UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
U NOVOM SADU

Sebastijan Kaplar

Proširivo procesno okruženje sa podrškom za upravljanje adaptabilnim poslovnim procesima

— doktorska disertacija —

Novi Sad, 2023.

УНИВЕРЗИТЕТ У НОВОМ САДУ **ОБРАЗАЦ – 5а**
НАВЕСТИ НАЗИВ ФАКУЛТЕТА ИЛИ ЦЕНТРА

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА[1]

Врста рада:	Докторска дисертација
Име и презиме аутора:	Себастијан Каплар
Ментор (титула, име, презиме, звање, институција)	Проф. др Мирослав Зарић, ванр. проф.
Наслов рада:	Прошириво процесно окружење са подршком за управљање адаптабилним пословним процесима
Језик публикације (писмо):	Српски (латиница)
Физички опис рада:	Страница: 127 Поглавља: 5 Референци: 104 Табела: 0 Слика: 78 Графикона: 0 Прилога: 0
Научна област:	Електротехничко и рачунарско инжењерство
Ужа научна област (научна дисциплина):	Инжењерство пословних процеса
Кључне речи / предметна одредница:	Пословни процеси, моделовање пословних процеса, радни ток, софтверско инжењерство
Резиме на језику рада:	Предмет истраживања дисертације припада области софтверског инжењерства и пословних процеса, односно управљања пословним процесима. Управљење пословним процесима припада области која је изузетно динамична како у погледу раста захтева и очекивања са којима се суочава са једне стране, тако и нових приступа и технологија који покушавају да премосте и реше уочене проблеме са друге стране. Модели процеса, података, као и алати на којима су засновани процесно оријентисани информациони системи имају задатак да обезбеде контролисано управљање и предвидиво извршавање пословних процеса. Основна премиса за функционисање оваквих система је да пословна функција може бити представљена као процес и/или радни ток – оркестрирани ланац активности (задатака), који када се изврше у дефинисаном редоследу дају успешан исход процеса.

[1] Аутор докторске дисертације потписао је и приложио следеће Обрасце:
5б – Изјава о ауторству;
5в – Изјава о истоветности штампане и електронске верзије и о личним подацима;
5г – Изјава о коришћењу.
Ове Изјаве се чувају на факултету у штампаном и електронском облику и не кориче се са тезом.

	Прилагођавање променама у једном пословном процесу је постала једна од најзначајнијих активности у савременим пословним информационим системима. Међутим, ефикасна примена ове активности у контексту савремених пословних процесно- оријентисаних система је донекле ограничена начинима имплементације самих процесних система, који најчешће праве стриктну дистинкцију између модела процеса и инстанци креираних на основу таквог модела. Оваква имплементација, иако потпуно валидна и широко прихваћена, ипак има и велики недостатак – слабу могућност адаптације већ покренутих инстанци процеса на нове околности, које се могу рефлектовати на неопходне измене у самом процесу. Редизајнирање, усаглашавање и потврђивање валидности и измењеног модела процеса, и имплементација нове верзије и даље је далеко бржи поступак него измена логике уграђене у сам код апликације. Циљ ове дисертације, на основу поменутог, је креирање техничког решења које ће пружати подршку за адаптабилност процеса у процесно-оријентисаним пословним системима, као и одговор на питање, како и у којој мери је могуће применити адаптабилност на текућим процесима у процесно оријентисаним пословним системима. Два проблема у значајној мери утичу на усвајање адаптабилних процесно- оријентисаних пословних система. Први проблем се односи на комплексност ове активности. Измена модела процеса и креирање нове верзије која ће се користити за будуће инстанце је поступак који је уобичајен, добро познат и у широкој употреби. Али адаптација која се пропагира тренутно и на већ покренуте инстанце процеса носи много више непознаница и потенцијалних проблема. Неопходно је сагледати тренутно стање извршавања сваке појединe инстанце, извршити евалуацију примењивости у контексту сваке од тих инстанци. Потребно знање за примењивање оваквог приступа, односно креирање решења са подршком за адаптабилност процеса подразумева и могућност аналитичког сагледавања последица које таква измена може имати на процесне инстанце које се већ извршавају. Неопходно је доста искуства како би се са сигурношћу предвидело да ли имплементација неке нове промене у већ покренутој инстанци не доводи у питање само извршавање процеса. Процесна окружења која подржавају адаптабилност би стога требала да пруже алат који омогућава да се ове провере спроведу пре него што до нежељених последица дође. Други проблем настаје због недостатка постојања конкретне парадигме која би описала начин примене ове активности, односно случајеви који захтевају примену адаптабилности се разликују и специфични су у зависности од циљаног пословног система. Циљ ове дисертације, на основу поменутог, је креирање техничког решења које ће пружати подршку за адаптабилност процеса у процесно-оријентисаним пословним системима, као и одговор на питање, како и у којој мери је могуће применити адаптабилност на текућим процесима у процесно оријентисаним пословним системима.
Датум прихватања теме од стране надлежног већа:	15.7.2021.
Датум одбране:	

Чланови комисије: (титула, име, презиме, звање, институција)	**Председник**: др Горан Сладић, редовни професор, Факултет техничких наука, Нови Сад **Члан**: др Бранко Маркоски, редовни професор, Факултет техничких наука, Нови Сад **Члан**: др Гордана Милосављевић, редовни професор, Факултет техничких наука, Нови Сад **Члан**: др Владимир Вујовић, ванр. професор, Електротехнички факултет, Источно Сарајево **Ментор**: др Мирослав Зарић, ванр. професор, Факултет техничких наука, Нови Сад
Напомена:	

UNIVERSITY OF NOVI SAD
FACULTY OR CENTER

KEY WORD DOCUMENTATION[2]

Document type:	Doctoral dissertation
Author:	Sebastijan Kaplar
Supervisor (title, first name, last name, position, institution)	Miroslav Zarić, PhD, assoc. prof.
Thesis title:	Extensible Workflow Engine With Support for Adaptable Business Process Management
Language of text (script):	Serbian language (latin)
Physical description:	Pages: 127 Chapters: 5 References: 104 Tables: 0 Illustrations: 78 Graphs: 0 Appendices: 0
Scientific field:	Electrical and computer engineering
Scientific subfield (scientific discipline):	Business Process Modeling
Subject, Key words:	Business processes, Workflow Management, BPMN
Abstract in English language:	Research topic of this dissertation is from the area of software engineering and business processes and business process management. An ever-increasing demand for information systems in the last few decades brought many new opportunities but also presented new challenges for companies in the landscape of doing business globally. In these terms, software systems need to adapt almost instantaneously to new requirements, opportunities, and customer expectations. Process-aware systems have become integral part in business information system solutions, finding their place at different locations. Process and data models, as well as tools on which process-oriented information systems are based, have the task of ensuring controlled management and predictable execution of business processes. The basic premise for the functioning of such systems is that the business function can be represented as a process and/or workflow - an orchestrated chain of activities (tasks), which when performed in a defined sequence, give a

[2] The author of doctoral dissertation has signed the following Statements:
5б – Statement on the authority,
5в – Statement that the printed and e-version of doctoral dissertation are identical and about personal data,
5г – Statement on copyright licenses.
The paper and e-versions of Statements are held at he faculty and are not included into the printed thesis.

	successful process outcome. Adapting to changes in a business process has become one of the most important activities in modern business information systems. However, the effective application of this activity in the context of modern business assessment-oriented systems is somewhat limited by the implementation methods of the process systems themselves, which usually make a strict distinction between process models and instances created on the basis of such models. Two problems significantly affect the adoption of adaptable process-oriented business systems. The first problem is related to the complexity of this activity. Modifying a process model and creating a new version to be used for future instances is a procedure that is common, well-known, and widely used. But the adaptation that propagates currently and to already started instances of the process carries many more unknowns and potential problems. It is necessary to look at the current state of execution of each individual instance, to evaluate the applicability in the context of each of those instances. A lot of experience is necessary in order to predict with certainty whether the implementation of a new change in an already started instance does not call into question the execution of the process itself. Process environments that support adaptability should therefore provide tools that enable these checks to be carried out before unintended consequences occur. Another problem arises due to the lack of existence of a concrete paradigm that would describe the way of applying this activity, that is, the cases that require the application of adaptability differ and are specific depending on the targeted business system. The goal of this dissertation, based on the aforementioned, is to create a technical solution that will provide support for process adaptability in process-oriented business systems, as well as an answer to the question of how and to what extent it is possible to apply adaptability to ongoing processes in process-oriented business systems.
Accepted on Scientific Board on:	15.7.2021.
Defended:	
Thesis Defend Board: (title, first name, last name, position, institution)	**President**: Goran Sladić, PhD, Full Professor, Faculty of Technical Sciences, Novi Sad **Member:** Branko Markoski, PhD, Full Professor, Faculty of Technical Sciences, Novi Sad **Member:** Gordana Milosavljević, PhD, Full Professor, Faculty of Technical Sciences, Novi Sad **Member:** Vladimir Vujović, PhD, Associate Professor, Faculty of Electrical Engineering, East Sarajevo **Mentor:** Miroslav Zarić, PhD, Associate Professor, Faculty of Technical Sciences, Novi Sad
Note:	

Sadržaj

Rezime

Predmet istraživanja disertacije pripada oblasti softverskog inženjerstva i poslovnih procesa, odnosno upravljanja poslovnim procesima.

Upravljenje poslovnim procesima pripada oblasti koja je izuzetno dinamična kako u pogledu rasta zahteva i očekivanja sa kojima se suočava sa jedne strane, tako i novih pristupa i tehnologija koji pokušavaju da premoste i reše uočene probleme sa druge strane.

Modeli procesa, podataka, kao i alati na kojima su zasnovani procesno orijentisani informacioni sistemi imaju zadatak da obezbede kontrolisano upravljanje i predvidivo izvršavanje poslovnih procesa. Osnovna premisa za funkcionisanje ovakvih sistema je da poslovna funkcija može biti predstavljena kao proces i/ili radni tok – orkestrirani lanac aktivnosti (zadataka), koji kada se izvrše u definisanom redosledu daju uspešan ishod procesa. Tokom sedamdesetih i osamdesetih godina prošlog veka, fokus je bio na podacima-upravljanom pristupu, gde je modelovanje podataka bila inicijalna aktivnost za kreiranje informacionog sistema. Fokus informacionih tehnologija je primarno bio na skladištenju, čitanju i prezentovanju podataka, odnosno informacija.

Pomenuti pristup doveo je do toga da je sama dinamika sistema - poslovni procesi koji se izvršavaju u organizacijama, bila donekle zanemarivana. Ovo je imalo za posledicu da je poslovna logika, koju informacioni sistem primenjuje transformišući uočene domenske podatke kako bi pratio izvršenje radnih operacija, bila razuđena u aplikacijama informacionih sistema. Ovakva implementacija rezultuje u informacionim sistemima koji dobro i detaljno modeluju domenske podatke, ali malo ili nimalo pažnje posvećuju sagledavanju i modelovanju procesa, i posledično, znatno otežavaju prilagođavanje sistema promenama koje nastaju u domenu samog poslovnog procesa.

Prilagođavanje promenama u jednom poslovnom procesu je postala jedna od najznačajnijih aktivnosti u savremenim poslovnim informacionim sistemima. Međutim, efikasna primena ove aktivnosti u kontekstu savremenih poslovnih procesno-orijentisanih sistema je donekle ograničena načinima implementacije samih procesnih sistema, koji najčešće prave striktnu distinkciju između modela procesa i instanci kreiranih na osnovu takvog modela. Ovakva implementacija, iako potpuno validna i široko prihvaćena, ipak ima i veliki nedostatak – slabu mogućnost adaptacije već pokrenutih instanci procesa na nove okolnosti, koje se mogu reflektovati na neophodne izmene u samom procesu. Redizajniranje, usaglašavanje i potvrđivanje validnosti

izmenjenog modela procesa, i implementacija nove verzije i dalje je daleko brži postupak nego izmena logike ugrađene u sam kod aplikacije. Ipak, nemogućnost adaptacije već pokrenutih instanci na ove nove uslove je limitirajući faktor za primenu procesno orijentisanih sistema u izrazito dinamičnim poslovnim okruženjima. Kako je u poslednje dve decenije došlo do znatnog ubrzanja u svim vidovima privrednih grana, ovo je uslovilo i relativno čestu potrebu izmene ustanovljenih poslovnih procesa, kako bi se poslovni subjekti što bolje adaptirali na izmene na tržištu. Ovo posledično dovodi i do sve veće potrebe za fleksibilnim procesno-orijentisanim sistemima.

Dva problema u značajnoj meri utiču na usvajanje adaptabilnih procesno-orijentisanih poslovnih sistema. Prvi problem se odnosi na kompleksnost ove aktivnosti. Izmena modela procesa i kreiranje nove verzije koja će se koristiti za buduće instance je postupak koji je uobičajen, dobro poznat i u širokoj upotrebi. Ali adaptacija koja se propagira trenutno i na već pokrenute instance procesa nosi mnogo više nepoznanica i potencijalnih problema. Neophodno je sagledati trenutno stanje izvršavanja svake pojedine instance, izvršiti evaluaciju primenjivosti u kontekstu svake od tih instanci. Potrebno znanje za primenjivanje ovakvog pristupa, odnosno kreiranje rešenja sa podrškom za adaptabilnost procesa podrazumeva i mogućnost analitičkog sagledavanja posledica koje takva izmena može imati na procesne instance koje se već izvršavaju. Neophodno je dosta iskustva kako bi se sa sigurnošću predvidelo da li implementacija neke nove promene u već pokrenutoj instanci ne dovodi u pitanje samo izvršavanje procesa. Procesna okruženja koja podržavaju adaptabilnost bi stoga trebala da pruže alat koji omogućava da se ove provere sprovedu pre nego što do neželjenih posledica dođe.

Drugi problem nastaje zbog nedostatka postojanja konkretne paradigme koja bi opisala način primene ove aktivnosti, odnosno slučajevi koji zahtevaju primenu adaptabilnosti se razlikuju i specifični su u zavisnosti od ciljanog poslovnog sistema.

Cilj ove disertacije, na osnovu pomenutog, je kreiranje tehničkog rešenja koje će pružati podršku za adaptabilnost procesa u procesno-orijentisanim poslovnim sistemima, kao i odgovor na pitanje, kako i u kojoj meri je moguće primeniti adaptabilnost na tekućim procesima u procesno orijentisanim poslovnim sistemima.

Posebnu zahvalnost za pomoć i podršku tokom rada na istraživanju i na pisanju ove teze dugujem svom mentoru dr Miroslavu Zariću. Takođe, zahvaljujem se uvaženom profesoru dr Stéphane Ducasse-u koji je nesebično podelio svoje znanje i ideje koje su ucrtale put ove disertacije. Zahvaljujem se i svojoj porodici na nesebičnoj podršci tokom izrade ove teze.

Ovu tezu posvećujem svojim ćerkama Sofiji i Emili.

Sebastijan Kaplar

Abstract

Research topic of this dissertation is from the area of software engineering and business processes, business process management.

An ever-increasing demand for information systems in the last few decades brought many new opportunities but also presented new challenges for companies in the landscape of doing business globally. In these terms, software systems need to adapt almost instantaneously to new requirements, opportunities, and customer expectations. Process-aware systems have become integral part in business information system solutions, finding their place at different locations.

