLocalContextHolder* (čuva definirane podatke), a životni vijek joj je jednak vremenu trajanja REST zahtjeva.
- bilo koja komponenta na BPMKS-u, u lancu poziva unutar REST zahtjeva, može koristiti podatke iz lokalnog konteksta.
- podaci za komponentu ***lokalni kontekst*** dolaze iz naslijeđenih aplikacija putem prilagođenog HTTP zaglavlja (engl. custom HTTP header) *bpm-context-header* koje se mora nalaziti na svakom REST zahtjevu.
- klasa *RequestLocalContext* i ime zaglavlja (*bpm-context-header*), dio su svakog prilagodnika ugrađenog u naslijeđene aplikacije da bi prilagodnik znao kreirati i navedeno prilagođeno zaglavlje.

Komponenta ***konfiguracije*** BPMKS-a sačinjena je od sljedećih komponenti:

- klase *RestInterceptor* koja ima ulogu presretanja svih REST zahtjeva koji dolaze na BPMKS pri čemu:
  - provjerava *bpm-context-header* prilagođeno zaglavlje, dešifrira vrijednost navedenog zaglavlja u *RequestLocalContext* objekt uz pomoć *customObjectMapper beana*. U slučaju da *bpm-context-header* prilagođeno zaglavlje ne postoji, zabranjuje pristup uz *AppNotAuthorisedException* pogrešku.
  - Za pristup do nekolicine resursa na BPMKS-u (npr. swagger dokumentacija i resursi) nije potrebno *bpm-context-header* prilagođeno zaglavlje, pa je stoga *RestInterceptor* klasa ima provjeru *isAllowedWithoutAccessHeader()* kojom se omogućava pristup bez navedenog prilagođenog zaglavlja.
  - Prema identifikatoru naslijeđene aplikacije provjerava je li određena aplikacija ovlaštena za pristup BPMKS-u (metoda *isAppAuthorised*).
- klase *RestConfig* koja definira sve komponente potrebne za uspostavu REST komunikacije. Tako metodom *createMappingJackson2HttpMessageConverter()* definiramo *MappingJackson2HttpMessageConverter* koji uz pomoć prilagođene *customObjectMapper* komponente radi pretvorbu, to jest čitanje/pisanje Java objekta iz/u JSON format. Metoda *createHttpClient()* služi kako bi kreirali te konfigurirali HTTP klijenta koji odrađuje komunikaciju preko HTTP-a.
- prilagođenih klasa za serijalizaciju/deserijalizaciju, npr.: *CustomProductDeserializer*, *CustomBpmDateSerializer* i druge.

- klase *SecurityConfiguration* koja služi za dodatno izuzimanje određenih resursa BPMKS-a iz sigurnosnih metoda pristupa
- klase *BpmCommunicationServiceApp* koja predstavlja glavnu Spring Boot klasu BPMKS aplikacije
- klase Swager konfiguracije (*SwaggerConfig)*
- klase *SoapWebServicesConfig* koja čini osnovnu konfiguracijsku komponentu izloženih SOAP servisa. Sadrži:
    - *ServletRegistrationBean* s *MessageDispatcherServlet* komponentom koju povežemo na osnovni link gdje će web servisi biti izloženi (*/ws/**). Uloga *MessageDispatcherServlet*-a kod SOAP web servisa je analogna ulozi *DispatcherServlet* komponente kod web aplikacija i REST servisa, a služi za identifikaciju SOAP *endpointova* i baratanje SOAP zahtjevima.
    - *DefaultWsdl11Definition bean* koji omogućava automatsko kreiranje i izlaganje WSDL opisa web servisa.
    - *XwsSecurityInterceptor* interceptor koji je zadužen za sigurnost.

5.3.7 Sigurnost i obrada pogrešaka

Kod izgradnje komponente za sigurnost BPMKS-a, potrebno je zadovoljiti sljedeće zahtjeve:

- Osigurati pristup BPMKS-u (SOAP, REST komunikacija) i izuzeti iz sigurnosti resurse za koje nije potrebna autorizacija (npr. Swagger dokumentaciju, WSDL definiciju, itd.)
- Odabrati čim jednostavnije, ali učinkovite metode sigurnosti
- Koristiti jedinstvene autorizacijske podatke za pristup cijeloj BPMKS komponenti

Sigurnost unutar BPMKS-a uključuje osiguravanje SOAP i REST komunikacije. U svrhu realizacije u ovom radu REST komunikacija osigurana je implementacijom Basic Authentification metode u kombinaciji s HTTPS (Secure HyperText Transfer Protocol) protokolom. Kao dodatna sigurnosna mjera pristupa BPMKS-u služi i obavezno prilagođeno zaglavlje *bpm-context-header* koje mora postojati u svakom REST zahtjevu, a dodatno je opisano u poglavlju 5.4.1.1.

Sigurnost SOAP servisa realizirana je korištenjem XWSS (XML and Web Services Security) implementacije, a koristi se sigurnost na razini poruke (engl. Message Level Security) temeljem WSS UsernameToken profila.

BPMKS mora moći "uhvatiti" sve pogreške koje se dogode te vratiti kvalitetnu i razumljivu informaciju o vrsti i uzroku pogreške. Sukladno navedenom definirajmo zahtjeve mehanizma obrade pogrešaka:

- Potrebno je definirati centralizirano mjesto obrade pogrešaka na BPMKS-u
- Potrebno je definirati unificiranu poruku o grešci koja sadrži osnovne elemente vezane uz informaciju o nastaloj pogrešci, a to su:
  - Kratka i jasna opisna poruka pogreške
  - Detalji pogreške ako su dostupni (npr. te ispisa traga stoga[49] pogreške)
  - Vremenska oznaka kada se pogreška dogodila
- Omogućiti oporavak od greške bez smetnji za funkcioniranje sustava (ako je to izvedivo).

Obrada pogrešaka unutra BPMKS-a realizirana je prema opisanim smjernicama. Klasa *CustomResponseEntityExceptionHandler* predstavlja središnje mjesto obrada grešaka dok klasa *ExceptionResponse* predstavlja univerzalan odgovor u slučaju greške.

Kako bi naslijeđene aplikacije, koje komuniciraju s BPMKS-om, imale točne informacije o pogreškama, klasa univerzalnog odgovora u slučaju pogreške (*ExceptionResponse*) mora se kreirati u sklopu adaptera integriranog u naslijeđenu aplikaciju.

5.4 Ugradnja prilagodnika i izlaganje funkcionalnosti naslijeđenih aplikacija

Da bi naslijeđene aplikacije mogle komunicirati s BPMKS-om i izložiti svoje funkcionalnosti, kao što je opisano u poglavlju 4.1.2.2, potrebno je u njih ugraditi prilagodnik.

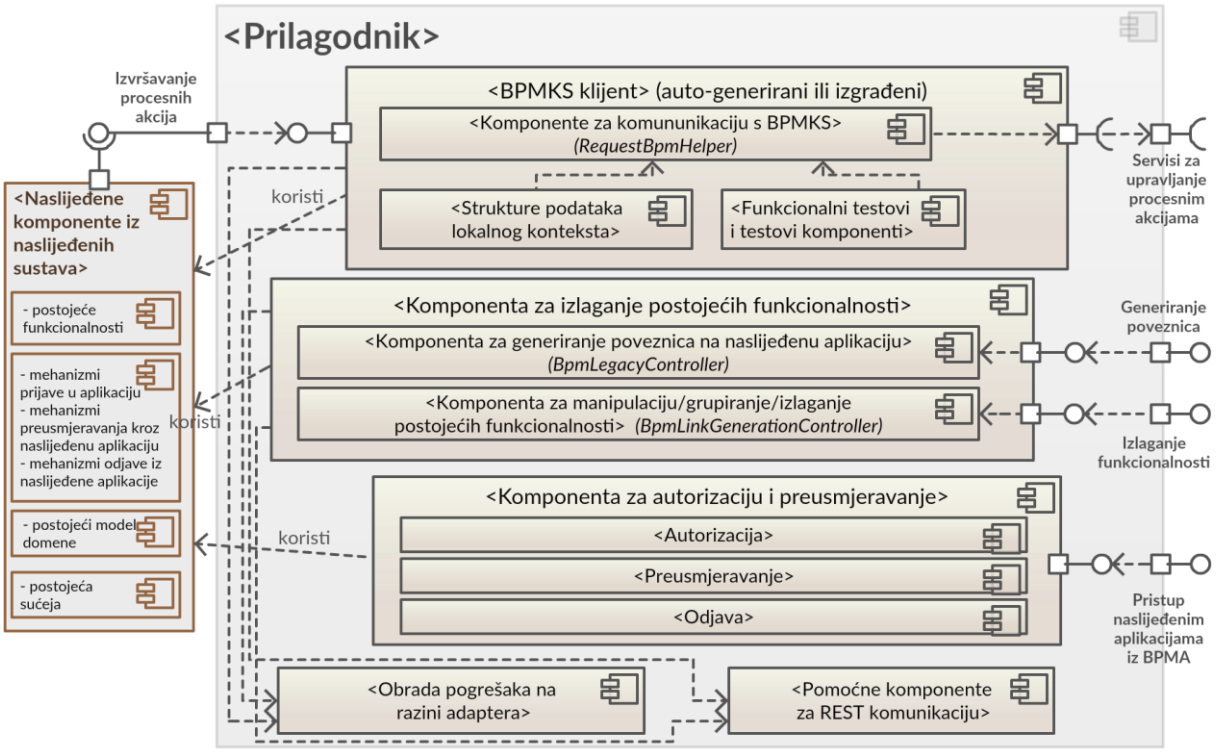
Kroz ovo poglavlje dan je detaljan opis metoda za izgradnju prilagodnika u naslijeđenim aplikacijama baziranim na SOA principima. Prikazan je dijagram komponenti prilagodnika (Slika 5.11) te je dana detaljna tehnička specifikacija svake od komponenti. Također, opisani su principi integracije prilagodnika s procesima unutar naslijeđenih aplikacija

5.4.1 Struktura i komponente prilagodnika

Svaki prilagodnik, ovisno o naslijeđenoj aplikaciji, može biti pisan u različitim jezicima, međutim način integracije te struktura prilagodnika je ista. Prema opisu danom u poglavlju

[49] **Trag stoga** (engl. **stack trace**, **stack backtrace** ili **stack traceback**) je izvješće o aktivnim okvirima stanja (engl. active stack frames) u određenoj točki vremena tijekom izvođenja programa. Okviri stanja nazivaju se još aktivacijski zapisi (engl. activation records) ili aktivacijski okviri (engl. activation frames), a to su zapravo podatkovne strukture koje sadrže informacije o slijednim pozivima potprograma i metoda tijekom rada programskog rješenja.

4.1.2.2, prikazana je detaljna struktura prilagodnika sa svim relevantnim komponentama (Slika 5.11).



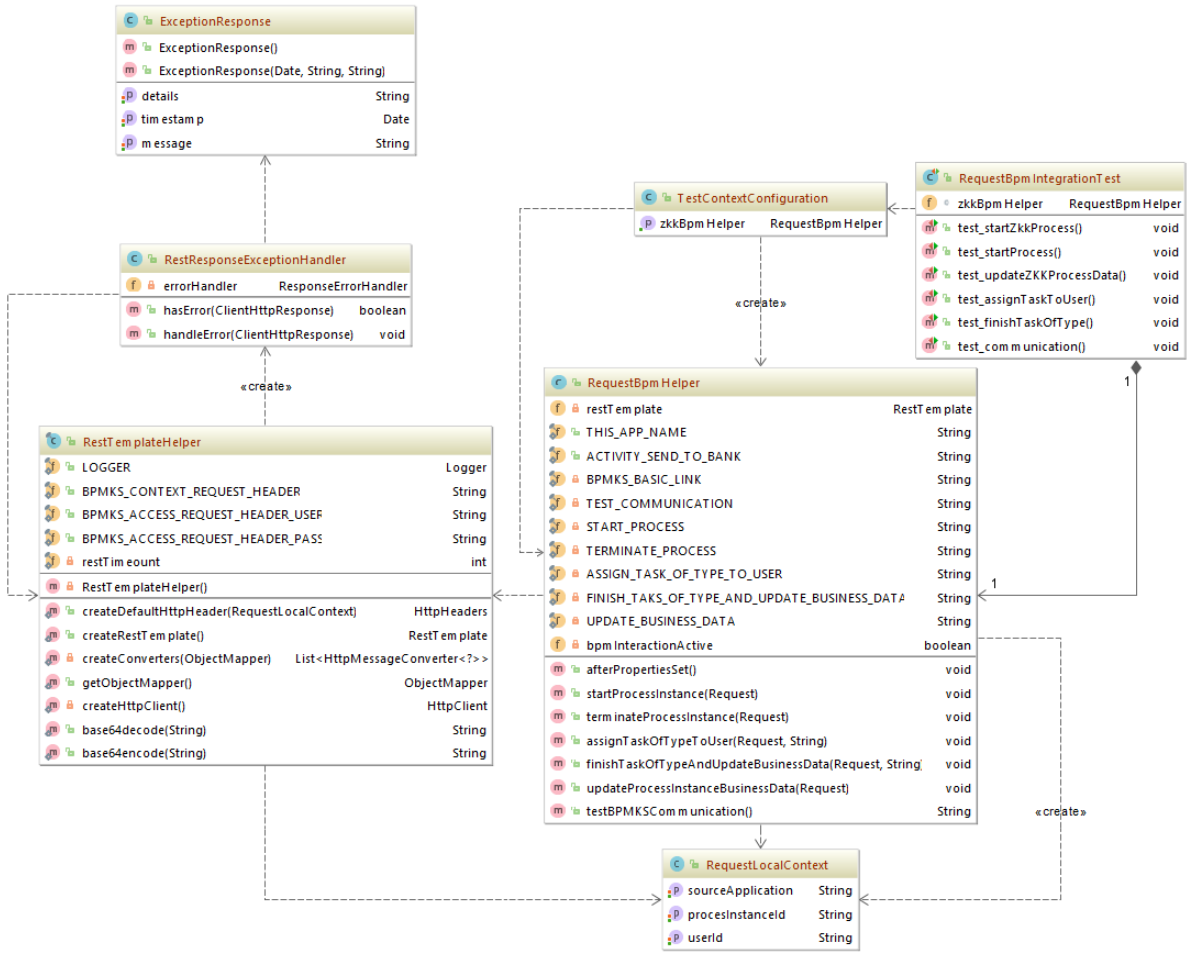
Slika 5.11 Dijagram komponenti prilagodnika

Važan čimbenik ugradnje prilagodnika je implementirati prilagodnik u zasebne, točno definirane komponente unutar naslijeđenih aplikacija, kako bi se izbjeglo dodatno kompliciranje programskog koda naslijeđenih aplikacija. U nastavku je razrađena svaka od komponenti prilagodnika uz prikaz detaljne tehničke specifikacije te su dani primjeri implementacije pojedinih komponenti prilagodnika.

5.4.1.1 BPMKS klijent

Komponenta ***BPMKS klijent*** omogućava uspostavu veze između naslijeđene aplikacije i BPMA u standardnim procesnim točkama PP-a.

U nastavku su dane preporuke za definiciju komponente ***BPMKS klijent*** s ciljem da se izgradi izdvojeni, zasebni, modul unutar naslijeđene aplikacije. Slika 5.12 prikazuje predloženi dijagram klasa komponente ***BPMKS klijent***.



Slika 5.12 Dijagram klasa BPMKS klijentske komponente

Na prikazanom dijagramu klasa (Slika 5.12) mogu se uočiti tri osnovne cjeline:

- Prvu cjelinu čine sve komponente za ostvarenje REST komunikacije (*RestTemplate*, *HttpMessageConverter*, *ObjectMapper*, itd.) sa svim potrebnim podacima (nazivima prilagođenih zaglavlja, podataka za autorizaciju, te adrese BPMKS-a), a implementirane su pomoćnom klasom *RestTemplateHelper*.
- Druga cjelina sastoji se od metoda za komunikaciju s BPMKS-om, koje su implementirane u zasebnoj klasi (*RequestBpmHelper*) uz pomoć komponenti iz *RestTemplateHelper* klase. Koristi se model domene podataka naslijeđene aplikacije, a na BPMKS-u se kreiraju identične domenske klase, sa smanjenim skupom poslovnih podataka (poglavlje 5.3.5). Za potrebe testiranja funkcionalnosti implementiranih metoda iz *RequestBpmHelper* klase kreirani su testovi (*RequestBpmIntegrationTest*) i testna konfiguracija koja omogućuje pokretanje testova (*TestContextConfiguration*).
- Treća cjelina obuhvaća klasu za obradu pogrešaka *RestResponseExceptionHandler* koja je posebno prilagođena postojećim principima obrade pogrešaka u naslijeđenoj aplikaciji te

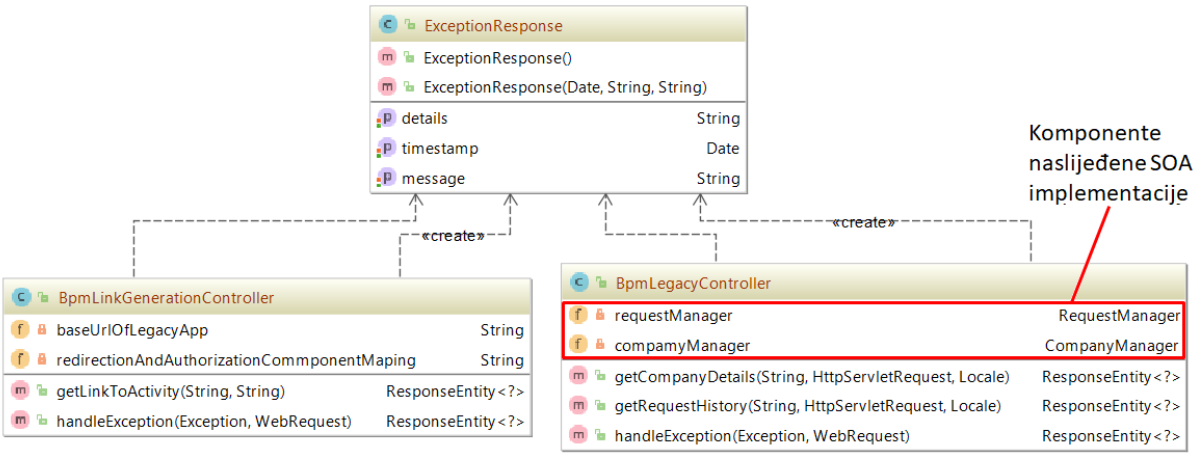
ExceptionResponse klasu koja je identična istoimenoj klasi na BPMKS-u, a sadrži detaljne informacije o pogrešci. Dodatno, ova cjelina obuhvaća i *RequestLocalContext* klasu pomoću koje se u prilagođenom HTTP zaglavlju (*bpm-context-header*) prenose podaci lokalnog konteksta. Lokalni kontekst detaljnije je objašnjen u poglavlju 5.3.6.

U slučaju da pojedine naslijeđene koriste velik dio funkcionalnosti BPMKS-a, zasebna implementacija i održavanje ***BPMKS klijent*** komponenti može biti zahtjevna i mukotrpna. U tom slučaju, komponentu ***BPMKS klijent,*** moguće je automatski generirati iz BPMKS komponente ***servisi za upravljanje procesnim akcijama***. Automatski generirana komponenta ***BPMKS klijent*** može se zatim izravno uključiti u naslijeđenu aplikaciju, a cijeli postupak detaljno je opisan u poglavlju 5.4.2. Nakon uključivanja komponente nije potrebno dodatno podešavati konfiguraciju unutar naslijeđene aplikacije jer komponenta sadrži predefinirane metode za ostvarivanja poziva. Metode BPMKS klijentske komponente se, iz programskog koda naslijeđene aplikacije, koriste kao normali lokalni poziv, dok se u pozadini odvijaju REST upiti.

5.4.1.2 Komponenta za izlaganje postojećih funkcionalnosti

Sastoji se od jednostavnih metoda koje izlažu postojeće funkcionalnosti naslijeđene aplikacije i/ili naslijeđenog SOA sloja. Funkcionalnosti naslijeđenih aplikacija izlažu se po potrebi i u željenim formatima, a nad podatcima mogu se odrađivati operacije kao što su grupiranje filtriranje itd. U kontekstu Java Spring naslijeđene aplikacije ova komponenta sastoji se od jednog ili više REST kontrolera.

Slika 5.13 prikazuje dijagram klasa ***komponente za izlaganje postojećih funkcionalnosti***, a komponenta sadrži metode za generiranje poveznica na naslijeđenu aplikaciju (*BpmLegacyController*), metode za manipulaciju/grupiranje/izlaganje postojećih funkcionalnosti (*BpmLinkGenerationController*) te klasu odgovora u slučaju pogreške (*ExceptionResponse*).



Slika 5.13 Dijagram klasa komponente za izlaganje postojećih funkcionalnosti

Klasa za izlaganje funkcionalnosti naslijeđenih aplikacija, *BpmLegacyController*, po potrebni koristi dijelove naslijeđenih aplikacija (npr. *requestManager* i *companyManger* - Slika 5.13), da bi izložila određene podatke i funkcionalnosti za korištenje s BPMA. Primjerice metoda *getCompanyDetails()* izlaže detaljne podatke o određenoj organizaciji temeljem implementacije funkcionalnosti u postojećoj aplikaciji (naslijeđena komponenta *companyManger*).

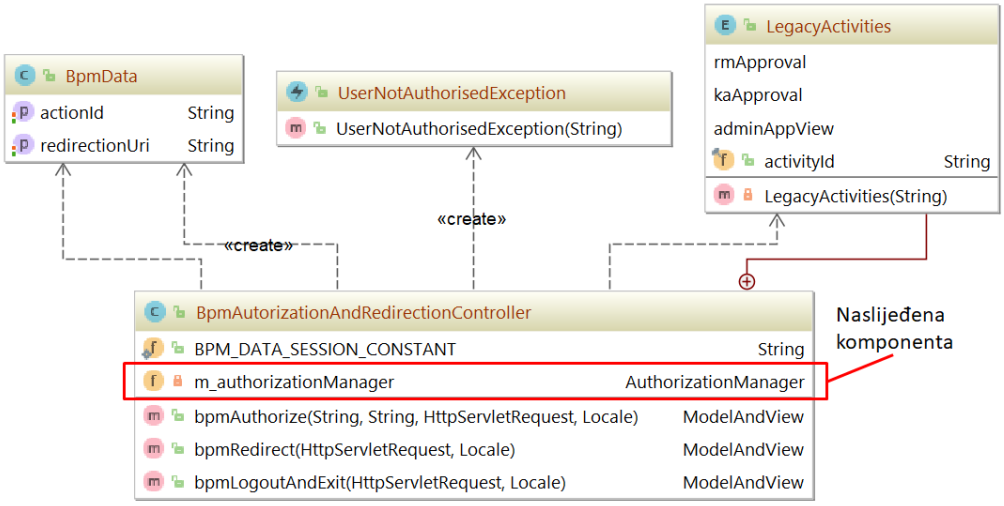
Klasa *BpmLinkGenerationController* generira poveznice za pristup do aktivnosti u naslijeđenoj aplikaciji, a koristi se od strane BPMA u pozadinskim procesnim točkama. Kreiranje poveznica najjednostavnije je odraditi u naslijeđenim aplikacijama, koje su ujedno i cilj tih poveznica, zbog dostupnosti potrebnih podataka. Naslijeđena aplikacija svjesna je svoje lokacije (imena poslužitelja, pristupne točke i *context root* oznake na kojoj se nalazi), razumije kontekst i barata značenjem konstanti i pojmova koji se mogu koristiti u poveznici, a dostupni su i podaci komponente za autorizaciju i preusmjeravanje koja se također nalazi u prilagodniku naslijeđene aplikacije.

Dodatno, korištenje metoda za generiranje poveznica u naslijeđenim aplikacijama sprečava da se podaci specifični za kreiranje poveznica nalaze izvan naslijeđenih aplikacija i time dodatno kompliciraju podatkovni model BPMKS-a ili BPMA. Struktura generiranih poveznica, te način obrade poveznica od strane komponente za autorizaciju i preusmjeravanje, detaljno su opisane u sljedećem poglavlju.

5.4.1.3 Autorizacija i preusmjeravanje

Kao što je već kratko opisano u poglavlju 4.1.2.2 ova komponenta autorizira korisnika kada dolazi s BPMA, pripremi potrebne poslovne podatke koristeći dostupne SOA funkcionalnosti naslijeđene aplikacije, te preusmjeri korisnika na ulaznu interakcijsku točku gdje se odvija

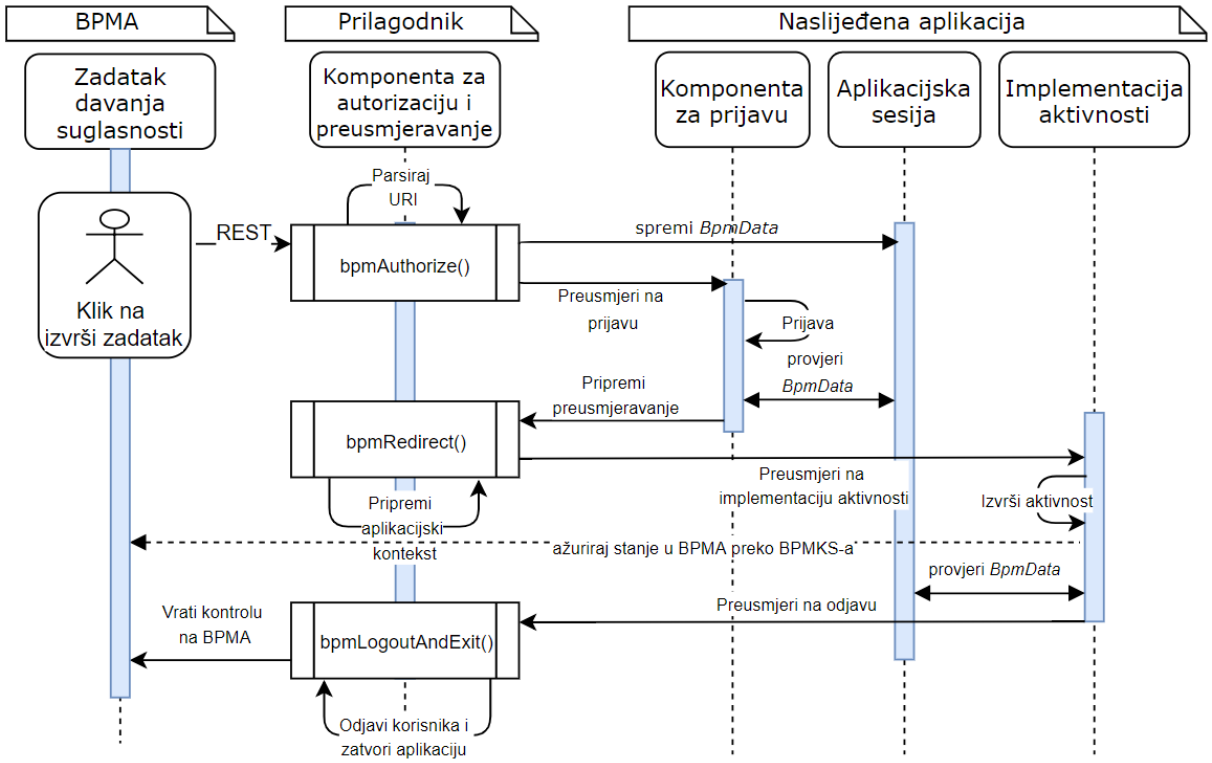
izvršavanje zadatka. ***Komponenta za autorizaciju i preusmjeravanje*** u naslijeđenoj aplikaciji sprema informaciju da trenutni korisnik dolazi s BPM-a. Tako naslijeđena aplikacija zna da u standardnim procesnim točkama treba uspostaviti komunikaciju prema BPMA. Isto tako zna da se u izlaznim interakcijskim točkama kontrola predaje natrag BPM okruženju. Slika 5.14 prikazuje dijagram klasa komponente za autorizaciju i preusmjeravanje.



Slika 5.14 Dijagram klasa komponente za autorizaciju i preusmjeravanje

Komponenta sadrži pobrojani tip *LegacyActivities* koji čuva listu svih aktivnosti dostupnih u određenoj naslijeđenoj aplikaciji te *BpmData* klasu koja sadrži potrebne podatke temeljem kojih će se raditi preusmjeravanje i priprema podataka za izvršavanje korisničkih aktivnosti u naslijeđenoj aplikaciji. Osnovna klasa ***komponente za autorizaciju i preusmjeravanje*** je *BpmAuthorizationAndRedirectionControler*, a sastoji se od tri metode, *bpmAuthorize* i *bpmRedirect* koje su karakteristične za ulazne interakcijske točke te *bpmLogoutAndExit* koja se koristi kod izlaznih interakcijskih točaka.

Pretpostavimo da u BPMA službenik klikne na poveznicu na aktivnost davanja suglasnosti te biva preusmjeren do naslijeđene aplikacije. Slika 5.15 ilustrira dijagram toka komunikacije koji zatim odrađuje ***komponenta za autorizaciju i preusmjeravanje*** unutar prilagodnika naslijeđene aplikacije.



Slika 5.15 Dijagram toka komunikacije komponente za autorizaciju i preusmjeravanje

Cijeli tok komunikacije, te uloga pojedinih metoda *BpmAuthorizationAndRedirectionControler* klase detaljnije je opisan u nastavku.

Metoda *bpmAuthorize* temeljem definiranog mapiranja (`@RequestMapping(value = "/secure/bpmAuthorizeAndRedirect.html", method = RequestMethod.GET)`) "hvata" i obrađuje zahtjev korisnika koji je u BPMA kliknuo na poveznicu za pristup do aktivnosti u naslijeđenoj aplikaciji. Poveznice za preusmjeravanje na funkcionalnosti naslijeđene aplikacije predefinirane su strukture i u osnovi se sastoje od tri glavna dijela.

- Prvi dio čini osnovna adresa naslijeđene aplikacije s imenom poslužitelja (***host:port***), *context root* oznakom na kojoj se aplikacija nalazi (***contextRoot***) te mapiranjem na novokreiranu komponentu za autorizaciju i preusmjeravanje (***bpmAuthorizeAndRedirect.html***): *https://**host:port/contextRoot/bpmAuthorizeAndRedirect.html***
- Drugi dio poveznice čini identifikator (***activityId***) koji govori o kojoj vrsti pristupa, te o kojoj aktivnosti se radi, npr.: *?**activityId**=rmApproval*. Temeljem identifikatora naslijeđena aplikacija može ispravno inicijalizirati aplikacijski kontekst kako bi korisnik aktivnost obavio.

- Treći dio sadrži dodatne parametre potrebne za preusmjeravanje nakon autorizacije, a sastoji se od kodirane URL strukture temeljem koje se radi preusmjeravanje unutar naslijeđene aplikacije: *&**redirectionUri**=redirect:%2encoded%2legacy%2uri*. Budući da ovaj dio ima oblik URL-a, kako se ne bi narušio cjelokupni format poveznice za preusmjeravanje, potrebno je obaviti kodiranje ovog dijela URL-a. Kodiranje se može odraditi korištenjem *URLEncoder* pomoćne klase, `URLEncoder.encode(redirectioUriUrlPart, "UTF-8")`.

Metoda *bpmAuthorize* čita parametre iz zaprimljene poveznice, instancira *BpmData* objekt kojeg puni pročitanim parametrima, te taj objekt sprema u naslijeđenoj aplikaciji, npr. stavlja ga u korisničku sesiju naslijeđene aplikacije (engl. user session). objekt *BpmData* govori naslijeđenoj aplikaciji da trenutni korisnik dolazi s BPMA, te sadrži potrebne podatke temeljem kojih će se raditi preusmjeravanje i priprema podataka za izvršavanje korisničke aktivnosti.

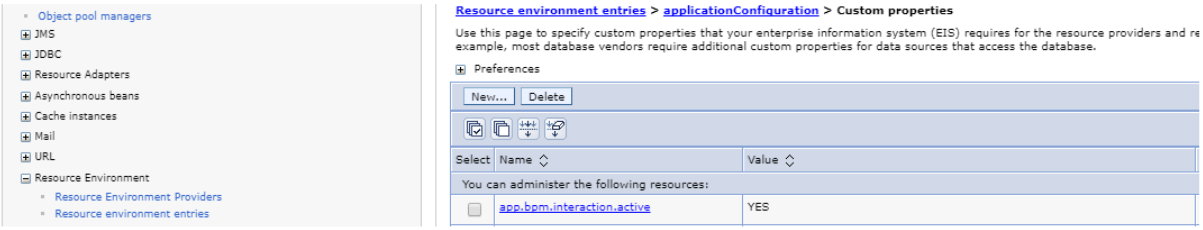
Metoda *bpmAuthorize* preusmjerava korisnika na postojeću funkcionalnost prijave u aplikaciju. Korisnik unosi pristupne podatke i odrađuje proces prijave. Budući da korisnik dolazi s BPMA (postoji *BpmData* objekt), postojeća komponenta za autorizaciju ne preusmjerava korisnika na početni ekran u naslijeđenoj aplikaciji, već poziva metodu *doRedirect* na komponenti za autorizaciju i preusmjeravanje.

Metoda *doRedirect* temeljem identifikatora aktivnosti (*BpmData.activityId*) postavi specifični aplikacijski kontekst potreban za izvršavanje aktivnosti, npr. odgovarajuću korisničku rolu. Slika 5.14 prikazuje primjer korištenja naslijeđene *m_authoritazionManager* komponente za popunjavanje specifičnog aplikacijskog konteksta potrebnog naslijeđenoj aplikaciji. U slučaju da prijavljena osoba u naslijeđenoj aplikaciji nema dodijeljenu ulogu koja može odraditi potrebnu aktivnost, baca se pogreška *UsetNotAuthorisedException*. Nakon prijave i postavljanja aplikacijskog konteksta, koristeći *BpmData.redirectionUri* podatak, napravi se preusmjeravanje na točan ekran unutar naslijeđene aplikacije gdje se aktivnost može izvršiti. Metoda *bpmLogoutAndExit* poziva se nakon što je aktivnost obavljena, a ima ulogu "čišćenja" podataka i odjave korisnika iz naslijeđene aplikacije.

5.4.1.4 Odvajanje BMA i naslijeđenih aplikacija

Kao što je već spomenuto u poglavlju 4.1.2.2, naslijeđene aplikacije mogu se odvojiti od BPMA čime se ostvaruje ***prilagodljivost*** vlastitog modela. Tako se isključuje interakcija s BPMA, a kompletan PP i dalje se obavlja u postojećim sustavima.

Na razini naslijeđene aplikacije definira se identifikator koji pokazuje je li interakcija s BPMA omogućena ili ne. Identifikator može biti konfiguracijski parametar naslijeđene aplikacije, a preporučeni način bio bi definirati ga kao parametar na poslužitelju koji se čita od strane jedne ili više naslijeđenih aplikacija. Tako omogućavamo promjenu parametra bez promjene koda aplikacije. Primjerice, u slučaju Java aplikacije i aplikacijskog poslužitelja kao što su Apache Tomcat, IBM WebSphere Application Server (IBM WAS) i slični, identifikator bi se definirao kao tzv. JNDI[50] resurs (Slika 5.16).



Slika 5.16 Identifikator BPMA interakcije kao JNDI parametar IBM WAS-a

U standardnim procesnim točkama kontrolira se vrijednost identifikatora. U slučaju da je identifikator pozitivan uspostavlja se interakcija s BPMA, dok se u slučaju negativne vrijednosti metode BPMKS klijenta ne pozivaju.

Alternativa bi bila dozvoliti interakciju za samo određene korisničke uloge naslijeđenih aplikacija. Unutar proizvoljne komponente prilagodnika, konfiguracijskim postavkama naslijeđene aplikacije ili u postavkama poslužitelja kao što je pokazano u prethodnom odlomku definira se lista korisničkih uloga koje mogu obavljati interakciju s BPMA. U slučaju da korisnik nema ulogu koja je ovlaštena za interakciju s BPMA, naslijeđena aplikacija ne koristi metode BPMKS klijenta.

Logiku odvajanja naslijeđene aplikacije od BPM-a, putem korisničkih uloga ili putem identifikatora na razini naslijeđene aplikacije, potrebno je složiti na jednom mjestu unutar prilagodnika. Idealno mjesto je komponenta s metodama za komunikaciju prema BPMKS-u, dakle *RequestBpmHelper* klasa iz navedenog primjera u poglavlju 5.4.1.1. U sklopu instanciranja *RequestBpmHelper beana* (prilikom podizanja naslijeđene aplikacije) učitava se identifikator ili lista korisničkih uloga, te se temeljem tih podataka zabranjuju ili dozvoljavaju pozivi prema BPMKS-u.

[50] **Java Naming and Directory Interface** API ili **JNDI** API omogućava aplikacijama pristup do programskih objekta koji se nalaze na drugim sustavima, kao što su poslužitelji ili baze podataka, temeljem naziva tih objekata.

5.4.2 Generiranje komponente *BPMKS klijent* za ugradnju u prilagodnik naslijeđenih aplikacija

U svrhu realizacije karakteristike ***prilagodljivosti*** i ***univerzalnosti*** vlastitog modela integracije (poglavlje 3.3.1) predložen je postupak automatskog generiranja komponente prilagodnika ***BPMKS klijent*** koji služi za pozivanje servisa komponente ***servisi za upravljanje procesnim akcijama*** na BPMKS-u. Generirani klijent ugrađuje se u prilagodnike naslijeđenih sustava. Kako bi podržali integraciju BPMKS-a s različitim naslijeđenim aplikacijama, koje mogu biti izgrađene u različitim tehnologijama i pisane u različitim programskim jezicima, potrebno je omogućiti izgradnju klijenta u proizvoljnim programskim jezicima.

Prednosti automatskog generiranja programskog koda klijenta su:

- Nije potrebno ručno pisati implementaciju REST klijenta.
- Osigurava se konzistentnost klijentskog koda klijenta s *endpontima*. U slučaju generiranja nove verzije klijenta te integracije klijenta u naslijeđenu aplikaciju, razvojni IDE prepoznaje promijene te javlja pogreške prilikom automatske provjere programskog koda.
- Olakšava korištenje klijenta tako da prikazuje moguće funkcije, nadopunu programskog koda (engl. code completion) te dokumentaciju metoda.
- Postoji mogućnost generiranja klijenta u različitim programskim jezicima te integraciju u različite naslijeđene aplikacije.