Business process management is in an extremely dynamic field, both in the terms of the growth of demands and expectations that it faces on the one hand, and new approaches and technologies that try to bridge and solve perceived problems on the other hand.

Process and data models, as well as tools on which process-oriented information systems are based, have the task of ensuring controlled management and predictable execution of business processes. The basic premise for the functioning of such systems is that the business function can be represented as a process and/or workflow - an orchestrated chain of activities (tasks), which when performed in a defined sequence, give a successful process outcome. During the seventies and eighties of the last century, the focus was on a data-driven approach, where data modeling was the initial activity for creating an information system. The focus of information technology was primarily on storing, reading and presenting data, that is, information.

The mentioned approach led to the fact that the very dynamics of the system - the business processes that are carried out in organizations - were somewhat neglected. This had the effect that the business logic, which the information system applies by transforming the observed domain data to track the execution of work operations, was dispersed in the information system applications. This kind of implementation results in information systems that model domain data well and in detail, but pay little or no attention to process visualization and modeling, and consequently make it much more difficult to adapt the system to changes that occur in the domain of the business process itself.

Adapting to changes in a business process has become one of the most important activities in modern business information systems. However, the effective application

of this activity in the context of modern business assessment-oriented systems is somewhat limited by the implementation methods of the process systems themselves, which usually make a strict distinction between process models and instances created on the basis of such models. This kind of implementation, although completely valid and widely accepted, still has a major drawback - a weak possibility of adaptation of already started instances of the process to new circumstances, which can be reflected in necessary changes in the process itself. Redesigning, reconciling and validating the changed process model, and implementing the new version is still a much faster process than changing the logic embedded in the application code itself. However, the inability to adapt already started(running) instances to these new conditions is a limiting factor for the application of process-oriented systems in highly dynamic business environments. In the last two decades there has been a significant progress in all types of economic branches, which has caused a relatively frequent need to change established business processes, in order for business entities to better adapt to changes in the market. This consequently leads to an increasing need for flexible process-oriented systems.

Two problems significantly affect the adoption of adaptable process-oriented business systems. The first problem is related to the complexity of this activity. Modifying a process model and creating a new version to be used for future instances is a procedure that is common, well-known, and widely used. But the adaptation that propagates currently and to already started instances of the process carries many more unknowns and potential problems. It is necessary to look at the current state of execution of each individual instance, to evaluate the applicability in the context of each of those instances. The necessary knowledge for applying this approach, that is, creating a solution with support for process adaptability, also implies the possibility of analytically looking at the consequences that such a change can have on process instances that are already being executed. A lot of experience is necessary in order to predict with certainty whether the implementation of a new change in an already started instance does not call into question the execution of the process itself. Process environments that support adaptability should therefore provide tools that enable these checks to be carried out before unintended consequences occur.

Another problem arises due to the lack of existence of a concrete paradigm that would describe the way of applying this activity, that is, the cases that require the application of adaptability differ and are specific depending on the targeted business system.

The goal of this dissertation, based on the aforementioned, is to create a technical solution that will provide support for process adaptability in process-oriented business systems, as well as an answer to the question of how and to what extent it is possible to apply adaptability to ongoing processes in process-oriented business systems.

Sebastijan Kaplar

Poglavlje 1

Uvodna razmatranja

Tokom poslednje decenije procesno-orijentisani poslovni sistemi dobili su na velikom značaju u kontekstu poslovnih informacionih sistema. Upravljanje poslovnim procesima i poslovno-orijentisani poslovni sistemi pripadaju oblasti koja je izuzetno dinamična. Dinamična u pogledu rasta zahteva i očekivanja sa kojima se suočava sa jedne strane, i novih pristupa i tehnologija koji pokušavaju da premoste i na kraju reše novonastale probleme sa druge strane. *Dumas* i *Aalst* navode da je jedan od najvećih izazova organizacija u današnjem okruženju da se ideje i koncepti transformišu u proizvode i usluge u sve bržem tempu današnjice [1]. Ovi sistemi spadaju u sisteme koji mogu da budu deo malih informacionih sistema, koji opslužuju nekoliko desetina korisnika dnevno, pa do ogromnih sistema koji opslužuju nekoliko desetina hiljada korisnika i zahteva na dnevnom nivou. Dakle, poslovni procesi predstavljaju ono što kompanije rade bilo da isporučuju uslugu ili proizvod svojim mušterijama, a način na koji su procesi dizajnirani i izvršeni utiče kako na kvalitet usluge koje opažaju mušterije, tako i na efikasnost sa kojom su usluge isporučene [2].

Pored toga poslovni procesi imaju direktan uticaj na privlačnost proizvoda i usluga, korisničko iskustvo kao i na prihod u slučaju korporacija [2]. Porast korišćenja interneta doneo je nove mogućnosti ali i nove izazove za korporacije odnosno njihove informacione sisteme. Veličina potencijalnih tržišta i količina relevantnih podataka koji se obrađuju zahtevaju inovativna rešenja u informacionim sistemima da bi mogli da se nose sa ogromnim količinama podataka koji se generišu iz dana u dan. Dodatno na globalnom tržištu od softverskih sistema se očekuje gotovo trenutno prilagođavanje novim zahtevima, prilikama i očekivanjima korisnika. Ta ekstremna dinamika je veoma nezgodan problem za tradicionalno napravljene softverske sisteme i zbog toga mnoga softverska rešenja gotovo uvek kasne barem jedan korak u odnosu na postojeće korisničke zahteve.

Zbog navedenog tradicionalni pristup razvoju softvera [3] u kome se detaljno diskutuje o poslovnoj logici između naručilaca softvera (korisnika) i softverskih inženjera, da bi se poslovni zahtevi (eng. *business requirements*) preveli u tehničku

specifikaciju za inženjere i onda putem implementacije (isključivo kroz kod) u krajnje rešenje - skoro sigurno neće doneti očekivane rezultate i biti održiv u dužem vremenskom periodu, jer će se vremenom količina neimplementiranih zahteva u izrazito dinamičkim sistemima nagomilavati.

Modelom upravljan (eng. *Model-Driven*) [4,5] pristup razvoju softvera smanjuje jaz između analize poslovnih zahteva i isporuke softvera tako što pruža alate koji lako mogu da prevedu modele iz poslovnog domena u izvršive softverske module - time skrativši vreme razvoja softvera. Međutim, pored domenskih modela, još jedan važan aspekt je poslovna logika koja definiše korake koji se u programu izvršavaju tokom obrade podataka iz domenskog modela. Poslovna logika je tradicionalno kodirana. Pokušaji da se poslovna logika izvuče iz koda i da se prikaže kroz prikladne, lako razumljive i u isto vreme izvršive modele su uslovili uspon procesno-svesnih informacionih sistema [6].

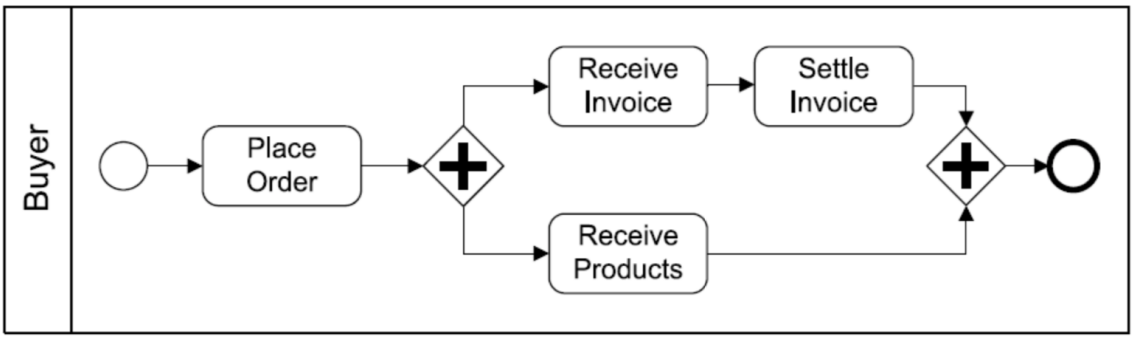
Modeli procesa, podataka, kao i alati na kojima su zasnovani procesno orijentisani informacioni sistemi imaju zadatak da obezbede kontrolisano upravljanje i predvidivo izvršavanje poslovnih procesa. Osnovna premisa za funkcionisanje ovakvih sistema je da poslovna funkcija može biti predstavljena kao proces i/ili radni tok (eng. *worfklow*) - orkestrirani lanac aktivnosti (zadataka), koji kad se izvrše u definisanom redosledu daju uspešan ishod procesa. Poslovni proces može biti modelovan na takav način da je njegova reprezentacija lako razumljiva naručiocima softvera kao i da se lako mašinski interpretira. Nekoliko jezika (notacija) za modelovanje poslovnih procesa su se pojavili tokom vremena, većina njih je zasnovana na grafovima. Modelovanje procesa, sa svojom bogatom grafičkom reprezentacijom omogućava naručiocima softvera da lako izraze korake koji su potrebni da se obave neke poslovne funkcije, identifikuju zadatke i da potvrda da je logika izvršavanja procesa tačna. Takav model se potom prevodi u izvršivi oblik za softverski sistem [7].

Na Slici 1.1. prikazan je primer jednog poslovnog procesa. Poslovni procesi najčešće su predstavljeni putem BPMN (*Business Process Modeling Notation*). BPMN predstavlja grafičku notaciju za specifikaciju poslovnih procesa i kreiran je od strane OMG[1](*Object Management Group*) grupe sa ciljem da bude lako razumljiv za sve poslovne korisnike, od poslovnih analitičara koji kreiraju inicijalne nacrte procesa do programera koji kreiraju tehnička rešenja koja će izvršavati te procese i na kraju do onih koji će vršiti nadgledanje i upravljanje tim procesima [8].

Pored toga, još jedan cilj BPMN-a je da obezbedi mogućnost lako razumljive notacije za sve korisnike koji su uključeni u postupku sprovođenja poslovnih procesa. BPMN je poslovno orijentisan ka XML (*Extensible Markup Language*) [9] jezicima koji mogu da skladište informacije o procesnim modelima i procesnim dijagramima, slično kao što je WSBPEL (*Web Services Business Process Execution Language*) [10] zasnovan na XML, a o kome će biti više reči kasnije.

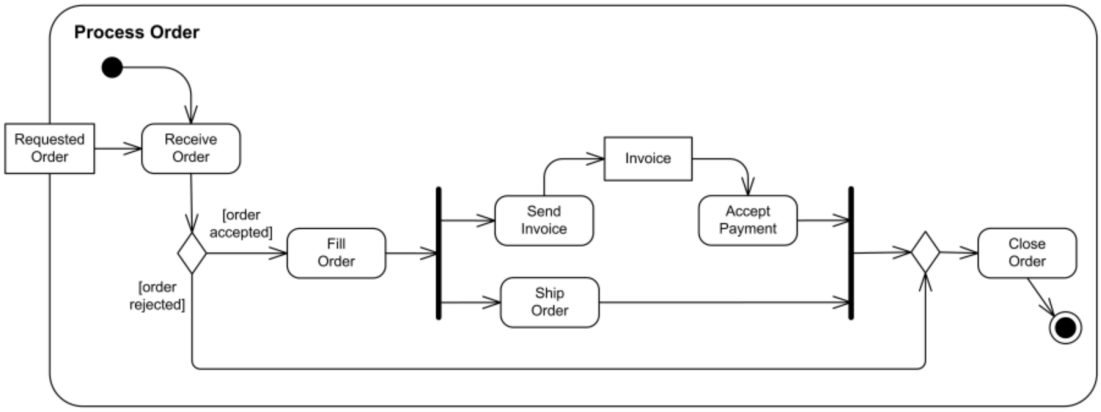
Pored BPMN-a postoje i druge notacije kojima se mogu opisati poslovni procesi kao

[1] www.omg.org



Slika 1.1: Primer poslovnog procesa

što su UML Activity Diagram, IDEF, LOVeM, Event-Process Chains (EPCs) i drugi. UML (*Unified Modeling Language*) je jezik za modelovanje opšte namene, takođe kreiran od strane OMG grupe. Dijagrami aktivnosti (eng. *Activity Diagram*) su dijagrami u UML jeziku, kojima se opisuju dinamički aspekti sistema i jako podsećaju na BPMN. U istraživanju [11] je izvršeno poređenje između BPMN i *UML Activity* dijagrama gde su dobijeni rezultati interesantni, jer istraživanje pokazuje da nema značajne razlike između dva jezika, sa akcentom na tome da je BPMN dizajniran tako da bude lako razumljiv. Primer dijagrama aktivnosti dat je na slici 1.2[2].



Slika 1.2: Primer Activity dijagrama

Proces opisuje način na koji će biti izvršeni određeni slučajevi ili kategorije slučajeva,

[2]Slika preuzeta sa: https://www.uml-diagrams.org/activity-diagrams-examples.html

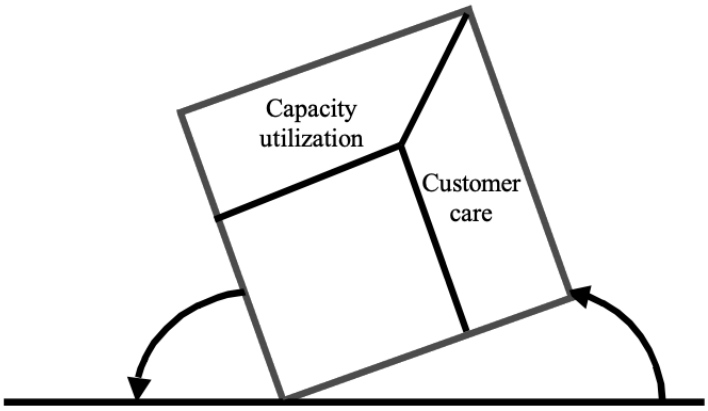
odnosno zadatke koji trebaju biti izvršeni i redosled u kome se oni izvršavaju. Ove aktivnosti zajedno dovode do poslovnog cilja. Poslovni proces jedne organizacije može biti u interakciji sa poslovnim procesima drugih organizacija, međutim poslovni proces se izvršava u jednoj organizaciji. Proces se takođe može posmatrati kao procedura, jer se sastoji od zadataka koji trebaju da se izvrše i skupa uslova koji određuju redosled tih zadataka, generalno jedan proces obrađuje mnogo različitih zadataka što omogućuje specifičan tretman zadataka na osnovu njegovih atributa, te se može reći da se proces sastoji od zadataka i uslova po kojima se određeni zadaci mogu obaviti [12].

Zadatak predstavlja posao koji treba obaviti, ili logičku jedinicu posla i kao takav je nedeljiv, odnosno uvek se izvršava u potpunosti. Zadaci su jedan od najvažnijih elemenata procesa i njihovom pravilnom identifikacijom moguće je strukturirati proces. Grupisanjem zadataka dobijamo složene aktivnosti. Ukoliko zadatak nije jasno definisan, odnosno ukoliko je to nedeljiva jedinica posla već podrazumeva izvršavanje niza jednostavnijih akcija kako bi se zadatak realizovao onda govorimo o složenoj aktivnosti.

Tipičan proces takođe sadrži i rutiranje. Rutiranjem se određuje koji zadatak treba da se izvrši i kojim redosledom, na taj način se određuje tok izvršavanja procesa. Proces se može rutirati na više različitih načina: sekvencijalno, paralelno ili selektivno. Svaki od ovih načina rutiranja objašnjen je u sledećem poglavlju. Na kraju, izvršavanje procesa dovodi do jednog ili više ishoda. Ishodi mogu biti uspešni ili neuspešni, međutim informacije o ishodu procesa bi trebalo da imaju svi učesnici istog.

Aalst navodi da je moderno društvo postalo toliko kompleksno da ga niko u potpunosti više ne može istraživati i da mnogi ljudi ne znaju kakvu ulogu njihov rad ima u celokupnoj šemi stvari (eng. *scheme of things*) [12] - generalni način na koji su stvari organizovane i međusobno povezane. Navedeno dalje povlači gubljene identiteta u poslovnim okruženjima, pogotovo u velikim korporacijama gde zaposleni često ne shvataju zašto moraju da rade stvari koje su im rečene. Takva vrsta otuđenja na poslu utiče na smanjenje produktivnosti i povećanju nezadovoljstva. Zbog navedenih problema, kompanije su počele da organizuju posao tako da njihovi zaposleni jasno razumeju zadatke koje obavljaju i da oni shvate da rade za određene klijente. Nova organizacija posla sa idejom povećanja zadovljstva i produktivnosti zaposlenih predstavlja tranziciju koju organizacije treba da usvoje. Ciljevi tranzicije na posao koji je orijentisan prema mušteriji je da se isprave prethodno uočeni problemi, odnosno da se poveća produktivnost zaposlenih. U [12] *Aalst* to objašnjava činjenicom da smo prešli iz zalihama-vođene ekonomije, gde su sredstva za proizvodnju (eng. *Means of Production*) bila oskudna, u potražnjom-vođenu ekonomiju, gde su zapravo klijenti ti koji su oskudni, odnosno da je došlo do "promene organizacione paradigme", gde se fokus prebacio sa sredstava za proizvodnju na klijente (vidi sliku 1.3). Zbog navedenog, kao rezultat rada na novim tehnologijama i organizaciji posla proistekli su WMS (eng. *Workflow Management Systems*), nova vrsta informacionih sistema koji omogućavaju direktnu vezu između posla koji u kontekstu radnog procesa obavlja određeni zaposleni

i računarskog programa (informacionog sistema) koji upravlja tim jednim procesom.



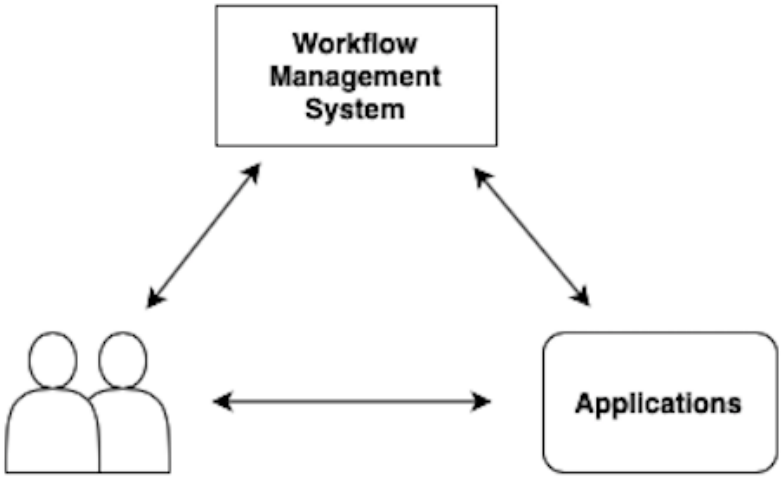
Slika 1.3: Promena organizacione paradigme. [12]

.

Kreiranje i strukturiranje ovakvih sistema nije jednostavan zadatak i većina ovih sistema se oslanja na mogućnosti dekompozicije poslovnih funkcija kao sekvence akcija, koje kad se izvršavaju kontrolisanim redosledom usmeravaju sistem ka uspešnom ishodu. Tokom sedamdesetih i osamdesetih godina prošlog veka, fokus je bio na podacima-upravljanom pristupu, gde je modelovanje podataka bila inicijalna aktivnost za kreiranje informacionog sistema [1]. Fokus informacionih tehnologija je primarno bio na skladištenju, čitanju i prezentovanju podataka, odnosno informacija. Pomenuti pristup je doveo do toga da je sama dinamika sistema - poslovni procesi koji se izvršavaju u organizacijama, bila donekle zanemarivana. Ovo je imalo za posledicu da je poslovna logika, koju informacioni sistem primenjuje transformišući uočene domenske podatke kako bi pratio izvršenje radnih operacija, bila razuđena u aplikacijama informacionih sistema. Ovakva implementacija rezultuje u informacionim sistemima koji dobro i detaljno modeluju domenske podatke, ali malo ili nimalo pažnje posvećuju sagledavanju i modelovanju procesa, i posledično, znatno otežavaju prilagođavanje sistema promenama koje nastaju u domenu samog poslovnog procesa.

Radni tok (eng. *Workflow*) možemo definisati kao automatizaciju radnog procesa u celini, ili delimično [13] - računarski potpomognutim olakšavanjem posla ili dela posla, gde se *workflow* tehnologija koristi da koordinira, kontroliše i izvršava predefinisane operacije na fleksibilan način [14]. *Workflow* sistemi omogućavaju specifikaciju, izvršavanje i kontrolisanje raznih scenarija i slučajeva upotrebe, ova tehnologija je primenjena u različitim domenima kao što su poslovni procesi, naučne aplikacije, *e-learning*, itd. WFMS (eng. *Workflow Management System*) omogućava vezu između korisnika i aplikacije delegirajući mu odgovarajuće zadatke na izvršavanje i to u momentu kada je u radnom procesu neophodan njegov angažman, omogućujući mu

pristup putem odgovarajućih formi za izvršenje specificiranog zadatka (vidi sliku 1.4) [1]. Upravljanje (menadžment) radnim tokom treba da podrži usmeravanje aktivnosti u okviru poslovne organizacije koja može pripadati nekom od navedenih domena. Ishod jednog radnog toka smatra se uspešnim kada je posao završen u predodređeno vreme, od strane zaposlenih pomoću odgovarajućeg softverskog alata. Upravljanje radnim tokom daje odgovor na pitanje "kako" jedan radni tok treba da radi, a ne "šta" treba da radi, dakle interesovanje je usmereno na strukturu radnog procesa. Šta treba da se izvršava u okviru jednog poslovnog procesa, odnosno sadržaj zadataka u okviru procesa ima potporu u različitim aplikativnim programima. Zadatak (eng. *task*) je jedan od najvažnijih koncepata u tehnologijama radnih tokova i njihovom pravilnom identifikacijom moguće je strukturirati radne tokove. Jedan zadatak predstavlja logičku jedinicu posla, nedeljiv je i uvek se izvršava u potpunosti [12]. Na zadatke, u okviru ove disertacije, posebno je obraćena pažnja tokom opisivanja elemenata poslovnog procesa.



Slika 1.4: WFMS kao veza između korisnika i aplikacija. [1]

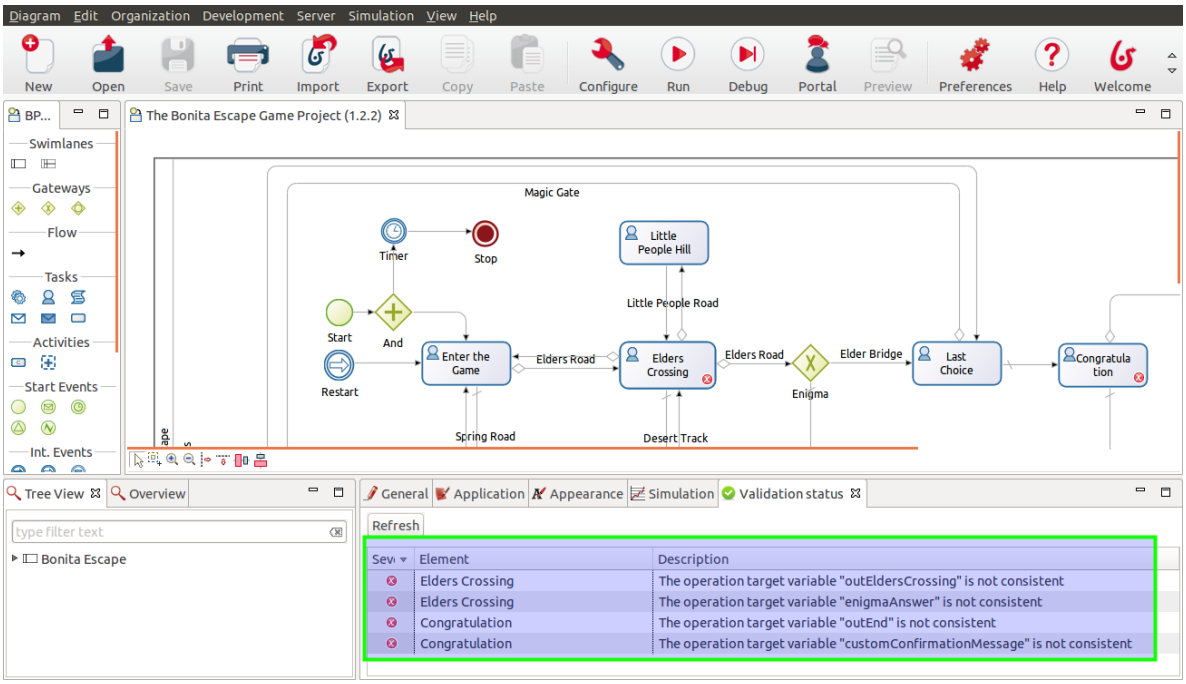
.

Informacioni sistemi koji su zasnovani na WFMS-u, odnosno sistemima koji podržavaju upravljanje poslovnim procesima nazivamo sistemima radnog toka, odnosno *workflow* sistemima. Specifikacija procesa opisuje tip tog procesa i služi kao šablon za kreiranje konkretnih instanci procesa. WFMS podržava poslovne procese kroz njihovo izvršavanje, odnosno WFMS kreira instancu procesa, na osnovu njegove specifikacije, gde instanca procesa predstavlja specifičnu pojavu ili izvršenje jednog poslovnog procesa.

Međutim domen WFMS-a danas je u potpunosti fragmentisan, za svaku svrhu postoje različita rešenja sa različitim specijalizacijama [15]. Pored domena, mnogo ograničenja dolazi i od platforme koja je odabrana za implementaciju jednog takvog sistema.

Kompanije koje se bave tehnologijama radnih tokova prate trendove tržišta i kreiraju rešenja koja omogućavaju modelovanje i izvršavanje različitih poslovnih procesa. Ti alati imaju za zadatak da stručnjacima iz oblasti omoguće da ih koriste samo sa svojim domenskim znanjem, bez potrebe detaljnog poznavanja tehničkih rešenja i njihovih implementacija.

Neki od trenutno najzastupljenijih BPM alata su Kissflow[3], Bonitasoft[4], ProcessMaker[5], Camunda[6]. Međutim većina ovih rešenja su komercijalna i besplatno dostupne verzije su sa veoma ograničenom funkcionalnosti. Izgled Bonitasoft BPM alata data je na Slici 1.5[7].



Slika 1.5: Izgled Bonitasoft BPM

Kreiranje alata za izvršavanje radnih tokova (eng. *Workflow Engine*) je kompleksan

[3]https://kissflow.com
[4]https://www.bonitasoft.com
[5]https://www.processmaker.com
[6]https://camunda.com
[7]Slika preuzeta sa: https://community.bonitasoft.com/bonita-bpm-63-available

zadatak, WfMC (eng. *Workflow Management Coalition*) propisuje referentnu arhitekturu, koja može da se iskoristi kao nacrt za razvoj jednog funkcionalnog takvog alata. Međutim, cilj istraživanja nije kreiranje još jedne implementacije *Workflow Engine*-a sa specifičnom specijalizacijom, nego dizajn jednog proširivog i fleksibilnog razvojnog okruženja koje omogućuje samostalan rad, kao i integraciju sa drugim alatima, kako bi bilo ispunjeno sve što je potrebno za izvršavanje jedne procesno-orijentisane aplikacije, sa mogućim različitim zahtevima/specijalizacijama.

Prilagođavanje promenama u jednom poslovnom procesu je postala jedna od najznačajnijih aktivnosti u savremenim poslovnim informacionim sistemima. Međutim, efikasna primena ove aktivnosti u kontekstu savremenih poslovnih procesno-orijentisanih sistema je donekle ograničena načinima implementacije samih procesnih sistema, koji najčešće prave striktnu distinkciju između modela procesa i instanci kreiranih na osnovu takvog modela. Ovakva implementacija, iako potpuno validna i široko prihvaćena, ipak ima i veliki nedostatak – slabu mogućnost adaptacije već pokrenutih instanci procesa na nove okolnosti, koje se mogu reflektovati na neophodne izmene u samom procesu. Redizajniranje, usaglašavanje i potvrđivanje validnosti izmenjenog modela procesa, i implementacija nove verzije i dalje je daleko brži postupak nego izmena logike ugrađene u sam kod aplikacije. Ipak, nemogućnost adaptacije već pokrenutih instanci na ove nove uslove je limitirajući faktor za primenu procesno orijentisanih sistema u izrazito dinamičnim poslovnim okruženjima. Kako je u poslednje dve decenije došlo do znatnog ubrzanja u svim vidovima privrednih grana, ovo je uslovilo i relativno čestu potrebu izmene ustanovljenih poslovnih procesa, kako bi se poslovni subjekti što bolje adaptirali na izmene na tržištu. Ovo posledično dovodi i do sve veće potrebe za fleksibilnim procesno-orijentisanim sistemima.

1.1 Polazne pretpostavke i ciljevi istraživanja

Dva problema u značajnoj meri utiču na usvajanje adaptabilnih procesno-orijentisanih poslovnih sistema.

Prvi problem se odnosi na kompleksnost ove aktivnosti. Izmena modela procesa i kreiranje nove verzije koja će se koristiti za buduće instance je postupak koji je uobičajen, dobro poznat i u širokoj upotrebi. Ali adaptacija koja se propagira tenutno i na već pokrenute instance procesa nosi mnogo više nepoznanica i potencijalnih problema. Neophodno je sagledati trenutno stanje izvršavanja svake pojedine instance, izvršiti evaluaciju primenjivosti u kontekstu svake od tih instanci. Potrebno znanje za primenjivanje ovakvog pristupa, odnosno kreiranje rešenja sa podrškom za adaptabilnost procesa podrazumeva i mogućnost analitičkog sagledavanja posledica koje takva izmena može imati na procesne instance koje se već izvršavaju. Neophodno je dosta iskustva kako bi se sa sigurnošću predvidelo da li implementacija neke nove promene u već pokrenutoj instanci ne dovodi u pitanje samo izvršavanje procesa.

Procesna okruženja koja podržavaju adaptabilnost bi stoga trebala da pruže alat koji omogućava da se ove provere sprovedu pre nego što do neželjenih posledica dođe.