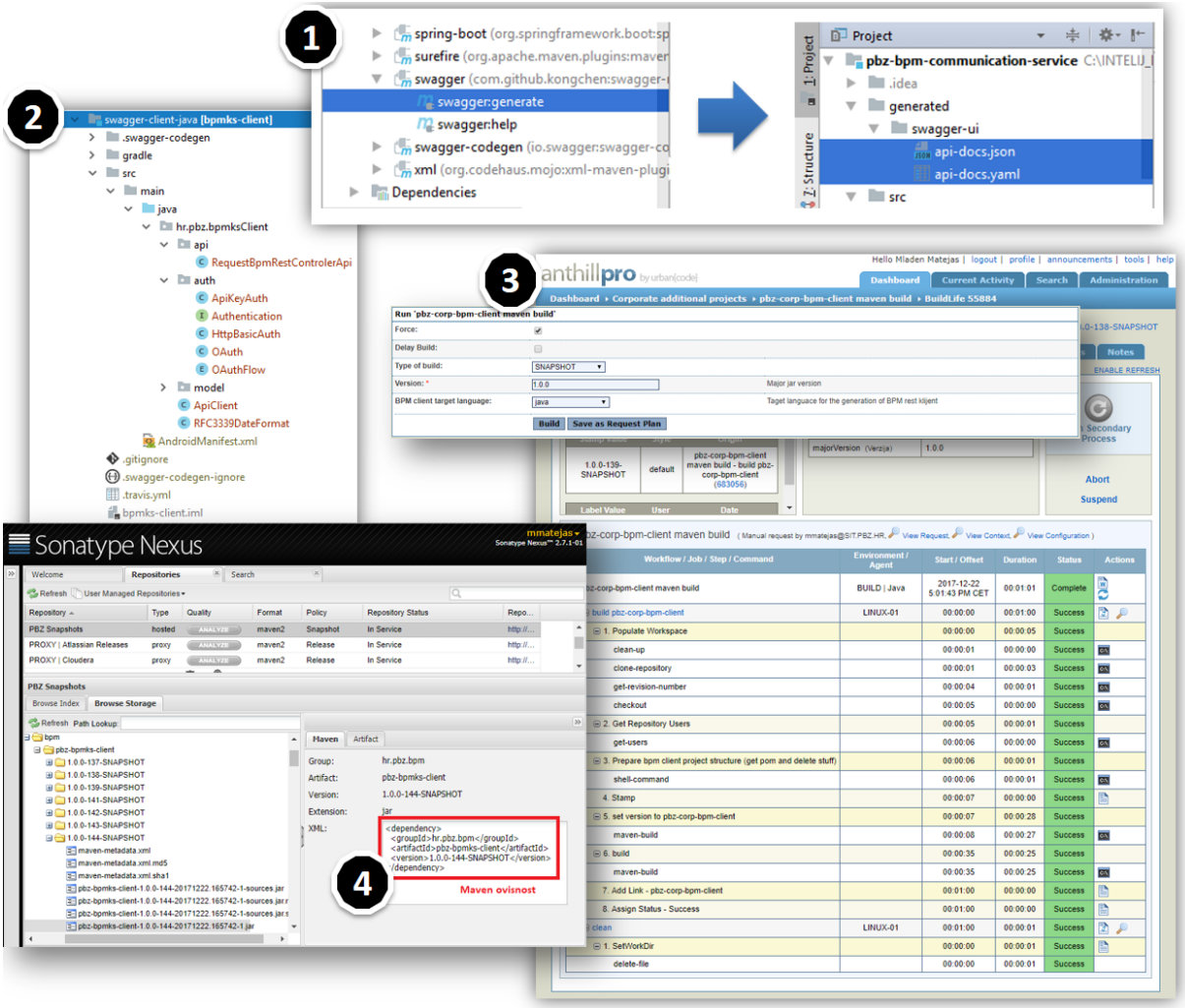
U svrhu automatskog generiranja klijenta predložen je sljedeći općeniti postupak:

1. *Generirati programsku dokumentaciju servisa komponente* ***servisi za upravljanje procesnim akcijama***
2. *Odabrati programski alat koji će omogućavati automatsko kreiranje klijenta temeljem generirane dokumentacije, u proizvoljnom programskom jeziku, te kreirati klase, programski kod, strukturu i komponente klijenta u programskom jeziku po potrebi*
3. *Kreiranog klijenta „zapakirati“ u artefakt, te ga napraviti dostupnim za naslijeđene aplikacije*
4. *Artefakt uključiti u prilagodnike naslijeđenih aplikacija.*

5.4.2.1 Tehnička specifikacija postupka automatskog generiranja

Pogledajmo primjer provedbe postupka automatskog generiranja komponente ***BPMKS klijent*** navedenog u prethodnom poglavlju. Prema tehničkoj specifikaciji (poglavlje 5.3.2) komponenta ***servisi za upravljanje procesnim akcijama*** sadrži REST *endpointe*, te je za nju

potrebno generirati REST klijenta, koji se koristi u prilagodnicima naslijeđenih aplikacija. Slika 5.17 prikazuje rezultate općenitog postupka generiranja komponente ***BPMKS klijent***, a prikazani koraci su u detalje objašnjeni u nastavku.

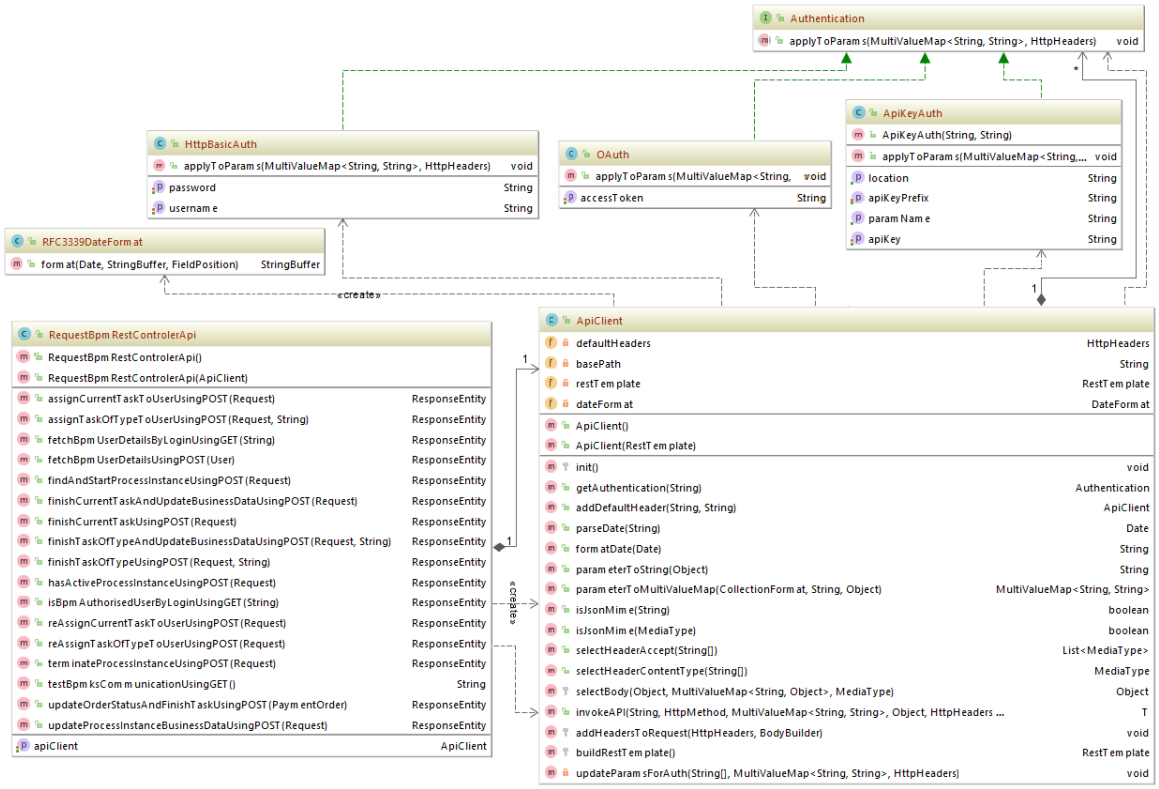


Slika 5.17 Postupak automatskog generiranja komponente BPMKS klijent

1. **Generiranje OpenAPI dokumentacije REST** *endpointa* komponente ***servisi za upravljanje procesnim akcijama***. Korišten je alat Swagger koji je uključen u BPMKS (poglavlje 5.1.2) i koji automatski izlaže OpenAPI dokumentaciju. Druga opcija je koristiti dodatak Maven (*swagger-maven-plugin*) za ručno kreiranje dokumentacije. Rezultat oba postupka je OpenAPI dokumentacija u obliku *api-docs.json* datoteke koja sadrži sve potrebne podatke za generiranje REST klijenta.

2. **Odabir alata i automatsko generiranje REST klijenta iz OpenAPI specifikacije**. Izabran je alat Swagger Codegen koji može generirati REST klijenta ili poslužiteljski *endpoint* iz OpenAPI specifikacije na više od 40 programskih jezika. Generiranje se može odraditi korištenjem aplikacije Swagger Editor (https://swagger.io/swagger-editor/),

pomoću Maven dodatka Swagger Codegen (*swagger-codegen-maven-plugin*) ili koristeći alat za kontinuiranu integraciju programskih rješenja (AnthillPro, Jenkins, i slični) i CLI[51] izvršne JAR verzije alata Swagger Codegen. Naprednom konfiguracijom alata Swagger Codegen moguće je uključiti dodatne klase u generirani model (primjerice klasu lokalnog konteksta) ili zamijeniti određenu klasu modela domene s klasom po želji. Također, određena specifična svojstva (primjerice prilagođena zaglavlja) potrebno je nakon generiranja prilagoditi u automatski kreiranom BPMKS klijentu. Slika 5.18 prikazuje UML dijagram klasa generiranog klijenta u programskoj jeziku Java.



Slika 5.18 Pojednostavljeni dijagram klasa REST API klijenta generiranog iz BPMKS endpointova

3. **Automatska izgradnja artefakta REST klijenta uz spremanje artefakta na centralni repozitorij organizacije.** U svrhu izgradnje artefakta korišten je sustavu AnthillPro gdje je napravljen ***pbz-corp-bpm-client maven build*** proces. Po definiciji parametra pokreće se *build* proces koji sa središnjeg repozitorija programskog koda organizacije uzima programski kod generiranog REST klijenta iz prethodnog koraka te pokreće izgradnju artefakta (Slika 5.17, 3. korak). Po završetku AnthillPro build procesa kreirani artefakt

[51] Sučelje naredbenog retka (engl. Command Line Interface) ili tumač komandnog jezika je konzolno korisničko sučelje koje pruža sredstva za interakciju s računalnim programom gdje korisnik izdaje naredbe programu u obliku uzastopnih redaka teksta (komandne linije).

kopira se na središnji repozitorij organizacije, a u prikazanom slučaju koristi se repozitorij Nexus kompanije Sonatype.

4. **Dostupnost generiranog artefakta.** Artefakt u repozitoriju Nexus dostupan je za korištenje naslijeđenim aplikacijama. U slučaju da naslijeđena aplikacija nema vezu prema centralnom repozitoriju organizacije, artefakt REST klijenta može se ručno s repozitorija prebaciti u željenu naslijeđenu aplikaciju.

Prilikom generiranja klijenta treba paziti na kompatibilnost s naslijeđenim aplikacijama u vidu korištenih klasa i biblioteka. Potencijalno može doći do konflikata između klasa generiranog klijenta i klasa u naslijeđenim aplikacijama (npr. različite verzije istih klasa). Jedno od rješenja u ovakvim slučajevima može biti korištenje druge biblioteke za generiranje REST klijenta koja ne sadrži konfliktne klase ili izjednačiti korištene verzije biblioteka u naslijeđenoj aplikaciji i generiranom klijentu.

Automatsko generiranje klijenata preporučljivo je u slučajevima kada određena naslijeđena aplikacija poziva veći broj BPMKS servisa koji su podložni čestim promjenama, te u slučaju implementacije cijelog niza procesa koje naslijeđena aplikacija podržava.

S druge strane, ako postoji nekoliko BPMKS servisa koji se pozivaju iz naslijeđene aplikacije te na kojima nisu očekivane modifikacije, preporučeni način implementacije bio bi izgraditi pozive korištenjem sredstava već dostupnih u naslijeđenim aplikacijama.

5.4.3 Principi izgradnje i postupak integracije prilagodnika

Struktura i ugradnja prilagodnika u naslijeđene aplikacije zavise o poslovnom procesu koji se tim aplikacijama obavlja.

Ako PP uključuje više naslijeđenih aplikacija, od kojih svaka ima određenu ulogu u procesu, prilagodnik je potrebno izgraditi u svakoj od navedenih aplikacija.

Međutim, struktura prilagodnika uvelike ovisi o ulozi koju određena naslijeđena aplikacija ima u poslovnom procesu, to jest o aktivnostima PP-a koje naslijeđena aplikacija obavlja. Iskustvo implementacije prilagodnika u naslijeđene aplikacije pokazalo je da nije uvijek potrebna implementacija svih definiranih komponenti prilagodnika kao što je navedeno u poglavlju 5.4.1. Primjerice, ako neka naslijeđena aplikacija ima zadatak samo pokretati instance procesa i/ili aktivnosti na BPMA, može se implementirati samo ***BPMKS klijent*** za izvršavanje procesnih akcija. U slučaju da je samo potrebno koristiti određene funkcionalnosti naslijeđene

SOA implementacije, može se izgraditi samo ***komponenta za izlaganje postojećih funkcionalnosti***.

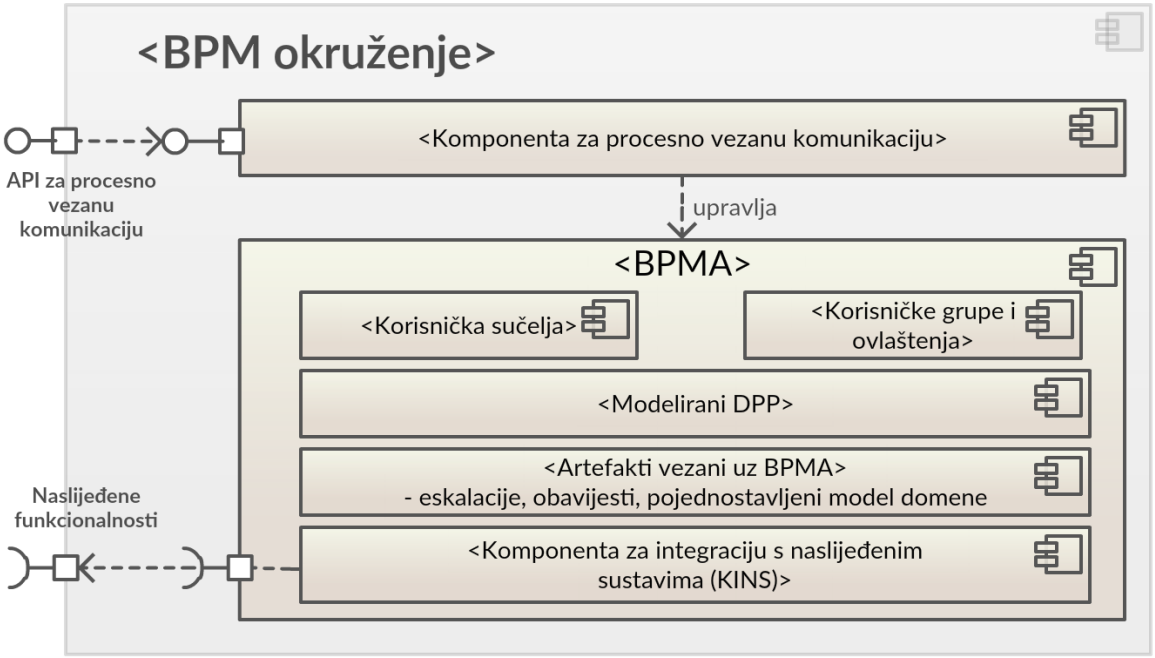
Kako bi znali koliko prilagodnika treba ugraditi i u koje naslijeđene aplikacije te koja je struktura pojedinog prilagodnika, potrebno je identificirati naslijeđene PP-e, razraditi detalje tih procesa te odrediti koje aktivnosti se obavljaju kojim naslijeđenim aplikacijama.

Kako bi se identificiralo navedeno, predložen je sljedeći postupak:

- *Odabrati ciljani postojeći PP*
- *Otkriti sve aktivnosti PP-a*
- *Identificirati sve naslijeđene aplikacije koje sudjeluju u izvršavanju PP-a*
- *Za svaku naslijeđenu aplikaciju:*
    - *Osigurati povezanost identificirane aplikacije s BPM okruženjem ugradnjom prilagodnika (podržati samo potrebne funkcionalnosti prema smjernicama iz uvoda ovog poglavlja)*
    - *Odrediti koje aktivnosti se određuju u dotičnoj naslijeđenoj aplikaciji, te za svaku aktivnost*
        - Ako *aktivnost sadrži specifične akcije, koje još nisu podržane od strane BPMKS-a (BPMKS komponenta* ***servisi za upravljanje procesnim akcijama****), potrebno je podršku za te akcije implementirati u BPMKS-u, te sukladno napraviti podršku u prilagodniku.*
        - *Identificirati procesne i interakcijske točke*
        - *Odrediti tip procesne točke i sukladno osigurati komunikaciju s BPMA preko prilagodnika*
        - *Odrediti tip interakcijske točke i sukladno osigurati kreiranje poveznice na ulaznu interakcijsku točku u prilagodniku naslijeđene aplikacije, te osigurati zatvaranje naslijeđene aplikacije u izlaznim interakcijskim točkama*
- *Izgraditi BPMA i integrirati BPMA s prilagodnicima naslijeđenih aplikacija, preko BPMKS-a, temeljem postupka izloženog u poglavlju 5.5.2.3.*

5.5 Izgradnja BPMA, MPP, izgradnja korisničkih te programabilnih sučelja

Ovo poglavlje prikazuje tehničke aspekte izgradnje BPMA, specifikaciju pojedinih komponenti te razradu metoda integracije s postojećim naslijeđenim sustavima preko BPMKS-a.



Slika 5.19 Dijagram komponenti BPMA u BPM okruženju

5.5.1 API za procesno vezanu komunikaciju

Predstavlja sučelje ***komponente za procesno vezanu komunikaciju*** koja je dio korištenog BPMS-a, te ovisi o implementaciji od strane proizvođača BPMS-a.

Sučelja su najčešće implementirana kao niz REST *endpointa* koja omogućavaju manipulaciju i upravljanje procesnim artefaktima na BPMA. U procesne artefakte BPMA spadaju:

- Instance procesa (engl. Process Instances)
- Definicije PP-a (engl. Business Process Definition)
- Zadaci (engl. Task)
- Servisi i aktivnosti (engl. Services and Activities)
- Korisnici i grupe (engl. Users ans Groups)

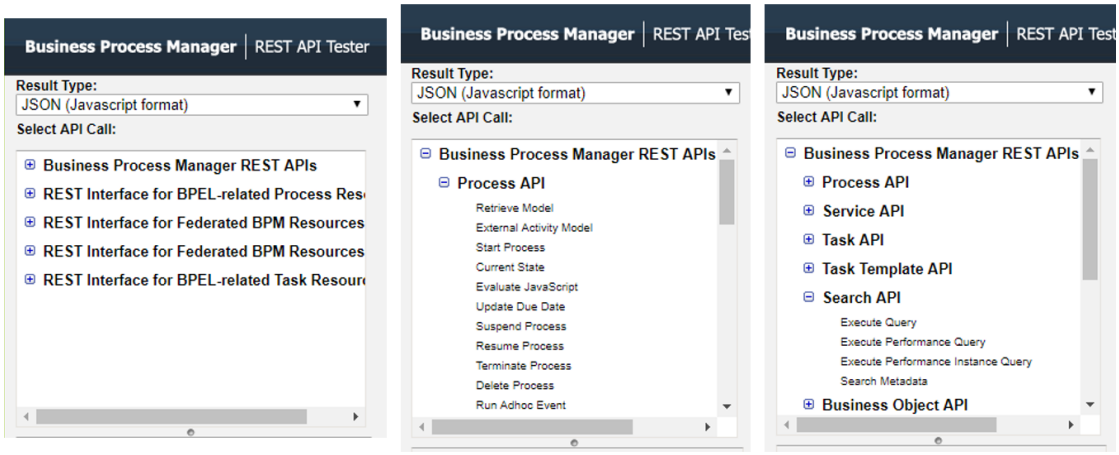
Budući da u ovom radu za izgradnju BPMA koristimo IBPM BPMS, demonstrirano je korištenje ***API-ja za procesno vezanu komunikaciju*** tog skupa alata koji se naziva IBPM REST API.

Kako bi pristupili nekom REST resursu IBPM REST API-ja potreban je URL[52] za pristup koji se sastoji se od predodređenih dijelova sa zajedničkim formatom [59]:

*http://**{host}**:**{port}**/**{context-root}**/v1/**{resource}**?**{query}***

Važniji dio URL-a čini ***{resource}*** koji predstavlja oznaku vrste resursa kojem pristupamo (npr. proces, aktivnost, itd.), te ***{query}*** koji definira skup popratnih parametra za definiciju dodatnih uvjeta upita (npr. parametri za pretragu).

Za pozivanje navedenih IBPM REST API-a može se koristiti *SoapUI* REST klijent (poglavlje 5.1.2), međutim kao dio IBPM skupa alata dolazi i internetska aplikacija *REST API Tester* koji sadrži kategoriziran popis svih REST *endpointa* dostupnih u IBPM REST API-ju (Slika 5.20). U ovom radu primarno se koristite *endpointi* kategorije "*Business Process Manager REST APIs*".

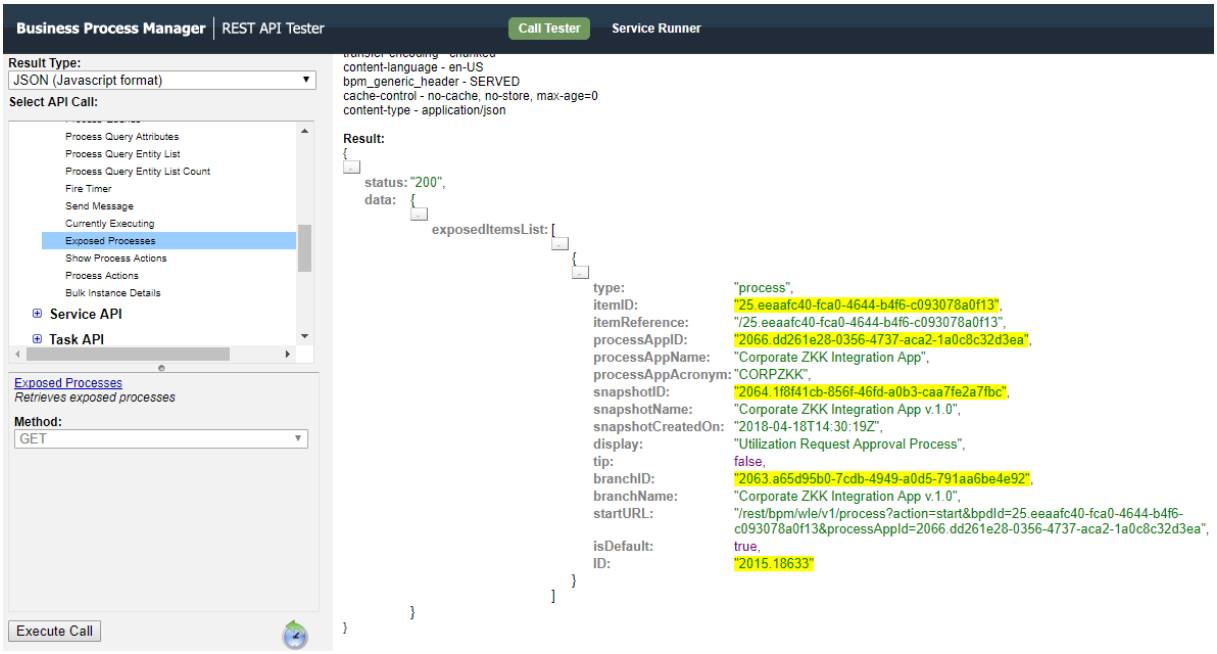


Slika 5.20 Kategorije REST zahtjeva IBPM REST API alata

Odabirom pojedinog servisa iz određene kategorije dobijemo grafički prikaz detalja REST servisa, kompletan URL, unosna polja za parametre, te mogućnost izvršavanja servisa. Odgovor servisa se, ovisno o želji, može prikazati u JSON ili XML obliku. Slika 5.21 ilustrira primjer poziva IBPM REST API servisa za dohvat izloženih procesa, a odgovor je prikazan u JSON formatu. Vidljivo je da odgovor sadrži jedinstvene identifikatore procesa (*itemID* i *snapshotId*) te čak i gotov upit kojim možemo pokrenuti instancu procesa (*startURL*). Upravo iz tog razloga IBPM REST API Tester koristan je u procesu razvoja BPM rješenja prema predloženom modelu integracije.

[52] **Ujednačeni ili usklađeni lokator sadržaja** (engl. **Uniform Resource Locator - URL**). predstavlja putanju do određenog sadržaja na Internetu te se obično naziva poveznica, a ponekad i mrežna adresa.

Kod komunikacije s IBPM REST API sučeljem, kao opciju autorizacije možemo koristi Lightweight Third-Party Authentication (LTPA) protokol ili HTTP Basic Authentification metodu. Pristupni podaci određuju razinu pristupa, to jest skup artefakata kojima korisnik može manipulirati koristeći metode IBPM REST API sučelja.



Slika 5.21 Primjer poziva REST servisa iz alata REST API Tester

Važno je napomenuti da se prilikom kreiranja GET REST zahtjeva prema IBPM REST API-ju može desiti da veličina URL-a zahtjeva prelazi praktično ograničenje dopuštenog broja znakova. U tom slučaju koristi se metoda tuneliranja naziva HTTP metoda unutar HTTP zaglavlja, tzv. *Verb tunneling* [59]. Zahtjev šaljemo kao POST, parametre zahtjeva stavljamo u tijelo zahtjeva, te postavljamo *x-http-method-override* HTTP zaglavlje u kojem predajemo stvarni naziv metode - GET.

5.5.2 Glavni dijelovi BPMA

BPMA je aplikacija koja podržava određeni PP, te je izgrađena uz pomoć alata za MPP korištenog BPMS-a (npr. u slučaju IBPM-a, to je alat Process Designer). Ostvaruje karakteristiku ***centralizacije*** predloženog modela (poglavlje 3.3.1). U nastavku su opisani osnovni dijelovi BPMA.

5.5.2.1 Definicija modela domene BPMA

Poslovni objekti (engl. Business Object) definirani u sklopu BPMA (BPMAPO) čine model domene, to jest podatkovni model BPMA (BPMAPM). BPMAPO-i se realiziraju programskim

klasama, te sadrže podatke o procesu. BPMAPM ovisi o PP-u modeliranom u BPMA, a primjer kreiranog BPMAPM-a možemo vidjeti u poglavlju 6.3.4 (Slika 6.6).

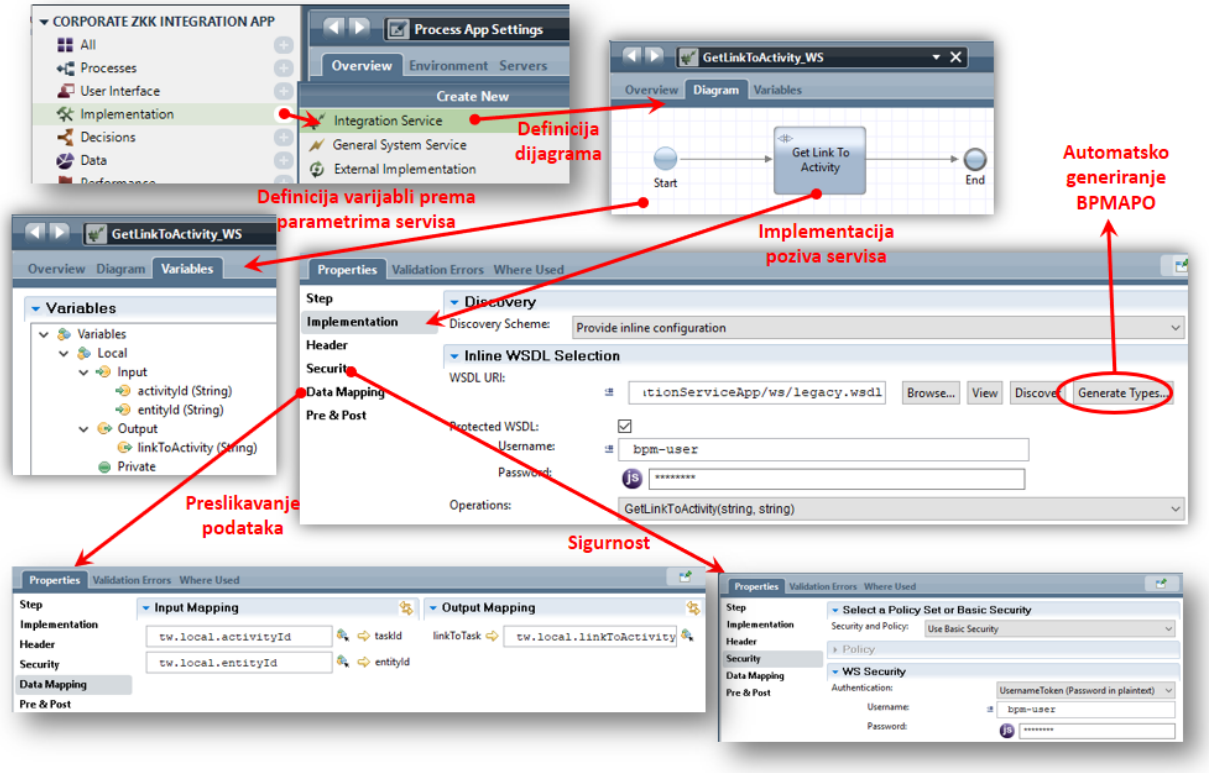
5.5.2.2 *Komponenta za integraciju s naslijeđenim sustavima*

BPMA ***KINS*** poziva BPMKS komponentu ***servisi za rad s postojećim funkcionalnostima*** (poglavlje 5.3.4). Karakteristična je za komunikaciju u pozadinskim procesnim točkama, a sadrži implementaciju servisa mehanizmima odabranog BPMS-a.

Osnovni dio ***KINS*** je servis za dohvat poveznica na aktivnosti koje se obavljaju u naslijeđenim aplikacijama, a osim toga može još sadržavati pozive metoda kojima dohvaća podatke potrebne za prikaz na BPMA. Slika 5.7 prikazuje primjer *endpointova* servisa koje poziva ***KINS***, gdje *endpoint getLinkToActivity* vrši dohvat poveznica na aktivnosti, dok *endpointovi getContactDetails* i *getRequestDetails* služe za dohvat ostalih potrebnih podataka.

Implementacija poziva i realizacija ***KINS*** ovisi o realizaciji u pojedinom BPMS-u, a ovdje ćemo prikazati principe implementacije temeljem IBPM skupa alata.

Zbog razloga opisanih u poglavlju 5.2 ***KINS*** je realizirana kao SOAP klijent koji poziva SOAP *endpointe* BPMKS komponente ***servisi za rad s postojećim funkcionalnostima***.



Slika 5.22 Ilustracija poziva SOAP servisa u BPMA

Implementacija poziva servisa svodi se na kreiranje nove implementacije integracijskog servisa u BPMA, definiranje poveznice do WSDL datoteke SOAP servisa, podešavanja niza parametara, mapiranja podataka na ulazne/izlazne argumente te definiciju sigurnosnih postavki. Slika 5.22 prikazuje osnovne korake implementacije SOAP klijenta ***KINS*** na primjeru metode za dohvat poveznica na aktivnosti u naslijeđenim aktivnostima.

Od verzije 8.5.7 standardnog paketa IBPM alata, na gotovo identičan način moguće je implementirati poziv vanjskih REST servisa. U tom slučaju, umjesto WSDL-a, u BPMA se definira poveznica do OpenAPI specifikacije REST endpointa.

Alternativa ručnom kreiranju BPMAPO (opisanom u poglavlju 5.5.2.1) je mogućnost automatskog generiranja poslovnih objekata iz WSDL definicije SOAP servisa (ili OpenAPI specifikacije REST endpointa od verzije 8.5.7) korištenjem opcije *"Generate Types"* (Slika 5.22). Ako BPMKS servisi za rad s postojećim sustavima koriste cijeli model domene BPMA, na navedeni način, moguće je "jednim klikom" konstruirati cijeli podatkovni model BPMA.

5.5.2.3 Ovlaštenja korisnika i kreiranje timova

Svaki korisnički rad u procesu, to jest aktivnost koja postoji na DPP-a procesne aplikacije mora biti smještena unutar određene staze, a za svaku stazu zadužen je određeni tim. Timovi predstavljaju skupove ljudi koji imaju ovlasti odraditi određene zadatke, a u pravilu su definirani u okviru pojedine BPMA.

Da bi korisnik mogao biti član tima, potrebno mu je dodijeliti prava u korištenom BPMS-u. Primjerice u IBPM BPMS-u korisnicima se ovlaštenja dodaju putem procesne administracijske konzole, a IBPM može biti podešen i da korisnike uzima s vanjskog sustava organizacije (npr. središnjeg mjesta za pohranu podataka o autorizaciji kao što je LDAP[53] repozitorij). Moguće je definirati i grupe korisnika koje su zatim vidljive u cijelom IBPM okruženju.

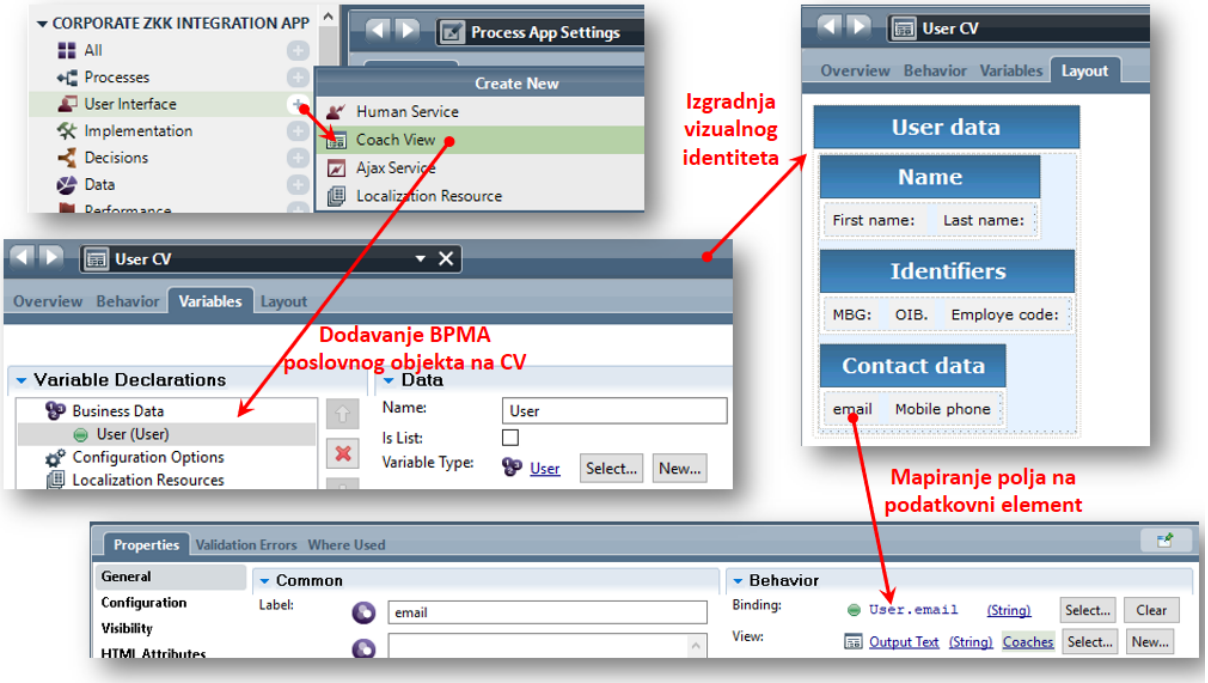
U svrhu kasnije simulacije i analize procesa (poglavlje 7), prilikom definicije korisničkog tima potrebno je odrediti određena svojstva tima, npr. cijenu rada određenog tima po satu (eng. *Cost per hour*) ili približnu veličinu tima (engl. *Capacity*).

5.5.2.4 Korisnička sučelja

Za prikaz podataka koristit ćemo *CV* komponente koje definiramo kao elementarna sučelja prema određenom BPMAPO, te ih tako možemo višestruko koristiti. Po potrebi, pojedini BPMAPO može imati više različitih *CV* komponenti od kojih svaka prikazuje različitu razinu

[53] **Lightweight Directory Access Protocol** (**LDAP**) je softverski protokol za pronalaženje resursa (kao što su datoteke ili uređaji) na mreži, bilo na javnom internetu ili lokalnoj mreži organizacije. Implementacija tzv. imenika u LDAP-u sadrži podatke o korisnicima, datotekama i aplikacijama, kao i njihove sigurnosne postavke.

detalja podatkovnog objekta. Za svaku *CV* komponentu definiraju se sučelja (elementi grafičkog oblikovanja, sekcije, elementi za unos, odabir i prikaz podataka, itd.), BPMAPO čije podatke će *CV* prikazivati (elementi za prikaz podataka povezuju se s poljima poslovnih objekata), te ponašanje (dodatna svojstva koja određuju dekoracije i ponašanje elemenata za prikaz kao što su CSS ili JavaScript). Slika 5.23 prikazuje izgradnju *CV*-a za *User* BPMAPO.

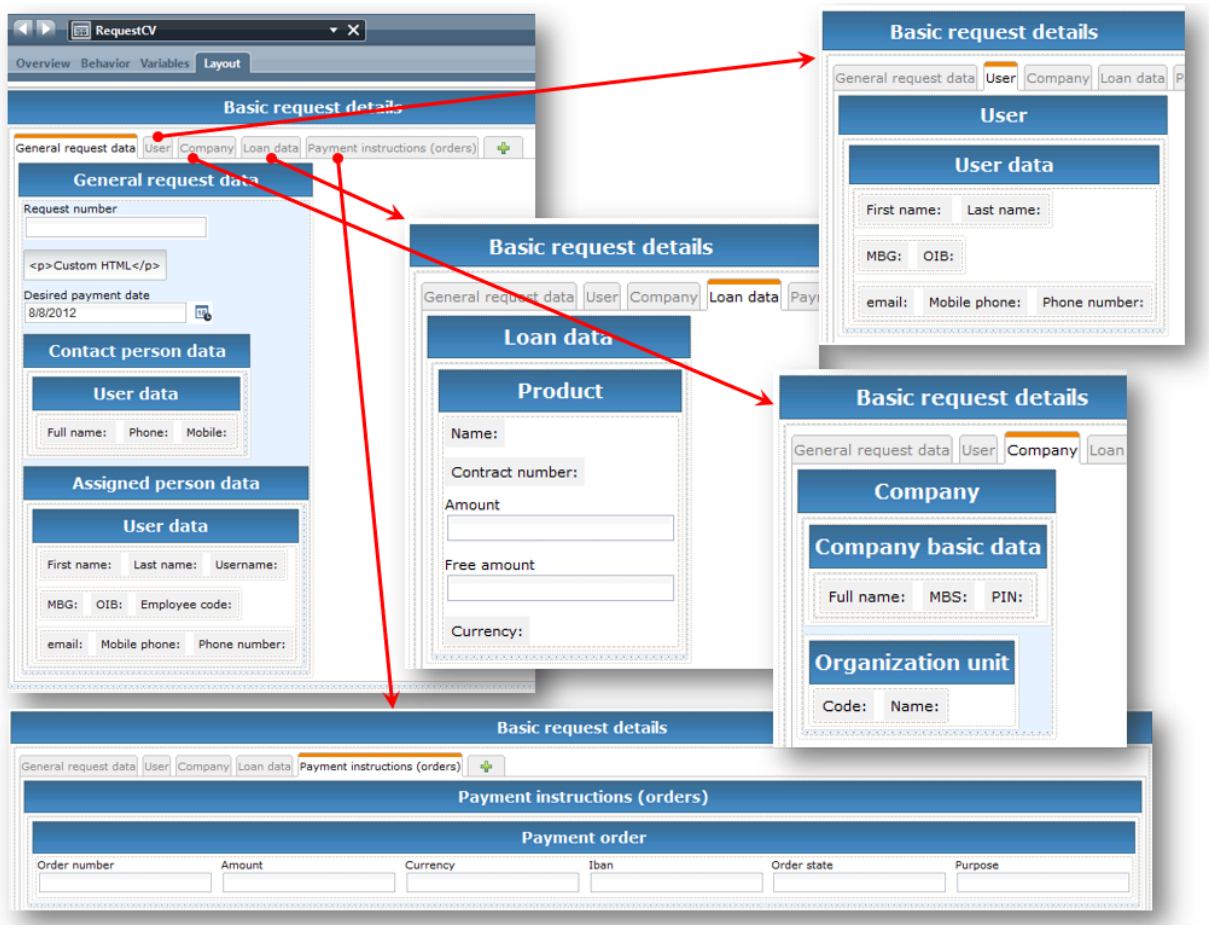


Slika 5.23 Izgradnja CV-a za objekt korisnika (User)

Ekrani na kojima se odvija interakcija korisnika i procesa uglavnom sadrže informacije iz više različitih BPMAPO.

Primjerice kod odobravanja zahtjeva opisanog u poglavlju 3.3 poželjno je da bankarski službenik osim informacija o zahtjevu ima i podatke o klijentu za kojeg je zahtjev zadan te korisniku koji je inicirao zahtjev. U svrhu ovakvog načina izgradnje korisničkih sučelja, koristit ćemo IBPM *Coach* komponente.

Coach predstavlja vizualno sučelje jednog korisničkog ekrana, a sastojat će se od kombinacije *CV* komponenti te dodatnih elemenata za prikaz podataka i izvršavanje akcija. Slika 5.24 prikazuje *coach* komponentu za prikaz informativnih detalja zahtjeva. Svaka kartica komponente sadrži *CV* komponentu za prikaz podataka BPMAPO, dodatne jednostavne komponente za prikaz pojedinih podataka ili njihovu kombinaciju.



Slika 5.24 Coach komponenta za prikaz informativnog ekrana detalja zahtjeva

Ako je kod prikaza poslovnih podataka potrebno primijeniti određenu logiku (npr. drugačiji prikaz za različite iznose kredita), i/ili ih je potrebno prikazati prema određenim uvjetima (npr. prikazati nazive statusa ovisno o šifri statusa) preporučljiv način je korištenje prilagođene HTML komponente gdje uz HTML možemo koristiti svu moć JavaScript programskog jezika. Slika 5.28 između ostalog prikazuje primjer definiranja HTML-a unutar prilagođene HTML komponente.

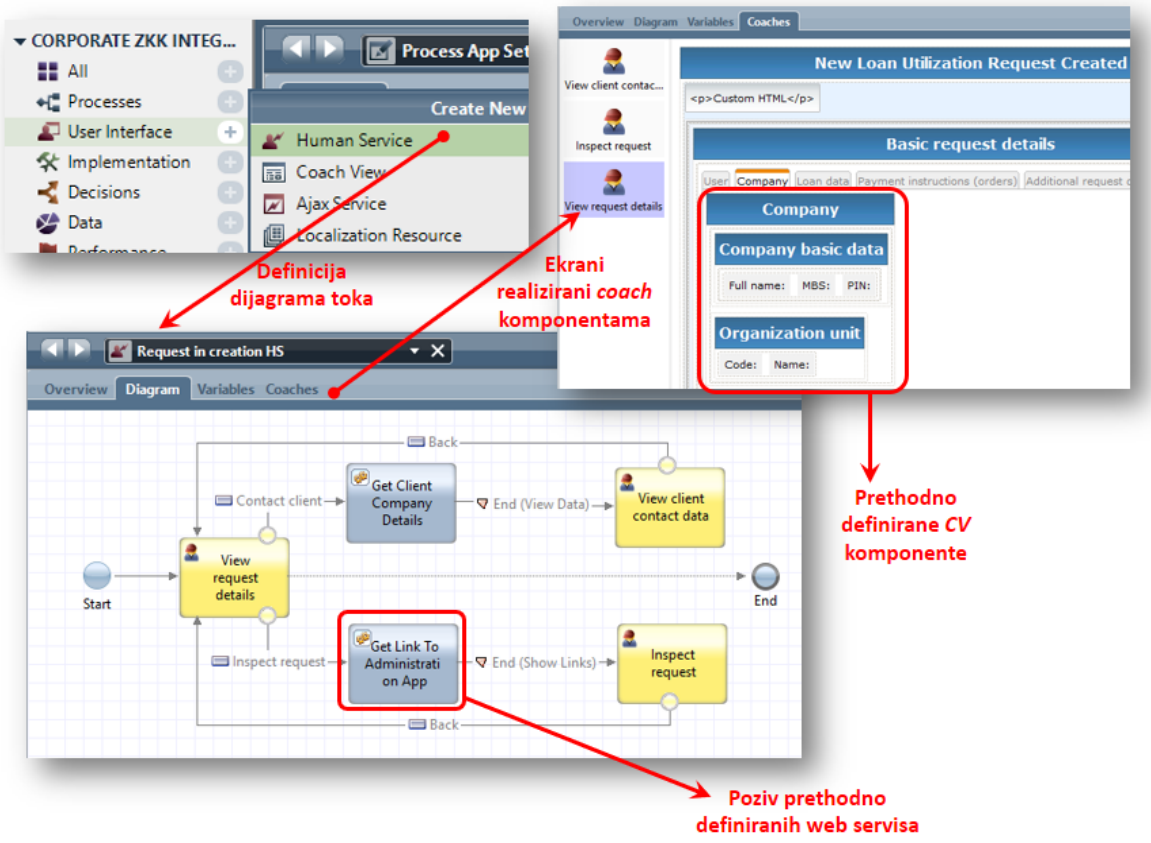
Coach i *CV* komponente također omogućuju manipulaciju nad podacima putem ugrađenog JavaScript programskog koda (engl. Inline JavaScript) koji se može definirati generalno za cijelu komponentu ili u sklopu određenih događaja (engl. Event Handlers). Dodatna opcija je korištenje skriptirane sistemske aktivnosti (engl. script task) za pripremu i željene operacije prije renderiranja *coach* ili *CV* komponente (npr. Slika 6.3 - *Initialize_Bpm_Contex_Data* aktivnost).

5.5.2.5 Dijagram poslovnog procesa

Izgradnja DPP-a u BPMS-u u pravilu je temeljena na BPMN standardu koji je opisan u poglavlju 2.2.2.1). Proces iterativne izgradnje DPP-a definiran je iteracijama u poglavlju 5.8, a prikazan u poglavlju 6 (potpoglavlje 6.3.2), te ovdje neće biti razmatran u detalje.

Svaka aktivnost poslovnog procesa sastoji se od određenih sučelja koje korisniku pomažu kod izvršavanja aktivnosti. Može sadržavati niz korisničkih ekrana s podacima PP-a, a osim toga potrebna je logika koja definira tok izvršavanja aktivnosti (kretanje po ekranima) te korisničke akcije.

Princip izrade aktivnosti u IBPM BPMS-u koristi komponente *ljudskih servisa*. Svaki *ljudski servis* sastoji se od niza povezanih korisničkih ekrana, servisnih i skriptiranih zadataka te popratne logike koja omogućava kretanje po ekranima i izvršavanje akcija. Korisnički ekrani realizirani su *coach* komponentama, a svaka *coach* komponenta izgrađena je od niza prethodno definiranih *CV* komponenti. Ključni elementi *ljudskih servisa* su ulazne i izlazne varijable koje čuvaju podatke, a povezane su s varijablama definiranim na razini procesa. Bez ulaženja u nepotrebne tehničke detalje, Slika 5.25 prikazuje opisani proces izgradnje *ljudskog servisa* na primjeru prve aktivnosti iz procesa opisanog u poglavlju 6.3.2.

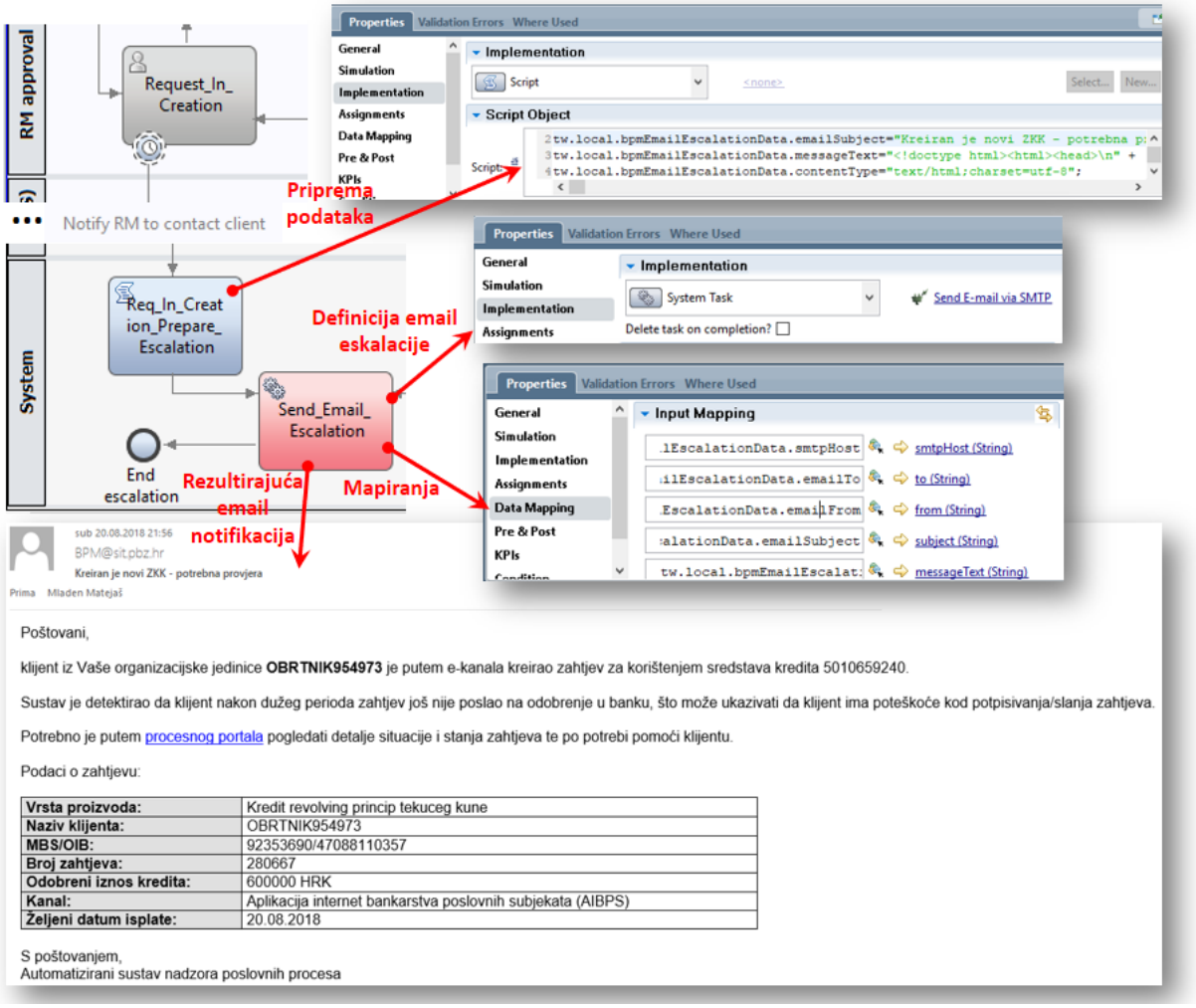


Slika 5.25 Izgradnja ljudskog servisa, dijagram i sučelja

5.5.2.6 *Eskalacije, notifikacije i ostali detalji implementacije*

Nad svakom od aktivnosti PP-a definiramo eskalacijski put uz pripadnu notifikaciju te, po potrebi, preraspodjelu osoba zaduženih za aktivnost. Vremenski događaj (engl. timer event), pripojen određenoj aktivnosti, po isteku predefiniranog vremenskog perioda (period je određen od strane poslovnih korisnika, kao iskustveno vrijeme u kojem je potrebno završiti određenu aktivnost) pokreće eskalacijsku aktivnost.

U sklopu ovog rješenja koristit će implementacija slanja e-mail obavijesti zaduženim osobama (izvorni integracijski servis *Send E-mail via SMTP* IBPM BPMS-a). Slika 5.26 prikazuje predloženi postupak kreiranja eskalacijskih puteva u procesu.



Slika 5.26 Princip izrade e-mail eskalacija nad aktivnostima

5.6 Specifikacija interakcije u procesnim i interakcijskim točkama

Razmjena informacija između BPMA i naslijeđene okoline odvija se u procesnim točkama, dok su za korisničku interakciju s BPM rješenjem karakteristične interakcijske točke. Ovo poglavlje sadrži tehničku specifikaciju te specifičnosti komunikacije u navedenim točkama.

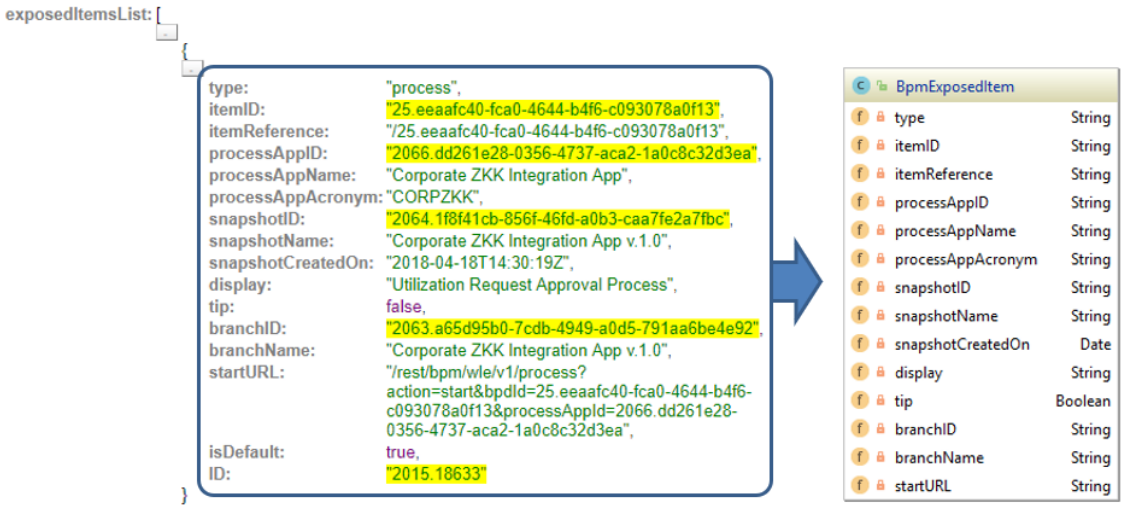
5.6.1 Standardne procesne točke

Važan koncept komunikacije u standardnim procesnim točkama, konkretno, komunikacije između BPMKS komponente ***klijenta za procesno vezanu komunikaciju*** i ***komponente za procesno vezanu komunikaciju*** BPM okruženja je kreiranje klasa na BPMKS-u koje će držati podatke o procesnim artefaktima specifičnim za BPMS u kojem je BPMA izgrađena.

Podaci vraćeni od strane API-ja ***komponente za procesno vezanu komunikaciju*** BPM okruženja nalaze se u predodređenim strukturama. Stoga je, temeljem struktura podataka o artefaktima iz BPMS-a, potrebno definirati odgovarajuće BPMKS klase, te napraviti preslikavanje podataka.

Primjerice, IBPM REST API sučelje IBPM BPMS-a, u pravilu definira informacije o procesnim artefaktima strukturom mapa (ili lista mapa) s podacima u obliku ključ-vrijednost parova. Sukladno navedenom, na BPMKS-u su kreirane klase u kojima je ključ mape naziv polja klase, dok je vrijednost mape zapravo podatak sadržan u polju klase. Slika 5.9 prikazuje klase procesnih artefakata IBPM-a koje su dovoljne su za implementaciju primjera u ovom radu. Kako bi realizirali preslikavanje podataka iz mapa u različite objekte (instance različitih klasa), na BPMKS-u je primarno korištena metoda serijalizacije mapa u JSON strukturu, te deserijalizacija dobivenog JSON podatka u odgovarajuće objekte kao što je prikazano na slici u nastavku.

Kao primjer, Slika 5.27 prikazuje strukturu BPMKS *BpmExposedItem* klase izgrađene temeljem strukture podataka o izloženoj BPMA koje vraća IBPM REST API.



Slika 5.27 Kreiranje klase BpmExposedItem temeljem podataka koje vraća IBPM REST API

Ako neki od parametara koje vraća IBPM REST API nisu potrebni mogu se izbaciti prilikom preslikavanja tako da se istoimena polja u *BpmExposedItem* klasi ne kreiraju (primjerice kao što je to napravljeno s *isDefault* i *ID* parametrom).

5.6.1.1 Ažuriranje poslovnih podataka instance procesa

Dodatnu pozornost kod standardnih procesnih točaka potrebno je posvetiti prijenosu poslovnih podataka na BPMA, to jest ažuriranju poslovnih podataka instance PP-a. Poslovni podaci instance procesa, ažuriraju se putem metoda API-ja ***komponente za procesno vezanu komunikaciju***, a pozornost je potrebno posvetiti na pravila ažuriranja, dakle definirati koji podaci se mogu ažurirati te na koji način.

Ažuriranje podataka vrši BPMKS koji uspoređuje zaprimljene poslovne podatke iz naslijeđenih aplikacija i podatke instance BPMA procesa, te radi uspoređivanje i ažuriranje temeljem predloženog postupka:

- ažuriraju se samo polja gdje postoji razlika između poslovnih podataka instance procesa i poslovnih podataka iz naslijeđenih sustava
- prazni objekti/polja poslovnih podataka naslijeđenih aplikacija ne smiju pregaziti poslovne podatke instance procesa
- ažuriraju se novi podaci iz naslijeđenih sustava koji još ne postoje na procesnoj instanci
- određeni poslovni podaci instance procesa prepišu se podacima iz naslijeđenih sustava, samo ako je tako definirano pravilima implementiranim u BPMKS-u
- očuvani su poslovni podaci instance procesa koji se koriste kao poveznice između identifikatora naslijeđenih sustava te identifikatora instance procesa. Principi navedenih identifikatora objašnjeni su u poglavlju 5.7, a definirani na konkretnom primjeru u poglavlju 5.7.1.

Ažuriranje poslovnih podataka na BPMA radi se BPMKS metodom *updateProcessInstanceBusinessData(BpmProcessInstance bpInstance, Request newRequest)* koja se nalazi u *BpmRestManagerImpl* klasi (poglavlje 5.3.1, Slika 5.4). Metoda radi dohvat trenutnih poslovnih podataka iz odgovarajuće instance procesa (metoda *getProcessInstanceBusinessData*).

Zatim, podaci instance procesa se ažuriraju s podacima pristiglim iz naslijeđene aplikacije korištenjem metoda *smartDataUpdate(...)* koje implementiraju prethodno navedeni postupak ažuriranja. Metoda *smartDataUpdate(...)* postoji na svakoj domenskoj klasi BPMKS-a, a metoda sadrži pravila ažuriranja poslovnih podataka te klase (poglavlje 5.3.5, Slika 5.8).

Nakon ažuriranja poslovnih podataka na BPMKS-u, uz pomoć REST poziva implementiranog u metodi *setProcessInstanceBusinessData(BpmProcessInstance bpInstance, Request request)*, ažurirani podaci šalju se na BPMA (poglavlje 5.3.1, Slika 5.4).

Poslovni podaci šalju se kao parametri IBPM REST API poziva. Java objekte s poslovnim podacima potrebno je serijalizirati u JSON format te na dobiveni format dodati naziv varijable BPMAPO koja će navedeni podatak sadržavati.

5.6.2 Pozadinske procesne točke

Prilikom dohvata podataka iz naslijeđenih aplikacija BPMA upućuje poziv prema BPMKS-u koji dalje radi poziv prema određenim naslijeđenim aplikacijama. Stoga BPMKS mora imati informaciju koja naslijeđena aplikacija sadrži koji dio podatka potrebnih za BPMA.

Najbolji primjer ove situacije je dohvat poveznica na aktivnosti naslijeđenih aplikacija. Temeljem identifikatora aktivnosti, koji šalje BPMA, BPMKS upućuje poziv prema ***komponenti za izlaganje postojećih funkcionalnosti*** prilagodnika naslijeđene aplikacije zadužene za navedenu akciju. Stoga BPMKS mora imati mapiranje, to jest popis aktivnosti koje odrađuje pojedina naslijeđena aplikacija.

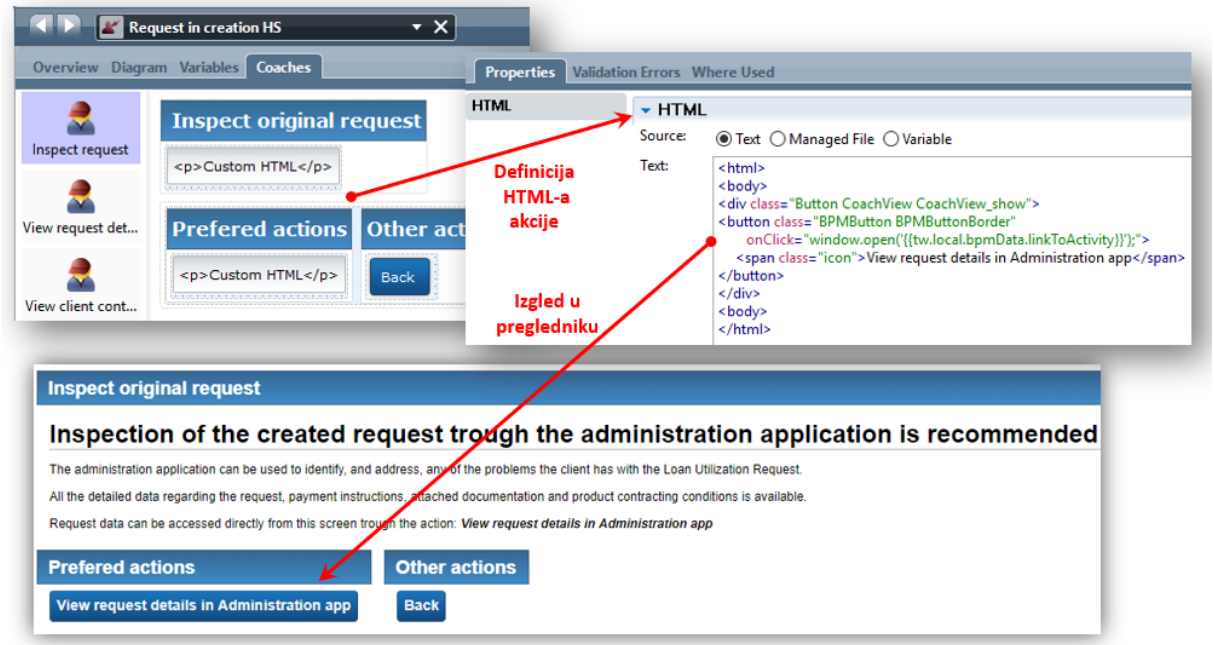
Navedeni postupak realizira BPMKS komponenta ***klijent za komunikaciju s naslijeđenim aplikacijama*** (poglavlje 5.3.3, Slika 5.6) metodom *mapActivityIdToLegacyAppUrl(...)* koja temeljem identifikatora aktivnosti određuje naslijeđenu aplikaciju zaduženu za tu aktivnost. Pobrojani tip *LegacyActivities* sadrži listu svih aktivnosti, a polje *LEGACY_APP_LINKS* popis osnovnih adresa naslijeđenih aplikacija. Na dijagramu klasa BPMKS komponente ***klijent za komunikaciju s naslijeđenim aplikacijama*** (Slika 5.6) vidljive su sve opisane metode i polja.

5.6.3 Interakcijske točke, preusmjeravanje na naslijeđene aplikacije

U interakcijskim procesnim točkama službenik putem poveznica iz BPMA pristupa naslijeđenim aplikacijama (ulazne interakcijske točke). Prije generiranja BPMA ekrana na kojem je prikazana poveznica do aktivnosti, u pozadinskoj procesnoj točci (opisanoj u prethodnom poglavlju) odradi se dohvat poveznice iz naslijeđenih sustava.

Osim samog prikaza poveznice na korisničkom ekranu, BPMA mora sadržavati BPMPO koji će čuvati identifikator pojedine aktivnosti te dohvaćenu poveznicu na aktivnost. U tu svrhu kreiran je *BpmData* BPMPO na BPMA. *BpmData* čuva vrijednost poveznice (polje *getLinkTOActivity* objekta), te naziv aktivnosti. Svaka aktivnost na BPMA ima svoj *BpmData* BPMAPO koja sadrži podatke specifične za tu aktivnost.

Na korisničkom ekranu, poveznica je prikazana prilagođenim HTML-om unutar HTML elementa *Coach* komponente. Slika 5.28 prikazuje osnovni proces kreiranja dijela sučelja koje se koristi u interakcijskim točkama.



Slika 5.28 Definicija interakcijskih točaka

5.7 Poveznice identifikatora procesa i identifikatora u naslijeđenim sustavima

Kako se postojeći PP-i protežu kroz različite naslijeđene sustave, česta situacija je da se identifikatori zahtjeva, instrukcija i artefakata vezanih uz PP mijenjaju od sustava do sustava. To znači da ne postoji jedinstveni identifikator koji bi proces jedinstveno definirao, već svaki naslijeđeni sustav ima zasebni identifikator za svoj dio procesa.

Pitanje koje se ovdje postavlja je kako sve identifikatore naslijeđenih sustava povezati s instancom procesa u BPM okruženju? Kako će određena naslijeđena aplikacija znati koja instanca PP-a odgovara podacima zahtjeva koji se trenutno obrađuje u konkretnoj naslijeđenoj aplikaciji?

Nekoliko opcija razmatrano je da bi se dao odgovor na postavljena pitanja:

- Inicijalno, postojala je ideja da se identifikator instance procesa sprema s podacima procesa u svaki od naslijeđenih sustava kroz koje se proces proteže. Međutim, taj pristup zahtijevao bi širenje podatkovnog modela naslijeđenih aplikacija, te prilagodbu naslijeđenih aplikacija da spremaju i barataju novim podatkom. Ujedno, novi podatak ne predstavlja poslovni podatak PP-a.

- Druga opcija bila je koristiti BPMKS kao centralno mjesto gdje bi se temeljem identifikatora zahtjeva, instrukcije ili relevantnog artefakta iz naslijeđenih sustava određivao identifikator procesa kojem zahtjev instrukcija ili artefakt pripadaju. U tom slučaju BPMKS bi trebao imati vlastitu bazu podataka gdje bi se držale poveznice navedenih identifikatora. Vlastita baza podataka BPMKS-a ukratko je razmatrana u poglavlju 4.2.1, međutim model podataka te baze sastojao bi se od niza podataka koji nemaju neko poslovno značenje, te bi uslijed gomilanja podataka kroz povijest potencijalno mogao stvoriti probleme pada performansi i zagušenja.

  Dodatna činjenica koja ne ide u prilog metodi spremanja identifikatora instanci procesa u bazu je činjenica da identifikator instance procesa ovisi o korištenom BPMKS-u, te ne mora biti u potpunosti jedinstven kroz vrijeme. Dobar primjer za ovaj slučaj je IBPM skup alata gdje se identifikatori procesa resetiraju i kreću od početka prilikom nadogradnje na novu verziju. Stoga, ako su korisnici upotrebljavali identifikator instanci procesa kao ključ u bazama podataka imat će poteškoće prilikom rada s novom verzijom proizvoda.

- Kao najprikladnija pokazala se opcija čuvanja identifikatora procesa iz naslijeđenih aplikacija u podacima instance procesa na BPM okruženju.

  Jedna od vrlo korisnih funkcionalnosti većine BPMS-ova je mogućnost pretrage po instancama procesa i podacima koje sadrže instance procesa. Uz pomoć navedene funkcionalnost, naslijeđena aplikacija koristeći vlastiti identifikator šalje upit prema BPMKS-u. BPMKS prema zaprimljenom identifikatoru pronalazi instancu procesa te nad njom određuje traženu akciju. Na navedeni način sve naslijeđene aplikacije, temeljem vlastitih identifikatora, znaju komunicirati s BPMA, a uloga BPMKS-a, kao posrednika te komunikacije, je znati koji identifikator koristi koja naslijeđena aplikacija te uz pomoć njega dohvatiti instancu procesa.

  Dodatno pitanje koje se ovdje postavlja je pitanje performansi pretrage instanci procesa temeljem identifikatora iz naslijeđenih aplikacija. Stvara li navedeni način pretrage opterećenje na BPM okruženju? Odgovor na ovo pitanje krije se u načinu na koji BPMS radi pretragu, te količini podataka koje mora pretražiti. BPMS pretragu moguće je ograničiti temeljem raznih kriterija (npr. pretraživati samo aktivne instance procesa) što sužava skup te olakšava pretragu. Dodatno, postoji mogućnost stvaranja indeksa nad željenim podacima instanci procesa, što uvelike ubrzava pretragu. Brzina također ovisi i o PP-u, odnosno prosječnom broju instanci procesa koje su istovremeno aktivne na BPM

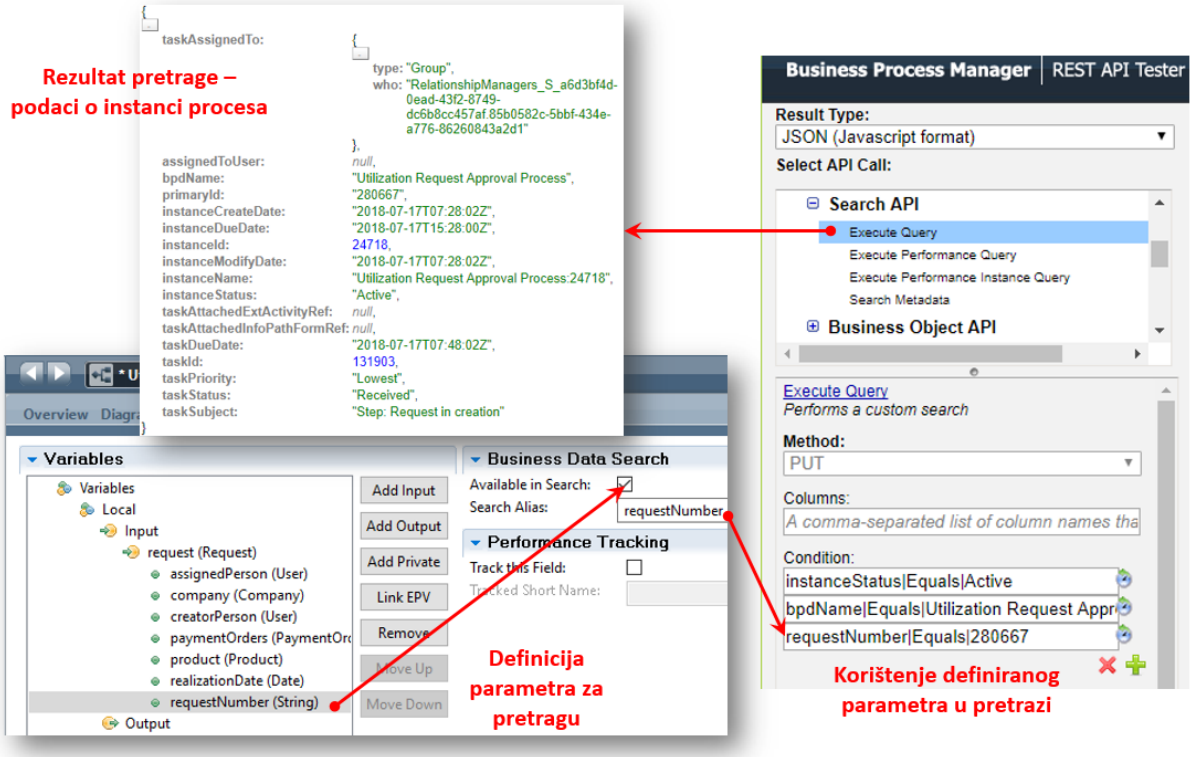
okruženju. Primjerice, za razmatrani primjer zadavanja i obrade ZKK-a (poglavlje 6) statistika je pokazala da se dnevno u prosjeku odobri 15-tak zahtjeva. To bi značilo da trenutno može biti aktivno najviše 15-tak instanci procesa na BPM okruženju, što je zanemariva brojka.

5.7.1 Tehnička realizacija identifikacije procesa

Realizacija opisanog koncepta prikazana je uz pomoć IBPM REST SEARCH API-a, koje omogućava pretraživanje instanci procesa prema:

- osnovnim podacima instance procesa (identifikator instance, aktivnosti, ime, itd…)
- poslovnim podacima instance procesa koji se eksplicitno moraju uključiti u pretragu (poslovni podaci koji predstavljaju identifikatore procesa iz naslijeđenih aplikacija).

Slika 5.29 prikazuje definiciju poslovnih podataka temeljem kojih se vrši pretraga instanci procesa, te proces pretrage. Polje identifikatora procesa iz naslijeđenih aplikacija (podatak *requestNumber*) dodaje kao parametar funkcionalnosti pretrage instanci procesa u definiranoj BPMA. Prikazano je i kreiranje pretrage temeljem novodefiniranog parametra koristeći IBPM REST API (Search API).



Slika 5.29 Omogućavanje i pretraga instanci procesa temeljem poslovnih podataka

U svrhu programske pretrage instanci procesa na BPMKS-u postoji pobrojani tip *BpmSearchConstants* (unutar *BpmRestManagerImpl* klase) koji sadrži sve potrebne uvjete pretrage. Klasa *BpmProcessInstance* BPMKS-a sadrži podatke o instanci procesa (Slika 5.9) koju dobijemo kao rezultat pretrage (Slika 5.29). Preslikavanje podataka iz rezultata REST upita u *BpmProcessInstance* objekt analogno je postupku opisanom u poglavlju 5.6.1.

Važno je napomenuti da pretragu instanci procesa možemo raditi i temeljem više različitih identifikatora iz različitih naslijeđenih aplikacija koji su vezani uz istu instancu procesa (skupni identifikator). Konkretna realizacija na poslovnom primjeru, uz odabir i identifikatora naslijeđenih sustava, te konstrukciju skupnog identifikatora prikazana je u poglavlju 6.3.5.

5.8 Procesno-podatkovna integracija, prijedlog iteracija razvoja i integracije BPMA

U nastavku je predložen strukturiran niz iteracija kojima se definira redoslijed te opisuju detalji koraka za uspješnu izgradnju BPMA, te integraciju BPMA s postojećim aplikacijama prema predloženom modelu integracije. Svaka od iteracija fokusira se na određene dijelove BPMA, te na ispunjavanje ciljeva koji će se detaljnije analizirati u nastavku.

Cjelokupni postupak prikazuje procesno-podatkovnu razinu integracije BPMA s procesima naslijeđenih sustava, dakle fokus je na toku procesa i interakciji s podacima u tom procesu.

Izgradnji procesne aplikacije pristupamo s pretpostavkom da je napravljena integracija na razini arhitekture (poglavlja 5.1 do 5.6).

5.8.1 Prva iteracija - definicija i otkrivanje karakteristika procesa

U prvoj iteraciji potrebno je pronaći odgovor na pitanja: „*Što od procesa želimo?*“ i „*Tko sudjeluje u procesu?*“. Kako bi izgradili BPMN DPP-a potrebno je sakupiti općenite informacije te metapodataka o procesu. Postavlja se pitanje zašto se proces modelira, određuje se ime te ciljevi procesa, otkriva se i definira pozadina koja obavlja funkcionalnosti procesa te se sažimaju sve sakupljene informacije. U suradnji s poslovnim sektorom potrebno je identificirati poslovne izazove i ustanoviti mjerljive poslovne ciljeve procesa [30], [31], te razraditi način na koji se ti ciljevi ostvaruju.

Zatim, razmišlja se o sudionicima procesa i određuje se koji sudionici su zaduženi za koje dijelove procesa. Sukladno navedenom, skiciraju se korisničke grupe koje imaju ovlasti nad pojedinim aktivnostima procesa. Definira se koji dijelovi procesa se odrađuju u kojim naslijeđenim aplikacijama te koje će se sve naslijeđene aplikacije koristiti. Ugrubo se skicira pojednostavljeni tok procesa kroz naslijeđene aplikacije te se definiraju statusi vezani uz proces

kao osnova grananja i toka procesa. Na kraju poželjno je obaviti informativni razgovor s vlasnikom procesa, osobom koja je odgovorna za proces te u čijem interesu je da proces uspije. Iteracija završava definicijom artefakta (kontejnera) procesne aplikacije u BPM alatu koji će sadržavati model PP-a i sve elemente vezane uz procesnu aplikaciju.

Prva iteracija izgradnje BPMA nije striktno vezana uz BPMS i MPP. Predstavlja zapravo kombinaciju analize naslijeđenih procesa i aplikacija u svrhu precizne definicije karakteristika procesa s pripremama za modeliranje analiziranog procesa u BPMS-u. Simulacija tijekom prve iteracije nije potrebna.

5.8.2 Druga iteracija - modeliranje toka procesa

„*Kako proces teče?*“ pitanje je na koje je potrebno dati odgovor u drugoj iteraciji. Potrebno je razraditi tok procesa, otkriti sve moguće puteve toka procesa te izgraditi BPMN model PP-a. Putevi procesa uključuju niz zadataka, aktivnosti, koje je potrebno izvršiti da bi proces postigao očekivani rezultat, to jest uspješno završio. Osim uspješnog završetka, tok procesa uključuje puteve koji rezultiraju negativnim rezultatima te pogreškama. Ugrubo se definiraju eskalacije i potrebne notifikacije.

Kroz administracijske alate BPM okruženja korisnicima koji sudjeluju u procesu dodjeljuju se prava pristupa BPM okruženju. Temeljem skiciranih korisničkih grupa iz prethodne iteracije, kreiraju se korisničke grupe na BPM alatu (prema opisima u poglavlju 5.5.2.3) te se modeliraju polja i staze za koje će pojedine korisničke grupe biti zadužene. Koncentrira se na modeliranje i ugradnju procesa na najvišoj razini te stvaranje općenite slike funkcioniranja BPMA, bez uključenih funkcionalnosti naslijeđenih aplikacija u svrhu izvršavanja aktivnosti procesa. U sklopu definicije toka procesa precizno se modelira logički tok aktivnosti unutar procesa. U ovoj iteraciji, informacije o načinu i mehanizmima izvršavanja aktivnosti procesa nisu od važnosti.

Dodatnu pozornosti treba obratiti na općeniti dogovor o načinu modeliranja PP-a te detaljnije razumijevanje opsega ugradnje. Potrebno je uskladiti metrike na razini procesa i s očekivanjima poslovne strane i osigurati prijenos znanja i iskustva između tima zaduženog za analizu te tima koji radi implementaciju procesnog rješenja.

Kao dio ove iteracije, nakon modeliranja toka PP-a na višoj razini, to jest prije implementacije svih detalja procesa, obavlja se inicijalna simulacija procesa. Inicijalna simulacija uključuje dvije provjere:

- Validaciju toka izvođenja modeliranog procesa kroz sve moguće scenarije kako bi osigurali ispravnost svih tokova procesa te detektirali potencijalne pogreške u logici grananja.
- Detekciju vremena čekanja i potencijalnih uskih grla procesa. Temeljem očekivanih vremena trajanja pojedinih aktivnosti, dostupnih resursa te iskustvenih zapažanja o poslovnom procesu definiramo simulacijske parametre na razini modeliranog DPP-a. Temeljem izvođenja automatske analize tako definiranog procesa, BPMS može aproksimirati vrijeme čekanja za pojedinu aktivnost te detekcije uskih grla.

Stoga, inicijalnom simulacijom otkrivamo pogreške u procesu te se izvlačimo zaključke za daljnji tok implementacije.

5.8.3 Treća iteracija - razrada integracije s naslijeđenim aplikacijama

Glavno pitanje koje se postavlja u ovoj iteraciji je: "*Na kojim dijelovima procesa postoji programska interakcija naslijeđene okoline i BPMA, te gdje se odvija korisnička interakcija i izvršavanje aktivnosti procesa kroz naslijeđene aplikacije?*"

Određuju se standardne i pozadinske procesne točke prema definicijama opisanim u poglavlju 4.4.1 te se identificiraju događaji u toku procesa koji zahtijevaju postojanje procesnih točaka. Osiguravaju se preduvjeti za ostvarenje komunikacije u tim točkama.