Drugi problem nastaje zbog nedostatka postojanja konkretne paradigme koja bi propisala način primene ove aktivnosti, odnosno slučajevi koji zahtevaju primenu adaptabilnosti se razlikuju i specifični su u zavisnosti od ciljanog poslovnog sistema. Iz ovoga proističu dva ključna istraživačka problema koje teza adresira:

- Kako kreirati tehničko rešenje koje će pružati podršku za adaptabilnost procesa u procesno-orijentisanim poslovnim sistemima?
- Kako i u kojoj meri primeniti adaptabilnost na tekućim procesima u procesno-orijentisanim poslovnim sistemima?

Polazna pretpostavka (hipoteza) je da se upotrebom modernih programskih alata i jezika, posebno uzimajući u obzir inherentna dinamička svojstva nekih jezika, može implementirati adaptabilni procesni sistem koji će efikasno odgovoriti na ova dva glavna istraživačka problema i podržavati:

- izražavanje dinamike sistema putem modela, po ugledu na postojeće procesno orijentisane sisteme,
- mehanizme i alate koji omogućavaju adaptaciju postojećih modela procesa,
- mehanizme i alate za opis unesenih izmena i analizu primenjivosti izmena na već instancirane procese

Bez obzira na velike napretke koji su se desili u polju informacionih tehnologija u poslednje dve dekade, i zbog promene tržišta sa lokalnog na globalno, koje je uslovilo velike promene u smislu dinamičnosti zahteva sa kojima se susreću korporacije, poslovni sistemi koji pružaju podršku izvršavanju poslovnih procesa, nisu u potpunosti pratili tempo ovih promena. U dostupnoj literaturi [16–19] mogu se naći različite analize i pristupi problemima adaptabilnosti procesa, ali i dalje postoji potreba i prostor za istraživanjima u ovoj oblasti, jer veliki broj radova ukazuje na nerešene probleme u domenu primene adaptabilnih procesa, dok većina dostupnih procesnih sistema i dalje ne pruža standardizovanu podršku za koncept adaptabilnosti. Imajući u vidu da je ova tema od velikog interesa za poslovne informacione sisteme, postoji potreba za istraživanjem koje će ponuditi koncept procesnog okruženja koje pruža podršku za adaptaciju procesa, proveru primenjivosti izmena na postojeće instance procesa i njihovu dinamičku rekonfiguraciju kada je ona moguća, te na taj način pružiti mogućnost odgovora na dinamičke zahteve koje se javljaju u modernim informacionim sistemima. Ovakvo istraživanje treba da prepozna ograničenja i probleme u postojećim procesno-orijentisanim poslovnim sistemima i ponudi odgovore kako te sisteme unaprediti u kontekstu dinamičke adaptabilnosti.

Cilj istraživanja doktorske disertacije jeste empirijski dokaz polazne hipoteze i ciljeva koji su iz nje izvedeni, odnosno:

- Kreiranje tehničkog rešenja (*New Wave* procesna platforma) i radnog okvira za razvoj alata za izvršavanje poslovnih procesa, koji će biti generalizovan, ali istovremeno i adaptabilan.

- Definisanje postupka adaptacije procesa i podrška na njegovo sprovođenje kroz radni okvir.

Tehničko rešenje i radni okvir moguće je primeniti u akademskom i korporativnom okruženju, gde se u akademskom okruženju tehničko rešenje sa radnim okvirom može iskoristiti za dalje napredovanje u oblasti poslovno-informacionih sistema, a u korporativnom za analizu i primenu prilagođavanja nad postojećim procesima.

Drugi rezultat istraživanja, definisanje postupka adaptacije koji je kompatibilan sa savremenim procesima razvoja softvera i procesno-orijentisanim poslovnim informacionim sistemima. Implementacija ovog postupka u radnom okviru ponudiće korisnicima fleksibilan pristup za predstavljanje izmena i razmatranje njihove primenjivosti u kontekstu postojećih procesa koje imaju. Konfigurabilnost procesa zajedno sa uputstvima koje će istraživanje pružiti omogućuje njegovu upotrebu kako za eksperimentalne procese, tako i za postojeće procese u korporativnim okruženjima.

Cilj istraživanja direktno doprinosi razvoju softvera koji podržava adaptabilne procese. Očekivani rezultati se mogu primeniti u akademskoj zajednici kao osnovna za dalji razvoj alata i radnih okvira koji podržavaju adaptabilne poslovne informacione sisteme, kao i u industriji za primenu adaptabilnosti u poslovnim-procesima korporacija.

Struktura disertacije je sledeća:

- Uvodna razmatranja i objašnjenje zašto postoji potreba za jednim ovakvim istraživanjem dati su u ovom poglavlju. Pored toga poglavlje takođe daje uvod i opisuje poslovne procese i ulogu poslovnih procesa u poslovnim informacionim sistemima. Uočeni problemi, kao i predložena rešenja su opisani kroz ciljeve disertacije u sklopu ovog poglavlja.

- Drugo poglavlje predstavlja sistematizaciju i analizu dostupne literature iz oblasti. U sklopu drugog poglavlja su date i opisane ideje sličnih sistema, kao i formalni zaključci šta jedno predloženo rešenje koje pripada ovoj oblasti treba da pruži.

- Treće poglavlje daje uvid u dinamička programska okruženja kao i njihove pogodnosti za realizaciju ciljeva predstavljenih u ovoj disertaciji.

- Četvrto poglavlje opisuje arhitekturu i formalno rešenje kao i opis radnog okvira za izvršavanje poslovnih procesa.

- Peto poglavlje opisuje formalan model za podršku adaptacija u okviru predloženog rešenja, kao i verifikaciju dizajna predloženog rešenja.
- Šesto poglavlje predstavlja zaključak, ostvarene rezultate kao i pravce daljeg istraživanja i razvoja.

Poglavlje 2

Teorijske osnove i pregled stanja u oblasti

2.1 Poslovni procesi

Poslovni procesi predstavljaju srž procesno-orijentisanih poslovnih sistema i od ključne važnosti je da su efikasni, pored toga poslovni procesi pripadaju oblasti koja je izuzetno dinamična i kao takvi moraju da budu spremni da odgovore na zahteve u poslovnim sistemima kompanija. Ukoliko poslovni procesi u sklopu neke kompanije nisu efikasni onda ne donose profit, već naprotiv, mogu da generišu gubitke za kompaniju. Poslovni procesi su ključni u organizovanju ovih aktivnosti i u razumevanju njihovih međusobnih povezanosti. *Hammer* u [20] predlaže radikalan pristup redizajnu poslovnih procesa kompanije. Poslovni proces je sagledan kao kolekcija aktivnosti sa jednim ili više ulaza da bi na kraju, odnosno izlazu proizveo novu vrednost za krajnjeg korisnika. Iako je originalno izdanje knjige napisano 1993 materija kojom se ona bavi je i dalje relevantna i aktuelna. Pored toga *Davenport* u [21] navodi da se procesi moraju dizajnirati i implementirati efikasno i da oslanjanje samo na formulisanje poslovne strategije nije dovoljno za postizanje poslovnih ciljeva. Uspeh je u promeni, u pristupu koji spaja informacione tehnologije i ljudske resurse, usmerene na izvršavanje određenih poslovnih zadataka, odnosno pristupa se inovaciji procesa.

Ključ za uspešnu implementaciju informacionog sistema koji pruža podršku izvršavanju poslovnih procesa je da se poslovni proces može modelovati na takav način da je njegova reprezentacija istovremeno jasna i lako razumljiva naručiocima softvera, inženjerima koji rade na razvoju softvera, i u kontekstu procesno orijentisanih softverskih sistema, da se takav model može lako mašinski interpretirati. Raznovrsna grafička reprezentacija omogućava korisnicima koji su domenski stručnjaci da na jednostavan način izraze korake koji su neophodni da se obave neke poslovne funkcije, identifikuju zadatke i verifikuju tačnost logike izvršavanja procesa.

Pored toga, poslovni procesi su izvesno vreme i u interesu dve grupe profesionalaca

koji su različitog obrazovanja i iz različitih interesnih sfera:

- Menadžera (pripadaju poslovnim upravljačkim strukturama)
- IT Sektora

Menadžeri poboljšavaju i unapređuju način poslovanja njihove kompanije, upravljanjem i organizacijom poslovnih procesa da bi ultimativno dobili bolje rezultate (veća satisfakcija korisnika, nova rešenja, smanjeni troškovi i optimizacija poslovanja).

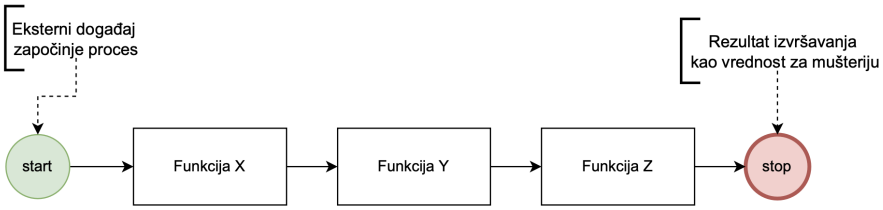
IT sektor je širok pojam, međutim naučni radnici/istraživači i softverski inženjeri su oni na koje se može odnositi ova podela. Zainteresovanost istraživača zasniva se na korišćenju modela i apstrakcija u cilju unapređivanja već postojećih rešenja i novih doprinosa u polju poslovnih procesa, dok su softverski inženjeri zainteresovani na primenu i implementaciju tih doprinosa u cilju kreiranja alata i softverskih sistema za podršku poslovnim procesima.

Razmatrajući poslovne procese dolazi se do zaključka da se svaki proces sastoji od niza *aktivnosti* i *događaja*. Aktivnosti su jedinice posla koje se izvršavaju tokom toka poslovnog procesa. U zavisnosti od stepena kompleksnosti, aktivnosti možemo posmatrati na više načina, ako je aktivnost jednostavna ili predstavlja jednu operaciju, tu aktivnost zovemo *zadatak*. Ako se aktivnost sastoji iz više operacija (zadataka), odnosno više jedinica posla, ona predstavlja složenu aktivnost. Dakle, zadatak je jasno definisana jedinica posla koju obavlja jedan učesnik u procesu, dok aktivnost može biti jasno ili grubo definisana jedinica posla. Događaji su atomičke prirode, odnosno nemaju vreme trajanja, gde pri tome svaki događaj može da uzrokuje lanac drugih događaja. Za razliku od događaja, aktivnosti su te kojima treba vreme za izvršavanje. Da bi tok izvršavanja procesa bio kontrolisan, koriste se tačke odluke (eng. *decision points*), odnosno mesta u procesu na kojima se donosi odluka a od koje zavisi kako će se proces dalje izvršavati.

U proces su uključeni i učesnici (eng. *Actor*), fizički objekti (eng. *Physical objects*) i informacioni objekti (eng. *Informational objects*). Pod učesnikom procesa ne podrazumeva se samo osoba kao učesnik, nego i organizacija ili drugi softverski sistem koji može nastupati kao učesnik procesa umesto osobe ili organizacije. Učesnici mogu biti interni ili eksterni, u smislu da su interni učesnici oni koji rade u organizaciji u okviru koje se izvršava proces. Konzumenti procesa su učesnici koji se nazivaju mušterije (eng. *customer*), koji takođe mogu biti interni ili eksterni za proces. Fizički objekti mogu biti razni dokumenti, proizvodi, oprema i slično, dok su informacioni objekti predstavljeni elektronskim dokumentima, zapisima ili sadržajem.

Na kraju izvršavanje procesa vodi ka jednom ili više ishoda (eng. *outcome*), gde ishod treba da donese neku vrednost učesnicima procesa. Dešava se da ishod procesa nije u potpunosti uspešan, odnosno može doneti parcijalnu vrednost, ili ne mora da donese vrednost uopšte. Ukoliko izvršavanjem procesa nije stvorena nikakva nova vrednost kažemo da proces ima negativan ishod, u suprotnom ima pozitivan ishod.

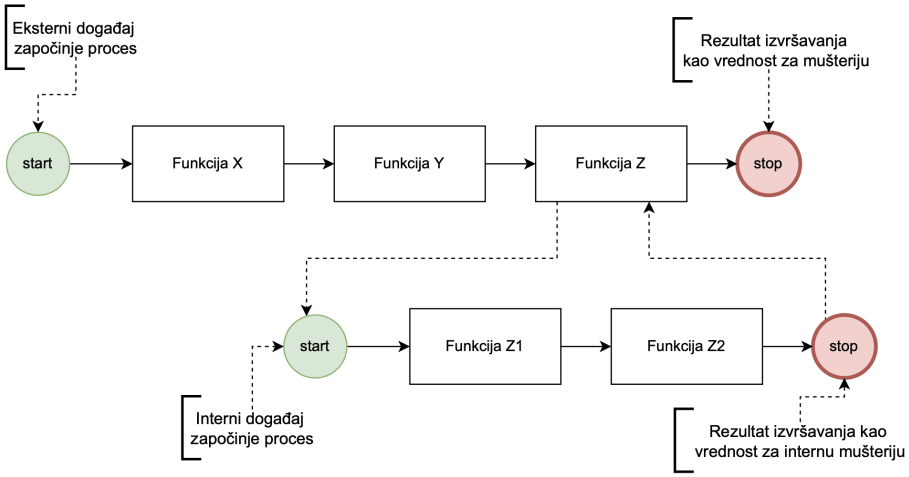
Na osnovu predočenog možemo definisati poslovni proces. *Dumas* i autori u knjizi [1] poslovni proces definišu kao kolekciju međusobno povezanih događaja, aktivnosti i odluka koji uključuju učesnike i objekte i kolektivno vode ka ishodu koji je od vrednosti klijentima. *Weske* u knjizi [22] navodi da se proces sastoji od skupa aktivnosti koje se koordinisano izvršavaju u organizacionom i tehničkom okruženju, i time postižu poslovni cilj. Poslovni proces se izvršava od strane jedne organizacije, međutim on može biti u interakciji sa poslovnim procesima koje obavljaju druge organizacije. *Kiršmer* [23] definiše proces kao set funkcija datih u specifičnoj sekvenci koje konačno isporučuju vrednost za interne ili eksterne klijente, sa naglaskom da je početak procesa definisan eksternim događajem [20, 24–26]. Na osnovu ove definicije moguće je dati i grafički prikaz na slici 2.1.



Slika 2.1: Grafički prikaz definicije procesa

Procesi mogu dodatno da se dekomponuju, odnosno ako je neka funkcija procesa kompleksna ili jednostavno ima više koraka čijim bi se posmatranjem kao individualnih aktivnosti jasnije stekla slika o zadacima koji trebaju da se obave, takva aktivnost se može posmatrati/interpretirati takođe kao poseban proces ili podproces, ovakva dekompozicija u mnogome pomaže detaljnom ispitivanju procesa u celosti (eng. *end-to-end*) [27, 28].Podprocese aktiviraju prethodno aktivni procesi i/ili podprocesi koji služe kao inicijatori (iniciraju neke događaje). Zadatak tih podprocesa je takođe da proizvedu rezultat za nekog učesnika u procesu, bilo da je to za neki naredni podproces ili je to za krajnji ishod procesa, odnosno kako je to navedeno krajnjeg klijenta, primer je ilustrovan na slici 2.2, ovakav pristup ilustruje dekompoziciju procesa na hijerarhije.