Definiraju se interakcijske točke kao preduvjet za izgradnju sučelja putem kojih će se interakcija ostvariti. Tako se određuje gdje i kada će korisnici pristupati podacima te gdje će se izvršavati aktivnosti modeliranog procesa.

Simulacija u ovoj točci nije potrebna, jer su u pitanju pripremne radnje za niz sljedećih iteracija. Nema izmjena po BPMA.

5.8.4 Četvrta iteracija - izgradnja poslovnog modela i korisničkih sučelja

Četvrta iteracija, daje odgovor na pitanje: "*Kojim poslovnim objektima su predstavljeni poslovni podaci procesa, te koja sučelja se koriste za prikaz i korisničku interakciju s procesom na BPMA?*"

Određuje se kako korisnici vide podatke modeliranog procesa na BPMA. Potrebno je postići dogovor oko svih potrebnih korisničkih sučelja, modela podataka koji će služiti kao podrška tim sučeljima i poslovnom odlučivanju.

U suradnji s poslovnim korisnicima, imajući u vidu mogućnosti postojećih sustava, definira se osnovni skup poslovnih podataka koji će se prikazivati na BPMA. Temeljem definiranog

skupa podataka kreiraju se poslovni objekti na BPMA, podatkovne strukture i objekti korišteni u definiciji PP-a, a koji su identičnim objektima modela domene na BPMKS-u.

Temeljem podatkovnih struktura izgrađuju se korisnička sučelja (primjerice *ljudski servisi* s *coach* i *CV* komponentama u IBPM alatu) za podršku interakcijskim točkama (poglavlje 4.4.1), te se određuje razina detalja i informacija o procesu koje će se tim sučeljima prikazivati.

Simulacija izvršena u ovoj iteraciji uključuje provjeru ispravnosti sučelja i prikaza podataka sadržanih u kreiranom poslovnom modelu.

5.8.5 Peta iteracija - implementacija procesnih i interakcijskih točaka

U ovoj iteraciji postavljaju se pitanja: "*Kako i putem kojih sredstava će se odvijati interakcija s naslijeđenom okolinom? Kako se izvršavaju aktivnosti procesa?*"

Pretpostavimo da su na BPMKS-u izgrađeni servisi za upravljanje procesnim akcijama koji se koriste u standardnim procesnim točkama te servisi za rad s postojećim funkcionalnostima koji su bitni za pozadinske procesne točke. Temeljem podatkovnog modela definiranog u četvrtoj iteraciji, te lokaciji procesnih točaka DPP-a (utvrđenih u trećoj iteraciji) radi se implementacija komunikacije s naslijeđenim sustavima u procesnim točkama. Provjerava se da servisi za upravljanje procesnim akcijama manipuliraju procesom na željen način, te se gradi implementacija servisa za dohvat podataka i korištenje funkcionalnosti naslijeđenih aplikacija (KINS opisana u poglavlju 5.5.2.2). Ako je nužno, zbog potrebe komunikacije s naslijeđenim aplikacijama, dorađuje se podatkovni model BPMA.

Zatim, osigurava se ispravnost podataka dohvaćenih u pozadinskim točkama koje prethode ulaznim interakcijskim točkama, konkretno, dohvat poveznica za preusmjeravanje na implementaciju funkcionalnosti u naslijeđenim aplikacijama. Po potrebi, dodaju se aktivnosti poziva servisa KINS. Dorađuju se detalji implementacije te sučelja *ljudskih servisa*. Određuju se identifikatori naslijeđenih sustava, koji će se koristiti kod dohvata instanci procesa prema principima opisanim u poglavlju 5.7.

Simulacija u ovoj iteraciji fokusira se isključivo na provjeru prijelaza između sučelja BPMA i sučelja naslijeđenih aplikacija u interakcijskim točkama. Provjeravaju se rezultati komunikacije u procesnim točkama, izvršavanje aktivnosti procesa, te ispravnost prikaza podataka o procesu kroz BPMA sučelja nakon promjene istih u naslijeđenim sustavima.

5.8.6 Šesta iteracija - integracija aktivnosti s tokom procesa i razrada detalja implementacije

Šesta iteracija pruža odgovor na pitanje „*Kako integrirati sve dijelove procesa i koji su završni detalji procesne implementacije?*“. Fokus iteracije leži na cjelokupnom toku procesa.

Modelirani tok procesa, izgrađena sučelja, definirani podatkovni model i implementaciju aktivnosti procesa potrebno je složi u cjelokupno rješenje koje je spremno za korisničko testiranje.

Osigurava se redoslijed i međusobna povezanost aktivnosti razrađenih u prethodnoj iteraciji. Temeljem rezultata izvršavanja aktivnosti procesa detaljno se razrade uvjeti i vrste grananja te se odrađuje implementacija grananja. Provjerava se sinkroniziranost podataka u BPMA i naslijeđenim sustavima kontrolom komunikacije u procesnim točkama. Definiraju se krajnji rokovi aktivnosti (engl. due date) u BPMA, vremena do eskalacije aktivnosti te se određuje cijena utroška pojedinih resursa aktivnosti kao preduvjet za analizu i optimiranje procesa. Dorađuju se detalji implementacije, osigurava automatizacija procesa te odrađuje obrada pogrešaka i definiraju se eskalacijske poruke. Nema uvođenja novih funkcionalnosti.

Simulacija u ovoj iteraciji fokusira se na ispravno izvršavanje cjelokupnog toka procesa kroz BPMA i naslijeđene aplikacije.

5.8.7 Sedma iteracija - otkrivanje pogrešaka i iznimaka u procesu uz finalnu verifikaciju

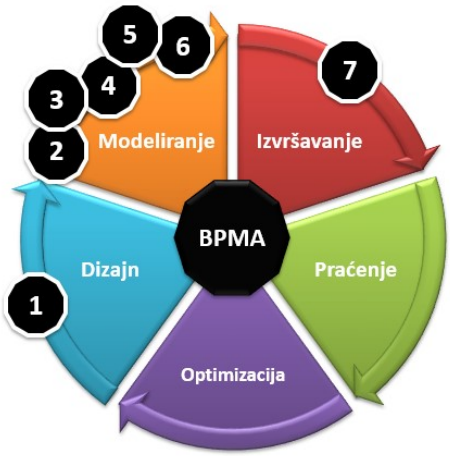
Glavni cilj i srž sedme iteracije je završna simulacija te ispravak pogreška koje proizađu iz zadnje simulacije. Potrebno je dati odgovor na pitanje*„Funkcionira li modeliran proces na način kako je zamišljen?“*. Provjeravaju se različiti uvjeti grananja te simuliraju različiti scenariji koji ispituju sve moguće tokove i sve moguće završetke procesa. Cilj je isproducirati funkcionalno i stabilno rješenje koje se predaje korisnicima na verifikaciju. Nakon isporuke korisnicima ostaje angažman oko podrške korisničkim testovima, moguće dorade te konstantno održavanje i usavršavanje. Po potrebi u ovoj iteraciji definiraju se dodatna *ad hock*[54] sučelja, izvještaji i dashboard-ovi[55] koji služe da bi se dobio jasniji uvid u proces [59].

[54] **Ad hoc** događaj, unutar IBPM-a, je mehanizam koji omogućuje nezavisno pokretanje niza aktivnosti u postojećem procesu**.** Ako se pridruži procesu, u opcijama instance dotičnog procesa pojavljuje se nova ulazna točka. Koraci koji se izvršavaju u sklopu *ad hoc* događaja imaju pristup svim podatcima unutar procesa, što može biti korisno za "usputno" obavljanje određenog posla.

[55] **Dashboard**, u teoriji upravljanja informacijskim sustavima, predstavlja čitljivo i jednostavno grafičko sučelje, koje u stvarnom vremenu, prikazuje trenutni status i povijesne podatke KPI-jeva određene organizacije.

5.9 Veza predloženih iteracija i životnog ciklusa BPM rješenja

Ako navedene iteracije pogledamo s aspekta životnog ciklusa BPM rješenja, koji je u detalje razrađen u mnogobrojnim u znanstvenim i komercijalnim radovima [12], [15], [44], [69], [70], itd., možemo identificirati u kojim fazama se nalaze pojedine predložene iteracije (Slika 5.30).



Slika 5.30 Raspored predloženih iteracija izgradnje BPMA kroz životni ciklus BPMA

Vidljivo je da najveći dio posla leži u fazi modeliranja s početnom iteracijom u fazi dizajna, a završnom u fazi izvršavanja procesa. Preostale dvije faze životnog ciklusa BPMA, faza praćenja i faza optimizacije, spadaju u domenu prikupljanja podataka o izvršavanju gotovog procesa u produkcijskim okruženjima te analizi tih podataka u svrhu otkrivanja poboljšanja i promjena u BPMA. Faze praćenja i optimizacije predstavljaju opširno područje izvan opsega doktorskog rada, te je u osnovama razrađeno u poglavlju 7.

6 Primjena modela integracije na proces zahtjeva za bankarske usluge

U ovom poglavlju prikazana je programski podržana strukturna inovacija i racionalizacija poslovnog procesa zaprimanja zahtjeva od strane klijenata te obrade i realizacije od strane davatelja usluga.

U svrhu demonstracije postupka odabran je PP ugovaranja korištenja kredita, to jest zadavanje i obrada zahtjeva za korištenje kredita (ZKK). ZKK je već ugrubo objašnjen u poglavlju 3.3.3.1. prilikom demonstracije ideje predloženog modela integracije. Proces je odabran zbog kompleksnosti, broju naslijeđenih aplikacija i korisnika koji sudjeluju u realizaciji procesa, pa je stoga prikladan za demonstraciju korisnosti predloženog modela.

Poglavlje 6.2 prikazuje nedostatke i probleme postojećeg stanja. Navedeni problemi i nedostaci otklanjaju se procesom izgradnje i integracije BPMA s naslijeđenim sustavima koji je prikazan u poglavlju 6.3 temeljem postupka definiranog u poglavlju 5.8. Objašnjeni su osnovni principi korišteni prilikom modeliranja procesa, tok i razmjena podataka između naslijeđenih aplikacija i BPMA, te korisnička interakcija s procesom. Poglavlje 6.4 prikazuje dodatne strukturne inovacije koje su postupkom izgradnje i integracije BPMA uvedene u poslovni proces.

6.1 Smjernice i plan razvoja BPM rješenja

Svaka tvrtka u razvoj rješenja za BPM kreće s različite početne točke i uslijed različitih potreba. Neki već imaju definirane procese, drugi žele naglasiti automatizaciju procesa, dok treći trebaju bolju transparentnost, mjerenja performansi, optimiranje itd. Međutim, cilj svake tvrtke je najprije razumjeti process te preslikati njegovo trenutno stanje u što fleksibilniji i razumljiviji model [27].

Dobar i sveobuhvatan plan razvoja rješenja omogućava krajnju fleksibilnost sustava spram nužnih promjena. Pristup razvoju BPM rješenja korištenjem modernih BPMS-ova u suprotnosti je s tradicionalnim razvojem poslovnih modela gdje je izrada svakog PP-a, iako modeliranog od strane poslovnih korisnika, ovisila o potpori stručnjaka IT. Informatička realizacija u tom slučaju često je bila sasvim druga stvar od zamišljenog modela iako je zapravo na temelju njega generirana. Zasebne izmjene na jednoj razini često nisu u bile vidljive na drugoj razini, primjerice izmjene implementacije nisu se vidjele u modelu i obrnuto [29]. Zbog toga je

potrebno slijediti dobro definiran holistički, iterativni, pristup gdje se približe poslovne potrebe te realizacija tih potreba od strane stručnjaka IT.

6.2 Poslovni proces i nedostaci postojećeg stanja kao ciljevi strukturnih poboljšanja

Proces ZKK proteže se kroz više naslijeđenih sustava, te uključuje veći broj međusobno neovisnih službenika koje obavljaju određene aktivnosti u sklopu procesa (Slika 6.1 ilustrira osnovni tok procesa).

Jedna od glavnih poteškoća u izvršavanju procesa ZKK je da ne postoji centralno mjesto gdje bi se vidjele precizne informacije o stanju procesa, dakle u kojem točno koraku se proces nalazi, koji zadaci slijede, a koji su završili. Te informacije trenutno se mogu dobiti spajanjem informacija dobivenih uvidom u proces kroz više različitih naslijeđenih sustava.

Korisnici pojedinih aplikacija su iz različitih organizacijskih dijelova te nemaju doticaja jedni s drugima.

Aktivnosti procesa nisu učinkovito povezane, to jest postoji nedostatak komunikacije između ljudi različitih organizacijskih dijelova koji aktivnosti izvršavaju. Iako nakon aktivnosti postoje e-mail obavijesti osobama zaduženim za sljedeću aktivnost u toku procesa, čest slučaj je da se te e-mail poruke zagube u hrpi s ostalim automatskim obavijestima ili završe u sandučićima s neželjenom poštom. U tom slučaju proces ostane "visjeti" dok ne dođe do eskalacije od strane klijenta ili intervencije osobe koja uoči problematičnu situaciju.

Cjelokupni tok procesa nalazi se u glavama ljudi koji na procesu rade. Svaki korisnik upoznat je s aktivnostima koje obavlja, bez jasne slike o cjelokupnom toku procesa. Pitanje „*Koji su koraci i aktivnosti procesa?*“ upućeno dvjema različitim osobama koje u procesu sudjeluju, često rezultira s dva različita odgovora. To osobito vrijedi za organizacije koje upotrebljavaju agilne razvojne metodologije te proizvedu minimalnu količinu dokumentacije. Kada procesi nisu jasno definirani ili kada nisu jasno definirane uloge ljudi u tom procesima, postoji mogućnost da ljudi aktivnosti, zadatke i probleme rješavaju zaobilaznim putevima, pronalazeći rješenja na načine na koje to ne bi trebalo raditi.

Ne postoji centralno mjesto gdje bi se mogla napraviti analiza izvršavanja procesa. Teško je otkriti uska grla te detektirati aktivnosti procesa koje bi treba ubrzati ili optimizirati.

Detaljnija statistika o izvršavanju procesa mora se složenim upitima, ručno, vući iz baza podataka, čime se opet ne dobije cijela slika procesa, jer zapisi o procesu postoje u više odvojenih sustava koje je često teško povezati.

U slučaju parcijalnog izvršavanja ili odbijanja zahtjeva klijent često nije u detalje ili u potpunosti informiran o razlozima donesene odluke.

6.3 Iterativna izgradnja programskog rješenja

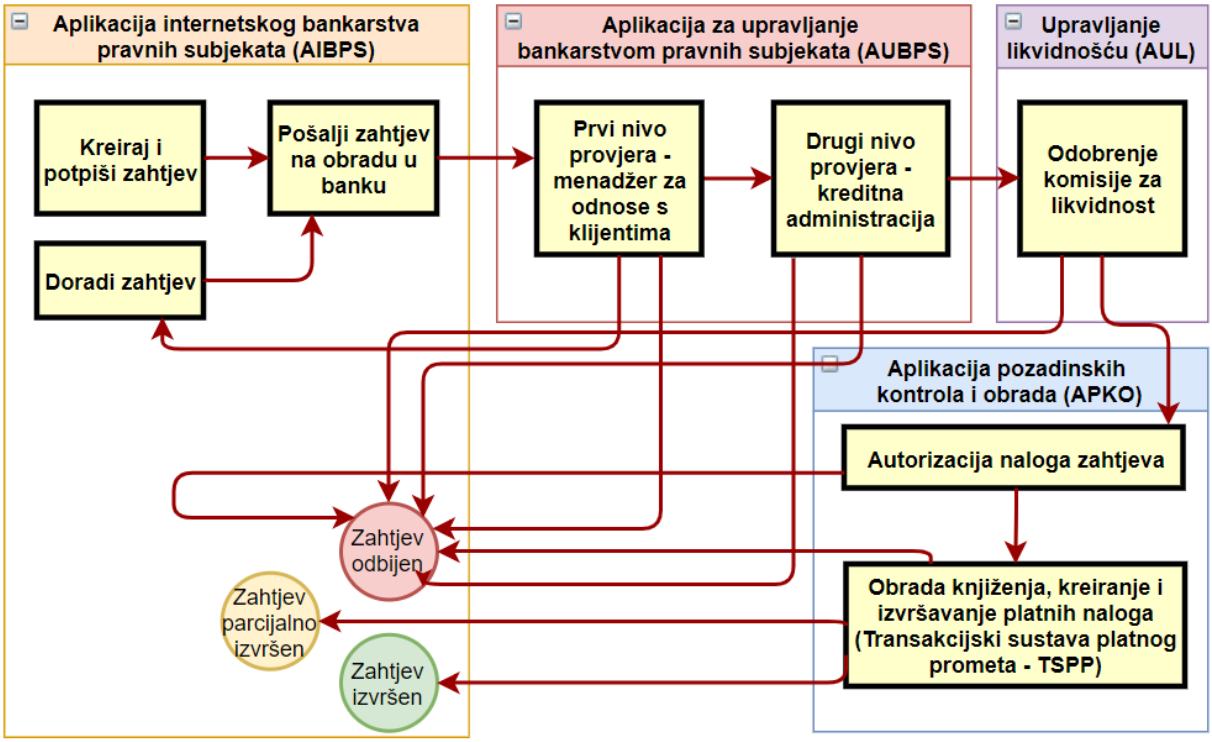
Temeljem iteracija izloženih u poglavlju 5.8 opisat će se izrada programske podrške zadavanju zahtjeva od strane klijenta, te obradi zahtjeva od strane službenih osoba, na primjeru procesa ZKK. Provedba i rezultat svake od navedenih iteracija u detalje je opisan u nastavku.

6.3.1 Definicija i otkrivanje karakteristika procesa

Nakon provođenja prve iteracije (poglavlje 5.8.1) sakupljene su relevantne informacije te su doneseni zaključci o poslovnom procesu koji postavljaju temelj za daljnji razvoj rješenja.

Proces se modelira s ciljem pružanja sistematiziranog i centraliziranog uvida u cjelokupni tok te detalje pojedinih aktivnosti. Želja je omogućiti centraliziranu interakciju i automatizaciju PP-a, postaviti temelje za lakše praćenje, mjerenje i optimiziranje pojedinih dijelova procesa.

U razgovoru sa sudionicima te vlasnikom procesa identificirano je ukupno 4 naslijeđenih sustava koji sudjeluju u izvršavanju procesa, utvrđene su aktivnosti procesa te je ugrubo skiciran tok procesa (Slika 6.1).



Slika 6.1 Pojednostavljeni tok PP-a ZKK-a

Aplikacija internetskog bankarstva pravnih osoba (AIBPS) ima ulogu kreiranja instance procesa sukladno akcijama klijenta koji kreiraju ZKK. AIBPS također zaduženoj osobi (službeniku u banci) dodjeljuje prvu aktivnost unutar procesa. Aplikacija za upravljanje bankarstvom pravnih subjekata (AUBPS) odrađuje najvažniji i najveći dio posla u procesu ZKK. Kroz dvije razine odobrenja, službenici provjeravaju ispravnost zadanih zahtjeva, priloženu dokumentaciju, svrhe isplate, zadane instrukcije za isplatu, itd., a po potrebi zahtjev vraćaju klijentu na doradu. Sljedeći korak je odobrenje komisije za likvidnost kroz aplikaciju za upravljanje likvidnošću (AUL). Zatim, slijedi autorizacija instrukcija plaćanja i pokretanje pozadinskih obrada kroz aplikaciju za pozadinske kontrole i obrade (APKO). Pozadinske obrade kreiraju platne naloge za isplatu sredstva klijentu te ih zatim izvršavaju uz pomoć transakcijskog sustava platnog prometa (TSPP). Izvršavanje kroz TSPP pokreće se automatikom iz APKO u sklopu pozadinskih obrada, traje zanemarivo kratko i prikazano je kao dio aktivnosti APKO.

Temeljem aktivnosti procesa, otkriveno je 5 osnovnih grupa korisnika koji sudjeluju u procesu. Prvu grupu čini korisnik AIBPS koji kreira zahtjev. U AUBPS, dvije grupe službenika rade odobrenja (RM-ovi te kreditni administratori). Zatim, član komisije za likvidnost odobrava isplatu sredstva, nakon čega slijede zadaci autorizacije te izvršavanja platnih instrukcija klijenta od strane službenika iz grupe osoba ovlaštenih za rad u APKO.

Grananje u BPMA radi se temeljem statusa zahtjeva te je stoga potrebno definirati i sistematizirati statuse zahtjeva koji odgovaraju statusima zahtjeva iz naslijeđenih sustava.

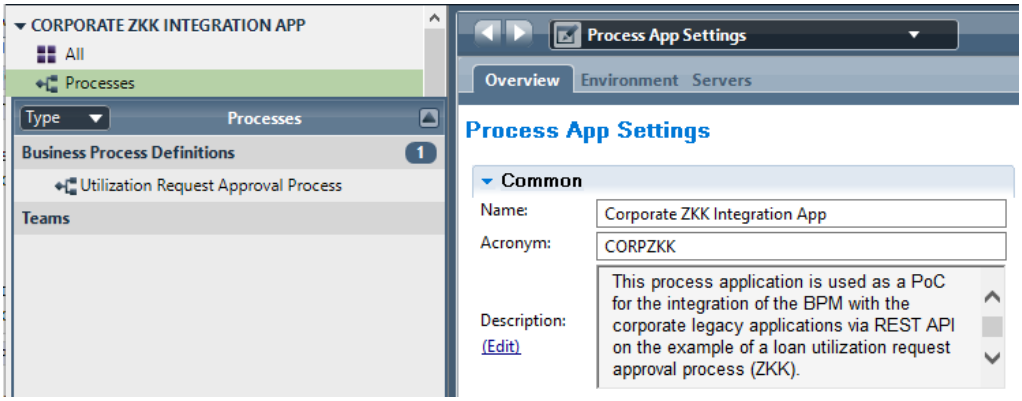
Moguće je uvesti nove, dodatne, statuse koji bi se koristili samo u BPMA. Ovaj princip koristan je ako unutar BPMA postoje dodatne aktivnosti koje nije potrebno odrađivati u sklopu naslijeđenih sustava, a kreirane su sa svrhom poboljšanja kvalitete PP-a (npr. dodatne kontrole, aktivnosti informiranja klijenta, itd.). U ovom radu, BPMA statusi pratit će statuse zahtjeva u naslijeđenim sustavima, nema uvođenja novih statusa.

Preslikavanja statusa razrađena su u tablici u nastavku. Podebljani nazivi statusa predstavljaju završne statuse.

Tablica 6.1 Razrada statusa ZKK-a

Šifra i naziv statusa	***Mogući statusi***	***Aplikacija prijelaza***
73: Created by client	51,49	AIBPS
51: Not yet signed by client	52, 53, 54	AIBPS
49: Deleted by client	-	AIBPS
52: Left signed by client	53, 54	AIBPS
53: Right signed by client	54, 40	AIBPS
54: Returned to update (by bank or client)	52, 53, 54	AIBPS / AUBPS
72: Revoked by client	-	AIBPS
40: Received from client	55, 72, 54, 45	AIBPS
55: Approved by Relationship Manager	56, 72, 54, 45	AUBPS
56: Approved by Credit Administrator	64, 45	AUBPS
64: Approved by Liquidity Managers	65	AUL
65: In back-office processing	66, 45, 47	APKO
66: In realization	45, 70, 47	APKO
45: Rejected	-	AUBPS / APKO
70: Partially realized	-	APKO
47: Realized (Completed)	-	APKO

Sljedeći korak ove iteracije predstavlja pripremu za sljedeće iteracije. U IBPM alatu definira se artefakt procesne aplikacije, određuje se naziv BPMA i naziv PP-a te se dodaju metapodaci i detalji o procesnoj aplikaciji (Slika 6.2).



Slika 6.2 Definicija BPMA s definicijom PP-a ZKK u IBPM alatu

6.3.2 Modeliranje toka procesa

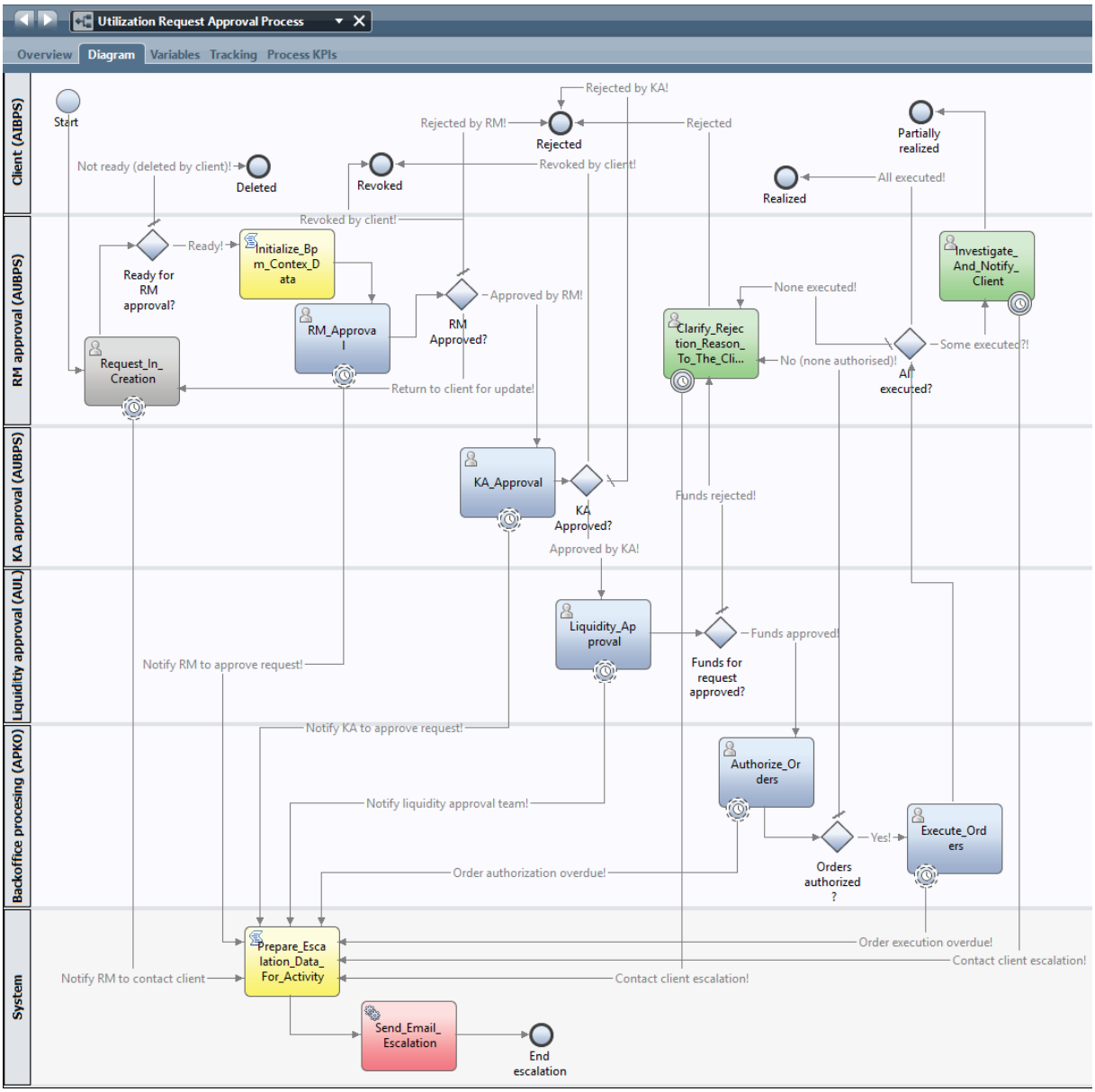
Glavna zadaća ove iteracije (poglavlje 5.8.2) je modelirati BPMN model PP-a na najvišem nivou te razraditi puteve pozitivnih i negativnih ishoda procesa.

Proces ZKK modeliran je u jednom polju kroz 6 staza te uključuje 4 tima ovlaštenih osoba koji rade na aktivnostima procesa. Zasebno dodjeljivanje ovlaštenja za rad u BPMS-u pojedinim korisnicima kroz procesnu konzolu IBM BPM alata nije potrebno, jer je alat inicijalno konfiguriran da dodijeli ovlaštenja svim korisnicima iz centralnog sustava organizacije. Tablica 6.2 prikazuje definirane staze te njima pridružene timove prema razradi iz poglavlja 5.5.2.3. Oznaka *sistemska staza* definira stazu s akcijama izvođenim isključivo od strane postojećih sustava, bez ljudske interakcije, te stoga ne sadrži zaduženi tim.

Tablica 6.2 Definirane staze i pridruženi timovi procesa ZKK

Staza	**Zaduženi tim**	**Sistemska staza**
Client (AIBPS)	**-**	NE
RM approval (AUBPS)	***RelationshipManagers***	NE
KA approval (AUBPS)	***CreditAdministrators***	NE
Liquiditiy approval (AUL)	***LiquidityAdministrators***	NE
Backoffice procesing (APKO)	***BackOfficeProcessingTeam***	NE
System	**-**	DA

Temeljem pojednostavljenog toga procesa ZKK-a modeliran je DPP sa svim ključnim aktivnostima, putevima grananja te eskalacijama (Slika 6.3).



Slika 6.3 Dijagram toka procesa ZKK

Obavlja se inicijalna simulacija izvođenja procesa u sklopu koje se provjerava valjanost toka procesa izvođenjem fiktivnih scenarija procesa. Scenariji se definiraju uz pomoć izvornih podataka postavljenih na razini pojedinih aktivnosti, a simulacija izvođenja obavlja se kroz "Inspector perspektivu" IBM Process Designer-a.

Osim validacije toka procesa, druga faza simulacije uključuje detekciju vremena čekanja, potencijalnih uskih grla te podešavanje ključnih parametara procesa prije implementacije detalja procesa. U svrhu detekcije potrebno je utvrditi vremena trajanja pojedinih aktivnosti, definirati veličine i trošak zaduženih timova (poglavlje 5.5.2.3) te odrediti raspodjelu toka procesa, to jest statistiku raspodjele grananja u modeliranom procesu. Vremena trajanja odrede se na temelju iskustva osoba koje u procesu sudjeluju dok se veličina timova vidi iz broja ljudi

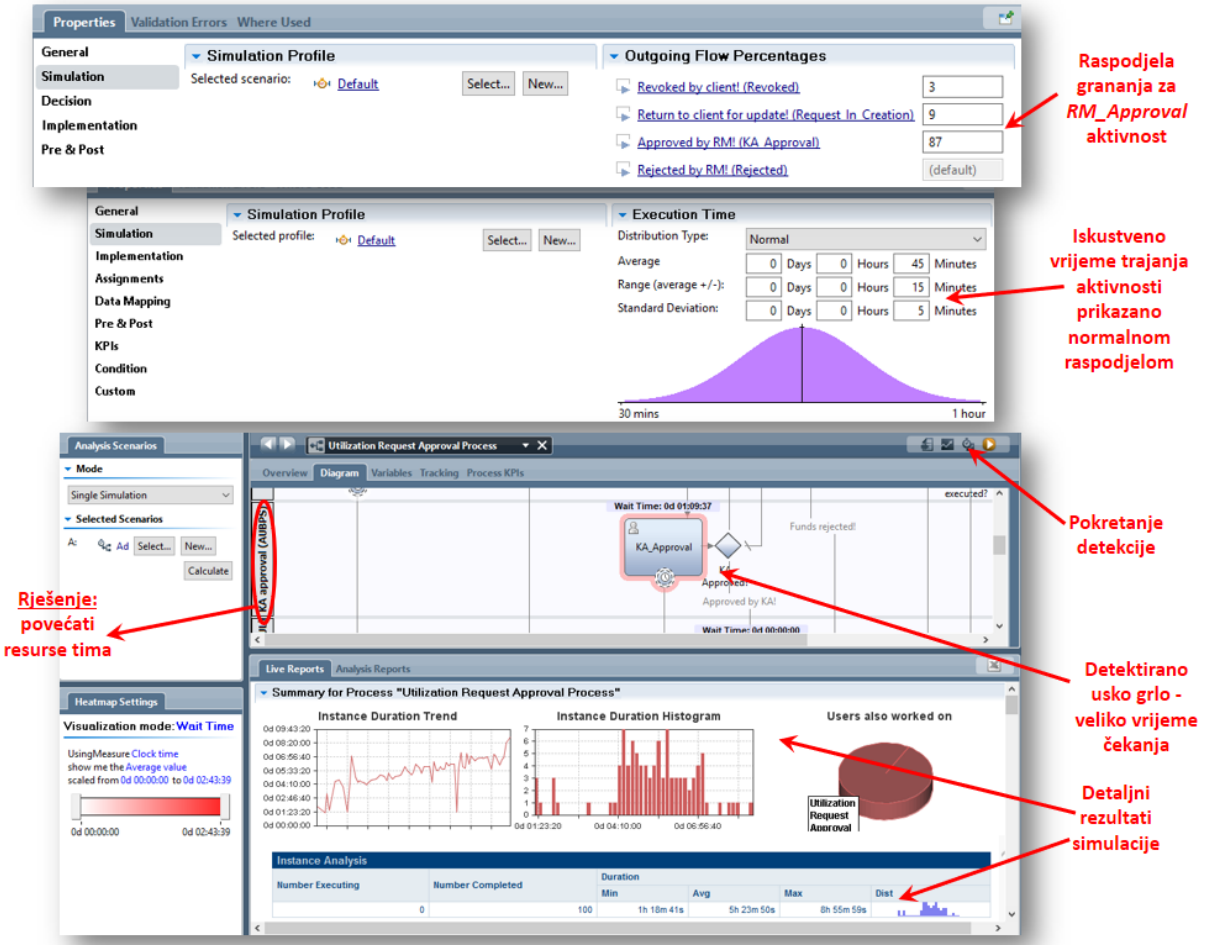
koji imaju ovlaštenja na pojedine aktivnosti procesa u naslijeđenim aplikacijama. Statistiku raspodjele grananja na određenim skretnicama određujemo analizom promjene statusa ZKK-a kroz povijesne podatke iz naslijeđenih aplikacija. Tablica 6.3 prikazuje analizu provedenu na uzorku od 1179 zahtjeva koji se nalaze u jednom od završnih statusa u naslijeđenoj okolini.

Tablica 6.3 Analiza grananja u toku procesa ZKK-a temeljem povijesnih podataka

Skretnica	**Relevantna promjena statusa (grana toka procesa)**	**Broj zahtjeva po grani skretnice**	**Postotak zahtjeva po grani skretnice**
Ready for RM approval?	* → 49: Deleted by client	45	3,50%
	* → 40: Received from client	**1240**	96,50%
RM Approved?	* → 72: Revoked by client - ***during RM approval***	36	2,90%
	* → 54: Returned to update ***- only the ones that are later approved by RM***	106	8,55%
	* → 45: Rejected – ***by RM***	16	1,29%
	* → 55: Approved by Relationship Manager	**1082**	87,26%
KA Approved?	* → 72: Revoked by client - ***during KA approval***	4	0,37%
	* → 45: Rejected – ***by KA***	25	2,31%
	* → 56: Approved by Credit Administrator	**1053**	97,32%
Funds for request approved?	* → 45: Rejected – ***by* Liquidity Managers**	2	0,19%
	* → 64: Approved by Liquidity Managers	**1051**	99,81%
Orders authorised?	* → 45: Rejected – ***by Backoffice team***	7	0,67%
	* → 65: In back-office processing	**1044**	99,33%
All orders executed?	* → 45: Rejected - **all orders rejected**	9	0,86%
	* → 70: Partially realized - **some orders rejected**	12	1,15%
	* → 47: Realized (Completed)	**1023**	97,99%

Slika 6.4 prikazuje postupak unosa prethodno otkrivenih vrijednosti u modelirani DPP te proces pokretanja simulacije, detekciju i reakciju na otkrivene nedostatke u procesu. Zbog mogućnosti unosa samo cjelobrojnih vrijednosti postotaka u simulacijskim svojstvima IBPM alata, vršimo zaokruživanje dobivenih vrijednosti, a za postotak manji od 1% zaokružujemo na 1% kako bi alat vršio izračun toka kroz sve grane skretnice. Postoje različiti scenariji simulacija (više u poglavlju 7), a za ovu iteraciju iskoristit ćemo jednokratnu simulaciju sa 100 instanci procesa, pri čemu je vremenski razmak između pokretanja instanci procesa u skladu s

normalnom razdiobom. Definirani su parametri normalne razdiobe, srednja vrijednost od 15 minuta uz standardnu devijaciju od 2 minute. Tako na primjer možemo očekivati da će razmak između pokretanja procesa biti u intervalima 15±2, 15±4 odnosno 15±6 minute u 68.25%, 95.45% odnosno 99.73% slučajeva. Definirana jednokratna simulacija računa vremena čekanja na aktivnostima, prosječno trajanje instanci procesa, utrošak resursa, te daje sveobuhvatan izvještaj i prikaz rezultata.



Slika 6.4 Priprema podataka i izvođenje simulacije

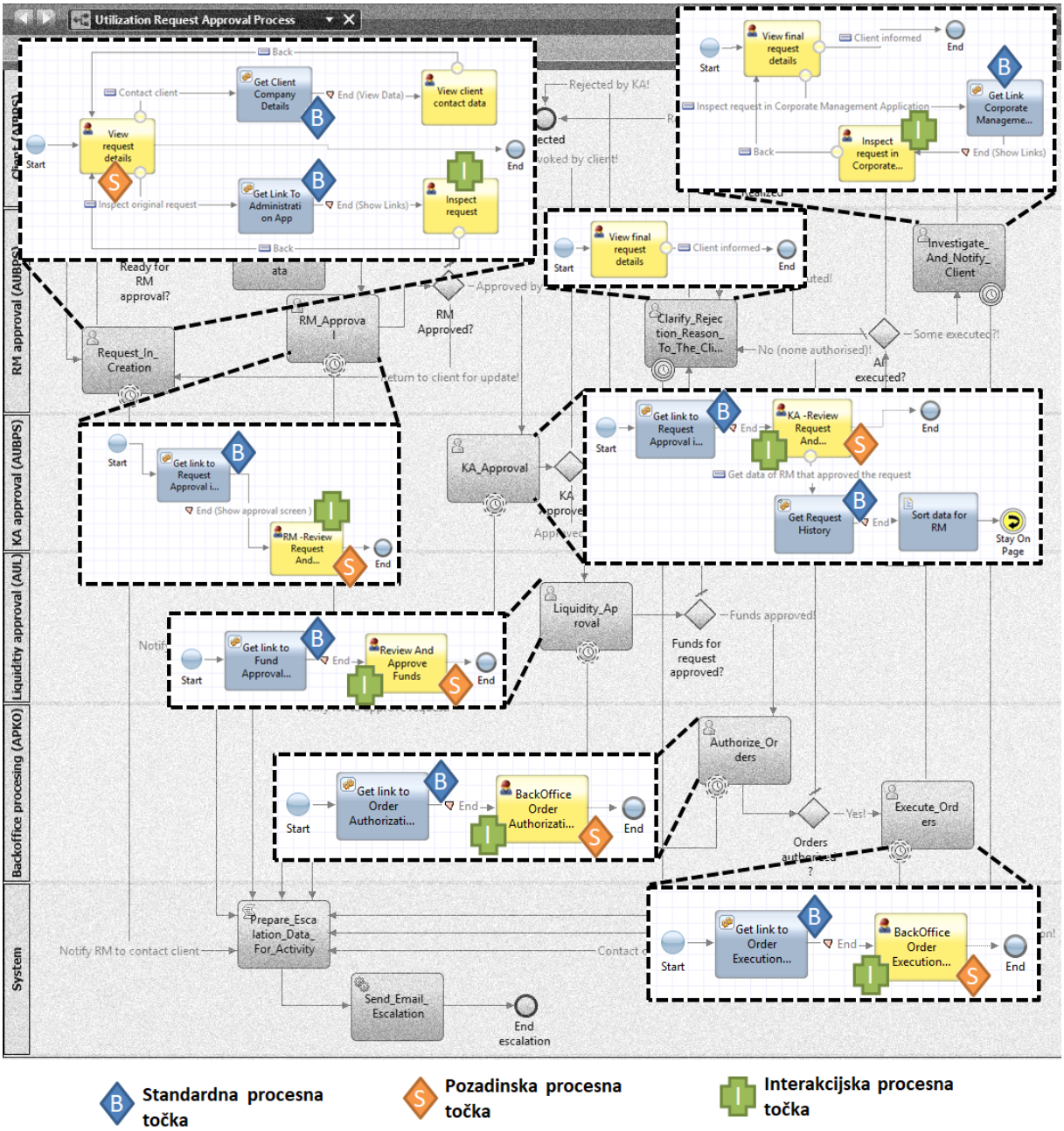
6.3.3 Razrada integracije s naslijeđenim aplikacijama

U sklopu ove iteracije (poglavlje 5.8.3) identificiramo poslovne događaje naslijeđenog procesa koji zahtijevaju postojanje standardnih procesnih točaka. Ti događaji u pravilu ukazuju na promjenu statusa zahtjeva vezanog uz PP i/ili određene izmjene nad podacima procesa u naslijeđenim sustavima. Temeljem pronađenog događaja definiraju se akcije koje je potrebno obaviti prema BPMA kako bi stanje u naslijeđenim sustavima i BPMA bilo konzistentno. Tablica 6.4 prikazuje otkrivanje događaja te definiranje pripadajućih akcija prema BPMA na primjeru procesa ZKK uz pomoć prethodne razrade statusa ZKK-a (Tablica 6.1).

Tablica 6.4 Identificirani događaji i odgovarajuće akcije

Zadužena aplikacija	Poslovni događaj	Promjena statusa	Akcije prema BPMA
AIBPS (klijent)	Kreiranje zahtjeva	→73	• Kreiranje instance procesa ZKK s podacima definiranim od strane klijenta • Dodjeli aktivnost *Request_In_Creation* RM-u zaduženom za klijenta
	Ažuriranje podataka zahtjeva	Nema	• Ažuriranje podataka instance procesa
	Brisanje zahtjeva	54/73→49	• Ažuriraj podatke instance procesa i završi aktivnost *Request_In_Creation*. • Instanca procesa završava sa statusom "***Deleted by client***"
	Slanje zahtjeva na potpis	73/54→51	• Ažuriranje podataka instance procesa
	Lijevi potpis zahtjeva	54/73→52	• Ažuriranje podataka instance procesa
	Desni potpis zahtjeva	52/54/73→53	• Ažuriranje podataka instance procesa
	Klijent vrati zahtjev na doradu	52/53→54	• Ažuriranje podataka instance procesa
	Pošalji zahtjev na verifikaciju	53→40	• Ažuriraj podatke instance procesa i završi aktivnost *Request_In_Creation*. • Dodjeli aktivnost *RM_Approval* RM-u zaduženom za klijenta
	Opozovi zahtjev	40/55→72	• Ažuriraj podatke instance procesa i završi aktivnost u kojoj je trenutno proces (*RM_Approval* / *KA_Approval*). • Instanca procesa završava sa statusom zahtjeva "***Revoked by client***"
AUBPS	RM vraća zahtjev na doradu klijentu	40 → 54	• Ažuriraj podatke instance procesa i završi aktivnost *RM_Approval*
	RM odbija zahtjev	40 → 45	• Ažuriraj podatke instance procesa i završi aktivnost *RM_Approval* • Instanca procesa završava sa statusom "**Rejected**"
	RM odobrava zahtjev	40 → 55	• Ažuriraj podatke instance procesa i završi aktivnost *RM_Approval*
	KA odbija zahtjev	55 → 45	• Ažuriraj podatke instance procesa i završi aktivnost *KA_Approval* • Instanca procesa završava sa statusom "**Rejected**"
	KA odobrava zahtjev	55 → 56	• Ažuriraj podatke instance procesa i završi aktivnost *KA_Approval*
AUL	Komisija za likvidnost odobrava sredstva za zahtjev	56 → 64	• Ažuriraj podatke instance procesa i završi aktivnost *Liquidity_Approval*
	Komisija za likvidnost ne odobrava sredstva za zahtjev	56 → 45	• Ažuriraj podatke instance procesa i završi aktivnost *Liquidity_Approval* • Instanca procesa završava sa statusom "**Rejected**"
APKO	Djelatnik autorizira sve naloge zahtjeva ili neke autorizira, a neke odbija	64 → 65	• Ažuriraj podatke instance procesa i završi aktivnost *Authorize_Orders*
	Djelatnik odbije sve naloge zahtjeva	64 → 45	• Ažuriraj podatke instance procesa i završi aktivnost *Authorize_Orders* Instanca procesa završava sa statusom "**Rejected**"
	Dolazi povratna informaciju o izvršenju instrukcije	65 →66/70/47/45 ili 66→70/47/45	• Ažuriraj podatke instance procesa, to jest status instrukcije procesa • Ako su sve instrukcije u završnom statusu završi aktivnost *Execute_Orders* i postavi završni status zahtjeva: - **Realized (Completed):** ako su sve instrukcije uspješno izvršene - **Partially realized**: ako su neke instrukcije uspješno izvršene - **Rejected**: Ako nijedna instrukcija nije uspješno završila

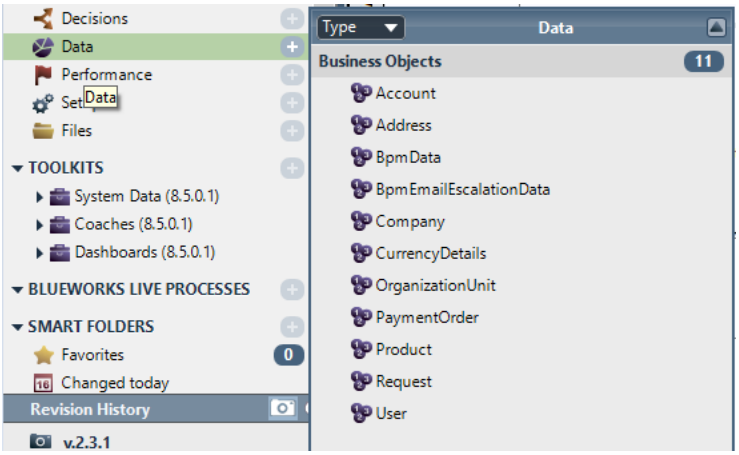
Osim standardnih identificiraju se pozadinske procesne točke karakteristične za BPMA te interakcijske točke. Na prethodno definiranom DPP-u ZKK (Slika 6.3) određuje se lokacija navedenih procesnih i interakcijskih točaka te se definiraju detalji implementacije ljudskih aktivnosti procesa (Slika 6.5). Detalji razrade komunikacije i interakcije u navedenim točkama fokus su pete iteracije (6.3.5).



Slika 6.5 Detalji implementacije aktivnosti te identifikacija procesnih i interakcijskih točaka

6.3.4 Izgradnja poslovnog modela i korisničkih sučelja

Temeljem kratke konzultacije s poslovnim korisnicima, koji definiraju osnovni set podataka potrebnih za prikaz na BPMA, izgrađen je BPMA poslovni model, to jest definirani su BPMAPO u IBPM alatu (Slika 6.6). *BpmData* BPMAPO služi za čuvanje tehničkih podataka potrebnih za BPMA, dok ostali objekti služe za čuvanje potrebnih poslovnih podataka, te su zapravo preslika podatkovnog modela BPMKS-a.



Slika 6.6 Podatkovni model BPMA

Tablica 6.5 prikazuje razinu ostvarenog pojednostavljenja podatkovnog modela BPMA spram podatkovnog modela naslijeđenih sustava, konkretno AIBPS i AUBPS koje dijele podatkovni model te predstavljaju najkompleksniji dio procesa. Usporedba je napravljena temeljem broja polja klasa te tipu polja klasa (složeni / jednostavni).

Tablica 6.5 Usporedba kompleksnosti podatkovnog modela naslijeđenih sustava i BPMA

Naslijeđeni sustavi		**BPMA/BPMKS**	
Klasa	Broj polja (složeni / jednostavni tip)	**BPMAPO**	Broj polja (složeni / jednostavni tip)
Request	49 (21 / 28)	*Request*	9 (6 / 3)
Product	158 (80 / 78)	*Product*	4 (1 / 3)
Company	62 (30 / 32)	*Company*	12 (3 / 9)
User	78 (28 / 50)	*User*	7 (0 / 7)
Account	83 (29 / 55)	*Account*	1 (0 / 1)
PaymentOrder	91 (26 / 65)	*PaymentOrder*	7 (3 / 4)
CurrencyDetails	7 (0 / 7)	*CurrencyDetails*	3 (0 / 3)
OrganizationUnit	11 (3 / 8)	*OrganizationUnit*	2 (0 / 2)
Address	22 (1 / 21)	*Address*	4 (0 / 4)

Određene su opće smjernice koje će se koristiti kod prezentacije podataka, izgradnje sučelja te definicije procesno relevantnih varijabli:

- Poslovni podaci prikazani na BPMA sučeljima isključivo su za čitanje (eng. read only), izbjegavamo bilo kakav unos i promjenu podataka koristeći BPMA sučelja. Ažuriranje podataka vrši se samo i isključivo kroz sučelja naslijeđenih aplikacija.
- Sukladno tehničkoj specifikaciji iz poglavlja 5.5.2.1 izgrađuju se *CV* komponente za prikaz podatka pojedinog BPMAPO (Tablica 6.6). Jedan BPMAPO može imati više *CV* komponenti, koje prikazuju različitu razinu detalja, a *CV* komponente mogu sadržavati druge *CV* komponente.
- Zbog jednostavnosti na razini modeliranog procesa definirana je varijabla *request* tipa *Request* BPMAPO koja će sadržavati sve poslovno relevantne podatke, a zapravo sliči strukturi klase zahtjeva iz naslijeđenih sustava. Dodatna varijabla *bpmData* tipa *BpmData* sadrži tehničke informacije potrebne u procesu, dok varijabla *bpmEmailEscalationData* tipa *BpmEmailEscalationData* sadrži podatke potrebne za slanje email eskalacijskih obavijesti.

Tablica 6.6 Popis CV komponenti izgrađenih za prikaz detalja BPMAPO

BPMAPO	**CV komponenta za prikaz podataka**	**Koristi CV komponente**
Request	***RequestCV***	***ProductCV, UserBasicCV, UserExtendedCV, EmployeeCV, CompanyBasicCV, PaymentOrderCV, AddressCV, OrganizationUnitCV***
Product	***ProductCV***	**-**
Company	***CompanyBasicCV***	**-**
	CompanyExtendedCV	***UserExtendedCV, AddressCV, OrganizationUnitCV***
User	***UserBasicCV, UserExtendedCV, EmployeeCV***	**-**
Account	**-**	**-**
PaymentOrder	***PaymentOrderCV***	**-**
CurrencyDetails	**-**	**-**
OrganizationUnit	***OrganizationUnitCV***	**-**
Address	***AddressCV***	**-**

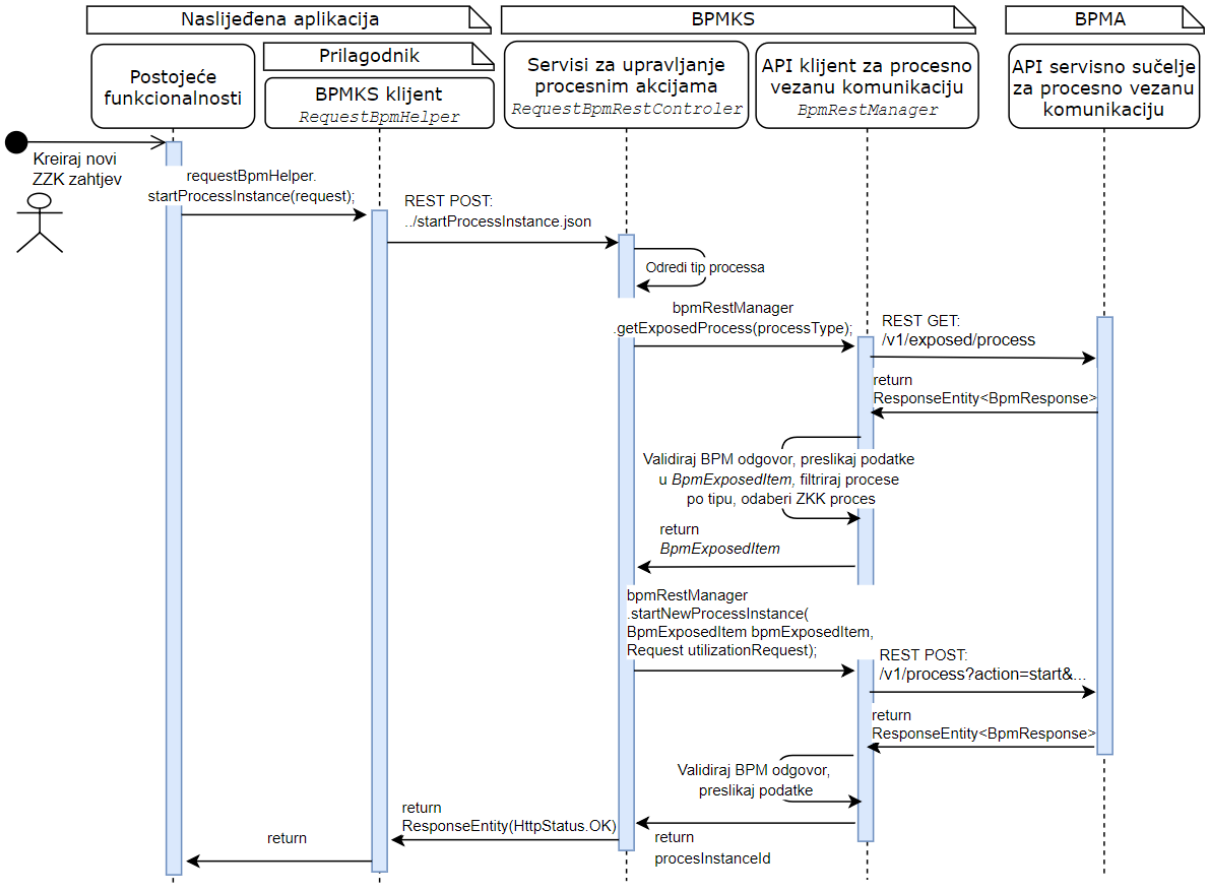
Zadnji korak uključuje parcijalnu simulaciju, provjeru izgleda i ispravnosti prikaza podataka za svaki od kreiranih CV sučelja. U svrhu simulacije koriste se imitirani, (engl. mock) podaci koji se postave kao izvorne (engl. default) vrijednosti na razini deklariranog BPMAPO ili se definiraju uz pomoć skriptirane sistemske aktivnosti koja čini prvi korak modeliranog procesa i/ili *ljudskog servisa*. Po definiciji podataka BPMAPO, funkcionalnost IBM Process Designer alata koja omogućava pokretanje odvojenih simulacija pojedinačnih *ljudskih servisa*, koristi se u svrhu zasebne provjere izgleda i ispravnosti podataka svake *coach* komponente te elementarnih *CV* sučelja.

6.3.5 Implementacija procesnih i interakcijskih točaka

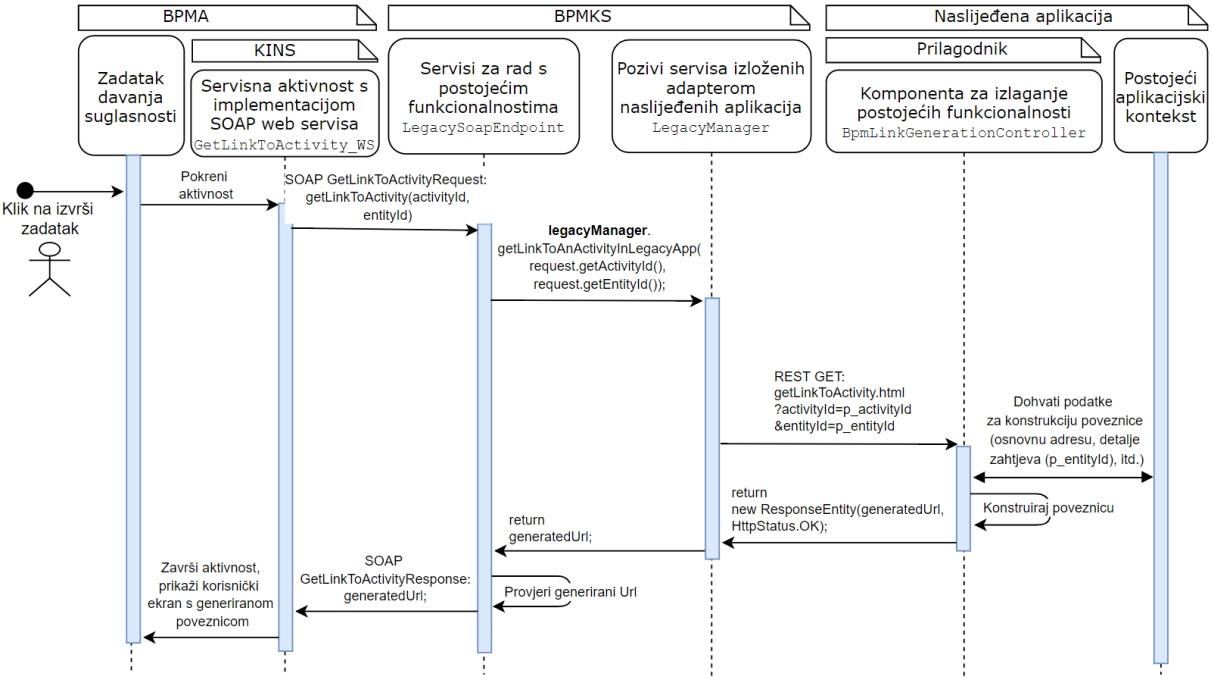
Prilikom svake interakcije s BPMA (u standardnim procesnim točkama) naslijeđena okolina mora znati koja instanca procesa odgovara određenom zahtjevu iz naslijeđene aplikacije. Sukladno tome, te prema razradi problema u poglavlju 5.7, potrebno je definirati identifikatore koji će omogućiti odgovarajuću vezu.

U svrhu podrške dohvata instanci procesa ZKK koristit će se 3 vrste identifikatora iz naslijeđenih sustava: broj zahtjeva (*requestNumber*) kojeg koriste AIBPS i AUBPS, broj ugovora kredita iz ZKK (*contractNumber*) kojeg koristi AUL te složeni identifikator (*paymentOrdersIds*) sastavljen od brojeva naloga (npr. "*43256|43262|...*") koji je potreban u završnom dijelu procesa (APKO). Polja *requestNumber* i *contractNumber* dio su ulaznog *request* BPMAPO objekta, dok *paymentOrdersIds* kreiramo kao zasebno polje unutar *BpmData* BPMAPO. Kod slanja zahtjeva na prvi korak verifikacije, skriptirana aktivnost *Initialize_Bpm_Contex_Data* (Slika 6.3) postavlja vrijednost *paymentOrdersIds* polja na opisani format, temeljem vrijednosti identifikatora svih naloga ZKK-a.

Zatim, se temeljem definicije iz poglavlja 4.4.1 i postupka razrađenog u poglavlju 5.6 implementiraju detalji komunikacije u procesnim točkama. Kao primjer, u nastavku je prikazan dijagram slijeda u standardnoj procesnoj točci prilikom kreiranja nove instance procesa (Slika 6.7). Dodatno, Slika 6.8 prikazuje dijagram komunikacije u *pozadinskoj* procesnoj točci prilikom dohvata poveznice na aktivnosti naslijeđene aplikacije.



Slika 6.7 Dijagram kreiranja nove instance procesa u standardnoj procesnoj točci

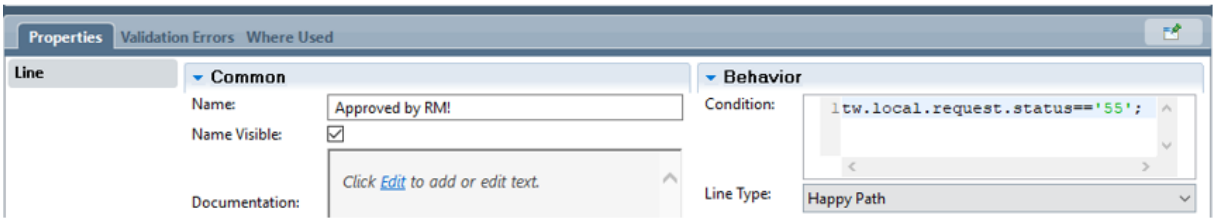


Slika 6.8 Dijagram dohvata poveznice na aktivnosti naslijeđenih aplikacija

Simulacijom u ovoj iteraciji otkrivamo pogreške kod dohvata podataka iz naslijeđenih sustava, probleme s manipulacijom BPMA iz naslijeđenih sustava, te nekonzistentno ponašanje u interakcijskim točkama.

6.3.6 Integracija aktivnosti s tokom procesa i razrada detalja implementacije

Po završetku određene aktivnosti, naslijeđene aplikacije u standardnim procesnim točkama ažuriraju statuse zahtjeva temeljem kojih se radi grananje u procesu. Ovisno o poslovnim događajima, te prema razradi prijelaza statusa zahtjeva (Tablica 6.1 i Tablica 6.4), svaka od grana određena je statusom zahtjeva. Slika 6.9 prikazuje definiciju uvjeta grananja na BPMN elementu protoka (skretnici) temeljem statusa ZKK-a koristeći primjer "*Approved by RM!*" grane koja spaja aktivnosti *RM_Approval* i *KA_Approval*.



Slika 6.9 Definiranje uvjeta grananja na skretnici

Zatim, dorađuju se detalji *Clarify_Rejection_Reason_To_The_Client* i *Investigate_And_Notify_Client* aktivnosti. Navedene aktivnosti detaljnije su opisane u poglavlju 6.4.4, a služe za podizanje kvalitete usluge u vidu komunikacije s klijentom u slučaju negativnog ishoda izvršenja zahtjeva. Ove aktivnosti se ne zatvaraju od strane naslijeđenih sustava, već ih zatvara korisnik iz Procesnog Portala nakon što procijeni situaciju i po potrebi obavijesti klijenta. Budući da se u ovom koraku ZKK već nalazi u završnom statusu, ako RM u odgovarajućem vremenu ne preuzme zadatak, definirana eskalacija nad aktivnošću ujedno aktivnost i zatvara kako bi se proces priveo kraju bez dodatnih čekanja.

Dodatno, definiramo detalje implementacije eskalacija, pripremu podataka *JavaScript* programskim jezikom unutar *Prepare_Escalation_Data_For_Activity* aktivnosti, te izgled i sadržaj eskalacijskih email poruka tehnikama opisanim u poglavlju 5.5.2.6.

6.3.7 Otkrivanje pogrešaka i iznimaka u procesu uz finalnu verifikaciju

Završna iteracija (opisana u poglavlju 5.8.7) rezultira finalnim provjerama kompletnog toka PP-a kroz različite poslovne scenarije, te ispravak pronađenih pogrešaka i nekonzistentnosti. Izrađuju se izvještaji, kao što je primjerice izvještaj o eskalacijama prikazan u poglavlju 6.4.5 (Slika 6.13).

6.4 Inovacije uvedene u proces

Integracijom BPMA s naslijeđenim procesom ugovaranja usluga, koji je prikazan na primjeru zadavanja i obrade ZKK-a, uklonjeni su nedostaci procesa navedeni u poglavlju 6.2. Dodatno, uvedeno je više inovacija u proces koje su ukratko objašnjene u nastavku.

6.4.1 Preventivne eskalacije putem pasivnih aktivnosti

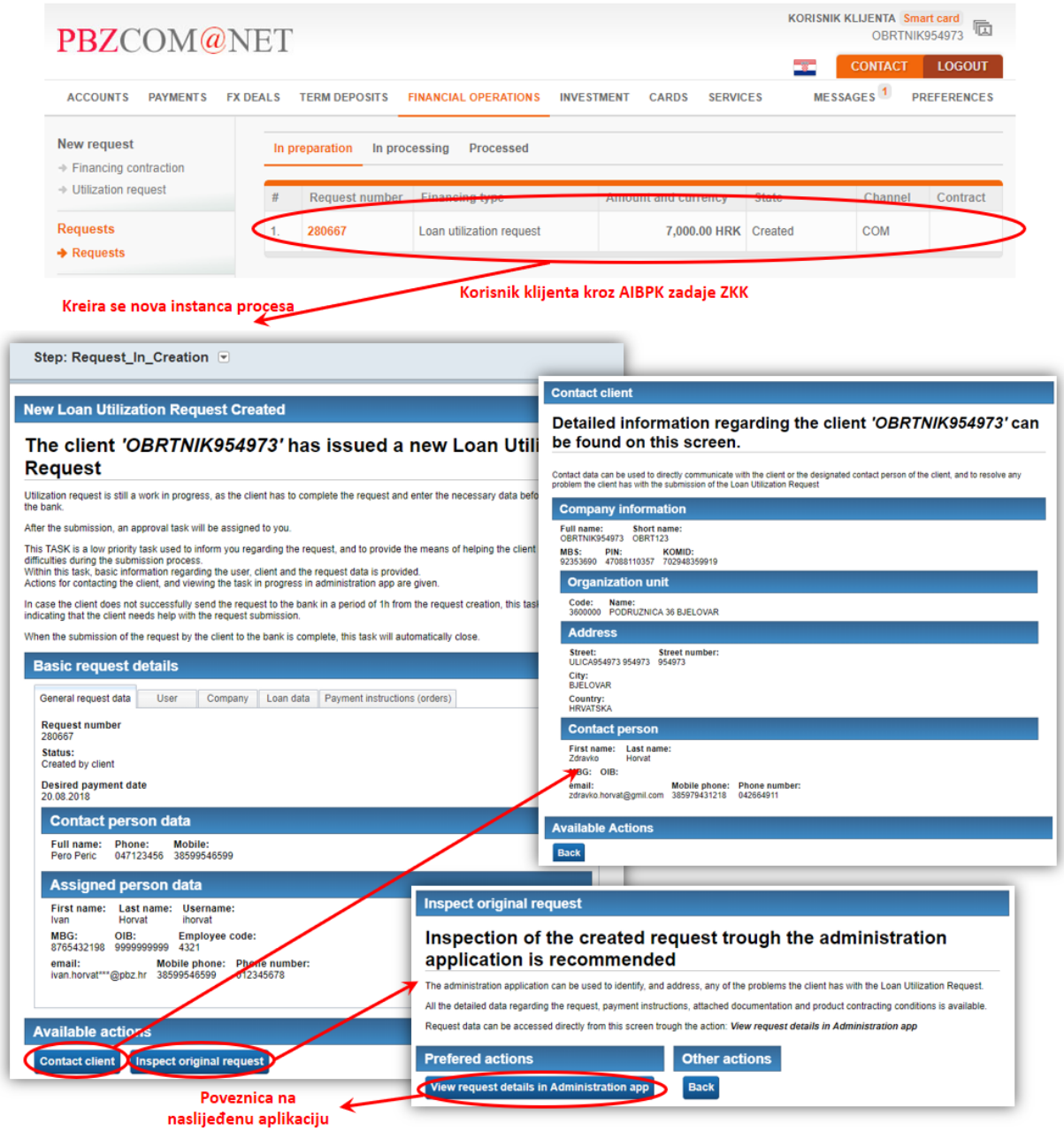
Kod zadavanja zahtjeva za usluge, klijent nakon kreiranja zahtjeva treba odraditi niz koraka u AIBPS da bi zahtjev poslao u banku na obradu. Ti koraci uključuju aktivnosti kao što su slanje na potpisivanje, potpisivanje hardverskim ili softverskim tokenima te certifikatima od strane jedne ili više osoba (lijevi i desni potpisnik - ovisno o ovlaštenjima), digitalno potpisivanje dokumenata (ugovora o usluzi) kvalificiranim FINA certifikatima od strane osoba koje su zakonski ovlaštene za zastupanje firme klijenta, itd.

Ponekad, osobito kod novih usluga, klijenti nisu dovoljno informirani o procesu i/ili nemaju sve preduvjete za njegovo provođenje (odgovarajuće certifikate i autorizacijske uređaje). Dodatno, zbog kompleksnosti sustava ili specifičnosti slučaja dotičnog klijenta može i doći i do nepredviđene pogreške u samoj AIBPS. U tom slučaju klijent "zapne" sa slanjem zahtjeva u banku na obradu, te prijavljuje problem službenim kanalima. Kod usluga koje je potrebno brzo realizirati (primjerice u, ZKK) prijava problema standardnim kanalima, zbog relativne sporosti nije učinkovita. Iz navedenih razloga osmišljen je model preventivnih eskalacija u sklopu pasivnih aktivnosti definicije PP-a.

Naziv "pasivna aktivnost" označava aktivnost PP-a koja služi za praćenje rada korisnika i preventivno djelovanje u vidu pomoći klijentu te ju zadužena osoba ne mora eksplicitno preuzeti i zatvoriti. Programska podrška ugrađena u naslijeđene aplikacije sama zatvara navedenu pasivnu aktivnost kada zahtjev u naslijeđenoj aplikaciji poprimi odgovarajući status, to jest biva poslan u banku na odobrenje. Nad pasivnim aktivnostima ugrađene su eskalacije koje podižu razinu važnosti aktivnosti te šalju e-mail notifikaciju zaduženim osobama po detekciji da klijent ima problema s kreiranjem i slanjem zahtjeva. Implementacija pasivne aktivnosti "*Request_In_Creation*" iz procesa ZKK-a ukratko je opisana u nastavku.

Radeći u AIBPS klijent kreira ZKK. U trenutku inicijalnog spremanja ZKK-a, AIBPS u pozadini, preko prilagodnika i BPMKS-a, pokreće instancu odgovarajućeg PP-a na BPM okruženju. Prva aktivnost procesa (*Request_In_Creation*) predstavlja pasivnu aktivnost te se dodjeljuje službeniku (RM-u) zaduženom za klijenta. Aktivnost je realizirana ljudskim zadatkom, a dijagram toka aktivnosti, te mogućih akcija, prikazan je u sklopu Slika 5.25.

Preuzimanjem aktivnosti na procesnom portalu, RM dobiva podatke o kreiranom zahtjevu te su mu ponuđene akcije u vidu preventivnog djelovanja i ukazivanja pomoći klijentu. Akcije uključuju pregled zahtjeva u administratorskoj naslijeđenoj aplikaciji (sustav automatski kreira poveznicu na detalje zahtjeva u naslijeđenoj aplikaciji) te mogućnosti kontakta klijenta (sustav automatski dohvaća kontakt podatke o klijentu iz naslijeđenih sustava). Slika 6.10 prikazuje opisana sučelja.



Slika 6.10 Sučelja i akcije Request_In_Creation pasivne aktivnosti kod kreiranja ZKK-a

Ako životni vijek pasivne aktivnosti prelazi određenu vrijednost (koja je predefinirana ovisno o tipu i vrsti PP-a) to je znak da klijent ima problema sa zahtjevom, te se generira eskalacija na zaduženog RM-a u svrhu potreba preventivnog djelovanja. Kada klijent uspješno

pošalje ZKK u banku na verifikaciju, *Request_In_Creation* pasivni zadatak automatski se zatvara (prilagodnik naslijeđene aplikacije poziva *finishCurrentTaskAndUpdateBusinessData(...)* ili *finishTaskOfTypeAndUpdateBusinessData(...)* REST *endpoint* BPMKS-a).

Temeljem analize pasivnih procesa izvlači se zaključak o teškoćama koje klijenti imaju kod zadavanja zahtjeva, te se može raditi na optimizaciji toka procesa i intuitivnosti zadavanja zahtjeva u naslijeđenim aplikacijama.

6.4.2 Modifikacija toka poslovnog procesa u BPM alatu

Predloženi model integracije omogućava promjenu toka procesa izmjenom redoslijeda aktivnosti modeliranog PP-a izravno u BPM alatu za slučajeve gdje je ta izmjena podržana u naslijeđenim sustavima.

Uzmimo za primjer naslijeđeni PP s 3 slijedne aktivnosti (A→B→C), koji je prema predloženom modelu integracije povezan s BPM rješenjem. Stoga, svaka od aktivnosti odrađuje se u određenoj naslijeđenoj aplikaciji, a BPMA radi preusmjeravanje na tu naslijeđenu aplikaciju. Pretpostavimo da se javi poslovna potreba zamjene redoslijeda aktivnosti B i C, dakle PP će izgledati A→C→B. Navedenu izmjenu moguće je napraviti zamjenom redoslijeda aktivnosti modeliranog procesa u BPMA. Međutim, uvjet koji mora biti zadovoljen (kako bi izmijenjeni proces bio funkcionalan) je da implementacija u naslijeđenim sustavima omogućava izvršavanje aktivnosti B prije aktivnosti C. Aktivnosti moraju biti neovisne, to jest izvršavanje aktivnosti C ne smije za preduvjet imati uspješan završetak aktivnosti B.

Stoga, modifikacije toka procesa, bez izmjena po naslijeđenim sustavima, moguće su u mjeri u kojoj to dozvoljava implementacija aktivnosti procesa u naslijeđenim sustavima. Prikazanim postupkom ostvarena je karakteristika "***mogućnost manjih, jednostavnijih izmjena tokova procesa kroz BPM alate***" vlastitog modela integracije navedena u poglavlju 3.3.1.

6.4.3 Agregacija podataka

U predloženom modelu BPMA obavlja agregaciju ključnih podataka PP-a koji se izvorno nalaze u pojedinim naslijeđenim sustavima. Kako proces teče, prolazi kroz različite naslijeđene aplikacije, svaka od aplikacija prilikom komunikacije s BPMA u standardnim procesnim točkama šalje u BPMA svoj dio podataka. Tako se u instanci procesa na jednom mjestu nalaze svi relevantni podatci vezani uz tok PP-a kroz naslijeđenu okolinu, umjesto da su isti raštrkani po različitim naslijeđenim aplikacijama.

Inkrementalno ažuriranje poslovnih podataka BPMAPO na jednostavan način omogućavaju *smartDataUpdate(...)* metode koje se nalaze na klasama modela domene BPMKS-a, a razrađene su i opisane u poglavlju 5.6.1.1.

6.4.4 Kvalitetnija komunikacija s klijentima

Uvođenjem aktivnosti *Clarify_Rejection_Reason_To_The_Client* i *Investigate_And_Notify_Client* poboljšana je komunikacija prema klijentu. Klijenti dobiju kvalitetniju povratnu informaciju o razlozima odbijanja ili parcijalnog izvršenja zahtjeva, što im omogućuje brže reakciju te daje dodatne informacije u slučaju ponovnog zadavanja zahtjeva.

Aktivnost *Clarify_Rejection_Reason_To_The_Client* uvedena je zbog specifičnosti poslovnog slučaja izvršavanja ZKK-a, to jest izvršavanja i autorizacije instrukcija plaćanja u postojećim sustavima (AUL i APKO) koji u slučaju pogrešaka i/ili odbijanja, ne nude kvalitetnu razumljivu povratnu informaciju prema klijentu. Iz navedenog razloga, RM koji je upoznat s pogreškama i arhitekturom tih sustava zamoljen je da komunicira i objasni razlog klijentu te mu preporučuje danji tok akcija. Sličan postupak predviđen je i kod aktivnosti *Investigate_And_Notify_Client* koja postaje aktivna u slučaju da su samo neke instrukcije uspješno izvršene.

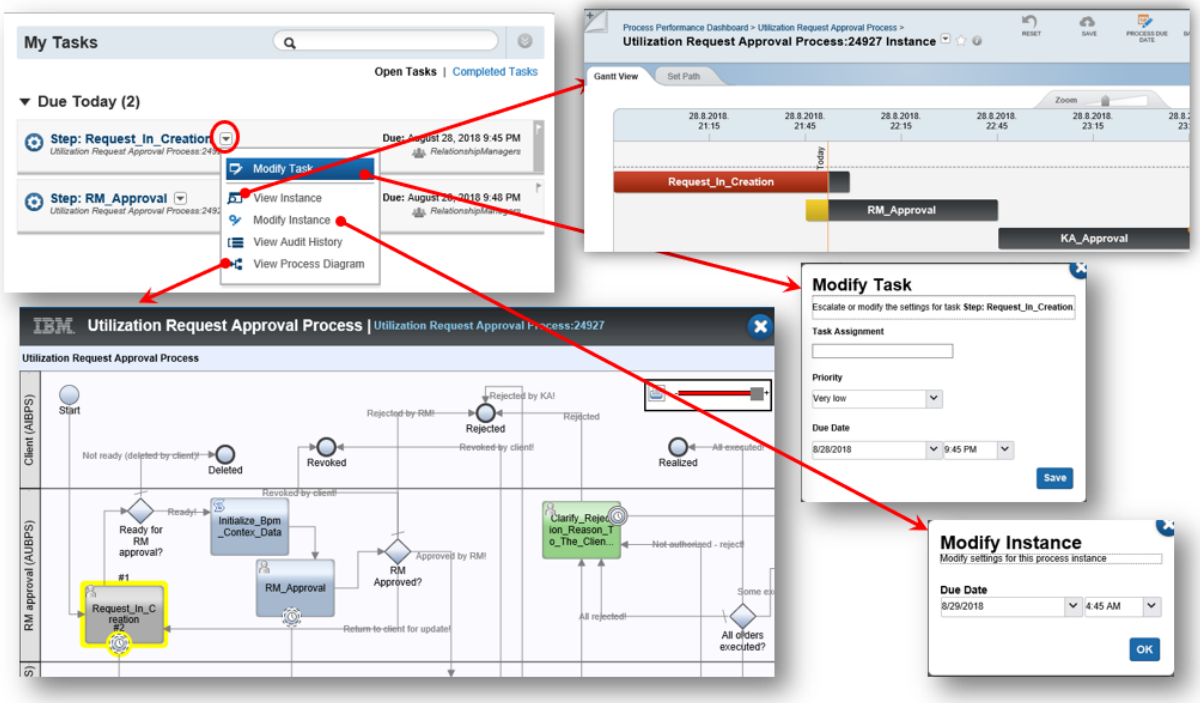
Obje aktivnosti sadrže potrebne informacije (linkove do naslijeđenih aplikacija, podatke o zahtjevu te povijesti zahtjeva) kako bi RM mogao identificirati uzrok odbijanja zahtjeva i/ili instrukcije plaćanja (račun upao u blokadu, korisnik na crnoj listi, itd.), obavijestio klijenta i informirao ga o daljnjim akcijama.

Iako zbog rigoroznih kontrola koje prethode samom procesu autorizacija i izvršavanja naloga te velikog stupnja automatizacije TSPP-a relativno mali broj zahtjeva završi u statusima "Odbijen" i "Parcijalno izvršen" (Tablica 6.3), navedene aktivnosti podižu kvalitetu PP-a i u tim slučajevima.