Na osnovu predočenih informacija o poslovnom procesu, BPM (*Business Process Management*), odnosno upravljanje poslovnog procesa možemo opisati kao skup raznih metoda i alata koji služe za identifikovanje, prepoznavanje, praćenje i na kraju modifikaciju samog procesa radi poboljšanja osobina, odnosno učinka poslovnog procesa. *Weske* kaže da BPM uključuje koncepte, metode i tehnike u cilju podrške dizajna, administracije, konfiguracije, izvršavanje i analizu poslovnih procesa [22], dok *Dumas* i autori definišu BPM kao telo metoda, tehnika i alata koji identifikuju, otkrivaju,analiziraju,redizajniraju,izvršavaju i nadgledaju poslovni proces u cilju optimizacije njegovih performansi [1].



Slika 2.2: Grafički prikaz definicije procesa sa podprocesom

Jedna od glavnih ideja BPM je da omogući detaljan prikaz poslovnog procesa, što kroz eksplicitnu grafičku reprezentaciju, što kroz alate koji su mu dati na raspolaganju. Detaljnim uvidom u poslovni proces mogu se primetiti njegove aktivnosti, njegov tok kao i ograničenja koja se nameću tokom njegovog izvršavanja u cilju njegovog upravljanja i organizovanja u sklopu jedne organizacije. Naravno, jednom definisani poslovni proces, zbog svoje kompleksne prirode podložan je promenama, odnosno analiziranju i redizajniranju, kao i na kraju izvršavanju, koristeći prethodno navedene postupke.

Tradicionalno, poslovni procesi su se izvršavali manuelno, odnosno velike korporacije se oslanjaju na svoje zaposlene i njihovo poznavanje procedura i regulativa koje su propisane od strane kompanije da bi izvršavali poslovne procese. Kompanije mogu da ostvare velike benefite ukoliko naprave prostor za automatizaciju poslovnih procesa uvođenjem softvera za koordinaciju i izvršavanje aktivnosti koje su sastavni deo poslovnog procesa. Da bi se ostvario benefit nije neophodno automatizovati celi poslovni proces (nekad je to i nemoguće jer postoje akcije koje npr. može samo da uradi ljudska osoba), nego samo jedan ili više delova poslovnog procesa koji su pogodni za tu akciju. Softverski sistemi koji koordinišu aktivnosti koje su uključene u poslovne procese nazivaju se BPMS (eng. *Business Process Management Systems*), odnosno sistemi za upravljanje poslovnim procesima. BPMS predstavljaju generički softverski sistem vođen eksplicitnom reprezentacijom procesa da koordiniše izvršavanje poslovnih procesa [22].

2.1.1 Modelovanje poslovnih procesa

Aktivnosti koje se nalaze u sklopu poslovnog procesa mogu se predstaviti na više načina, npr. tekstualnom reprezentacijom, međutim tekstualna reprezentacija često nije najpogodnija da jasno prikaže redosled izvršavanja aktivnosti (iako je dobar komplement grafičkoj notaciji), mnogo češće se koriste grafičke notacije koje su pogodne za prikazivanje redosleda izvršavanja. Postoji više grafičkih notacija za modelovanje poslovnih procesa, čija je suština veoma slična, međutim u okviru ove disertacije fokus je na BPMN (eng. *Business Process Modeling Notation*) kao jednoj od najzastupljenijih i opšteprihvaćenih notacija.

Da bi se shvatila potpuna slika poslovnog procesa, odnosno njegov kompletan domen, on uključuje različite poddomene, odnosno modelovanje funkcija, podataka, organizacije i modelovanje čitavog pejzaža operativnih informacionih tehnologija [22]. Pomenuti poddomeni pripadaju vertikalnoj apstrakciji u modelovanju poslovnih procesa, odnosno pripadaju (integrišu se) modelovanju poslovnih procesa.

Kao što je već pomenuto veliki benefiti se dobijaju automatizacijom poslovnih procesa. Međutim, automatizacija nije jedina pogodnost koja se dobija modelovanjem poslovnih procesa. Modelovanje omogućava organizacijama jasan uvid u poslovne procese i načine na koji se oni koriste. Različite reprezentacije za opisivanje modela poslovnih procesa omogućavaju analitički pristup poslovnim procesima koji se može iskoristiti za njihovo lakše shvatanje kao i za njihovu optimizaciju, te sama implementacija poslovnog procesa predstavlja glavni nusproizvod njegovog modela.

2.2 Petrijeve mreže

Petrijeve mreže služe kao sredstvo za prikaz i modelovanje dinamičkih sistema, kao i za opisivanje i proučavanje sistema koji su opisani kao konkurentni, asinhroni, distribuirani, paralelni, nedeterministički i/ili stohastički [29]. Karl Adam Petri je jedan od pionira ove oblasti, iz njegove disertacije *Kommunikation mit Automaten* [30] proistekle su Petrijeve mreže, koje predstavljaju usmereni bigraf koji ima dve vrste elemenata, mesta i tranzicije. Na osnovu rada i doprinosa Karla A. Petrija, Petrijeve mreže našle su primenu u raznim sistemima, međutim od posebnog značaja za ovu disertaciju su i procesno-orijentisani poslovni sistemi.

Kao što je pomenuto, Petrijeve mreže su sredstvo koje je primenljivo na više različitih sistema. Grafička reprezentacija Petrijevih mreža može se iskoristiti da opiše dinamičku prirodu ovih sistema, odnosno tokeni kao deo ove mreže se koriste za predstavljanje dinamičkih aktivnosti sistema.

Petrijeva mreža je usmereni, ponderisani bigraf koji se sastoji od dve vrste elementa, mesta (stanja) p i tranzicija t(prelaza stanja). Mesta i tranzicije su naizmenično povezani usmerenim lukovima, odnosno luk povezuje mesto i tranziciju, odnosno tranziciju i mesto. Generalno, u grafičkoj reprezentaciji stanja su opisana krugovima,

tranzicije kvadratima, a lukovi su obeleženi njihovim težinama (prirodnim brojevima). Obeležavanjem dodeljujemo svakom stanju prirodan broj, odnosno ako obeležavanjem dodelimo mestu p prirodan broj n, onda kažemo da je p obeležen sa n *tokena*. Na grafičkoj reprezentaciji se tokeni obeležavaju sa tačkama u nekom mestu S. Na osnovu pomenutog, klasična Petrijeva mreža opisana je kao trojka (P, T,F):

- P - konačan skup mesta (stanja)
- T - konačan skup tranzicija: $P \cap T = \emptyset$
- F - skup lukova: $F \subseteq (P \times T) \cup (T \times P)$

U Petrijevim mrežama jedno od najbitnijih pravila je pravilo omogućavanja tranzicija i njihovog aktiviranja (okidanja), ovo pravilo se koristi da bi se prikazalo stanje u Petrijevoj mreži, odnosno dinamičko ponašanje sistema. Pravilo tranzicije kaže da:

- Neka tranzicija t je omogućena ako svako njeno ulazno mesto p sadrži najmanje $w(p,t)$ tokena, gde je $w(p,t)$ težina usmerenog luka od p ka t (kod elementarnog tipa Petrijeve mreže težina grane je 1, a mesta mogu sadržati samo jedan token).
- Omogućena tranzicija može a ne mora da se aktivira, u zavisnosti od toga da li se događaj zaista izvršio
- Aktiviranjem omogućene tranzicije t uklanja se $w(p,t)$ tokena iz svakog ulaznog mesta p od t i dodaje $w(t,p)$ tokena na svako izlazno mesto p od t, gde je w(t,p) težina usmerenog luka od t ka p.

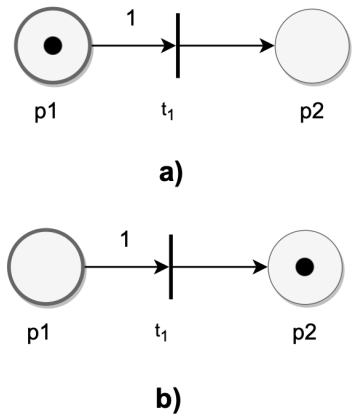
Dakle, mesta su pasivni element mreže i mogu da sadrže tokene, pod određenim uslovima aktiviranjem tranzicije prenosi se određen broj tokena iz prethodnog ka sledećem mestu, gde broj prenesenih tokena zavisi od transportnog kapaciteta, odnosno težine povezujućih lukova. Izvorna (eng. *source*) i završna (eng. *sink*) tranzicija su posebne, jer je izvorna tranzicija ona koja nema ulaznih mesta, a završna ona koja nema izlaznih.

Prilikom modelovanja, koristeći koncepte uslova i događaja, mesta predstavljaju uslove, a tranzicije predstavljaju događaje, odnosno tranzicija (događaj) ima određen broj *ulaznih* i *izlaznih* mesta koji predstavljaju preduslove i post-uslove događaja [29]. Prisustvo tokena u mestu se može posmatrati na više načina, ali najčešće da prisustvo tokena u mestu označava istinitost (ispunjenost) uslova koji je povezan sa tim mestom.

Aalst u svom radu povezuje Petrijeve mreže i upravljanje radnim tokom, gde se Petrijeve mreže već kao oprobani alat za modelovanje i analiziranje procesa koriste kao dizajn jezik za specifikaciju kompleksnih radnih tokova, pored toga Petrijeve mreže pružaju razne tehnike za analizu koje se mogu iskoristiti za verifikaciju korektnosti

procedura radnog toka [31]. Aalst takođe daje najmanje tri dobra razloga za korišćenje Petrijevih mreža za analizu i modelovanje i to su: formalna semantika uprkos grafičkoj prirodi, zasnovane na stanju umesto na događajima i veliki broj tehnika za analizu [32]. Iako su Petrijeve mreže grafički jezik koji dozvoljava modelovanje radnih tokova, njihova semantika je definisana formalno, a stanje slučajeva (eng. *case*) u Petrijevim mrežama se može modelovati eksplicitno, za razliku od nekih drugih tehnika za modelovanje procesa.

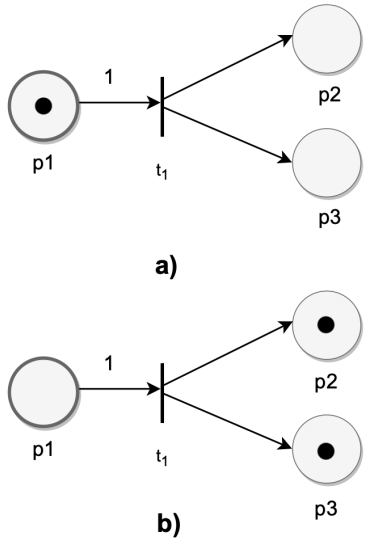
Primer koji ilustruje osnovne elemente Petrijeve mreže prikazan je na slici 2.3. Na primeru videmo mrežu koja se sastoji iz dva mesta *p1, p2* i jedne tranzicije *t1*. Token se nalazi u mestu *p1*, odnosno mesto *p1* je obeleženo sa jednim tokenom. Težina takođe iznosi *1*, odnosno da bi se aktivirala tranzicija *t1*, potrebno je da mesto *p1* sadrži jedan token. Slika 2.3 a), prikazuje mrežu pre, a slika 2.3 b) nakon aktiviranja tranzicije.



Slika 2.3: Osnovni elementi Petri mreže pre i posle okidanja tranzicije

Grananje u Petrijevim mrežama je ilustrovano na slici 2.4. Na slici su prikazana tri mesta *p1,p2,p3* i jedna tranzicija *t1*. Mesto *p1* je obeleženo sa jednim tokenom, a težina neophodna za aktiviranje tranzicije, takođe je jedan. Nakon aktiviranja tranzicije, mesto *p1* više ne sadrži token, dok će mesta *p2,p3* sadržati po jedan token, kao što je prikazano na slici 2.4 a) koja ilustruje stanje pre tranzicije, i 2.4 b) koja ilustruje stanje nakon tranzicije. Bitno je napomenuti da grananja u Petri mrežama mogu polaziti iz tranzicija, čime se implicitno dobija *AND* grananje, ili iz mesta (čime se u osnovnoj verziji impicitno postiže *XOR* grananje)

Poslednji primer ilustruje spajanje u Petrijevim mrežama i ilustrovano je na slici 2.5. Na slici su prikazana tri mesta *p1,p2,p3* i jedna tranzicija *t1*. Mesto *p2* sadrži dva tokena, dok mesto *p3* sadrži jedan token. Da bi se tranzicija aktivirala, potrebna su dva tokena iz *p2* i jedan iz *p3*, nakon čega će se u mestu *p1* sadržati jedan token, a *p2,p3* neće sadržati tokene, kao što je to prikazano na slika 2.5 a) i 2.5 b) respektivno.

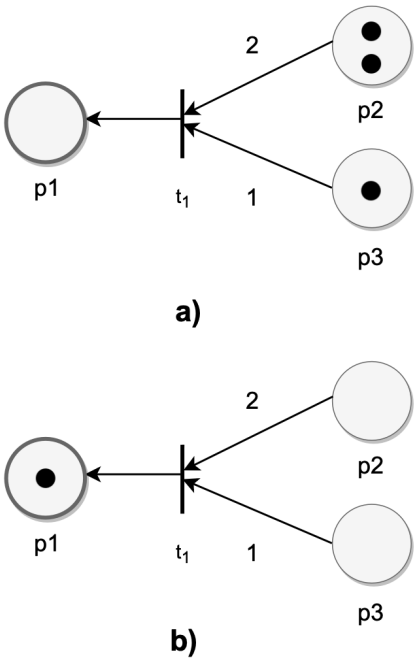


Slika 2.4: Razdvajanje u Petri mrežama

Petrijeve mreže dolaze i sa određenim ograničenjima, zbog čega su se pojavile razne modifikacije elementarnih Petrijevih mreža, koje rešavaju određena ograničenja. Osnovne Petrijeve mreže podrazumevale su jedan token na jednom mestu, gde tokeni nisu imali identitet. Da bi se rešio problem nepostojanja identiteta tokena uvedene su obojene Petrijeve mreže (eng. *Colored Petri Nets*) [33], koje dozvoljavaju tokene različitih tipova i čiji tokeni mogu da sadrže vrednosti različitih tipova podataka. Pomenuta modifikacija je rešila identitet tokena, gde se u smislu poslovnih informacionih sistema jednim tokenom može predstaviti jedna instanca procesa, što omogućava njihovo praćenje. WF mreže (eng. *Workflow nets*), odnosno mreže radnog toka su u suštini elementarne Petrijeve mreže sa jednim ulazim i izlaznim mestom, gde sva mesta i tranzicije moraju biti na jednoj dostupnoj putanji između ulaza i izlaza. Koriste se kao apstrakcija za analiziranje određenih svojstava koje garantuju odsustvo živih zaključavanja (eng. *livelock*), mrtvih zaključavanja (eng. *deadlock*) i drugih anomalija koje se mogu detektivati bez domenskog znanja [34]. Pored navedenih postoje i hibridne [35], rasplinute (eng. *fuzzy*) [36], stohastičke [37] i druge Petrijeve mreže.