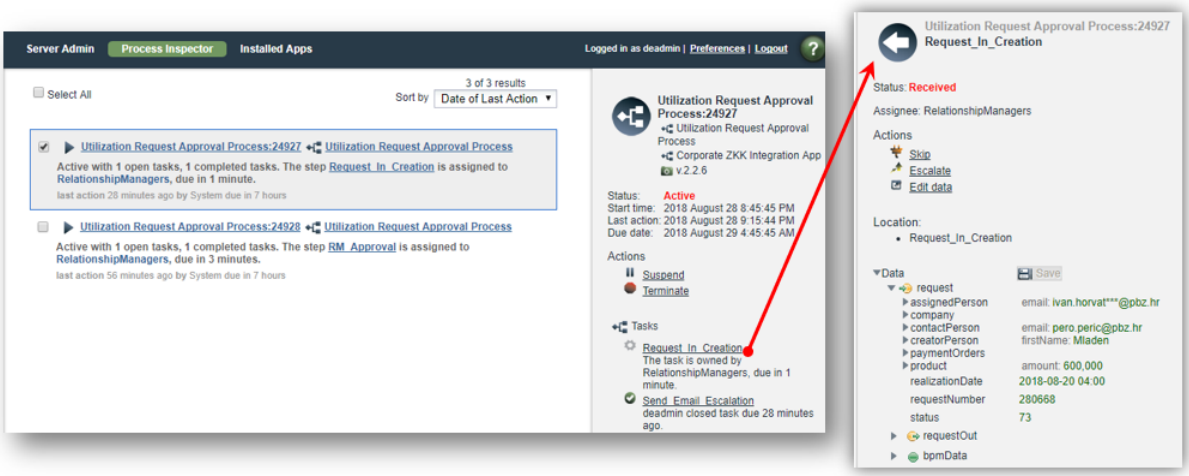
6.4.5 Pregled i kontrola nad procesom

Iskorištavajući alate BPM okruženja, predloženi model integracije, omogućava transparentnost stanja, toka i podataka procesa te dostupnost relevantnih informacija o aktivnostima i zaduženim osobama kroz sve korake PP-a. Korisnik može vidjeti rezultate odrađenih aktivnosti te aktivnosti koje tek slijede te u stvarnom vremenu ima opciju preraspodjele zadataka, promjene zaduženih osoba, vremenskih rokova i ostalih procesno relevantnih parametara.

U opisanoj implementaciji koristeći skup IBPM alata, korisnik kroz *procesni portal* prati proces i mijenja ograničeni skup parametra procesa i aktivnosti (Slika 6.11)., dok administrator sustava kroz *procesnu administratorsku konzolu* nadzire, mijenja procesne podatke i utječe na cjelokupni tok procesa (Slika 6.12).



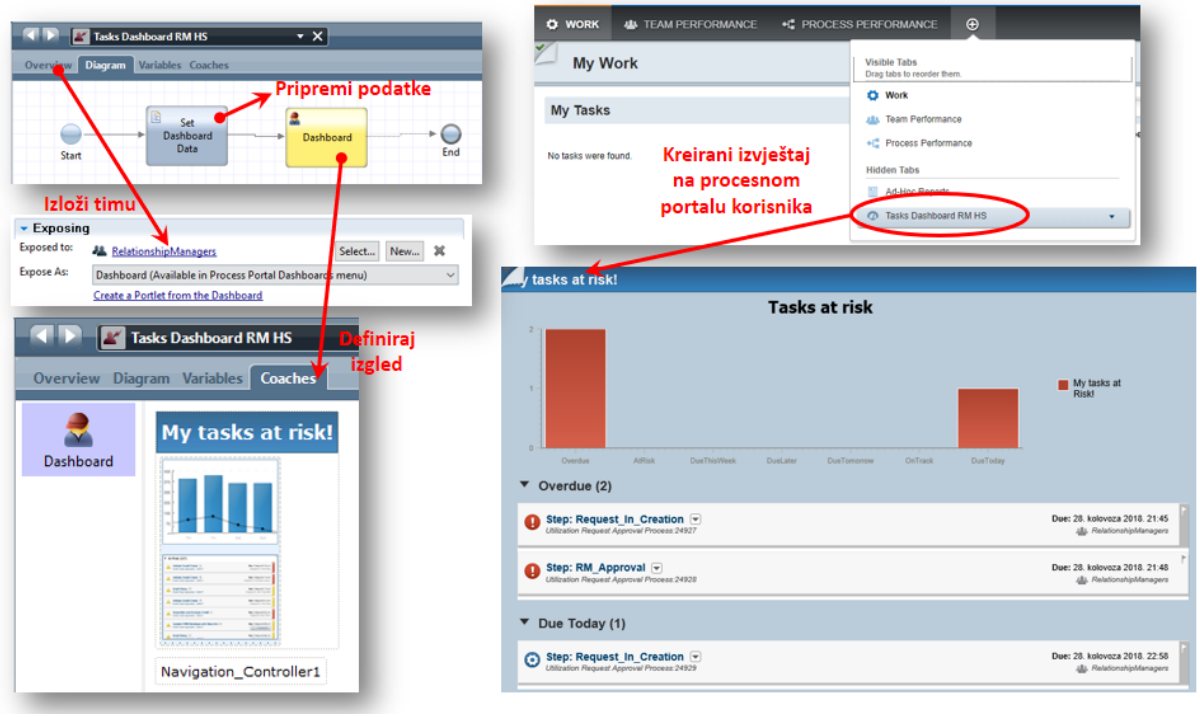
Slika 6.11 Nadzor procesa kroz Procesni Portal skupa alata IBPM



Slika 6.12 Kontrola i izmjene po procesu kroz procesnu administratorsku konzolu

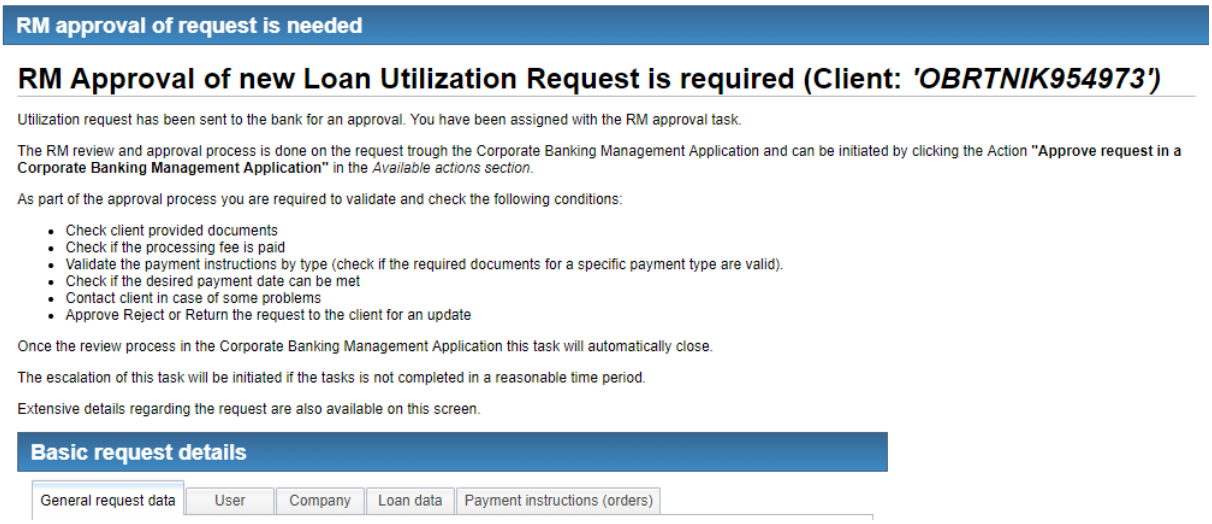
U svrhu sistematiziranog pregleda podataka procesa, primjerice za određenu grupu korisnika, možemo kreirati nadzornu ploču (engl. dashboard) koja prikazuje interesni skup informacija o procesu. Kreira se izgled izvještaja (koristeći IBPM ljudski servis i *Dashboard Toolkit* komponente), odredi se skup podataka za prikaz te se odabire tim unutar BPMA koji izvještaj može vidjeti. Član odabranog tima, na stranici procesnog portala zatim ima mogućnost

pregleda kreiranog izvještaja. Slika 6.13 prikazuje opisani postupak na primjeru izvještaja za pregled kritičnih aktivnosti člana tima *RelationshipManagers*.



Slika 6.13 Izvještaj o eskalaciji zadataka za korisnika tima RelationshipManagers

Dodatno, BPMA ekrani ljudskih zadataka sadrže detaljne upute o provjerama i akcijama koje zadužena osoba obavlja u naslijeđenim aplikacijama u sklopu određene aktivnosti (npr. podaci o zadacima RM-a koje prikazuje Slika 6.14).



Slika 6.14 Primjer popisa zadataka u aktivnosti odobrenja zahtjeva od strane RM-a

Službenik je dužan u detalje poznavati redoslijed i vrste provjera, međutim ako naslijeđena aplikacija eksplicitno ne prisiljava službenika da evidentira i provede sve provjere, moguće je

da se zahtjev neće u potpunosti provjeriti. Popisom zadataka na BPMA dodatno informirano službenike o potrebnim provjerama, te podižemo kvalitetu procesa odobrenja zahtjeva.

6.4.6 Postavljeni temelji za optimizaciju

Predloženi model integracije, koristeći mogućnosti BPMS-a, omogućava definiranje metrika, automatsko provođenje mjerenja i sakupljanje izmjerenih rezultata o poslovnom procesu, te pruža sredstva analize PP-a. Modeliranje naslijeđenog procesa u BPMS-u, kroz iteracije predložene u poglavlju 5.5.2.3, omogućuje otkrivanja defekata i nedostataka procesa kao što je prikazano kroz rezultate izvođenja simulacija iz poglavlja 6.3.2. Dodatno, uz pomoć vlastitog postupka preslikavanja mjera definiranog u poglavlju 7.1, te metoda i principa mjerenja opisanih u poglavlju 7 postavljeni su temelji za daljnju optimizaciju PP-a.

6.5 Prepoznatljivi dizajn sučelja i kvaliteta korisničkog iskustva

Kod izgradnje BPM alata, proizvođači se često primarno fokusiraju na brzinu i lakoću modeliranja PP-a, dok razvoj i intuitivnost korisničkih sučelja često ostane u drugom planu. Stoga je stiliziranje elemenata korisničkih sučelja i prilagođavanje dizajna procesne aplikacije te kreiranje intuitivnih i responzivnih korisničkih ekrana, koji se pravilno prikazuju na različitim uređajima, često vrlo složen zadatak.

Kao što je opisano u 6. poglavlju, za izgradnju korisničkih sučelja na BPMA korištene su izvorne kontrole i stilovi IBPM alata koji nisu dovoljno robusni i ne odgovaraju prepoznatljivom dizajnu organizacije koja alat koristi. Da bi postigli intuitivnost, interaktivnost i vizualnu usklađenost elemenata *Coach* i *CV* komponenti sučelja IBPM alata možemo koristiti kombinaciju CSS, HTML i JavaScript radnih okvira kao što su Bootstrap, Semantic UI ili slični.

Za podizanje razine kvalitete iskustva korisničke interakcije često je dovoljno koristiti najnovije verzije BPM alata. Odličan primjer je IBPM skup alata čija trenutna verzija (8.6 za vrijeme pisanja ovog rada) donosi mnogo inovacija spram verzije 8.5.0.1 korištene u ovom radu. Novi *BPM UI skup alata (engl. Toolkit)* (poznat kao *SPARK UI toolkit*) sadrži više od 80-tak kontrola za izgradnju i konfiguraciju korisničkih sučelja (za usporedbu verzija 8.5.0.1 sadržavala ih je 15-tak), a same kontrole su robusnije, integriranije s procesnim dijelom alata te pružaju više opcija vizualne i funkcionalne konfiguracije.

7 Postupak praćenja, mjerenja i optimizacije poslovnog procesa

Iako pojam metodologije BPM aludira na automatizaciju procesa jedna od važnijih karakteristika je sposobnost optimizacije. Koristeći predloženi model integracije, optimizacija PP-a u vidu raspodjele resursa, toka aktivnosti te detekcije nepravilnosti odrađuje se velikim dijelom već u fazi modeliranja procesa parcijalnim simulacijama (poglavlje 6.3). Međutim, da bi tijekom izvođenja bila dostupna informacija o stanju i problemima PP-a, te kako bi se dobio uvid u stanje poslovanja u cjelini, potrebno je redovito obavljati određena mjerenja.

Postupak mjerenja zahtijeva definiranje skupa mjerljivih vrijednosti ili KPI-jeva koji na višoj razini ocjenjuju napredak tvrtke u ostvarenju strateških i operativnih ciljeva te služe za usporedbu uspješnosti organizacije u odnosu na tržišne konkurente u okviru interesne djelatnosti [27], [30].

Metrika je skup vrijednosti, pojedinačnih mjera, koje mjerimo da bi pratili određeni aspekt poslovne aktivnosti. S druge strane KPI-jevi su metrike u službi poslovnih ciljeva, to jest metrike koje se odnose na određeni poslovni ili strateški cilj i odražavaju uspješnost poslovanja u postizanju tog cilja. KPI možemo smatrati metrikom, međutim, metrika nije nužno KPI, metrika stavljena u kontekst određenog poslovnog cilja zapravo postaje KPI. U ovom poglavlju pretežno se koristi izraz metrika.

Metrike se mogu definirati na različitim razinama, cilj mjerenja je povezati mjere na najnižoj razini (sredstva tehničke implementacije kao što su servisi, procedure, itd.) s podacima srednje razine (razina aktivnosti i PP-a), te promatrati kako promjene na navedenim razinama utječu na poslovne ciljeve na najvišoj organizacijskoj razini. Kako opisano povezivanje metrika imalo smisla, izmjerene mjere moraju biti međusobno kompatibilne na različitim razinama, moraju se moći preslikati s jedne razine na drugu ili imati jasnu međuovisnost. Dobar primjer kompatibilnih mjera bilo bi vrijeme ili trošak izvođenja pojedinih akcija.

Metrika na razini PP-a mogu uključivati vrijeme trajanja procesa, prosječnu starost procesa, postotak i cijenu resursa prekinutih procesa, itd. S druge strane KPI-jevi na razini SOA opisuju tehničke karakteristike servisa kao što su vrijeme odziva, dostupnost, količina korištenih resursa, propusnost, itd. [42].

U ovom poglavlju u osnovama se razrađuju principi definiranja metrika na različitim razinama, ispituju se metode međusobnog povezivanje metrika i njihov utjecaj na poslovne ciljeve organizacije, te se tako postavljaju temelji za faze ***praćenja*** i ***optimizacije*** PP-a (Slika 5.30).

7.1 Vlastiti postupak mjerenja i optimizacije procesa

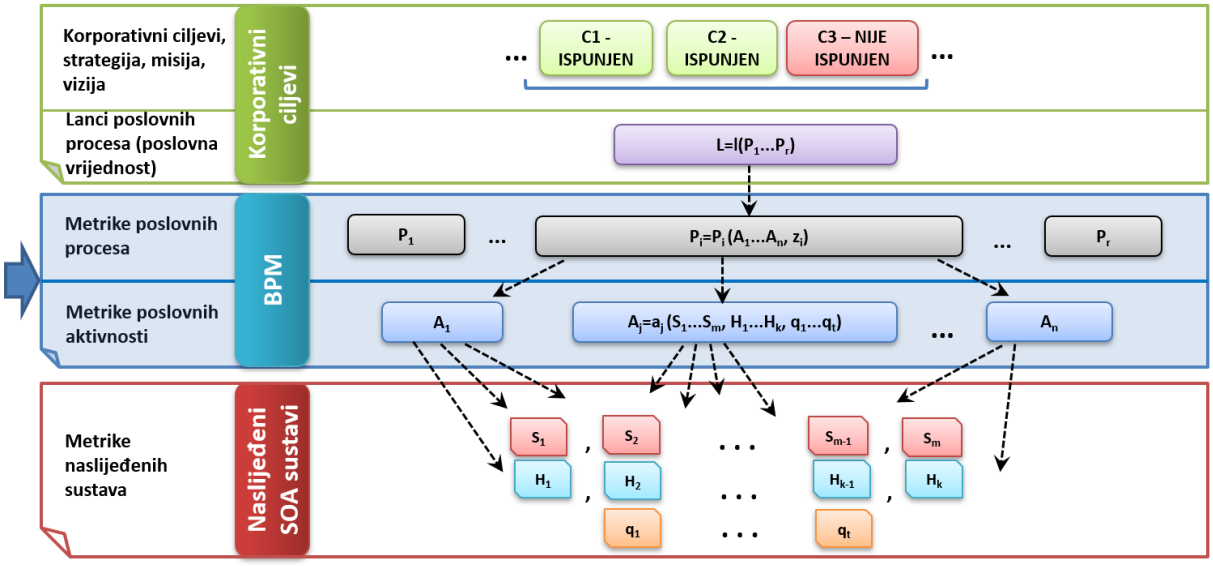
U predloženom modelu integracije, mjerenja se provode BPMS-om što rezultira izmjerenim podacima na razini ***aktivnosti i razini PP-a*** (Slika 7.1). U nastavku je razrađen općeniti model preslikavanja KPI-a, koji, temeljem mjernih podataka razina ***aktivnosti i PP-a***, omogućuje aproksimaciju zadovoljenja poslovnih ciljeva na najvišoj razini i pruža uvid u detalje implementacije aktivnosti u naslijeđenim sustavima na najnižoj razini.

Prikaz preslikavanja metrika ilustriramo na primjeru realizacije cilja C_1 određenog poslovnom vrijednošću koja se ostvaruje kroz:

- jedan lanac poslovnih procesa L
- r procesa $P_1, \dots, P_r$
- r vanjskih faktora $z_1, \dots, z_r$ koji utječu na izvršavanje aktivnosti
- n aktivnosti $A_1, \dots, A_n$
- m servisnih poziva $S_1, \dots, S_m$
- k ljudskih akcija $H_1, \dots, H_k$
- t vanjskih faktora $q_1, \dots, q_t$ koji utječu na izvršavanje aktivnosti.

Radi jednostavnosti notacije, navedene oznake za lanac procesa, procese, aktivnosti, itd. ujedno koristimo i za njihove metrike.

Slika 7.1 prikazuje vlastiti postupak preslikavanje metrika, a u nastavku su opisani detalji preslikavanja metrika na svakoj razini.



Slika 7.1 Općeniti model preslikavanja metrika u predloženom modelu integracije

Predloženi vlastiti model preslikavanja metrika, možemo promatrati kroz razradu metrika na nekoliko razina:

- **Razina lanaca poslovnih procesa (poslovna vrijednost i ostvarenje poslovnih ciljeva)**
  Na ovoj razini potrebno je utvrditi je li zadovoljen određeni poslovni cilj organizacije? Mjerenja provedena u sklopu vlastitog modela rezultiraju izmjerenim metrikama na ***razini poslovnih procesa***, međutim da bi se na razini ***lanaca poslovnih procesa*** stvorila određena poslovna vrijednost i zadovoljio određeni poslovni cilj potrebno je analizirati više PP-a.

  Ako se lanac procesa sastoji od r procesa, njegova metrika je matematička funkcija procesnih metrika $P_1 \dots P_r$, dakle:

  $$L = l(P_1, \dots, P_r)$$

  gdje je l realna funkcija najviše r varijabli.

  Metrike lanaca procesa predstavljaju složene poslovne pokazatelje s konkretnim značenjem te se mogu usporediti s očekivanim poslovnim ciljevima za određeno poslovno područje, uslugu ili proizvod. U slučaju odstupanja izmjerenih složenih poslovnih pokazatelja od definiranih poslovnih ciljeva potrebno je obaviti detaljnu analizu s vrha prema dolje (top-down), što je ujedno i glavna prednost ovog pristupa. Kada neki poslovni cilj organizacije nije zadovoljen, analiza se usredotoči na lanac poslovnih procesa, procese i aktivnosti koji sudjeluju u ostvarivanju tog poslovnog cilja. Identificira se izvor nepravilnosti otkrivanjem problematičnih procesa i analizom metrika tih procesa.

- **Razina poslovnih procesa**
  Analiza problematičnog procesa uključuje analizu aktivnosti od kojih se poslovni proces sastoji, pa je potrebno metriku procesa prikazati preko metrika aktivnosti. Svaku procesnu metriku $P_i,\ i = 1, \dots, r$, možemo izraziti kao matematičku funkciju metrika aktivnosti $A_1, \dots, A_n$ (ne nužno svih) i sumarnog vanjskog faktora z_i,

  $$P_i = p_i(A_1, \dots, A_n, z_i), \qquad i = 1 \dots r$$

  gdje je svaki p_i realna funkcija od najviše $n + 1$ varijabli.

  Osim samih metrika aktivnosti, procesna metrika može uključivati vanjske faktore z_i koji utječu na izvršavanje procesa, a nisu direktno vezani uz metrike aktivnosti. Tipični primjeri

takvih faktora mogu biti kašnjenja zbog pripreme i unosa podataka, nedostupnost korisnika, problemi kod ljudske interakcija, itd..

- **Razina aktivnosti**

  Kada prethodna analiza pokaže nepravilnosti u određenoj aktivnosti, fokus se seli na naslijeđenu aplikaciju koja je za tu aktivnost zadužena. Aktivnost A_j u naslijeđenoj aplikaciji možemo opisati kao kombinaciju metrika servisnih poziva $S_1, \dots, S_m$ i metrika ljudskih akcija $H_1, \dots, H_k$. Ljudske aktivnosti mogu uključivati usporedbu podataka u naslijeđenoj aplikaciji, ne automatizirane provjere, kontrolu dokumentacije, itd.. Metrika na razini aktivnosti također može uključivati i neke vanjske faktore koji utječu na izvršavanje aktivnosti $q_1, \dots, q_t$, kao na primjer pozadinska obrada koja dodatno opterećuje sustav i stvara kašnjenja. Izražen matematičkom funkcijom, opisani odnos izgleda ovako:

  $$A_j = a_j(S_1, \dots, S_m, H_1, \dots, H_k, q_1, \dots, q_t), \qquad j = 1,2, \dots, n$$

  gdje je svaki a_j realna funkcija od najviše $m + k + t$ varijabli.

  Optimizacijom izvođenja servisa, identifikacijom potencijalnih nepredvidivih situacija, uklanjanjem nepotrebnih provjera koje korisnici obavljaju ili pojednostavljenjem toka aktivnosti u naslijeđenim aplikacijama mijenjamo metrike aktivnosti te tako preko metrika procesa i lanaca procesa direktno utječemo na poslovne ciljeve organizacije.

- **Razina metrika naslijeđenih sustava**

  Uključuje mjerenje metrika servisnih poziva, ljudskih akcija i dodatnih vanjskih faktora u naslijeđenim sustavima.

Rezultat ovog općenitog modela preslikavanja metrika je izravna povezanost između svakodnevnih zadataka i globalnih ciljeva tvrtke.

Identifikacija uzroka nepravilnosti određuje opseg promjena i razinu na kojoj treba uvesti promjene. Ako promjene uključuju izmjenu poslovne strukture, tema nadilazi SOA te BPM područje i odnosi se upravljanje poslovanjem i menadžment. S druge strane, promjene na razini procesa (preraspodjela aktivnosti, upravljanje resursima, redizajniranje potprocesa ili izmjena poslovnih pravila) te optimizacija implementacije i toka aktivnosti u naslijeđenim sustavima,

mogu se provesti i nadzirati. Međutim, nakon svake izmjene potreban je niz simulacija koje će potvrditi pozitivan ishod napravljene promjene.

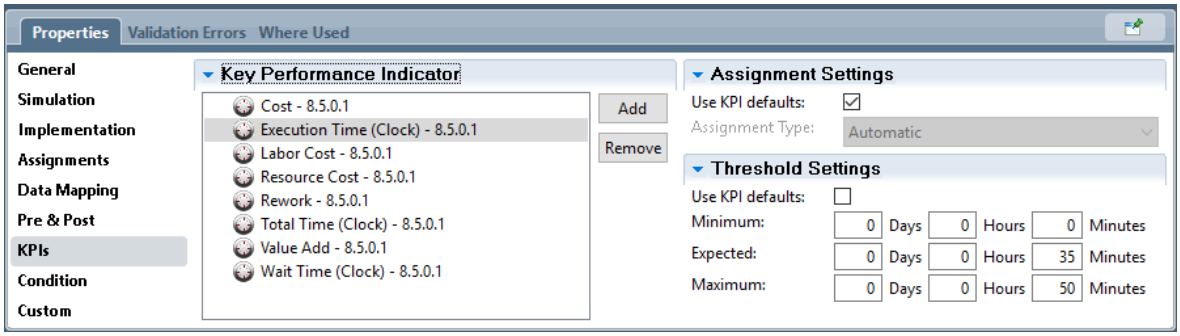
Osim neslaganja s konkretnim poslovnim ciljevima, postoje i drugi pokazatelji koji mogu ukazivati na potrebu za poboljšanjima poslovnog sustava, npr. veliki troškovi, veliki broj iteracija procesa, česte odgode, pogreške, itd. Za otkrivanje različitih pokazatelja poboljšanja potrebno je imati skup referentnih mjera ili KPI-a koji će se koristiti za usporedbu. U tom slučaju, vlasnik procesa ili poslovni korisnik koji je upoznat s karakteristikama određene aktivnosti, mora imati iskustveno utemeljene predodžbe o prihvatljivim vrijednostima tih mjera.

7.1.1 Definicija metrika i mjerenje uz pomoć BPMS-a

Kako bi podržali metodologiju opisanu u predloženom postupku mjerenja (poglavlje 7.1) potrebno se je usredotočiti na definiciju metrika na razini procesa i razini aktivnosti. Metrikama se mjere razni aspekti performansi BPMA, a definiraju se u BPMS-u. U pravilu, BPMS sadrži skup preddefiniranih metrika i točaka na kojima se vrše mjerenja uz mogućnost definicije dodatnih, prilagođenih, te složenih metrika.

Primjerice, u ovom radu korištene su predefinirane metrike iz IBPM alata, vrijeme i trošak izvršavanja servisa/aktivnosti/procesa, a IBPM BPMS ih automatski mjeri na razini procesa i aktivnosti ako uključimo funkciju automatiziranog praćenja ("*autotrack*") procesa [31].

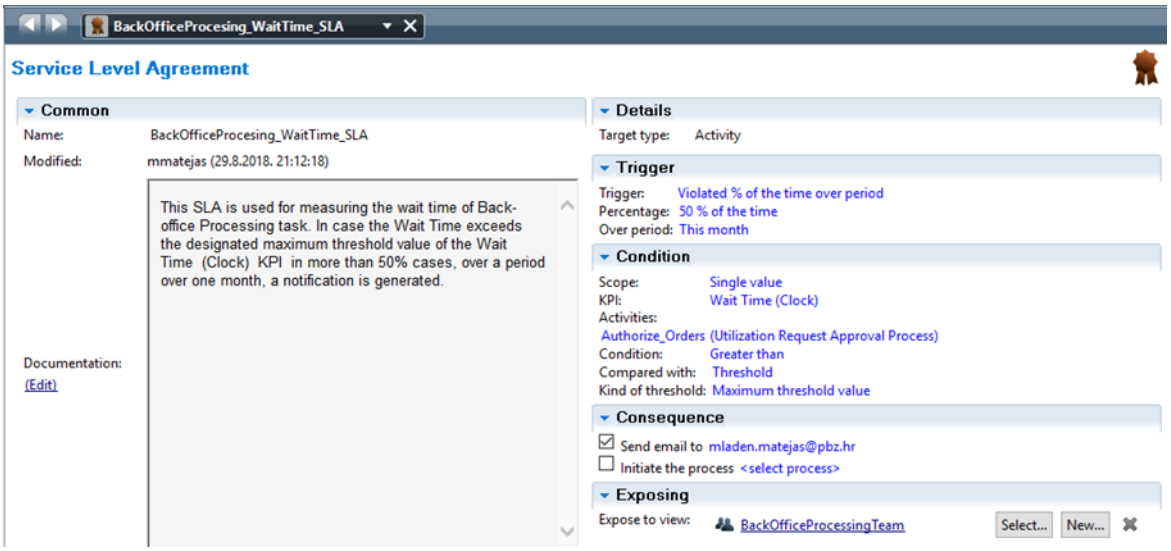
Navedene metrike možemo smatrati KPI-jevima, budući da se mjere u svrhu definicije cilja vezanog uz poslovnu aktivnost i/ili proces. IBPM *System Data Toolkit* sadrži veliki broj standardnih KPI-jeva koji su automatski povezani s aktivnostima (Slika 7.2). Međutim, temeljem specifičnih potreba moguće je definirati dodatne KPI-jeve. Poslovni korisnici, vlasnici i dioničari igraju važnu ulogu u definiciji KPI-jeva, osobito u određivanju graničnih vrijednosti (engl. treshold values) pojedinog KPI-a, pružajući tako uvid u očekivanja i ciljeve PP-a te strategiju organizacije.



Slika 7.2 Izvorni KPI-jevi definirani nad aktivnostima u IBPM alatu

Zanimljiva i korisna osobina KPI-jeva je agregacija ili pripadnost određenom "višem" KPI-u temeljem težinskog faktora. Primjerice KPI-jevi troškova rada i troškova resursa mogu se agregirati u KPI ukupnih troškova [64]. Agregacijom KPI-jeva na razini procesa izražavamo procesne metrike prikazane u predloženom vlastitom modelu mjerenja (poglavlje 7.1).

Mogućnost agregacije KPI-jeva korisna je kod definicije SLA indikatora, koji se obično sastoje od skupa KPI-jeva, omogućuju postavljanje uvjeta za jednu ili više aktivnosti te definiciju posljedica, događaja (slanje poruke ili pokretanja procesa), u slučaju nezadovoljenja tog uvjeta. Primjerice, uočeno je da trajanje aktivnosti autorizacije naloga (*Authorize_Orders* - Slika 6.3) zna biti puno veće od očekivanog, a sama aktivnost nije toliko zahtjevna. Sumnja pada na učestalo produženo vrijeme čekanja na aktivnost (vrijeme koje prođe od završetka prethodne aktivnosti do trenutka preuzimanja sljedeće aktivnosti od strane zadužene osobe). U svrhu provjere definiramo SLA indikator, koji će tijekom mjesec dana pratiti KPI vrijeme čekanja na aktivnost autorizacije naloga. Ako je u 50% slučajeva vrijeme čekanja na aktivnost veće od granične vrijednosti definirane nad KPI-em (Slika 7.2), definiramo obavijest u obliku e-mail poruke (Slika 7.3).



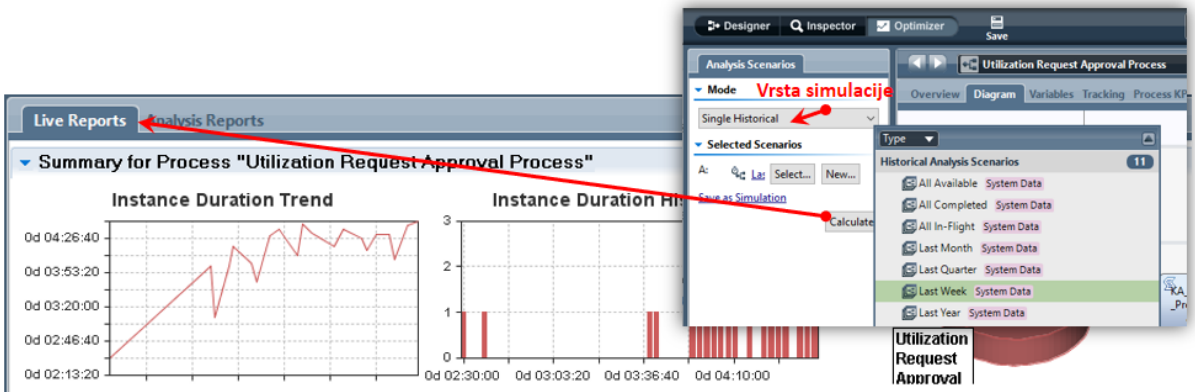
Slika 7.3 Definicija SLA identifikatora za kontrolu vremena čekanja

SLA indikatori su korisni u svrhu definiranja, te praćenja, određenih uvjeta nad aktivnostima i/ili procesom te pomažu osigurati da proces pridonosi ostvarenju ciljeva [27], [30].

Naprednije opcije praćenja omogućavaju tzv. grupe praćenja (engl. tracking groups) i vremenski intervali (engl. timing intervals). Grupe praćenja omogućavaju agregaciju metrika kroz različite vrste PP-a, dok se vremenski intervali koriste za praćenje željenog dijela jednog PP-a.

Tijekom izvršavanja procesa BPMS automatski prati definirane metrike te ih sprema radi naknadne analize procesa. BPMS također sadrži skup programskih rješenja za obradu, tumačenje i vizualni prikaz sakupljenih performansnih podataka o procesu.

Primjerice, IBPM skup alata asinkrono bilježi KPI i SLA identifikatore u točkama praćenja (engl. tracking points), koje se izvorno nalaze na ulazu i izlazu svakog elementa DPP-a (servisa, aktivnosti, skretnice, itd..), te ih sprema u PDW. Omogućava kreiranje prilagođenih izvještaja koji pružaju pogled na aktivnosti i performanse procesa u stvarnom vremenu, npr. kao što je izvještaj o eskalacijama prikazan u poglavlju 6.4.5 (Slika 6.13).



Instance Analysis

Number Executing	Number Completed	Duration			
		Min	Avg	Max	Dist
0	26	2h 29m 58s	4h 10m 18s	4h 42m 37s	

KPI Analysis

Cost

Activity Name	Number Of Occurrences	Sum	Average/Instance
Authorize_Orders	23	513,45 $	19,75 $
Clarify_Rejection_Reason_To_The_Client	2	34,82 $	1,34 $
Execute_Orders	23	111,89 $	4,30 $
Initialize_Bpm_Contex_Data	26	0,00 $	0,00 $
Investigate_And_Notify_Client	1	16,01 $	0,62 $
KA_Approval	26	1.371,89 $	52,77 $
Liquidity_Approval	24	1.066,38 $	41,01 $
Prepare_Escalation_Data_For_Activity	64	0,00 $	0,00 $
RM_Approval	26	978,31 $	37,63 $
Request_In_Creation	26	1.644,78 $	63,26 $
Send_Email_Escalation	64	312,78 $	12,03 $
Total		6.050,32 $	232,70 $

Activity Analysis

Activity Name	Counts			Execution Times			
	Waiting	Executing	Completed	Min	Avg	Max	Dist
Authorize_Orders	0	0	23	34m 43s	44m 38s	53m 3s	
Clarify_Rejection_Reason_To_The_Client	0	0	2	20m 9s	20m 53s	21m 36s	
Execute_Orders	0	0	23	7m 33s	9m 43s	11m 8s	
Initialize_Bpm_Contex_Data	0	0	26	0s	0s	0s	
Investigate_And_Notify_Client	0	0	1	19m 12s	19m 12s	19m 12s	
KA_Approval	0	0	26	39m 24s	45m 13s	55m 53s	
Liquidity_Approval	0	0	24	18m 38s	26m 39s	33m 43s	
Prepare_Escalation_Data_For_Activity	0	0	64	0s	0s	0s	
RM_Approval	0	0	26	36m 34s	45m 9s	54m 58s	
Request_In_Creation	0	0	26	58m 3s	h 15m 54s	1h 29m 22s	
Send_Email_Escalation	0	0	64	15m 56s	29m 19s	36m 55s	

Slika 7.4 Primjer rezultata analize povijesnih podataka razvojne okoline

Process Optimizer komponenta izlaže mogućnost različitih analiza povijesnih podataka (Slika 7.4) temeljem kojih dobijemo osnovne vrijednosti potrebne za korištenje u predloženom vlastitom modelu preslikavanja metrika (poglavlje 7.1). Također je moguće izvoditi složene analize kao što je usporedba simulacija izvođenja procesa sa stvarnim performansama procesa i povijesnim podacima ili istovremena analiza više različitih tipova procesa i instanci procesa.

7.1.2 Analiza i optimizacija aktivnosti naslijeđenih sustava

Temeljem predloženog vlastitog modela preslikavanja metrika i provedene analize procesa u BPM alatu koja je prikazana u prethodnom poglavlju, dolazi se do analize problematičnih aktivnosti u naslijeđenim sustavima. Zbog složenosti analize naslijeđenih sustava koji mogu biti različite složenosti, različitih arhitektura i pisani u različitim tehnologijama u ovom poglavlju dat će se općenite smjernice za pristup analizi s obzirom na principe prezentirane u vlastitom modelu preslikavanja metrika.

- Prvi korak analize aktivnosti je određivanje granica aktivnosti u naslijeđenoj aplikaciji temeljem interakcijskih točaka. Aktivnost se proteže od točke ulaza u naslijeđenu aplikaciju, do točke izlaza iz aplikacije po obavljanju aktivnosti.
- Zatim, potrebno je utvrditi sve korake/provjere/podaktivnosti koje službenik/ca mora obaviti u sklopu granica aktivnosti, te izmjeriti vezane metrike. Primjerice aktivnost odobrenja od strane kreditnog administratora (*KA_Approval* - Slika 6.3) uključuje kontrolu namjene korištenja kredita, provjeru jesu li ispunjeni svi uvjeti za korištenje kredita, provjeru naplate naknade, ažuriranje kreditne mape, donošenje konačne odluke (Odobri/Odbij).
- Svaki od identificiranih koraka analizira se s 3 različita aspekta te se utvrđuju metrike vezane uz pojedini aspekt:
    - **Ljudski faktor** uzima u obzir interakciju korisnika s naslijeđenom aplikacijom. Gledamo intuitivnost obavljanja posla, ergonomiju, lakoću i raspored akcija, provjera, koje korisnik mora obaviti prije donošenja konačne odluke. Uključuje razgovor s ljudima koje aktivnosti odrađuju, uzimanje u obzir prijedloga, želja i savjeta u svrhu poboljšanja aktivnosti. Metrike definirane u ovom koraku opisuju ljudsku interakciju (trajanje, trošak, itd.).
    - **Analiza sistemskih aktivnosti** podrazumijeva pozive servisa, spremljenih procedura, komunikaciju s vanjskim sustavima, kontrole pozadinskih obrada, te identifikaciju potencijalnih problema u navedenim točkama. Mjere se metrike navedenih sistemskih aktivnosti. Po potrebi pažnju usmjeravamo na baze podataka te kao rješenje radimo

optimizacije na bazama u svrhu bržeg dohvata podataka, primjerice particioniranje tablica, kreiranje indeksa, seljenje podataka u arhivske instance, itd.

- **Vanjski utjecaj** uključuje vanjske faktore i nedostatke koji uzrokuju smetnje u radu naslijeđenih sustava. Učestale pogreške, privremene nedostupnosti, pozadinska opterećenja i usporenja troše resurse i vrijeme te stvaraju probleme kod izvršavanja aktivnosti u naslijeđenim sustavima. Metrike iz ovog aspekta unose navedeni vanjski utjecaj u metrike izvršavanja aktivnosti naslijeđenih aplikacija.

Navedenim koracima, optimizacijom akcija iz aktivnosti naslijeđenih sustava, utječemo na metrike izmjerene BPM alatom (Slika 7.4), poboljšavamo rezultate izvršavanja procesa te pospješujemo postizanje poslovnih ciljeva.

7.2 Zaključci praćenja i optimiziranja procesa

Treba napomenuti da je, usprkos detaljnoj analizi, pogled na funkcioniranje procesa ograničen sve dok proces nije stavljen u produkciju. Usprkos različitim testnim scenarijima, referentnim iskustvenim podacima, stvarno trajanje određene aktivnosti i reakcija korisnika na proces bit će poznata tek nakon određenog perioda rada u stvarnom okruženju.

Dodatno, usprkos dobroj pripremi i definiciji KPI i SLA indikatora, stvarna slika mjera i potrebnih prilagodbi na mjerama također je vidljiva nakon perioda kontroliranog rada sustava u produkcijskom okruženju.

Proces optimizacije ovisi o mogućnostima korištenog BPMS-a u području praćenja, sakupljanja podataka te analize sakupljenih podataka (poglavlje 2.2.3.2). Dodatan faktor čini i kvaliteta naslijeđenih sustava te mogućnost pronalaska i otklanjanja nedostataka u tim sustavima.