2.3 BPMN

Business Process Modeling Notation (BPMN) je inicijalno predstavljen 2004. godine (verzija 1.0), kao standardni jezik za modelovanje poslovnih procesa. Kreiranje BPMN-a je doprinelo smanjenju jaza u fragmentaciji između postojećih alata za modelovanje



Slika 2.5: Spajanje u Petri mrežama

poslovnih procesa i notacija, što je i bio jedan od ciljeva za kreiranje BPMN-a. Od njegovog kreiranja pa do danas BPMN je široko prihvaćen u akademiji, kao i u industriji. Od njegovog začetka, BPMN je prošao kroz jednu značajniju reviziju 2013. godine, kada je predstavljena verzija 2.0, nakon čega je bilo još nekoliko manjih revizija, gde je trenutno aktuelna verzija 2.0.2 koja je predstavljena 2014. godine.

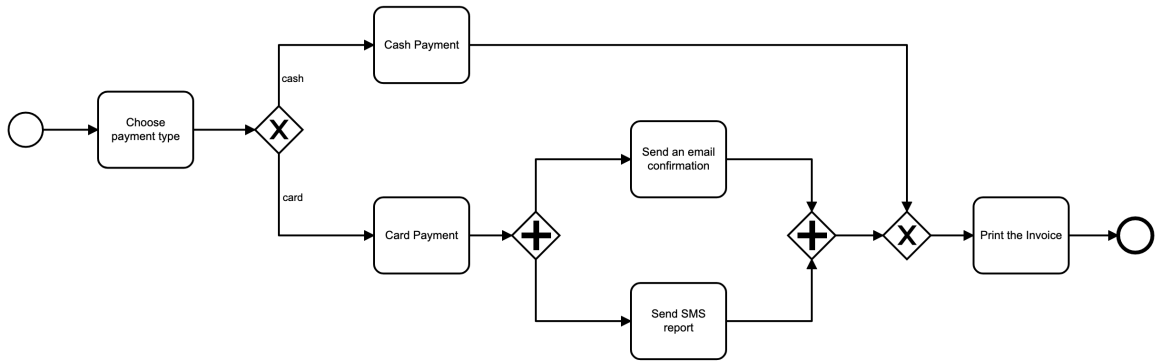
OMG grupa, definisala je za primarni cilj BPMN-a da pruži lako razumljivu notaciju za sve korisnike koji su uključeni u postupak uvođenja poslovnih procesa, od poslovnih korisnika koji su domenski stručnjaci za poslovne procese, tehničkog osoblja koje je zaduženo da sprovede postupak uvođenja softvera koji će izvršavati te procese, i na kraju stručno osoblje koje će nadgledati i rukovati tim procesima [8]. BPMN, kao i UML (eng. *Unified Modeling Language*) koji je kreiran od strane iste grupe, samo za objektno-orijentisani dizajn i analizu, kreirani su sakupljanjem najboljih praksi i prihvaćenih ideja, sa namerom da se te ideje i prakse spoje odnosno integrišu u generalno prihvaćene jezike.

Radni okvir u okviru ove disertacije dizajniran je i implementiran uz oslonac na BPMN, koji kao što je pomenuto predstavlja grafičku notaciju za specifikacija poslovnih procesa, dok je BPMN kreiran sa idejom da bude lako razumljiv za sve korisnike, od prethodno pomenutih pa do svih onih koji su interakciji sa poslovnim procesima. Skup

elemenata koji čine BPMN je ogroman i većina rešenja, uključujući i NewWave alat (koji implementira radni okvir iz ove disertacije) koji se oslanja na BPMN ne podržava sve navedene elemente. Međutim funkcionalno softversko rešenje mora da uključi minimalan skup elemenata koji je neophodan za modelovanje i izvršavanje procesa. U narednim pasusima prikazani su i opisani neki od najbitnijih koncepata BPMN-a.

BPMN je građen tako da njegovo jezgro (eng. *core*) omogućava kreiranje poslovnih procesa jednostavnijih struktura, dok se dodavanjem elemenata omogućava građenje kompleksnijih poslovnih procesa. BPMN je kreiran tako iz razloga što je njegovo jezgro, odnosno set osnovnih elemenata jednostavan i lak za razumevanje, a da se dodavanje novih elemenata može vršiti sekvencijalno, u zavisnosti od potreba procesa, tako da i njegov dizajner može da sekvencijalno usvaja znanje nakon što je u potpunosti ovladao prethodno iskorišćenim elementima procesa.

Na slici 2.6 prikazan je proces koji ilustruje primer plaćanja i sastoji se od glavnih elemenata jezika, a to su: aktivnosti, kapije/grananje (eng. *gateway*), događaji(eng. *events*) i toka sekvence (eng. *sequence flow*).



Slika 2.6: Ilustracija primera plaćanja

Na modelu procesa vidimo da započinje početnim događajem (eng. *start event*), nakon čega sledi aktivnost koja analizira tip plaćanja, odnosno da li će se plaćanje vršiti karticom ili gotovinom, pre nego što se izvrši isključivo grananje (eng. *exclusive gateway*). Isključivo grananje je kapija koja usmerava proces na jednu od svojih izlaznih grana, ova kapija je obično obeležena sa markerom u obliku slova **X**. Paralelno grananje je kapija koja je obeležena znakom + i označava kapiju koja aktivira sve svoje izlazne grane. Iz navedenog primećuje se da je grafička notacija obogaćena nizom atributa koji mogu biti povezani sa određenim elementima procesa, međutim za većinu atributa postoje smernice ali ne i propisani način grafičke reprezentacije dodatnih atributa i to je ostavljeno dizajnerima procesa da prate smernice i ispoštuju data pravila.

Bitnu ulogu u odlučivanju u sklopu BPM-a igraju izrazi (eng. *expressions*), koji se između ostalog koriste da se napišu logički izrazi za odlučivanje koja će grana biti odabrana kod isključivog grananja, odnosno kojom granom će izvršavanje biti nastavljeno. BPMN ne specificira obavezan jezik za izraze (najčešće su to skript jezici), međutim ograničenja u pogledu jezika mogu postojati na nivou dijagrama i alata u kome se rukuje poslovnim procesom, gde važi pravilo da se tu koristi samo jedan jezik koji opisuje izraze (eng. *expression language*). Na osnovu primera datog na slici 2.6 i na osnovu izraza je određeno odabrano plaćanje, odnosno gotovina ili kartica. Elementi i različite kategorije elemenata iz notacije prikazne su na slici 2.7.

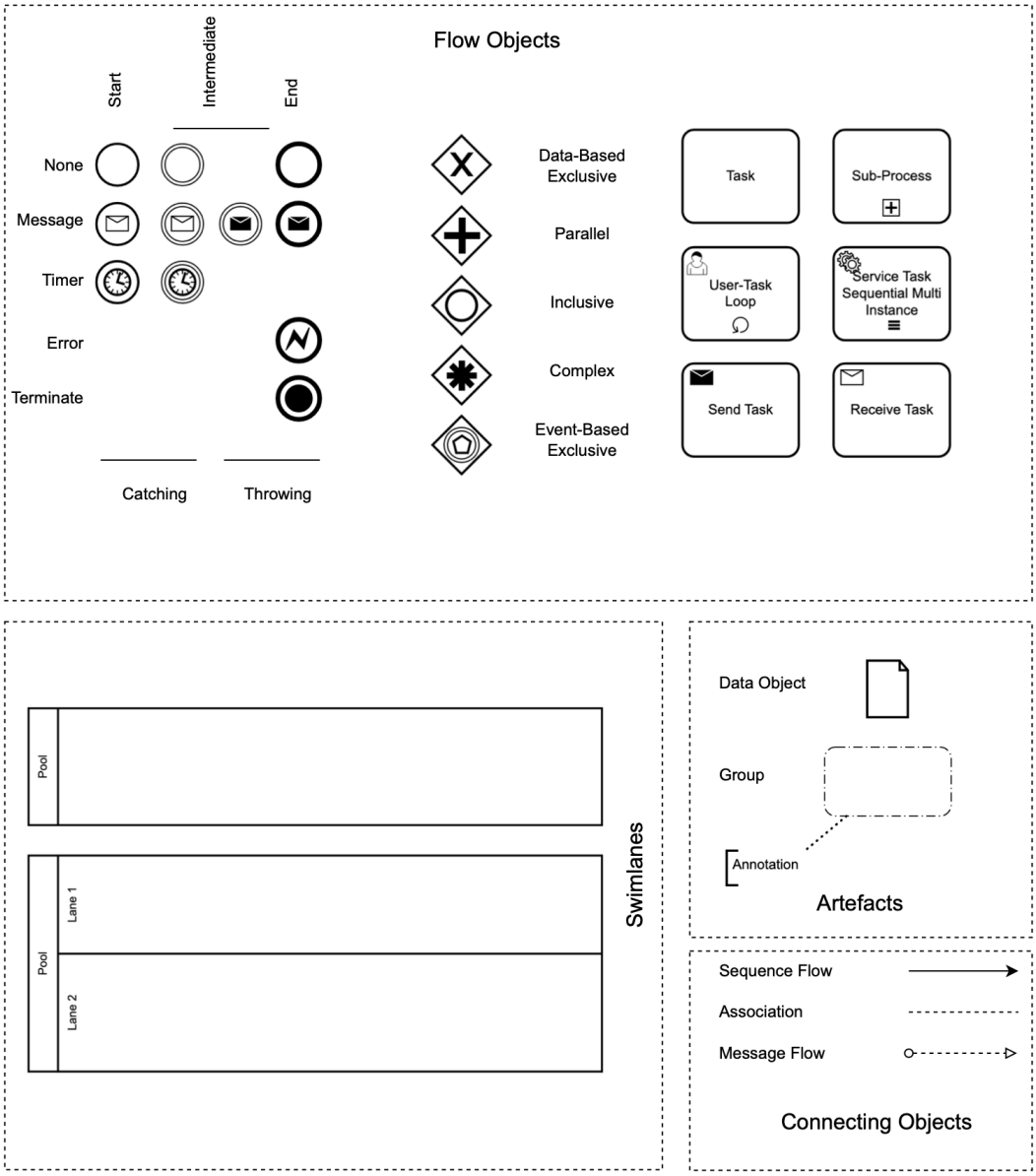
Osnovni gradivni elementi poslovnog procesa pripadaju objektima toka (eng. *flow objects*), objekti toka kao što je to naznačeno na slici 2.7 su događaji, kapije(grananja) i aktivnosti. Događaji u poslovnom procesu opisuju nešto što se desi tokom procesa, često je to neka pojava iz stvarnog sveta koja je relevantna za taj proces ili neki događaj koji se desio u samom procesu, a koji je relavantan i za njegovo okruženje.Postoje tri vrste događaja: početni (eng. *start*), posredni (eng. *intermediate*) i krajnji (eng. *end*).

Na osnovu prikazane slike 2.7 zaključujemo da su plivačkim stazama (eng. *swimlanes*) prikazani organizacioni aspekti poslovnog procesa. Plivačka staza se sastoji iz vertikalnog i horizontalnog dela, vertikalni deo je bazen (eng. *pool*) koji predstavlja organizaciju u okviru koje postoji interakcija između poslovnih procesa i staze (eng. *lane*) koja predstavlja različite entitete te organizacije na primer sektor ili odeljenje u organizaciji, traka se koristi da organizuje i kategorizuje aktivnosti i najčešće jedna staza predstavlja jednu identifikovanu rolu u procesu.

Artefakti (eng. *Artefacts*) služe samo u informacione svrhe, odnosno semantički obogaćuju proces dodatnim informacijama. Artefakti, kao sto je navedeno na slici 2.7 su objekti podataka (eng. *Data Objects*), grupe (eng. *Groups*) i anotacije (eng. *Annotations*) i oni mogu biti pridruženi bilo kom elementu toka procesa jer ne utiču na njih. Data objekti pripadaju celini koja opisuje podatke (eng. *Data*) koja se sastoji od već pomenutih objekata podataka, ulaznih podataka (eng. *Data Inputs*), izlaznih podataka (eng. *Data Outputs*) i skladišta podataka (eng. *Data Stores*). Objekti podataka pružaju informaciju o tome šta aktivnosti zahtevaju da bi se izvršili i/ili šta proizvode, objekat podataka može da predstavlja jedan ili kolekciju objekata, dok Izlazni i ulazni podaci pružaju iste informacije za procese [8].

Tekstualne anotacije, kao što im ime sugeriše dokumentuju određene delove poslovnog procesa u tekstualnoj formi, gde je ta anotacija grafički asocirana (povezana) sa objektom na dijagramu. Grupe nemaju formalno značenje, međutim koriste se da grupiše elemente procesa u dokumentacione svrhe, mogu se prožimati kroz plivače trake i bazene po potrebi.

Povezujući objekti (eng. *Connecting Objects*) povezuju prethodno opisane objekte odnosno objekte toka, plivačke staze i artefakte. Tok sekvence (eng. *Sequence Flow*) prikazuje redosled u kome se aktivnosti izvršavaju u procesu, odnosno prikazuje redosled objekata toka, dok tok poruka prikazuje razmenu poruka između dva učesnika koja se



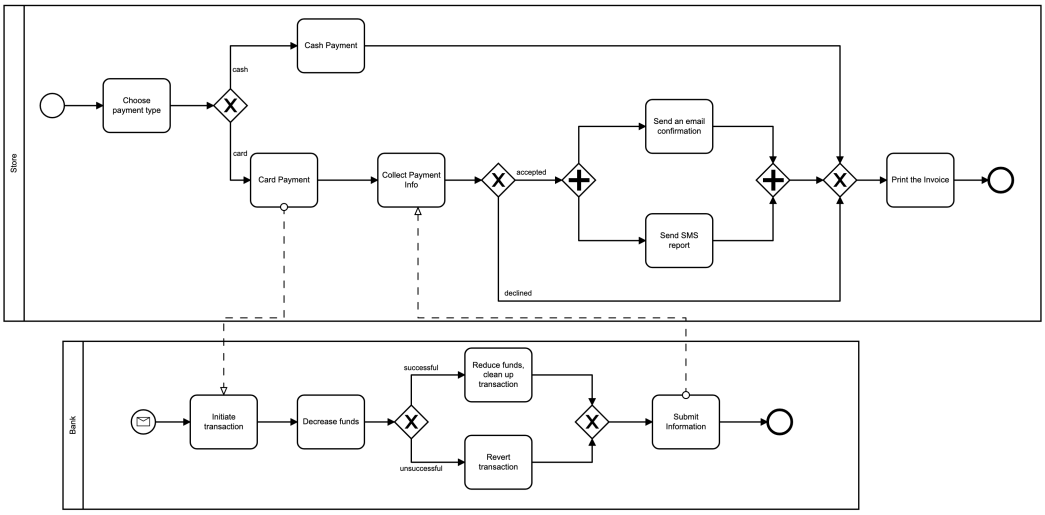
Slika 2.7: BPMN: Elementi i različite kategorije elemenata

nalaze u dva različita bazena i koja mogu da pošalju i da prime poruku. Asocijacije se koriste da povežu artefakte i informacije odnosno elemente poslovnog procesa, gde asocijacija ukoliko je potrebno može biti usmerena strelicom da naznači smer toka informacija.

Ukoliko se osvrnemo na primer koji je prikazan na slici 2.6 primećujemo da ovaj proces ima niz aktivnosti koje se izvršavaju u sekvenci. Aktivnost može biti atomička

ili ne-atomička odnosno složena. Dakle, ako bi u datom primeru aktivnost *Card Payment* prikazali kao niz manjih aktivnosti, npr. komunikacija sa bankom, provera dostupnih sredstava na računu, umanjenje sredstava na računu, onda bi to bila kompleksna, odnosno ne-atomička aktivnost. Postoji nekoliko tipova aktivnosti, a to su: proces, podproces i zadatak. Procesi mogu, a ne moraju biti ograničeni u okviru procesnog bazena.

Pomenuto je da tok sekvence prikazuje redosled kojim se izvršavaju aktivnosti u procesu, to u suštini predstavlja ograničenje u izvršavanju koje se primenjuje između aktivnosti, jer jedna aktivnost ne može da preskoči drugu, ukoliko to naravno nije prikazano drugačije. Ograničenje putem toka sekvence je jedno od najjednostavnijih oblika ograničenja, kompleksnija ograničenja se pojavljuju kod kapija, odnosno kod grananja (eng. *split* i spajanja (eng. *join*). Grananja mogu imati jednu ili više izlaznih grana (eng. *outgoing edges*) dok su spajanja ta koja imaju dve ili više ulaznih grana. Dati primer prikazuje četiri kapije, odnosno dva grananja i dva spajanja, međutim kao što je to objašnjeno ranije prikazano je jedno ekskluzivno i jedno paralelno grananje. Ekskluzivno grananje usmerava proces na jedan od dva moguća izbora kako je to prikazano na primeru, dok paralelno grananje, kao što mu ime naznačava ima zadatak da u paraleli izvršava dve aktivnosti *Send an email confirmation* i *Send SMS report* i tek kad se sve aktivnosti u paralelnim granama obavljene, proces iz tačke spajanja nastavlja dalje sa svojim izvršavanjem.



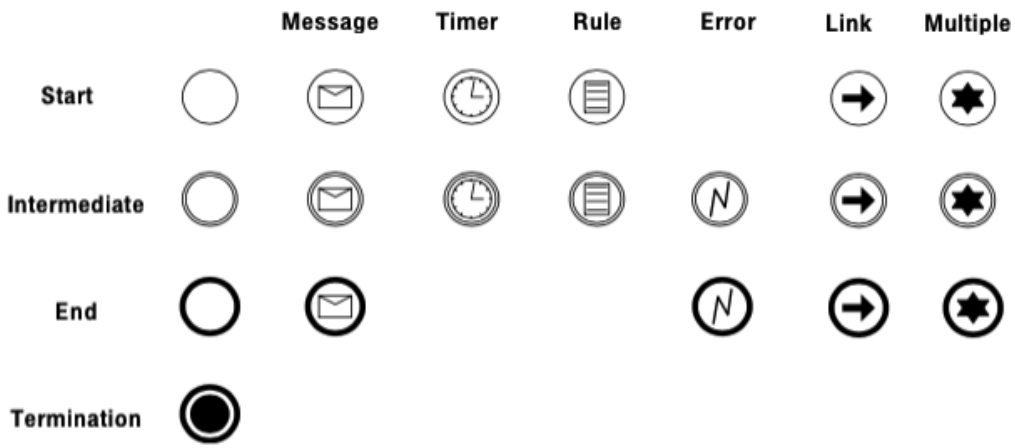
Slika 2.8: Ilustracija proširenog primera plaćanja

Na slici 2.8 ilustrovan je proširen primer plaćanja koji uključuje dodatne elemente i detaljnije pojašnjava komunikaciju između procesa. Na primeru se vide dva procesna bazena Prodavnica (eng. *Store*) i Banka (eng. *Bank*), gde je već rečeno da procesni bazen predstavlja učesnika u procesu, odnosno u ovom slučaju organizacije.

Proces počinje kad mušterija dođe na kasu da plati račun, a prva započeta aktivnost je izbor tipa plaćanja, odnosno gotovina ili kartica. Primalac poruke je događaj koji je relevantan za proces, odnosno primanje ove poruke je opisano početnim događajem procesa banke, odnosno uzrok ovog događaja za posledicu ima kreiranje drugog procesa. U ovom proširenom primeru, datom na slici 2.8 prikazana su još dva događaja, pet aktivnosti i isključivo grananje, odnosno kapija u posebnom procesnom bazenu, u kome je na osnovu grananja određeno da li mušterija ima dovoljno sredstava na računu za obavljanje transakcije i na osnovu čega se šalje odgovor. Učesnici koji sarađuju u okviru poslovnog procesa komuniciraju razmenjivanjem poruka, odnosno njihovim slanjem i primanjem. Tok poruke (eng. *message flow*), kao što je to prikazano na primeru obično povezuje aktivnost jednog učesnika u procesu sa odgovarajućim događajem/aktivnosti drugog učesnika, gde učesnici kao što je to već pomenuto mogu biti neke poslovne uloge, entiteti i slično.

2.3.1 Događaji

Događaji zauzimaju bitnu poziciju u BPM-u, jer oni osim što su glavni okidač za pokretanje, odnosno kreiranje novih instanci procesa služe kao elementi koji povezuju procese u poslovnim organizacijama i okidaju se ukoliko dođe do situacija koje zahtevaju interakciju između pomenutih procesa, odnosno njihovih učesnika. Kao što je pomenuto i prikazano na slici 2.7 događaji se dele na tri glavna tipa: početni, posredni i krajnji događaj. Tipovi događaja, odnosno njihove notacije u BPMN-u, koji su dati od strane OMG grupe, prikazani su na slici 2.9.



Slika 2.9: Tipovi događaja u BPMN, po OMG-u

Događaje karakteriše njihova pozicija u poslovnom procesu, ukoliko su na njegovom početku to su početni (eng. *start*) događaji i njihova pojava služi za pokretanje

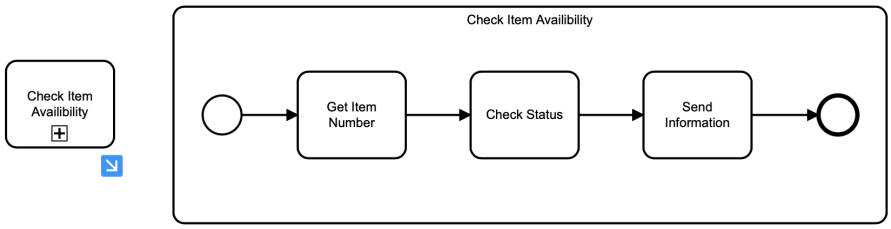
izvršavanja procesa, posredni (eng. *intermediate*) događaji su ti koji se dešavaju u toku procesa i oni mogu da zadrže, odnosno uspore njegovo izvršavanje i naravno krajnji (*end*) događaji se nalaze na kraju procesa i označavaju njegov završetak. Na slikama 2.7, 2.9 prikazani su različiti okidači koje mogu imati događaji.

Najopštiji tip je okidač bez specifičnog tipa (eng. *None*). Za početni događaj ovaj okidač znači da nije dat specifičan tip i koristi se prilikom pokretanja podprocesa od strane roditeljskog procesa, dok kod posrednih događaja ovaj okidač najčešće služi da nagovesti da se desila promena stanja u okviru procesa. Krajnji događaj bez specifičnog tipa okidača označava završetak procesa, odnosno podprocesa bez nekih dodatnih ili značajnih informacija. Ukoliko je korisnik okidač, on je taj koji inicira odnosno kreira početni događaj i pokreće proces. Poruke su već pominjanje i ovi okidači mogu da pokrenu proces kada se u sistemu registruje prijem poruke odgovarajućeg tipa, dok za posredne događaje poruka označava da se proces izvršio do neke tačke gde dalji napredak zavisi od prijema poruke, kad ta poruka stigne proces može da se nastavi sa izvršavanjem. Vremenski brojači (eng. *Timer*) mogu da posluže kao okidači procesa u zavisnosti od zadatih pravila (tačno naveden datum i vreme, ponavljajući ciklusi i slične operacije koje su dostupne vremenskim brojačima), slično je i za posredne događaja, gde se oni aktiviraju na osnovu vremenskog brojača. Greške (eng. *Error*) se javljaju u posrednim i krajnjim tipovima događaja, dešavaju se tokom izvršavanja procesa i služe za obradu izuzetaka, odnosno kad se desi greška podigne se izuzetak koji treba da se uhvati (eng. *catch*) i obradi, odnosno startuje se odgovarajuća aktivnost za obradu izuzetka. Događaj za prekid (eng. *Termination*) se koristi da odmah prekine izvršavanje svih aktivnosti procesa.

2.3.2 Aktivnosti

Aktivnosti predstavljaju jedne od glavnih elemenata poslovnog procesa, i kao što je već napomenuto mogu biti atomične (zadaci) ili složene (podprocesi ili pozivani procesi), zbog čega je BPMN kreiran sa hijerarhijskom podrškom za njihovo ugnježdavanje u zavisnosti od potrebe procesa. Notacija podržava prikazivanje podprocesa sa oznakom $+$ u donjem centralnom delu aktivnosti, kada je ta aktivnost u sažetom (minimizovanom) prikazu (eng. *collapsed*) i na toj aktivnosti nisu izložena njegova unutrašnja svojstva. Prošireni podproces (eng. *expanded* se razlikuje u tome što on prikazuje svoju unutrašnju strukturu, kao što je to prikazano na slici 2.10, odnosno primeru koji ilustruje dostupnost nekog artikla koji se provera u podprocesu sa nazivom *Check Item Availability*.

Postoji nekoliko različitih tipova zadataka u sklopu aktivnosti, od kojih su neki pominjani. Aktivnosti koje se zasnivaju na ručnom radu (eng. *Manual Activites*) se izvršavaju bez pomoći softverskog sistema. Odnosno njih najčešće izvršava osoba kao učesnik poslovnog procesa, kao npr. dostava pošiljke, iako se ovo izvršava van softverskog sistema, nakon završene aktivnosti ručnog rada, sistem za upravljanje



Slika 2.10: Primer skupljenog i proširenog podprocesa

poslovnim procesom treba da se obavesti o uspešnosti izvršenja zadatka, kako bi dalje nastavio izvršavanja, na primer pošiljka je uspešno isporučena.

Servisni zadaci (eng. *Service task*) su zadaci koje implementira i izvršava softver, odnosno neki veb servis (eng. *Web Service*) koji može biti javno ili privatno dostupan, i obično su delovi nekog većeg softverskog sistema. Zadaci za skriptovanje (eng. *Script Task*) su po ponašanju slični servisnim zadacima, odnosno postoji neka automatizacija koju izvršava softver, ali su po strukturi dosta jednostavniji od servisnih zadataka. Zadaci za skriptovanje koriste skriptne jezike (npr. groovy) koji su podržani od strane okruženja koje izvršava poslovne procese, izvršavanjem skripte završava se i zadatak.

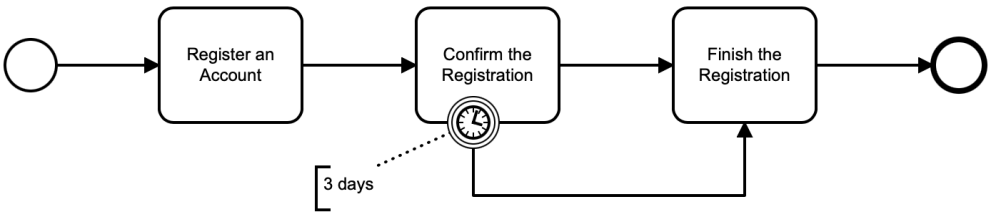
Interakcije između poslovnih procesa modeluju se zadacima za primanje (eng. *receive task*) i slanje (eng. *send task*) i zajedno čine komunikacione zadatke (eng. *communication tasks*). Ovi zadaci imaju atribut koji se zove poruka (eng. *Message*) koji zapravo specificira poruku koja treba da se pošalje ili primi. Zadaci za primanje čekaju na dolazak poruke, kad poruka stigne taj zadatak se završava, dok zadaci za slanje čekaju trenutak u poslovnom procesu da se aktiviraju i onda šalju poruku nakon čega se završavaju.

2.3.3 Tok sekvence

U BPMN-u tok sekvence kontroliše redosled izvršavanja elemenata, odnosno tok sekvence povezuje odredišni i izvorišni element koji pripada elementima toka, koji su već opisani. Tok sekvence je prikazan usmerenom strelicom između objekata aktivnosti, događaja i grananja. Pored toga što povezuje različite elemente toka, postoje i različiti tokovi sekvence, odnosno: normalni, izuzetak i *ad-hoc* tok. Normalni tok predstavlja ono što se očekuje od procesa, odnosno započinje se svojim start događajem i bez prekida,izuzetaka prolazi kroz poslovni proces dok se ne izvrši, odnosno dok ne dođe do poslednjeg elementa toka, koji je krajnji događaj. Za razliku od normalnih tokova, koji definišu regularan tok izvršavanja, postoje i alternativni tokovi koji otpočinju u izuzecima. Proces nastavlja ovim alternativnim tokom kada se tokom izvršavanja nekih aktivnosti dese određeni događaji posle kojih normalno izvršavanje i uspešan završetak

procesa više ne bi bio moguć.

Exception flow se kreira od strane posrednog događaja koji su pominjani, posredni događaji se dešavaju tokom procesa, a mogu biti i povezani na granici (eng. *boundary*) aktivnosti. Posredni događaji utiču na proces ali ga direktno niti pokreću niti završavaju, mogu se koristiti za više namena, a ovde se posredni događaji koriste da prekinu, odnosno promene normalan tok procesa kroz obradu izuzetaka. Na slici 2.11 prikazan je primer u kome se izvršava izuzetni tok, odnosno posredni događaj uslovljen brojačem inicira taj tok. Na primeru je prikazan proces registracije naloga, nakon što korisnik unese svoje podatke, treba da potvrdi registraciju. Registraciju može potvrditi na primer klikom na link koji dobije u elektronskoj poruci, ukoliko korisnik to ne uradi u roku od tri dana, aktivira se brojač koji se nalazi na granici aktivnosti i završava proces registracije.



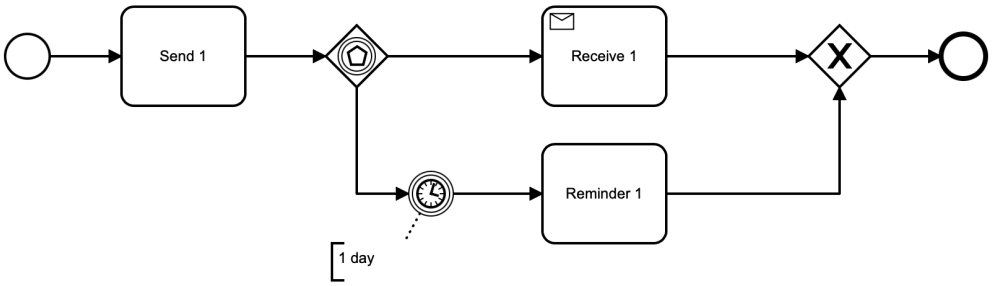
Slika 2.11: Primer skupljenog toka uslovljenog izuzetkom

2.3.4 Kapije / Grananja

Kapije su još jedan elemenat toka, bez kojih se ne može zamisliti nijedan iole ozbiljniji proces. Kapije mogu biti tačke grananja ili tačke spajanja, grananja mogu imati više izlaznih i najčešće jednu ulaznu granu, dok spajajuće kapije imaju tačno jednu izlaznu a imaju više ulaznih grana. Iako je moguće kreirati i grananja/spajanja sa više ulaza i izlaza, moguće je da semantika takvog čvora ne bude najjasnija, ili da ne funkcioniše na način na koji bi se na prvi pogled pretpostavilo, te stoga treba težiti jednostavnosti i jednoznačnosti grananja/spajanja. Različiti tipovi kapija su prikazani na slici 2.7.