U skladu sa svim navedeni faktorima, vidljivo je da je praćenje i optimizacija naslijeđenih PP-a u predloženom modelu integracije složen postupak koji otvara prostor za buduća istraživanja i dodatne razrade.

8 Evaluacija predloženog modela

U ovom poglavlju detaljno je prikazana evaluacija predloženog modela integracije te ideje koja stoji iza modela, temeljem ispitivanja mišljenja skupa ispitanika iz više srednjih i većih organizacija koje djeluju na području RH te na međunarodnoj razini. Neke od organizacija su: Privredna Banka Zagreb, Zagrebačka Banka, SV Group, ETNA d.o.o., CROZ d.o.o., TIS grupa, Infoart Group, mStart d.o.o., OMCO Croatia d.o.o, Ministarstvo Financija RH, Ekonomski Fakultet Zagreb, Podravska Banka, Franck d.o.o., itd.

Evaluacija je provedena u dva dijela:

- Provjera ideje na kojoj je temeljen predloženi model integracije, to jest ispitivanje mišljenja ispitanika o važnosti BPM-a i viđenja kako bi se BPM trebao implementirati u organizaciji te bi li se, i u kojoj dozi, trebao oslanjati na postojeće naslijeđene sustave. Ispitivanje je provedeno anketom na širom skupu ljudi različitih struka koji svakodnevno rade u složenim poslovnim okolinama te sudjeluju u PP-ima svojih organizacija. Više o istraživanju u poglavlju 8.1.
- Detaljnom evaluacijom karakteristika predloženog modela integracije od strane kompetentnih stručnjaka iz BPM područja. Od stručnjaka je traženo da daju svoja mišljenja, kritike i prijedloge. Detalji evaluacije s prikaznim rezultatima prezentirani su u poglavlju 8.2.

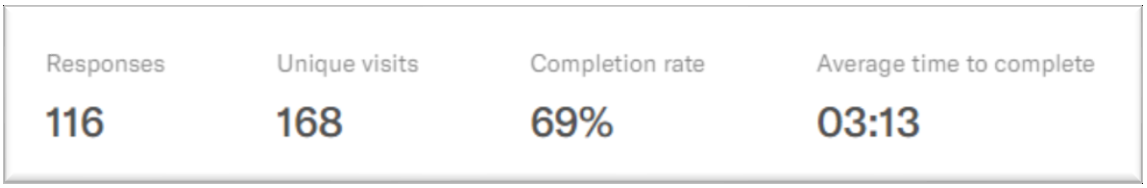
8.1 Važnost BPM-a i integracija s naslijeđenim sustavima

U ovom poglavlju, sažeti su rezultati istraživanja provedenog upitnikom nad širim spektrom praktičara i stručnjaka iz BPM i IT područja te ljudi različitih kompetencija koji se svakodnevno bave složenim poslovnim okolinama i procesima.

Cilj upitnika je dobiti mišljenje o važnosti integracije sustava za upravljanje poslovnim procesima u postojeću okolinu tako da se podrže postojeći procesi i očuvaju postojeće aplikacije. Upitnik se ne bavi analizom i validacijom korisnosti i karakteristika predloženog modela integracije, nego ispituje principe koji stoje iza ideje samog modela (opisane u poglavlju 3.3) te primjenjivosti BPM-a na kompleksne sustave poput primjera opisanog u poglavlju 6.

Upitnik se sastoji od deset pitanja koja su podijeljena u četiri skupine, a cilj svake skupine pitanja je dobiti mišljenje o određenom području/tezi/ideji. Ispitanici su na svako pitanje mogli odgovoriti s ocjenama od 1 do 5, gdje je ocjena 1 (*nimalo se ne slažem*) krajnje negativan odgovor, dok ocjena 5 (*u potpunosti se slažem*) predstavlja krajnje pozitivan odgovor.

Slika 8.1 prikazuje statistiku odaziva ispitanika koji su pristupili anketi.



Slika 8.1 Statistika odaziva na anketu o validaciji ideje predloženog modela integracije

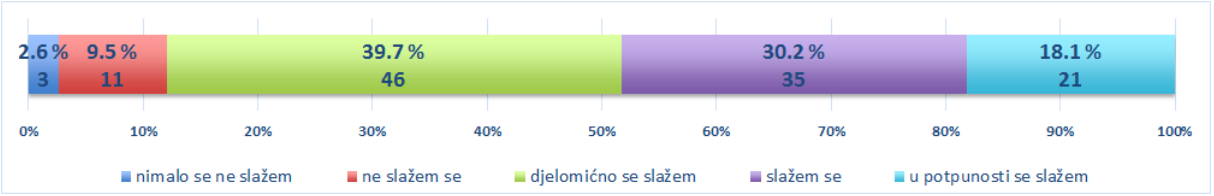
U nastavku su navedena pitanja i ciljevi pojedinih skupina pitanja, te su u detalje prikazani rezultati.

Prva skupina pitanja za cilj ima saznati koliko dobro su ispitanici upoznati s BPM područjem.

Pitanje:

1. *Upoznat sam s pojmom BPM-a (Business Process Management ili Upravljanje Poslovnim Procesima) i BPM alata koji omogućavaju modeliranje dijagrama poslovnih procesa te izradu procesnih aplikacija.*

Rezultati:



Prvo pitanje je ujedno i eliminacijsko pitanje. Ispitanici koji su odgovorili ocjenama 1 (*nimalo se ne slažem)* ili 2 (n*e slažem se*), nisu ubrojeni u danje rezultate jer se smatra da dovoljno ne poznaju područje te stoga ne bi dali relevantne odgovore na druga postavljena pitanja. Od ukupno 116 ispitanika, poznavanje područja iskazalo je 86%, to jest 102 ispitanika. Daljnja analiza upitnika napravljena je na tom uzorku.

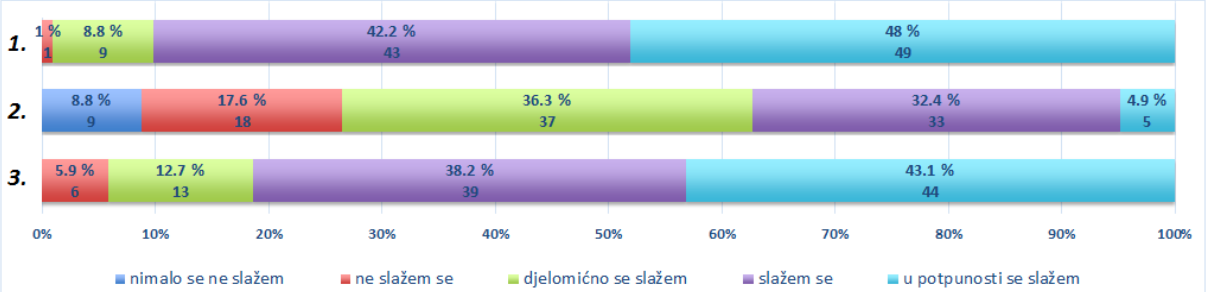
Druga skupina pitanja donosi saznanja u kojoj dozi ispitanici smatraju da je BPM važan koncept, te koliko misle da može biti koristan u njihovim organizacijama

Pitanja:

1. *Kvalitetno praćenje, kontinuirano optimiziranje i poboljšanje poslovnih procesa jedan od ključnih faktora za postizanje poslovnih ciljeva organizacije.*
2. *Poslovni procesi u vašoj organizaciji jasno su definirani i razumljivo implementirani u postojećim aplikacijama.*

3. *Implementacija BPM-a u vašoj organizaciji doprinijela bi poboljšanju postojećih poslovnih procesa.*

Rezultati:



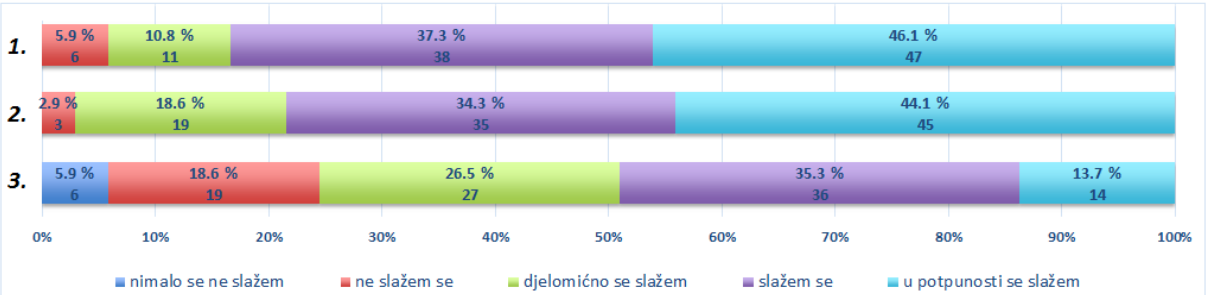
Iz navedenih rezultata vidljivo je kako gotovo svi ispitanici smatraju da BPM igra važnu ulogu u postizanju poslovnih ciljeva te u velikoj većini vjeruju da bi pomogao u poboljšanju PP-a u njihovoj organizaciji, u kojima ima mjesta za poboljšanje.

Treća skupina za cilj ima preispitati mišljenje o izvedivosti, te složenosti ponovne implementacije postojećih funkcionalnosti i procesa uz pomoć BPM alata.

Pitanja:

1. *Kompletne funkcionalnosti postojećih aplikacija iz Vaše organizacije, koje pružaju podršku postojećim poslovnim procesima, iznova izgraditi u adekvatnom BPM alatu bio bi veoma složen zadatak.*
2. *Prepisivanje funkcionalnosti postojećih aplikacija i izgradnja postojećih poslovnih procesa, uz pomoć BPM alata, bilo bi vremenski i resursno zahtjevno.*
3. *Ekrane, wizarde te interaktivna korisnička sučelja iz postojećih aplikacija, bilo bi veoma teško izgraditi u BPM alatima koji su primarno orijentirani na podršku tokovima poslovnih procesa i izvorno nude ograničen spektar mogućnosti za realizaciju korisničke interakcije.*

Rezultati:



Iako su ispitanici podijeljeni u mišljenju oko razine kompleksnosti korisničkih sučelja koja je moguće konstruirati u BPM alatima, prevladava stav da bi izgradnja podrške postojećim

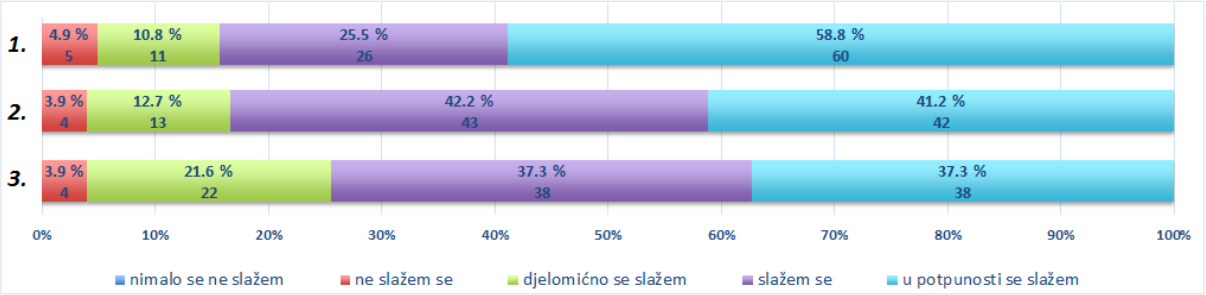
procesima iznova, uz pomoć BPM alata, bio kompleksan i vremenski zahtjevan pothvat. Jedna od zanimljivosti istraživanja može se vidjeti u rezultatima 3. pitanja ove skupine, gdje su podijeljena mišljenja o razini kompleksnosti sučelja koja se mogu realizirati BPM alatima. Otkriveno je da uzrok ove raspodjele mišljenja leži u činjenici da ljudi koriste različite BPM alate i/ili različite verzije istih alata koji uvelike variraju u količini i kvaliteti elemenata za izgradnju i vizualno oblikovanje korisničkih sučelja. Ova problematika djelomično je razrađena u poglavlju 6.5.

Četvrta skupina preispituje mišljenje ispitanika o ideji koja se obrađuje u ovom radu, a to je da se postojeće aplikacije, servisni moduli te postojeća poslovna logika maksimalno iskoriste kod izgradnje BPM rješenja u organizaciji.

Pitanja:

1. *BPM treba implementirati u organizaciji tako da se maksimalno iskoriste postojeći sustavi, funkcionalnosti, postojeća integracija s fizičkim uređajima (PKI), vanjskim pružateljima usluga (FINA, HROK, EaDS, itd.), te da se izbjegne dupli razvoj istih poslovnih funkcionalnosti.*
2. *Model poslovnog procesa izgrađen u BPM alatu moguće je povezati s postojećim poslovnim procesima koji se obavljaju u postojećim aplikacijama, na način da tok poslovnog procesa u BPM-u prati korake procesa u postojećim aplikacijama.*
3. *Dobro je poslovni proces modelirati u BPM alatu, koristiti BPM alat kao centralnu točku interakcije i u svrhu praćenja toka poslovnog procesa, a pojedine akcije tog procesa odrađivati korištenjem funkcionalnosti, sučelja, i poslovne logike koja već postoji implementirana u postojećim aplikacijama i servisnim modulima.*

Rezultati:



Rezultati ove skupine, posebice 1. pitanja, potvrđuju potrebu za što opsežnijim iskorištavanjem postojećih sustava kod izrade BPM rješenja, te opravdavaju problematiku kojom se ovaj rad bavi. Sama ideja ovog doktorskog rada, opravdana je prikazanim razmišljanjima ispitanika.

Analiza rezultata provedenog upitnika pokazala je postojanje problematike koja se obrađuje u ovom radu te potvrdila temelje ideje na kojoj se zasniva predloženi model integracije.

8.2 Detaljna evaluacija predloženog modela od strane relevantnih stručnjaka iz industrije

Predloženi model integracije predstavljen je ispitanicima iz više prethodno navedenih organizacija. Obuhvaćena je skupina od 53. ispitanika, koja se sastojala od stručnjaka različitih profila; korisnika i projektanata poslovnih IS-a, razvojnih stručnjaka, voditelja projekata te poslovnih korisnika, a zamoljeni su da daju svoje mišljenja, komentare i kritike. Svi odabrani ispitanici dobro su upoznati s BPM područjem, ali se razlikuju po količini iskustva stečenog u radu na različitim BPM projektima.

Ispitanici s ograničenim praktičnim znanjem i iskustvom u radu s BPM alatima i razvoju BPM rješenja dobro poznaju osnovne koncepte i principe BPM-a. Iskustvo tih ispitanika primarno se sastoji od proučavanja konkretnih gotovih primjera BPM inicijativa, završenih višednevnih osnovnih tečaja korištenja određenih BPM alata ili ograničenih praktičnih ispitivanja provedenih u svrhu dokazivanja koncepata i provjera mogućnosti BPM alata.

Ispitanici s opsežnim praktičnim iskustvom redovito rade na dizajnu i održavanju sustava za upravljanje poslovnim procesima, poznaju ulogu i rad BPM alata, te imaju uvid u praktične prednosti i nedostatke BPM rješenja. Istraživanje je provedeno u dvije faze. Tehnika istraživanja se temeljila na principima metode Delphi[56]. Budući da zbog geografskih udaljenosti te obaveza ispitanika nije bilo moguće sve stručnjake u isto vrijeme okupiti na istome mjestu, istraživanje je obavljeno individualnom usmenom i/ili elektroničkom komunikacijom.

Prva faza zahtijevala je najviše truda i vremena u cijelom istraživanju. U ovoj fazi provedena je prezentacija modela integracije stručnjacima kojima je prikazan sažeti opis predloženog modela integracije. Stručnjaci su iznijeli svoje mišljenje o modelu, mogući scenarij korištenja, te zaključak ili opis situacije u kojoj bi se predloženi model mogao pokazati korisnim.

[56] **Metoda Delphi** je strukturirana tehnika komunikacije, razvijena kao interaktivna metoda predviđanja koja se oslanja na skupinu stručnjaka. Stručnjaci odgovaraju na upite i donose neke zaključke u dvije ili više faza. Nakon svake faze ispitivanja, osoba zadužena za organizaciju i koordinaciju stručnjaka daje anonimni sažetak zaključaka. Taj sažetak se zatim potvrđuje, rangira prema važnosti ili korigira u sljedećoj fazi. Postupak se završava na temelju predefiniranog kriterija, npr. broja krugova, postizanja konsenzusa, stabilnosti rezultata, itd. Cilj cijelog procesa je filtrirati donesene zaključke te postići konvergenciju prema "ispravnim" odgovorima.

Sakupljeni scenariji korištenja su sistematizirani i sažeti čime je prva faza istraživanja privedena kraju.

Druga faza sastojala se od provjere sakupljenih scenarija. Svaki od ispitanih stručnjaka dobio je sažetak u vidu upitnika, te je zamoljen da ocjeni primjenjivost i učinkovitost svakog od scenarija korištenja s obzirom na predloženi model integracije. Tako su dobiveni najrelevantniji scenariji korištenja opisanog modela integracije, što je i cilj metode Delphi. Detaljna objašnjenja sakupljenih scenarija i rezultati evaluacije prikazani su u poglavlju 8.2.1.

Tijekom prve faze istraživanja, osim zaključaka o primjenjivosti modela, stručnjaci su bili zamoljeni da iskažu eventualne probleme i nedostatke koje vide u predloženom modelu integracije. U poglavlju 8.2.2. je napravljena analiza pronađenih nedostataka te su izneseni prijedlozi za daljnja poboljšanja kojima bi te ti nedostatci uklonili. Dodatno, kao "popratni" rezultat istraživanja nastala su i poglavlja 8.3 i 8.4, dok je temeljem pitanja i nejasnoća koje su stručnjaci imali vezao uz sličnosti predloženog modela i koncepata kao što su EAI i ESB napisano poglavlje 4.5.

Svi materijali korišteni u evaluaciji modela, zajedno sa sakupljenim rezultatima, dolaze kao dodatak ovome radu (poglavlje 12).

8.2.1 Prikupljeni scenariji korištenja i ocjene stručnjaka

U ovom poglavlju detaljno su razrađeni i objašnjeni sakupljeni scenariji korištenja predloženog modela integracije te situacije u kojima bi se predloženi model pokazao korisnim. Kao što je već navedeno, scenariji su proizašli iz ispitivanja mišljenja stručnjaka tijekom prve faze evaluacije modela.

Nakon objašnjenja svakog od scenarija, navedena je ocjena stručnjaka. Ocjena pokazuje koliko je prema mišljenju stručnjaka, opisani scenarij prikladan i koristan za primjenu predloženog modela integracije. Kao sredstvo ocjenjivanja odabran je upitnik koji je proveden u drugoj fazi evaluacije, a stručnjaci su mogli odabrati ocjenu na ljestvici od 1 do 5 gdje:

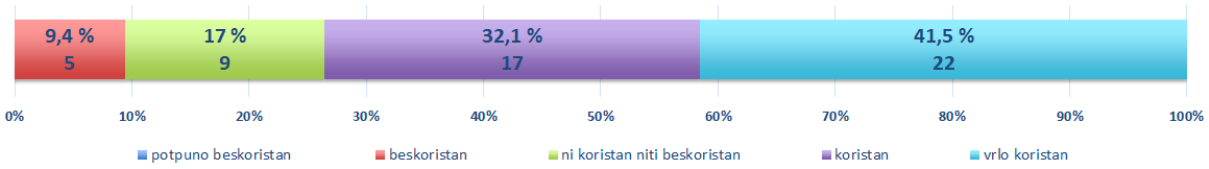
1 - *rangira scenarij kao* ***potpuno beskoristan*** *za primjenu predloženog modela*

5 - *rangira scenarij kao* ***vrlo koristan*** *za primjenu predloženog modela*

Ukupno, 53 stručnjaka pristupilo je upitniku ocjenjivanja scenarija i u potpunosti ga završilo. Detaljni opisi predloženih scenariji korištenja, te ocjene scenarija od strane stručnjaka (prema broju ocjena stručnjaka po pojedinom odgovoru, te postotnoj razdiobi) dani su u nastavku:

1) **Provođenje BPM pilot projekta u organizaciji -** Ispitanik iz organizacije koja je u vrijeme intervjua bila u procesu razmatranja uvođenja sustava za upravljanje poslovnim procesima, te su radili istraživanje tržišta i analizu alata i načina integracije, prezentirani model integracije vidio je kao potencijalni način provođenja pilot BPM projekta. Zbog iskorištavanja postojećih funkcionalnosti uz minimalne prilagodbe postojećih sustava, što povlači niže troškove te smanjuje vrijeme potrebno za razvoj, opisani model može se koristiti kao pilot projekt integracije BPM rješenja. Organizacije mogu izgraditi BPMA kao podršku za jedan od postojećih procesa, koristeći postojeće sustave, što bi im pomoglo da s obzirom na svoje potrebe sagledaju dobre i loše strane BPM-a, te odluče hoće li krenuti u daljnju implementaciju. U slučaju da organizacija odluči koristiti BPM, izvršavaju se daljnji integracijski napori. U protivnom, BPM se jednostavno razdvaja od postojećih sustava. Pilot projekt također omogućava organizacijama da poprave eventualne nedostatke u modelu integracije te služi kao uzor za izgradnju podrške ostalim procesima u organizaciji.

Ocjena stručnjaka:



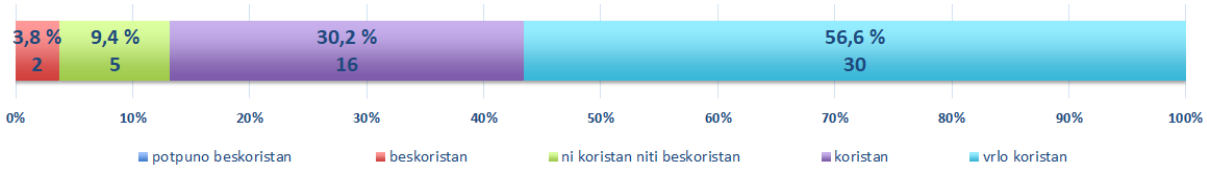
2) **Praktična procjena / usporedba karakteristika BPM alata na konkretnom primjeru -** Kod odabira BPM alata osim analize mogućnosti i ograničenja alata (poglavlje 2.2.3.2), potrebna je i njihova praktična validacija. Opisani model integracije pogodan je za ispitivanje mogućnosti i karakteristika različitih BPM alata. Budući da BPMKS odvaja naslijeđene aplikacije od BPM alata, te pruža univerzalan način komunikacije s BPM alatom, implementaciju istog procesa možemo relativno lagano povezati sa BPMA izgrađenim u različitim BPM alatima.

Ocjena stručnjaka:

3) **Implementacija u stabilnom, pouzdanom i kvalitetnom naslijeđenom aplikacijskom okruženju -** Model integracije koji maksimalno iskorištava postojeće sustave, dobar je ako

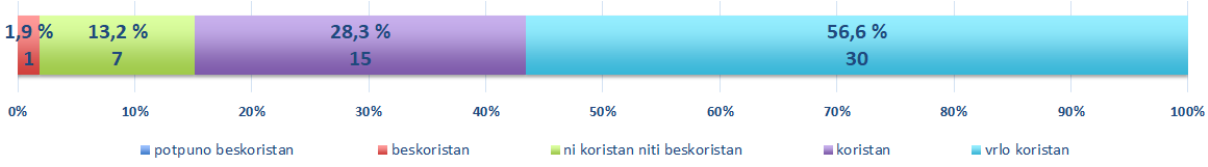
organizacija ima čvrsti i dobro izgrađen poslovni sustav. Stabilan i čvrst poslovni sustav podrazumijeva dobro izgrađene naslijeđene aplikacije koje uspješno podržavaju poslovanje organizacije. Dodatno, predviđa se da će dobro izgrađene naslijeđene aplikacije zadovoljiti potrebe organizacije u nadolazećoj budućnosti te nema naznaka za potrebom većih redizajna postojećih sustava. U takvoj okolini, poželjno je koristiti postojeće aplikacije kao dijelove BPM rješenja. Zaposlenici koji već rade na postojećim procesima ne trebaju puno dodatne obuke za rad s BPM rješenjem, jer bi uglavnom koristili postojeće aplikacije.

Ocjena stručnjaka:



4) **Situacija gdje organizacije imaju za cilj dodatno razvijati postojeće sustave i nove procese graditi u njima -** predloženi model omogućit će daljnji razvoj postojećih ili novih PP-a u postojećim aplikacijama. Imajući u vidu cjelokupnu sliku procesa, te podatke o performansama procesa prikupljene pomoću BPM alata, mogu se otkriti problematične aktivnosti. Na temelju otkrivenih problematičnih aktivnosti, identificiraju se odgovarajuće postojeće aplikacije koje te aktivnosti izvršavaju te se sukladno tome poduzmu korektivne mjere po aplikacijama. Temelji postavljeni u naslijeđenim aplikacijama, definirana arhitektura, razrađen model podataka, jasni principi i način rada, te iskustvo i znanje koje programeri imaju s postojećim tehnologijama i aplikacijama omogućilo bi lakšu i bržu implementaciju. Nakon implementacije novog poslovnog procesa u naslijeđenoj aplikaciji slijedi izgradnja odgovarajućeg procesnog modela (BPMA) te integracija BPMA s implementacijom. Opisani model omogućava daljnji razvoj postojeće okoline uvođenjem novih tehnologija, radnih okvira i alata, te omogućuje korištenje BPM-a iz novoizgrađenih sustava.

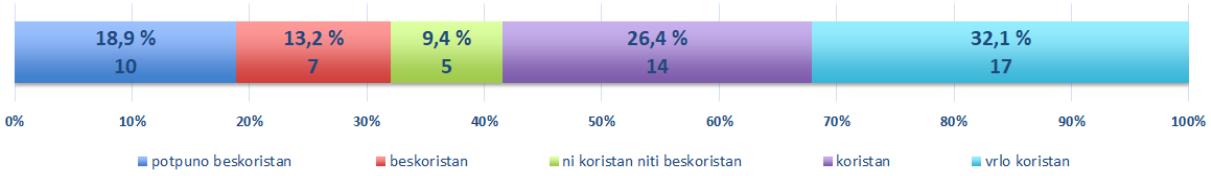
Ocjena stručnjaka:



5) **Potencijalni način izgradnje iBPM sustava -** iBPM je definiran od strane Gartner Inc. istraživačke kompanije kao proširenje konvencionalnog BPM alata (poglavlje 2.2.3.1).

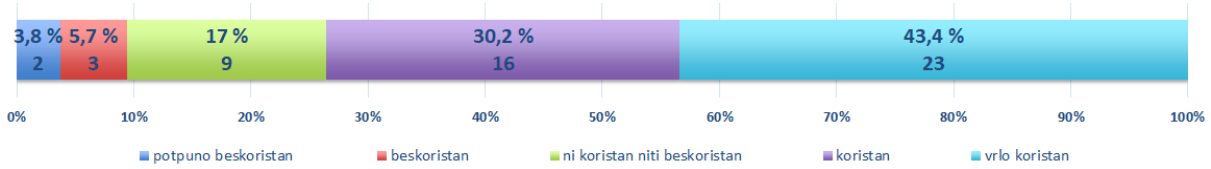
iBPM omogućuje povezivanje postojećih poslovnih procesa s metodama poslovnog planiranja koristeći umrežavanje sustava, računarstvo u oblacima, mehanizme za donošenje odluka u stvarnom vremenu, itd. iBPM predstavlja novi pojam, ali koncept već postoji kod organizacija koje u svrhu optimizacije procesa koriste alate za upravljanje poslovnim procesima u vezi s drugim alatima i sustavima. Prikazani model integracije je mogući način postizanja opisane veze.

Ocjena stručnjaka:



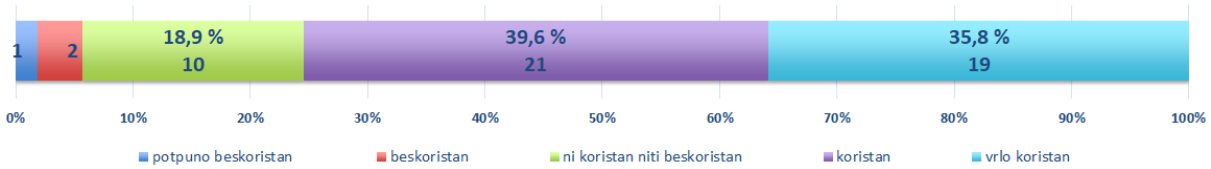
6) **Poslovna okruženja, organizacije s ograničenim resursima -** Organizacije u kojima ograničeni resursi sprječavaju reinženjering postojećih aplikacija, ali postoje inicijative za analizu i poboljšanje PP-a.

Ocjena stručnjaka:



7) **Implementacija u okruženjima temeljenim na aplikacijama s „debelim“ web slojem -** Ako su naslijeđene aplikacije izgrađene tako da se velik dio poslovne logike i programskog koda nalazi na web sloju aplikacije, gdje je usko vezan uz korisnička sučelja te prezentaciju poslovnih podataka, takvu aplikaciju nije lagano izdvojiti u zasebne servisne module. Direktno korištenje gotovih dijelova funkcionalnosti te korisničkih sučelja predstavlja lakši način djelovanja.

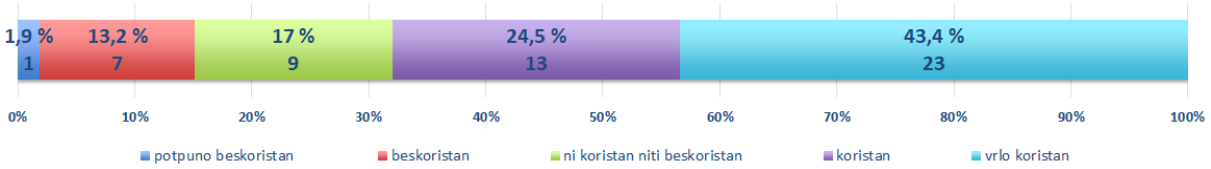
Ocjena stručnjaka:



8) **Iskorištavanje postojećih procesa kroz naslijeđene aplikacije ne zahtijeva detaljnu razradu poslovnih procesa u BPM alatu.** Ako koristimo postojeće funkcionalnosti, i

postojeće PP-e, svi detalji, akcije i funkcionalnosti potrebne za izvršavanje procesa su već implementirane u naslijeđenim aplikacijama. Prilikom modeliranja procesa u BPM alatu nije potrebno modelirati i razraditi svaki detalj, zasebno implementirati funkcionalnost svake aktivnosti, te razrađivati procese u detalje. Modelirani proces je može biti „grublji“, sadržavati kompletan tok procesa s osnovnim aktivnostima, dok naslijeđene aplikacije brinu za detalje i specifičnosti implementacije pojedinih aktivnosti.

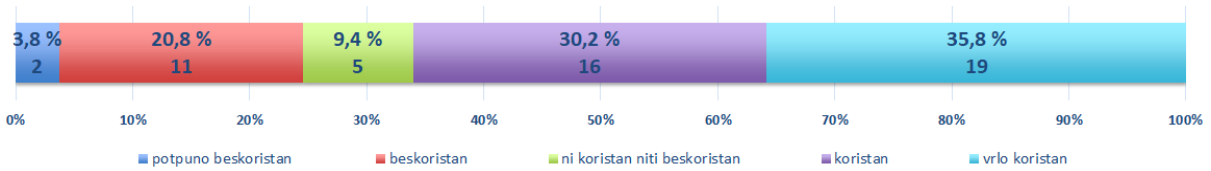
Ocjena stručnjaka:



9) **Specifičnosti i tehnologije ugrađene u postojeće sustave bolje odgovaraju poslovnom modelu i bolje služe poslovnim potrebama organizacije od mogućnosti koje pruža odabrani BPM alat.**

Specifičnosti poslovanja organizacije ukorijenjene su u postojećim informacijskim sustavima koji to poslovanje podržavaju. Korištenje alata, izgrađenog sa svrhom da rješava skupinu općenitih problema i podržava općenite situacije, u okolini sa specifičnim zahtjevima, može biti mukotrpna. Kako bi se specifičnosti premostile, često je potrebno posegnuti za dodatnim alatima ili izraditi vlastita programska rješenja što može zakomplicirati cjelokupnu situaciju. U navedenom slučaju, opisani model integracije može ostvariti optimalne rezultate kombinacijom osnovnih funkcionalnosti za upravljanje poslovnim procesima, koje pruža BPM alat, te specifičnih rješenja i sučelja implementiranih u postojećim aplikacijama.

Ocjena stručnjaka:



8.2.2 Dodatni komentari i zaključci evaluacijom stručnjaka

Kao nadopuna scenarija 3) iz poglavlja 8.2.1 zanimljiv zaključak iznesen od strane jednog stručnjaka je da predloženi model integracije ne bi bio najbolji izbor u okolini gdje postoji mnogo malih i nezavisnih naslijeđenih aplikacija, od kojih svaka upravlja manjim brojem procesa ili dijelova procesa. U takvom okruženju potrebno je implementirati više različitih

prilagodnika (jedan po aplikaciji) i povezati sve male naslijeđene aplikacije s BPM rješenjem. Zbog većeg broja različitih prilagodnika, povećava se vrijeme i složenost implementacije, pa je u opisanom slučaju pametnija odluka izdvojiti funkcionalnosti manjih aplikacija u skup servisa. Ti servisi se zatim koriste od strane novoizgrađene procesne aplikacije kako bi podržali PP-e. Stoga, predloženi model integracije je pogodniji za poslovno okruženje s nekoliko velikih aplikacija koje kontroliraju većinu PP-a u organizaciji.

Kao nastavak na scenarij 4) iz poglavlja 8.2.1 postavlja se pitanje što učiniti kada se implementiraju novi procesi koji nisu uopće povezani s postojećim aplikacijama, te opisuju poslovni model koji još ne postoji u organizaciji. Implementacija tih procesa u postojećim aplikacijama nije dobar izbor zbog neslaganja te različitosti novih procesa i postojećih aplikacija. Također, izrada zasebnih aplikacija koje će podržavati nove procese, te povezivanje tih aplikacija s BPM rješenjem nije isplativa. U opisanom slučaju bilo bi razumno izgraditi procesne aplikacije u BPM alatu te nove servise koji će podržavati funkcionalnosti procesnih aplikacija.

Upotreba postojećih aplikacija kao osnovnih dijelova BPM rješenja loša je ideja ako su naslijeđene aplikacije bazirane na zastarjelim tehnologijama ili procesi u postojećim aplikacijama nisu adekvatni te zahtijevaju sveobuhvatni redizajn. Bolji tijek djelovanja bio bi novih izgradnja servisnih modula koji će se koristiti od strane BPMA.

8.2.3 Pitanja i potencijalni nedostatci na koje su ukazali stručnjaci

Jedna od zabrinutosti izražena od strane nekoliko stručnjaka je da, ako se integracija ne provede ispravno, mogao bi nastati novi, složeni sloj izgrađen na postojećim aplikacijama. Novonastala situacija mogla bi dodatno zakomplicirati cjelokupnu informacijsku infrastrukturu organizacije te stvoriti dodatne probleme.

Stručnjaci koji su bili više zainteresirani za tehničke detalje implementacije, izrazili su zabrinutost zbog dupliciranja poslovnih pravila i modela podataka u BPMA i naslijeđenim aplikacijama. BPMKS, BPMA rade sa suženim skupom podataka naslijeđenih aplikacije, stoga je određeni nivo redundancije nažalost neizbježan. Međutim, cilj modela integracije je zadržati redundanciju na čim manjoj mjeri i osigurati čistu implementaciju.

Nadalje, u vlastitom modelu korisnici pristupaju naslijeđenim aplikacijama putem generiranih poveznica koje potencijalno sadrže tajne, "osjetljive" podatke (primjerice osobne identifikatore korisnika, klijenata, itd.) što može predstavljati problem sigurnosti. Da bi se to spriječilo, u predloženom modelu korištene su postojeće politike zaštite informacija te postojeće metode autorizacije koje se provode u organizaciji i u naslijeđenim aplikacijama [14].

U slučaju da organizacija nema valjane sigurnosne politike, može se implementirati Single Sign On[57] (SSO) metoda prijave pomoću LDAP-a. Različite aplikacije i usluge mogu se povezati s LDAP poslužiteljem i potvrditi autentičnost korisnika.

Neki od stručnjaka zanimali su se o detaljima ovlaštenja korisnika, konkretno kako i gdje odrediti koje aktivnosti korisnici mogu izvršavati. Kao rezultat ovog pitanja nastalo je poglavlje 4.4.2 gdje su razjašnjena korisnička ovlaštenja i načini pristupa.

Dodatno pitanje bilo je kako BPM zna koje aktivnosti nad procesom korisnik smije izvršavati u određenoj točci procesa, gledano s poslovne strane. Primjerice, korisnik može izdati naredbu opoziva zahtjeva samo dok je zahtjev u određenim statusima, to jest u ranim fazama procesa. Mora li BPM biti svjestan svih statusa u kojima se može odraditi opoziv? Odgovor na ovo pitanje je: Ne, BPM ne mora biti svjestan statusa u kojima je opoziv dozvoljen! Razlog leži u činjenici da naslijeđeni sustav izdaje naredbu opoziva, a taj naslijeđeni sustav već ima implementiranu logiku koja određuje kada i pod kojim uvjetima klijent može izdati naredbu opoziva. Uloga BPM-a je da, kada dobije naredbu za opoziv, završi proces s ispravnim statusom zahtjeva. Na ovaj način izbjegava se redundancija postojeće poslovne logike na BPM rješenju.

8.3 Osvrt na prosječnu poslovnu okolinu organizacija na domaćem tržištu

Temeljem razgovora s ispitanicima proizašla su neka zapažanja o stanju, potrebama i izazovima poslovnih okruženja u kojima ispitanici djeluju, pa ćemo ih u ovom poglavlju kratko prokomentirati.

Poslovna okruženja organizacija koje primarno djeluju na hrvatskom tržištu, u usporedbi s kompanijama svjetskih razmjera, su vrlo mala i dosta stabilna u smislu da nema mnoštvo učestalih promjena. Procesi koji se postave, većinom žive na način na koji su inicijalno postavljeni, bez većih naknadnih modifikacija. Volumen korisnika usluga u pravilu ne zahtijeva detaljne analize, opsežne prilagodbe i optimizacije procesa, a česta je i situacija da utrošak vremena i resursa nije isplativ spram koristi koje bi donijelo opsežno refaktoriranje. Implementacija BPM-a u takvom okruženju ne iskorištava cijeli potencijal metodologije. Postoje situacije gdje se koriste samo osnovne funkcionalnosti BPM alata, korištenje generalnih principa, te se BPM zapravo degradira i pretvara u opsežno praćenje toka PP-a.

[57] **Single sign-on** (**SSO**) je svojstvo kontrole pristupa do više srodnih, ali neovisnih, softverskih sustava. Korisnici se prijavljuju s jednim korisničkim imenom i lozinkom da bi pristupili do jednog ili više sustava, a u nekim konfiguracijama prijava je neprimjetna. SSO se obično postiže pomoću LDAP protokola i pohranjenih LDAP baza podataka na raznim poslužiteljima.

Dodatno, prisutan je problem u komunikaciji između poslovnih korisnika i vlasnika procesa te IT stručnjaka, što rezultira krivo postavljenim "temeljima" implementacije. Modeli PP-a kreirani od strane poslovnih korisnika znaju biti preopćeniti, bez potpuno razrađenih detalja potrebnih za tehničku implementaciju. Tako dolazi do razvoja procesnog modela koji zapravo nije reprezentacija stvarnog stanja, te je potrebno utrošiti dodatne resurse u njegovu razradu. Često se inicijalno zamišljeni proces uvelike razlikuje od krajnje implementacije. Rješenje takvih situacija bilo bi eliminiranje uzroka problema, rupe između poslovne strane i IT-a, dakle dodatna sinkronizacija između procesnih dizajnera te razvojnih stručnjaka koji rade na realizaciji tih procesa.

Javlja se trend gdje organizacije ne ulažu u BPM, već se ulaže u BPMS. Kupnjom precijenjenog BPMS-a koji podržava općenita rješenja, zaboravlja se da BPMS zapravo treba biti podrška procesu. BPM je proces optimizacije PP-a i ti PP-i moraju biti fokus cijele BPM inicijative. Odabir BPMS-a potrebno je prilagoditi potrebama PP-a, čega smo se detaljnije dotakli u poglavlju 2.2.3.2.

8.4 Neuspjeh inicijativa BPM-a, iskustva stručnjaka i komentari iz prakse te literature

Prema okvirnim podacima internetske zajednice *BPMInstitute.org*, koja okuplja preko 50 tisuća BPM stručnjaka iz raznih sfera, 9 od 10 BPM inicijativa osuđeno na propast. Tragom ovog alarmantnog podatka, u razgovoru s ispitanicima kroz navedeno istraživanje te kroz analizu slučajeva iz literature, pokušalo se otkriti i opisati neke od razloga tih neuspjeha. Dodatno, s obzirom na identificirane razloge neuspjeha, dan je osvrt na predloženi model integracije, te su istaknute karakteristike modela koje bi pomogle uspješnijoj realizaciji BPM inicijative.

Prema [12] česti uzrok neuspjeha BPM inicijativa je veliki i dugotrajni napor koji je potrebno uložiti u promjenu kulture. BPM zahtjeva promjenu u načinu rada organizacije, izmjene u upravljačkim strukturama, te opsežne edukacije i zaokret u načinu razmišljanja zaposlenika. Osim tehničkih izazova implementacije, poput razvoja novog informacijskog sustava koji će podržati promjenu, organizacije moraju razmišljati i o ljudskom aspektu implementacije; dakle jesu li zaposlenici spremni i osposobljeni za suočavanje s promjenama.

Model BPM integracije predložen u ovoj disertaciji omogućava spoj postojećeg i novog. Time se ublažavaju drastične promjene u kulturi i sposobnostima zaposlenika, te tako olakšava prihvaćanje i prilagodba na BPM. Nije potrebno raditi opsežne edukacije zaposlenika jer oni

većinu posla odrađuju u postojećim aplikacijama. Povezujući postojeće aktivnosti, predloženi model doprinosi interakciji različitih odjela i ljudi koji sudjeluju u poslovnom procesu. Zaposlenici dobivaju uvid gdje spada njihov dio posla, koji doprinos ima te kako se oni uklapaju u cjelokupnu sliku stvaranja poslovne vrijednosti.

Dodatan razlog neuspjeha BPM projekata je neadekvatan odabir tehnologije korištene u realizaciji. Prema [71] jedna od važnijih odlika BPM alata je da bude razumljiv i ne previše kompleksan. Isto obrazloženje možemo naći i u [23] gdje se dodatno spominje još niz tehnoloških faktora BPM alata koji otežavaju BPM implementaciju. Usprkos tome, uslijed neadekvatnog istraživanja tehnologija i alata dostupnih na tržištu u skladu s vlastitim potrebama, organizacije često potroše velike količine novčanih sredstava i vremena na nešto što je nerazumljivo, kompleksno i nepotrebno.

Odgovor na ovaj problem djelomično je dan u poglavlju 2.2.3.2 gdje su razrađene smjernice za odabir adekvatnog BPMS-a, međutim, pogledajmo kako predloženi model može pomoći u ovoj situaciji. Predloženi model omogućuje korištenje postojećih tehnologija i prilagođenih rješenja kao dio BPM rješenja. Mogućnost izvršavanja dijela aktivnosti u postojećim sustavima, smanjuje kompleksnost BPMS-a. Traženi BPMS ne mora podržati sve tehničke zahtjeve aktivnosti procesa jer je to već ispunjeno u postojećim sustavima. Primjerice, ako je u toku procesa potrebno digitalno potpisati dokument kvalificiranim certifikatom korisnika, u sklopu BPM rješenja nije potrebno podržati i razviti svu podršku za potpisivanje (PKI integracija, komunikacija s čitačima kartica ili USB tokenima, itd.), nego se koristi postojeća aplikacija. Tako se smanjuje sveukupna kompleksnost BPMS-a, te se olakšava proces BPM implementacije.

Prema iskustvima nekolicine intervjuiranih stručnjaka, inicijative BPM propadnu zbog neadekvatnog odabira prvog procesa. Odabir prvog procesa uvelike postavlja temelje razvoja cijelog BPM rješenja. Povučene entuzijazmom, organizacije inicijalno često odaberu tzv. *high effort – high business value* proces („*big bang*“ pristup), dakle proces u koji je potrebno uložiti mnogo resursa i za koji se smatra da će donijeti veliku dodatnu vrijednost organizaciji. Implementacija takvog procesa ima tendenciju zakomplicirati se i odužiti, prvenstveno zbog neiskustva uključenih timova, nedostatka komunikacije poslovne strane i razvojnih stručnjaka te dodatne složenosti procesa koja naknadno ispliva na vidjelo. U takvoj situaciji, vlasnici i zaduženi poslovni korisnici, ponukani velikim utroškom resursa i izostankom povratne vrijednosti projekta, izgube strpljenje. Rezultat je odustajanje od BPM rješenja uz stav da je to

nepotrebno te neisplativo. Odustaje se bez da je BPM isproban na barem jednom procesu, temeljem čega bi se mogli donijeti konkretni zaključci.

Kako bi navedenu situaciju izbjegli, stručnjaci predlažu odabir početnog procesa gdje je potrebno uložiti minimalan trud, a pruža "uočljivu" poslovnu vrijednost (*low effort – noticeable business value*). Dakle, jednostavan proces s jednostavnim grananjima koji po mogućnosti utječe na dovoljan broj korisnika kako bi postignuta korisnost ipak bila vidljiva. Upravo to jedan je od navedenih scenarija korištenja predloženog modela integracije kao što je i ustanovljeno u prvoj točci poglavlja 8.2.1 (realizacija BPM „pilot projekta“ uz nisku potrošnju resursa putem iskorištavanjem postojećih funkcionalnosti).

Prema Gartneru bit BPM-a nije tehnologija, već promjena. Promjene utječu na zaposlenike i na krajnje korisnike, a ako zaposlenici nisu na vrijeme uključeni u BPM integraciju, to može dovesti do otpora, negativnih posljedica, pa čak i propasti BPM inicijative. Analiza jednog rigoroznog primjera navedenog slučaja može se naći u [21] gdje je opisana implementacija sustava za upravljanje procesima obrade slika (engl. workflow and image processing system). Sustav je izgrađen, dobro je radio u testnom okruženju te je stoga implementiran u organizaciji. Međutim, zaposlenici su ga odbili koristiti i provedba nije uspjela. Rezultat neuspjeha identificiran je u needuciranosti i neuključivanju zaposlenika u tijek razvoja projekta. Zaposlenici, koji prethodno nisu bili kontrolirani ni direktno odgovorni za svoj dio posla, gotovo su rješenje doživljavali kao prijetnju zato što je mjerilo učinkovitost i posao koji su oni u procesu obavljali.

Jedan od načina sprječavanja ovakvog scenarija je iterativni pristup ugradnji BPM rješenja koji je također razmatran u ovom radu. Iterativni razvojni ciklusi, popraćeni simulacijama parcijalno dovršene procesne aplikacije omogućavaju upoznavanje i sudjelovanje zaposlenika na projektu već u ranim fazama razvoja.

9 Zaključak

Rad sadrži pregled BPM područja te istražuje vezu između poslovnih procesa i funkcionalnosti postojećih (naslijeđenih) sustava, temeljenih na SOA, koji te procese podržavaju.

Analiziraju se klasični modeli integracije sustava za upravljanje poslovnim procesima u postojeća aplikacijska okruženja te se otkriva da su oni prvenstveno usredotočeni na redizajn naslijeđenih sustava, uz minimalno iskorištavanje postojećih funkcionalnosti i oslanjanje na postojeće strukture. Praktična studija provedena u sklopu rada potvrđuje velik utrošak vremena i resursa te problematičnost redizajna složenih naslijeđenih aplikacijskih okruženja, uz negativan utjecaj na poslovanje organizacije u periodu redizajna. Zbog toga se mnoge organizacije ne odlučuju za uvođenje sustava za upravljanje poslovnim procesima.

Temeljem uočenih nedostataka klasičnih modela integracije izložena je ideja originalnog, vlastitog, modela integracije te su prikazane njegove prednosti.

Definiran je originalni model integracije čije osnovne karakteristike su očuvanje postojeće aplikacijske arhitekture i postojećih poslovnih procesa organizacije. Prilagodljivost modela osigurava labavu međuovisnost BPM rješenja i postojećih aplikacija, funkcionalnost postojećih aplikacija bez BPM rješenja te jednostavnu prilagodbu modela za rad s različitim BPMS-ovima. Omogućen je centraliziran pristup poslovnim procesima u naslijeđenim sustavima, podržane su manje izmjene tokova procesa promjenom modela procesa te je osigurana sinkronost podataka u BPM rješenju i naslijeđenim aplikacijama. Olakšano je uključivanje različitih postojećih aplikacija u BPM rješenje, neovisno o tehnologijama tih aplikacija. Model omogućava da poslovne potrebe na najvišoj razini diktiraju daljnji razvoj naslijeđenih aplikacija, definiranjem dodatnih funkcionalnosti prema želji.

Tehnička specifikacija radnog okvira za implementaciju novog modela definira postupke integracije na arhitekturnoj i procesno-podatkovnoj razini. Postupci na arhitekturnoj razini identificiraju poželjne karakteristike alata i tehnologija radnog okvira, specificiraju osnovne programske komponente modela, način njihove interakcije te opisuju principe povezivanja BPMA s postojećim aplikacijama. Predložen je strukturirani, iterativni, postupak procesno-podatkovne integracije koji obuhvaća otkrivanje postojećih poslovnih procesa, modeliranje

tokova tih procesa i njihove integracije s BPMA te prikaz rukovanja s podacima poslovnih procesa kombinacijom jednostavnih BPMA sučelja te složenih korisničkih ekrana postojećih aplikacija.

Postupci radnog okvira za implementaciju novog modela primijenjeni su na poslovnom primjeru iz bankarstva, a izgrađeni prototip dokazuje praktičnu primjenjivost modela i izvedivost pojedinih postupaka predloženog radnog okvira.

Strukturna inovacija i racionalizacija poslovnog procesa ostvarena je centraliziranim pogledom na postojeći poslovni proces koji se odvija kroz više različitih naslijeđenih aplikacija. Centralizirano izvršavanje aktivnosti, vidljivi sudionici procesa te točne informacije o stanju procesa podignule su razinu kontrole nad procesom. Ostvarena je agregacija ključnih poslovnih podataka procesa iz različitih naslijeđenih sustava u BPMA. Promjene u toku poslovnog procesa omogućene su promjenom modela procesa, do razine do koje to dozvoljava implementacija u naslijeđenim sustavima. Postupci preventivnih eskalacija detektiraju problematične situacije u toku poslovnih procesa te potiču preventivno djelovanje, prije prijave problema od strane klijenta. Postignuto je bolje razumijevanje poslovne okoline, povećana je kvaliteta komunikacije s klijentima, a sakupljeni podaci o procesu predstavljaju temelj za daljnju optimizaciju.

Originalni postupak preslikavanja metrika, u sustavu temeljenom na predloženom modelu integracije, povezuje metrike definirane u sklopu BPM rješenja s metrikama na razini naslijeđenih aplikacija te tako pruža temelje za praćenje, analizu i optimizaciju procesa.

Sveobuhvatno ispitivanje mišljenja o ideji koja stoji iza predloženog modela integracije, provedeno na širem skupu ispitanika, opravdava temelje na kojima je model izgrađen. Korisnost vlastitog modela u različitim scenarijima iz stvarnog života potvrđena je od strane stručnjaka iz BPM područja. Provedena je rasprava o najprikladnijim poslovnim scenarijima i okruženjima za primjenu modela, a navedeni su i potencijalni nedostaci predloženog modela integracije. Analizirani su osnovni uzroci neuspjeha BPM inicijativa te je ukazano kako predloženi model može pomoći u navedenim slučajevima.

Rad je također predstavio djelomičan odgovor na vječno pitanje, treba li, kada i u kojoj mjeri koristiti postojeće sustave, te u kojim slučajevima se javlja potreba za redizajnom.

Predloženi model integracije implementiran je i provjeravan isključivo u razvojnim i testnim okolinama. Iako je uključivao stvarne produkcijske procese te kopiju produkcijskih podataka, potreban je period kontinuiranog rada u produkcijskim okruženjima za kvalitetniju predodžbu funkcioniranja modela u različitim poslovnim scenarijima.

Za očekivati je da će u budućnosti sustavi za upravljanje poslovnim procesima podržavati razne oblike integracije s drugim sustavima, postojećim poslovnim arhitekturama i strategijama organizacije, tvoreći tako iBPM sustave. Koristit će se usluge poput računarstva u oblacima, analiza procesa u stvarnom vremenu u svrhu podrške inteligentnih poslovnih operacija, interaktivna integracija s društvenim mrežama te mobilnim platformama. Opisani model integracije uspješno se može koristiti u pravcu ostvarenja takvog iBPM rješenja.

Na kraju naglasimo da provedeno istraživanje omogućuje organizacijama da iskoriste predloženi model integracije kao platformu za postizanje veće poslovne vrijednosti, te im tako pomaže da postanu fleksibilnije i usmjerenije na ostvarenje svojih poslovnih ciljeva.

10 Popis literature

[1] Imran Sarwar Bajwa: *SOA Embedded in BPM: A High Level View of Object Oriented Paradigm*, WASET 2011 Spring International Conference, page 304-308. Tokyo, Japan, WASET, (2011)

[2] Alexandre Perin de Souza, Ricardo J. Rabelo: *An Approach for a more Agile BPM-SOA Integration supported by Dynamic Services Discovery*, 2010 14th IEEE International Enterprise Distributed Object Computing Conference Workshops, page 186-195, DOI: 10.1109/EDOCW.2010.42

[3] Nan Wang, Vincent Lee, *An Integrated BPM-SOA Framework for Agile Enterprise*, ACIIDS'11 Proceedings of the Third international conference on Intelligent information and database systems - Volume Part I, Pages 557-566, Springer-Verlag Berlin, Heidelberg 2011. DOI: 10.1007/978-3-642-20039-7_56

[4] Jasmine Noel, "*BPM and SOA: Better Together*", IBM White Paper, 2005, pp. 1-12.

[5] J. Huewitz, R. Bloor, C. Baroudi, M. Kaufman, "*Service Oriented Architecture for Dummies*", Wiley Publishing, Inc., Indianapolis, Indiana, 2007

[6] Gheorghe MATEI; *SOA and BPM, a Partnership for Successful Organizations*, Informatica Economică vol. 15, no. April 2011.

[7] Imran Sarwar Bajwa, Rafaqut Kazmi, Shahzad Mumtaz, M. Abbas Choudhary, and M. Shahid Naweed: *SOA and BPM Partnership: A paradigm for Dynamic and Flexible Process and I.T. Management*, World Academy of Science, Engineering and Technology 45, 2008.

[8] Wen ZhenHua, Huang Yousen, Deng ZiYun, Zhang Wei: *SOA – BPM Based Information System for Promoting Agility of Third Party Logistic*, International Conference on Automation and Logistics, Shenyang, China 5-7 Aug. 2009, Page(s): 248 – 252, DOI: 10.1109/ICAL.2009.5262920

[9] Fuhua Ge, Shaowen Yao: "*Architecture combining SOA and BPM*", Institute of Electrical and Electronics Engineers, Jan 27, 2011.

[10] Anirvan Saha, Rohan Kudav: "*Business Process Management for Successful Core Banking Implementations*", Cognizant Business Consulting, http://www.cognizant.com/InsightsWhitepapers/Business-Process-Management-for-Successful-Core-Banking-Implementations.pdf, srpanj 2013.

[11] Howard Smith & Peter Fingar: "*Workflow is just a Pi process*", BPTrends January, 2004.

[12] Paul Harmon: *The Scope and Evolution of Business Process Management*; Handbook on Business Process Management 1, 2010, Part I.

[13] Stephen A. White : *Using BPMN to Model a BPEL Process*, BPTrends, March 2005.

[14] Nivedita P Deshmukh: *Leveraging BPM Discipline To Deliver Agile Business Processes In Emerging Markets*, 2013 IEEE International Conference on Business Informatics; 2013; 338-345; DOI:10.1109/CBI.2013.55

[15] Michael Hammer: *What is Business Process Management?*; Handbook on Business Process Management 1, 2010, Part I.

[16] Salman Akhtar, Haleem Vaince: *Getting the Process of BPMS Right: "The Need for an Implementation Methodology"*, www.bptrends.com, svibanj 2008.

[17] Christopher Hahn, Till J. Winkler; Fabian Friedrich, Gerrit Tamm, Konstantin Petruch: *How to Choose the Right BPM Tool: A Maturity-Centric Decision Framework with a Case Evaluation in the European Market.* In: EMISA. 2012. p. 109-122.

[18] Laura Mooney: *5 Steps to Choosing the Right BPM Suite.* Metastorm Inc. 2009 http://www.i-nnovate.com/docs/5_Steps_to_Choosing_the_Right_BPM_Suite.pdf, veljača 2017.

[19] Scott Simmons, Michael Steele: *BPM Voices, Synchronicity: An agile approach to business process management*, IBM DeveloperWorks, February 2012.

[20] Nathaniel Palmer: *Building a Business Case for BPM – a Fast Path to Real Results*, OPENTEXT White Paper, ožujak 2012, https://www.opentext.com/file_source/OpenText/en_US/PDF/ot-bps-building-business-case-BPM-whitepaper.pdf, rujan 2017.

[21] John Jeston, Johan Nelis: *Business Process Management - Practical Guidelines to a sucesfull implementation*, ISBN-13: 978-0750686563, ISBN-10: 0750686561

[22] Claus Torp Jensen: *BPM Voices: Insight at the edge*, IBM developer works, Veljača 2013.

[23] Imanipour Narges, Talebi Kambiz, Rezazadeh Siavash; *Obstacles in BPM implementation and adoption in SMEs*; University of Tehran, 2012. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1990609 [17 February 2016]

[24] Hemant Kogekar: *Think Tank: Tips for a successful Business Process Management implementation*,

http://www.cio.com.au/article/320253/think_tank_tips_successful_business_process_management_implementation/?, svibanj 2013.

[25] H. Smith, P. Fingar: *Business Process Management (BPM): The Third Wave*, Tampa, FL: Meghan-Kiffer Press, 2003, ISBN: 0929652339

[26] Jeremy Westerman: *The Case for Business Process Management*, BPTrends, travanj 2009.

[27] Derek Miers: *Best practice BPM*, ACM QUEUE March 2006, https://dl.acm.org/citation.cfm?id=1122688, travanj 2017.

[28] Anirvan Saha, Rohan Kudav: *Business Process Management for Successful Core Banking Implementations*, Cognizant Business Consulting, http://www.cognizant.com/InsightsWhitepapers/Business-Process-Management-for-Successful-Core-Banking-Implementations.pdf, srpanj 2013.

[29] Tiziana Margaria, Bernhard Steffen: *Business Process Modelling in the jABC: The One-Thing-Approach*, Handbook of Research on Business Process Modeling str. 1-27, 2009.

[30] *IBM Business Process Management: A Prescriptive Guide to Solution Design and Implementation*, ftp://www.redbooks.ibm.com/redbooks/REDP4543/2009_08_31_BPM_Prescriptive_Guide.pdf, svibanj 2013.

[31] John Bergland, Luc Maquil, Kiet Nguyen, Chunmo Son: *BPM Solution Implementation Guide*, IBM RedBook, www.ibm.com/redbooks, lipanj 2013.

[32] Doug McIlroy: *Mass Produced Software Components;* 1st International Conference on Software Engineering, January 1968.

[33] Wei-Dong Zhu, Eric Adel, William Benjamin, Imtiaz A Khan, Mike Marin, Mark Yingling: *Introducing IBM FileNet Business Process Manager*, IBM Redbooks, http://www.redbooks.ibm.com/redbooks/pdfs/sg247509.pdf, lipanj 2013.

[34] A Short History of BPM, Column 2 https://column2.com/category/bpmhistory/, veljača 2017.

[35] Elaine A. Carvalho, Tatiana Escovedo, Rubens N. Melo; *Using Business Processes in System Requirements Definition*, 33rd Annual IEEE Software Engineering Workshop, 2009. 125-130, DOI: 10.1109/SEW.2009.8

[36] Rosenzweig B., Rakhimov E. (2009); *Oracle PL/SQL by Example Fourth Edition*, Boston: Pearson Education, Inc.

[37] P. Trkman; *The Critical Success Factors of Business Process Management*, International Journal of Information Management, No. 30(2):125-134, April 2010; DOI: 10.1016/j.ijinfomgt.2009.07.003

[38] Marinela Mircea: "*Adapt Business Processes to Service Oriented Environment to Achieve Business Agility*", Journal of Applied Quantitative Methods; Vol. 5 Issue 4, p679, December 2010

[39] Li, Qian; Chen, Yu; and Zhang, Lanfang; *An Apparel Trade Quotation Architecture Based on BPM and SOA*; PACIS 2009 Proceedings. Paper 115. http://aisel.aisnet.org/pacis2009/115, Veljača 2016.

[40] Bazán Patricia, Roxana Giandini, Gabriela Perez, Javier Diaz; *Process-Service Interactions using a SOA-BPM-based Methodology*, Chilean Computer Science Society (SCCC), 2011 30th International Conference, 9-11 Nov. 2011, Page(s): 100 – 107; DOI: 10.1109/SCCC.2011.14

[41] Matej Hertiš, Matjaž B. Jurič, *Ideas on Improving the Business-IT Alignment in BPM Enabled by SOA*, 2013 International Conference on Information and Communication Technology (ICoICT), 20-22 March 2013, Page(s):55-60, DOI: 10.1109/ICoICT.2013.6574549

[42] S. Al Aloussi: *SLA Business Management Based on Key Performance Indicators*, Proceedings of the World Congress on Engineering 2012 Vol III WCE 2012, July 4 - 6, 2012, London, U.K

[43] Indeevari Ramanayake: *Service Oriented Architecture (SOA)*, https://www.slideshare.net/IndeevariRamanayake1/soa-71539009, ožujak 2017.

[44] Wil M.P. van der Aalst, Arthur H.M. ter Hofstede, and Mathias Weske: *Business Process Management: A Survey*, in Business Process Management, Proceedings 1st Int. Conference. Springer Verlag, 2003.

[45] Fred A. Cummins: *BPM Meets SOA*; Handbook on Business Process Management 1, 2010, Part I.

[46] OASIS SOA Reference Model TC, *https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm*, ožujak 2017.

[47] Razvan Daniel Zota, Liviu Ciovica: *Designing Software Solutions Using Business Processes*, Procedia Economics and Finance Volume 20, 2015, Pages 695-699, DOI: 10.1016/S2212-5671(15)00125-2

[48] J.B. Hill, M. Cantara, M. Kerremans, D.C. Plummer: *Magic Quadrant for Business Process Management Suites*. Technical report, Gartner Research, 2009.

[49] Jim Sinur, Janelle B. Hill: *Magic Quadrant for Business Process Management Suites*, Gartner RAS Core Research Note G00205212, 18 October 2010, RA2 10182011

[50] Rob Dunie, W. Roy Schulte, Marc Kerremans, Michele Cantara: *Magic Quadrant for Intelligent Business Process Management Suites*, Gartner Research Note, ID: G00276892, 18 August 2016

[51] Clay Richardson with Alex Cullen, Shaun McGovern, Diane Lynch: *The Forrester Wave™: BPM Platforms For Digital Business*, Q4 2015, November 18, 2015.

[52] Clay Richardson: *The Forrester Wave™: Business Process Management Suites*, Q3 2010, August 24, 2010

[53] FinancesOnline: *Comparison of 15 Leading Business Process Management Software Products*, https://financesonline.com/comparison-15-leading-business-process-management-software-products/, Kolovoz 2017.

[54] Capterra: *Top Business Process Management Software Products*, http://www.capterra.com/business-process-management-software/, Kolovoz 2017.

[55] Gartner: *Reviews for Business Process Management Platforms*, https://www.gartner.com/reviews/market/BusinessProcessManagementPlatforms, Kolovoz 2017.

[56] Solutions Review: *All New 2017 BPM Buyer's Guide*, https://solutionsreview.com/business-process-management/free-bpm-buyers-guide/, Kolovoz 2017.

[57] Wikipedia, *The Free Encyclopedia: Business Process Management*, https://en.wikipedia.org/wiki/Business_process_management, Rujan 2017.

[58] [21-N] IBM Company: *Overview of IBM Business Process Manager*, ftp://ftp.software.ibm.com/software/integration/business-process-manager/library/ibpm_overview_pdf.pdf, April 2017.

[59] Niel Kolban: *Kolban's Book on IBM Business Process Management*, December 2014.

[60] Ahmed Abdel-Gayed, Kulvir Singh Bhogal, Don Carr, Richard Davies, Aditya P Dutta, Marcelo Correia Lima, Agueda Martinez, Hernandez Magro, Yuka Musashi, Michael Norris, Felix Pistorius: *Implementing an Advanced Application Using Processes, Rules, Events, and Reports*, IBM RedBook, www.ibm.com/redbooks, rujan 2012., SG24-8065-00

[61] John Bergland, Luc Maquil, Kiet Nguyen, Chunmo Son: *BPM Solution Implementation Guide*, IBM RedBook, www.ibm.com/redbooks, lipanj 2013.

[62] Object Management Group: *Business Process Model and Notation (BPMN) V2*, http://www.omg.org/spec/BPMN/2.0, listopad 2016.

[63] M. Owen, J. Raj: *BPMN and Business Process Management An Introduction to the New Business Process Modeling Standard*, Telelogic White papar, Version 1.1, 26 July 2005.

[64] Lisa Dyer, Flournoy Henry, Ines Lehmann, Guy Lipof, Fahad Osmani, Dennis Parrott, Wim Peeters, Jonas Zahn: Scaling BPM Adoption: From Project to Program with IBM Business Process Manager, IBM RedBook, www.ibm.com/redbooks, lipanj 2013.

[65] Hausotter, A., Koschel, A., Zuch, M., Busch, J., Hödicke, A., Pump, R., Seewald, J., Varonina, L.: *Applied SOA with ESB, BPM, and BRM - Architecture Enhancement by Using a Decision Framework*, in: Westphall, C.M., de Barros, M. (Hrsg.), Proc. SERVICE COMPUTATION 2016: The Eighth International Conferences on Advanced Service Computing, IARIA: Rome, Italy, S. 20-27. March 2016.

[66] Nihan Çatal, Daniel Amyot, Wojtek Michalowski, Mounira Kezadri-Hamiaz, Malak Baslyman, Szymon Wilk, Randy Giffen: *Supporting process execution by interdisciplinary healthcare teams: Middleware design for IBM BPM*; Procedia Computer Science, Volume 113, 2017, Pages 376-383, https://doi.org/10.1016/j.procs.2017.08.350, September 2017.

[67] Ananias Laftsidis: *Enterprise Application Integration*, IBM Sweden, DOI: 10.1.1.200.6603, Chapter 15, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.200.6603&rep=rep1&type=pdf, ožujak 2018

[68] Florence Lin: *Enterprise Application Integration (EAI) Techniques*, March 2005., http://www0.cs.ucl.ac.uk/staff/ucacwxe/lectures/3C05-04-05/EAI-Essay.pdf, ožujak 2018.

[69] Michael Rosemann, Jan vom Brocke: The Six Core Elements of Business Process Management; Handbook on Business Process Management 1, 2010, Part I.

[70] Wil M.P. van der Aalst, Mariska Netjes, Hajo A. Reijers: Supporting the Full BPM Life-Cycle Using Process Mining and Intelligent Redesign; Eindhoven University of Technology, Department of Technology Management, 2007.

[71] Vince Kellen: Business Performance Measurement: At the Crossroads of Strategy, Decision-Making, Learning and Information Visualization, E-Commerce DePaul University, Chicago, IL, DOI:10.1.1.105.7194

[72] Milosavljević, G., Sladić, G., Milosavljević, B., Zarić, M., Gostojić, S., Slivka, J.: Context-sensitive Constraints for Access Control of Business Processes. Computer Science and Information Systems, Vol. 15, No. 1., DOI: 10.2298/CSIS160628037M, 2018.

[73] Stevan Gostojić, Goran Sladić, Branko Milosavljević & Zora Konjović: Context-Sensitive Access Control Model for Government Services, Journal of Organizational Computing and Electronic Commerce, 22:2, 184-213, DOI: 10.1080/10919392.2012.667717, 2012.

[74] Srini Penchikala: Domain Driven Design and Development In Practice, InfoQ, https://www.infoq.com/articles/ddd-in-practice, sječanj 2019.

[75] Gartner: SSA Research Note SPA-401-068, 12 April 1996, 'Service Oriented' Architectures, Part 1, https://www.gartner.com/doc/302868?ref=ddisp, sječanj 2019

[76] Gartner: SSA Research Note SPA-401-069 , 12 April 1996, 'Service Oriented' Architectures, Part 2, https://www.gartner.com/doc/302869?ref=ddisp, sječanj 2019

11 Popis kratica

AdES - *Advanced Electronic Signature*

API - *Application Programming Interface*

B2Bi - *Business To Business integration*

BAM - *Business Activity Monitoring*

BPDM - *Business Process Definition Metamodel*

BPM - *Business Process Management*

BPMA - *Business Process Management Application*

BPEL - *Business Process Execution Language*

BPMI - *Business Process Management Initiative*

BPML - *Business Process Modeling Language*

BPMN - *Business Process Model and Notation*

BPMS - *Business Process Managment Suite/Software/Systems*

BPSS - *Business Process Specification Schema*

BRM - *Business Rules Management*

CMM - *Capability Maturity Model*

COBAC - *Context-sensitive Access Control Model for business procesess*

CORBA - *Common Object Request Broker Arhitecture.*

COM - *Component Object Model*

CRM - *Customer Relationship Management*

CSS - *Cascading Style Sheets*

DDD - *Domain-driven design*

DMS - *Document Management System*

DPP – *Dijagram Poslovnog Procesa*

EAI - *Enterprise Application Integration*

EPML - *Event-Driven Process Chains Markup Language*

ESB - *Enterprise Service Bus*

FINA - *Financijska Agencija*

HAL - *Hypertext Application Language*

HNB - *Hrvatska Narodna Banka*

HTTP - *Hypertext Transfer Protocol*

IBPM - *IBM Business Process Management*

iBPM - *Intelligent Business Process Management*

IDE - *Integrated Development Environment*

IS – *informacijski sustavi*

IT – *informacijske tehnologije*

JAR - *Java ARchive*

JAXB - *Java Architecture for XML Binding*

JDK - *Java Development Kit*

JPA - *Java Persistence API*

JSP - *JavaServer Page*

KPI - *Key Performance Indicator*

LDAP - *Lightweight Directory Access Protocol*

LTPA - *Lightweight Third-Party Authentication*

MPP - *Modeliranje Poslovnih Procesa*

MVC - *Model View Controller*

OASIS - *Organization for the Advancement of Structured Information Standards*

OIT – *Odjel informacijskih tehnologija*

ORM - *Object to Relational Mapping*

PP - *Poslovni Proces*

RBAC - *Role-based Access Control*

REST - *Representational State Transfer*

RDBMS - *Relational database management system*

RMI - *Remote Method Invocation*

RPC - *Remote Procedure Call*

SLA - *Service Level Agreement*

SOAP - *Simple Object Access Protocol*

SoD - *Separation of Duty*

SPA – Single Page Application

SSO - *Single Sign On*

OAS - *OpenAPI Specification*

SMTP - *Simple Mail Transfer Protocol*

UDDI - *Universal Description Discovery and Integration*

URL - *Uniform Resource Locator*

VA poslužitelj - *Validation Authority poslužitelj*

WAS - *WebSphere Application Server*

WSDL - *Web Services Description Language*

XML - *EXtensible Markup Language*

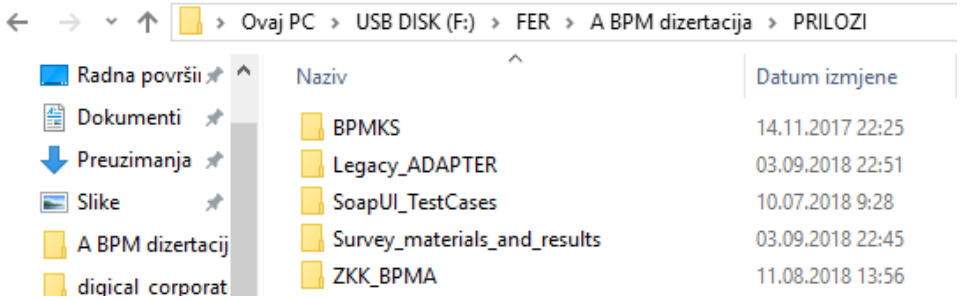
XSD - *XML Shema Definition*

XWSS - *XML and Web Services Security*

ZKK - *Zahtjev za korištenje kredita*

12 Prilozi

Ovo poglavlje sadrži pregled važnijih materijala, programskih rješenja, te koji su korišteni i proizvedeni tijekom izrade ovog doktorskog rada i provedbe pripadnih istraživanja. Materijali dolaze kao prilog ovoj disertaciji na USB mediju (Slika 12.1).



Slika 12.1 Važniji materijali kao prilozi na CD mediju

Materijali uključuju:

- Implementaciju BPMKS sloja (***BPMKS*** mapa). Sadrži cjelokupni programski kod BPMKS aplikacije sa svim integriranim alatima (Maven, Swagger, Codegen, jaxb2, itd.), modulima, komponentama i metodama koje su opisane u sklopu ovog rada (poglavlje 5.3). Dolazi u strukturi projekta programskog alata IntelliJ IDEA.
- Opsežan skup testnih scenarija za SOAP i REST module BPMKS programskog rješenja (***SoapUI_TestCases*** mapa). Testovi su pisani korištenjem SoapUI alata, te uključuju pozive servisa izloženih na BPMKS-u. Strukturirani su u skladu s primjerom PP-a ZKK te sadrže poslovne podatke relevantne za navedeni proces. Uz pomoć SoapUI testova, te BPMA ZKK-a, moguće je simulirati izvođenje dijelova procesa ZKK bez korištenja naslijeđenih aplikacija.
- Procesnu aplikaciju procesa ZKK modeliranu u IBM BPM Designeru (v.8.5.0.1) alatu (***ZKK_BPMA*** mapa). Aplikacija je eksportirana u dva različita fromata; *.twx* formatu koji sadrži sve procesno relevantne artefakte (definiciju PP-a s DPP-a, *coach* i *CV* komponentama, aktivnostima, BPMAPO, servisima, itd.) te u *BPMN 2.0* format-u (*.zip* datoteka) koji nudi prikaz toka procesa na višoj razini te se može koristiti u alatima koji podržavaju uvoz *BPMN 2.0* modela procesa.
- Primjer prilagodnika iz legacy aplikacije sa svim komponentama, klasama, te primjerima osnovnih metoda opisanih u poglavlju 5.4 (***Legacy_ADAPTER*** mapa). Uključeni su primjeri dva prilagodnika:

- ručno izglađenog temeljem principa, metoda i primjera koji su prikazani u poglavlju 5.4 (***BUILT_adapter_AIBPS*** mapa)
- Automatski generiranog uz pomoć Swagger Codegen alata na način opisan u poglavlju 5.4.2 (***GENERATED_Swager_Codegen_adapter*** mapa)

- Materijale korištene u provođenju istraživanja (obrazac za prezentaciju), te detaljne rezultate anketa i istraživanja iz 8. poglavlja (***Survey_materials_and_results*** mapa).

13 Životopis

Mladen Matejaš rođen je 26. svibnja 1985. u Zaboku. Nakon završene matematičke gimnazije u Krapini, 2003. godine upisuje studij na Fakultetu elektrotehnike i računarstva u Zagrebu. U listopadu 2008. diplomira na profilu diplomirani inženjer računarstva obranom diplomskog rada pod nazivom: "*Komponentni pristup razvoju aplikacija u heterogenim sustavima*".

Po završetku studija zapošljava se u Privrednoj Banci Zagreb, u sektoru za aplikativnu podršku, gdje sudjelujući na raznim projektima te uz svakodnevne izazove radi sve do danas. Tijekom zaposlenja pohađao je na nekoliko konferencija u zemlji i inozemstvu, sudjelovao na projektu Europske Unije "*HEUREKA-spoznajom do uspjeha*", te završio više internih te vanjskih edukacija i radionica iz raznih područja.

Poslijediplomski doktorski studij na Fakultetu elektrotehnike i računarstva u Zagrebu upisuje u studenom 2011. godine ponukan zanimljivošću i željom za upoznavanjem područja upravljanja poslovnim procesima, što odabire i kao temu istraživanja. 2013. godine odrađuje kvalifikacijski doktorski ispit, a 2014. polaže zadnji ispit, odrađuje javni razgovor te mu je odobrena teme doktorskog rada.

2015. objavljuje konferencijski rad pod naslovom "*Using Agility in Building Business Process Management Solutions*" s međunarodnom recenzijom, te obavlja prezentaciju rada na WASET konferenciji u Londonu. Tijekom danjeg istraživanja objavljuje rad u časopisu A kategorije vezan uz područje doktorskog rada pod nazivom "*Building a BPM Application in an SOA-based Legacy Environment*".

Vrlo se dobro služi engleskim jezikom u govoru i pismu, i u osnovama koristi njemački jezik.

Popis radova:

Science Citation Index (SCI) indeksiran rad:

Matejaš Mladen, Fertalj Krešimir: ***Building a BPM Application in an SOA-Based Legacy Environment***. Computer Science and Information Systems, Vol. 16, No. 1, 45-74. (2019), https://doi.org/10.2298/CSIS171005010M (članak, znanstveni).

Konferencijski rad s međunarodnom recenzijom:

Fertalj, Krešimir, Matejaš, Mladen: ***Using Agility in Building Business Process Management Solutions***, 17th International Conference on Computer Science and Information Technology. WASET, 2015. 1788-1794 (predavanje, međunarodna recenzija, objavljeni rad, znanstveni).

14 Biography

Mladen Matejaš was born on May 26, 1985 in Zabok. After completing the mathematical high school in Krapina, in 2003 he enrolled at the Faculty of Electrical Engineering and Computing at the University of Zagreb. He received dipl. eng. of computing degree, in the field Computer Science in October 2008, by defending his graduate thesis entitled: "*Component Approach to Application Development in Heterogeneous Systems*".

Since February 2009. he has been employed at the Privredna Banka Zagreb in the Application Development department, where he participated on different software development and system integration projects. During her employment, he attended several conferences domestic and international, participated in the European Union funded project "*HEUREKA-knowledge to success*", and completed several workshops in various fields.

In November 2011 he enrolled doctoral studies of Computer Science at the Faculty of Electrical Engineering and Computing with an interest in studying the field of Business Process Management. In 2013, he passed the doctoral qualification exam and in 2014. held a public presentation of his doctoral thesis, where the doctoral thesis title was approved.

In 2015. he published an international reviewed article entitled "*Using Agility in Building Business Process Management Solutions*" and held a presentation of the research at WASET conference in London. During his further research, he published a research paper on "*Building a BPM Application in an SOA-based Legacy Environment*" in an A category journal.

He uses English very well in and possesses basic communication skills in German language.