Ekskluzivno ILI (eng. *OR*) grananje je podržano u BPMN-u na nekoliko načina u zavisnosti da li je izlazna grana određena podacima ili događajima. Ukoliko je grananje zasnovano na podacima, to znači da postoji uslov koji vezan za kapiju i koji se izvršava na osnovu podataka. Na osnovu zadovoljenosti uslova, koji se najčešće piše nekim jezikom za izraze u sklopu alata koji se koristi (eng. *expression language*) kad uslov postane zadovoljen, odnosno logički tačan, grana koja je povezana sa tim uslovom se odabira za izvršavanje. Pošto se provera uslova izvršava nekim programskim jezikom,

redosled navođenja provere uslova je bitan, odnosno traži se prvi zadovoljavajući uslov, nakon čega se ostali zanemaruju. Ukoliko nijedan uslov nije zadovoljen, postoji opcija za podrazumevanim tokom (eng. *default flow*) koji će se izvršiti samo ukoliko nijedan od eksplicitno navedenih uslova se nije ispunio. Kao i u programiranju ovo zavisi od kreatora procesa, odnosno da li će dati opciju korišćenja podrazumevanog toka, ili će kroz izraze obezbediti da se uvek bar jedan od postojećih tokova izvrši. Grananje zasnovano na događajima je drugačije i ono vrši aktivaciju grane, odnosno izbor na osnovu tipa događaja koji je primljen. Razlika je u semantici, odnosno između aktivnog i reaktivnog ponašanja, dok kapija zasnovana na podacima aktivno pravi izbor na osnovu definisanih uslova, kapija zasnovana na događajima reaktivno reaguje na osnovu tipa primljene poruke. Na slici 2.12. prikazan je primer kapije koja je zasnovana na događajima, u zavisnosti od zadatka *Send 1* izvršava se *Receive 1* uz aktivaciju posrednog događaja koji će se aktivirati nakon jednog dana i koji će izvršiti *Reminder 1*.

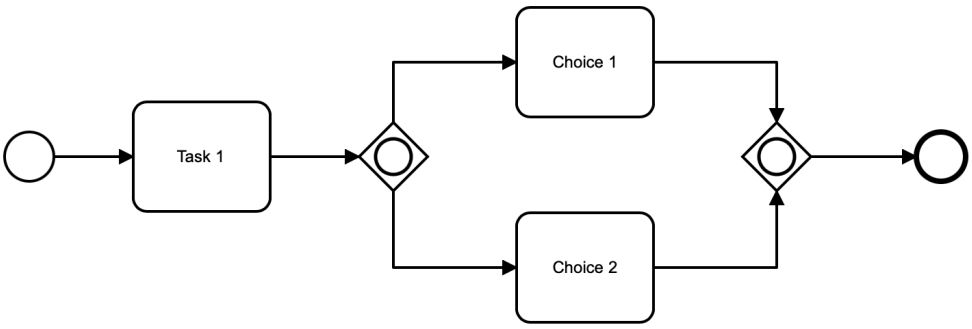


Slika 2.12: Primer kapije zasnovane na događajima

Inkluzivna ILI (eng. *OR*) kapija se koristi slično kao i ekskluzivna ILI kapija, u slučajevima gde se odabira proizvoljan broj izlaznih grana, kao što je to prikazano na slici 2.13, a kompleksna kapija (eng. *complex gateway*) omogućava definiciju kombinovanog ponašanja grananja i spajanja.

2.3.5 Interakcije i modelovanje interakcija u poslovnim procesima

Poslovni procesi mogu da vrše interakciju jedni sa drugima u okviru jedne organizacije ili između više različitih organizacija. Plivačke trake se koriste da grafički prikažu procese ili delove procesa i njihovih organizacionih entiteta koji su uključeni u poslovnim procesima, gde procesni bazeni često prikazuju učesnike u procesu, uglavnom kompanije, a plivače trake prikazuju odeljenja u okviru tih kompanija [22]. Elementi toka koji se nalaze u okviru jednog procesnog bazena mogu biti međusobno povezani, odnosno može



Slika 2.13: Primer inkluzivne kapije zasnovane na događajima

se uspostaviti sekvenca između njih, dok komunikacija između bazena se odvija, kao što je pomenuto, slanjem i primanjem poruka.

BPMN se uglavnom koristi za prikazivanje procesa u sklopu jedne organizacije, dok za prikazivanje interakcije procesa između različitih organizacija odnosno *koreografije* procesa ne postoji jasna notacija, odnosno uglavnom se svodi na dogovor između kreatora procesa iz različitih organizacija. Iako je modelovanje koreografije u BPMN-u moguće postoje određeni problemi poput *redundantnosti* koja zbog različitih ograničenja može dovesti do dodatnog napora za modelovanje i do nevažećih modela, kao i do *potencijalno nekompatibilnog ponašanja* ukoliko ne dođe do poklapanja u strukturama sekvenciranja procesa, što može izazvati zastoje (eng. *deadlock*) u procesu [38].

Barros i autori u [39] predlažu skup šablona interakcije servisa (eng. *Service Interaction Patterns*) koje dozvoljavaju interakciju između servisa u smislu koreografije i orkestracije. Ovaj skup šablona je pogodan za evaluaciju i poređenje sa nekim poznatim scenarijima odnosno njihovim apstraktnim oblicima [40]. Uzrok za kreiranje ovog skupa šablona je već sada skoro sveprisutna tražnja za standardizacijom jezika za modelovanje i platformi koje podržavaju upravljanje poslovnim procesima u sklopu upravljanja poslovnim procesima.

Postoje različiti predlozi za ovakvu vrstu modelovanja, kao što je WS-CDL(eng. *Web Service Choreography Description Language*), odnosno jezik zasnovan na XML-u koji opisuje saradnju između učesnika tako što definiše njihovo zajedničko ponašanje [41]. Da bi saradnja između pomenutih učesnika funkcionisala pravila interakcije moraju biti poznata, što je najveći nedostatak koreografije procesa jer standardizovan način za definisanje ovih interakcija ne postoji i WS-CDL je jedan od jezika koji pokušava to da reši. Glavna spona za komunikaciju između ovih učesnika, odnosno razmenu poruka predstavlja specifikacija veb servisa između različitih okruženja koje treba da opslužuju organizacije u okviru kojih se izvršavaju. WS-CDL dolazi sa tekstualnom sintaksom na

implementacionom nivou i ono što mu nedostaje je razdvajanje meta-modela i sintakse, odnosno nema striktne razlike između semantičkih i sintaksnih aspekata kao i nedostaci za formalnu verifikaciju [42].

Xianpeng i autori u [43] nastavljaju se na WS-CDL, međutim i oni uočavaju nedostatke ovog jezika. Njihov predlog uključuje predlog malog jezika sa nazivom CDL koji bi služio kao formalni model pojednostavljenog WS-CDL-a. Međutim cilj ovog jezika nije da pokrije celi WS-CDL, nego njegova najbitnija svojstva kako bi na osnovu formalnog modela bilo moguće kreirati i generisati prikaze orkestracija za neku zadatu koreografiju i rasuđivanje o svojstvima koja se moraju ispoštovati. Predložena je i funkcija za projekciju za generisanje orkestracije i alat za automatsko prevođenje koji može da pretvori koreografiju u ulazni jezik za proveru modela pod nazivom SPIN [44]. Svojstva koja CDL uključuje su uloge, promenljive, aktivnosti i koreografija, međutim neka napredna svojstva kao što su detalji o kanalu, blokovi za izuzetke i finaliziranje nisu uključeni. Pored toga druga svojstva poput baznih tipova, tokena i izraza nedostaju u modelu i autori navode uključivanje ovih svojstava u budućem radu.

Pomenuto je da se modelovanje interakcija može vršiti i u BPMN-u, međutim zbog navedenih nedostataka BPMN ne podržava sve što je potrebno za modelovanje interakcija, ali zbog mogućnosti njegovog proširenja i ekstenzija koje se mogu nasloniti na BPMN, autori u [38], predstavljaju *iBPMN*, odnosno ekstenziju BPMN-a za modelovanje interakcija. Osnovni gradivni elementi ove ekstenzije predstavljaju atomske (nedeljive) interakcije koje su delovi modela, kontrole i toka podataka između njih. U okviru *iBPMN* događaj poruke predstavlja elementarnu interakciju koja je deo (prilog) toka poruke u okviru ove ekstenzije. Iako je moguće izraziti većinu šablona za interakciju servisa (*Service Interaction Patterns*) kroz iBPMN još ostaje detaljno da se ispita validnost algoritma koji to radi, jer se ispostavilo da neke koreografije nije moguće lokalno izvršiti, zbog čega su autori najavili validaciju algoritma kroz buduća istraživanja.

Malo drugačiji pristup napravili su *Zaha* i autori u [45] koji su predstavili jezik za koreografiju *Let's Dance* na osnovu utvrđene potrebe za opisivanje koreografija na konceptualnom nivou, time pružajući podršku većini šablona za interakciju servisa. Ovaj jezik takođe prati pristup za modelovanje interakcije, odnosno interakcije predstavljaju gradivne elemente modela koreografije [38]. *Let's Dance* ima grafičku notaciju i nije zasnovan na imperativnom stilu programiranja, takođe ovaj jezik podržava lokalna i globalne preglede interakcije servisa, odnosno interfejse ponašanja i koreografija, gde je pogodnost jezika demonstrirana na osnovu 13 prethodno identifikovanih interakcionih šablona. Jezik opisuje i apstraktnu sintaksu u vidu statičkog meta-modela, kao i neformalnu semantiku, dok autori u [46] su predstavili alat koji implementira algoritme za statičku analizu koreografija ovog jezika i generisanje lokalnih modela.

2.3.6 Drugi tipovi modela

Pored navedenih, postoje i drugi tipovi modela, koji nužno ne omogućavaju veliki stepen fleksibilnosti i interakcija u poslovnim procesima, ali su svakako poslužili kao osnova i inspiracija za kreiranje navedenih modela, te ih je bitno pomenuti.

Notacija za specificiranje ponašanja poslovnih procesa zasnovanih na veb servisima ili kompozicija veb servisa može se izvršiti putem BPEL4WS (*Business Process Execution Language for Web Services* ili skraćeno BPEL [47]. BPEL je jezik za kreiranje radnih tokova zasnovan na XML-u, kreiran sa ciljem da se uklopi u veb servise i arhitekture koje ih podržavaju.

WSFL (eng. *Web Services Flow Language*) bio je preteča BPEL, jezik koji je takođe zasnovan na XML-u za opisivanje kompozicija web servisa, odnosno za za opisivanje dva tipa kompozicija, poslovnih procesa koji su deo veb servisa i za opisivanje interakcija između njih [48]. WSFL se može posmatrati kao XML serijalizacija jezika za definiciju toka, odnosno IBM-ov proizvod za radne tokove koji podržava koncepte za pristup veb servisima i koji je kreiran na osnovu procesnih jezika baziranih na grafovima u kojima su aktivnosti poređane u aciklični oblik [22].

XLANG je BCL (eng. *Block Structured Language*), odnosno jezik visok nivoa koji je sačinjen od blokova. Ovaj jezik korišćen je u BizTalk-u, softveru koji je Microsoft razvio za integraciju velikih poslovnih aplikacija. Glavni fokus XLANG-a je integracija heterogenih *back-end* sistema koji koriste procese, odnosno sistema koji tipično nisu dostupni korisniku [49]. BCL jezici koriste ugnježdavanje blokova za kontrolu toka, kako bi opisali i strukturirali poslovni proces [22].

Dok je WSFL bio IBM-ova realizacija, XLANG je bio Microsoft-ova, te su ove kompanije iskoristili najbolje od oba jezika i njihovom integracijom kreiran je BPEL.

BPEL koristi blokovsku strukturu u cilju organizovanja kompozicije servisa. BPEL je jezik koji može da opiše apstraktne i konkretne procese, gde apstraktni procesi opisuju ponašanje procesa koji je vidljiv spolja (odnosno eksterno ponašanje procesa) dok konkretni procesi sadrže sve informacije koje su neophodne za izvršavanje veb servisa iz prethodno pomenute kompozicije, dok je BPEL proces spolja izložen kao veb servis, odnosno može mu se eksterno pristupiti. Iako je BPEL jezik prihvaćen i široko korišćen, primarno se koristi za kompoziciju veb servisa i donosi mnogo zanimljivosti, ali jezik kao takav neće biti dalje razmatran u okviru ove disertacije.

Događajem upravljanji procesni lanci (eng. *Event Driven Process Chain*) predstavljaju tipove dijagrama toka za modelovanje poslovnih procesa i razvijeni su u okviru ARIS (eng. *Architecture of Integrated Information Systems*) platforme [50]. EPC predstavljaju uređeni graf događaja i funkcija, kao i raznih konektora koji dozvoljavaju alternativno i paralelno izvršavanje procesa, i kao takvi su pogodni za korišćenje zbog svoje jednostavnosti i lako razumljive notacije [51].

2.4 Adaptacije u poslovnim procesima

Tradicionalno, upravljanje sa radnim tokovima je imperativno, odnosno sistem kontroliše izvršavanje poslovnih procesa u skladu sa modelima na osnovu kojih su ti procesi definisani. Zasnovani su na striktnoj separaciji dve glavne aktivnosti, vreme kreiranja procesa (eng. *build time*) i vreme izvršavanja procesa (eng. *runtime*), gde jednom pokrenuti procesi, odnosno instance koje su u momentu kreiranja nastale interpretacijom modela procesa, nakon pokretanja više nemaju direktnu zavisnost od samog modela. Model je interpretiran u trenutku instanciranja, dobijeni su izvršni artefakti neophodni da se proces izvršava, i naknadno stanje modela je irelevantno za pokrenuti proces.

Istorijski, kao što je pominjano ovaj postupak je pogodan za podržavanje poslovnih procesa u kompanijama sa dobro utvrđenim procesima, na statički način. Odnosno procesi čiji se model napravi jednom i čiji se procesi kasnije izvršavaju na identičan način, sa predviđenim i očekivanim ishodima. Međutim, u današnjim poslovnim okruženjima gde su promene zahteva česte, postoji potreba da se odgovori na nove zahteve i da se proces prilagodi novim promenama, odnosno promenama koje nisu očekivane u vremenu kreiranja procesa. Ove promene vode ka novim zahtevima i traženju novih mogućnosti od sistema za upravljanje poslovnim procesima.

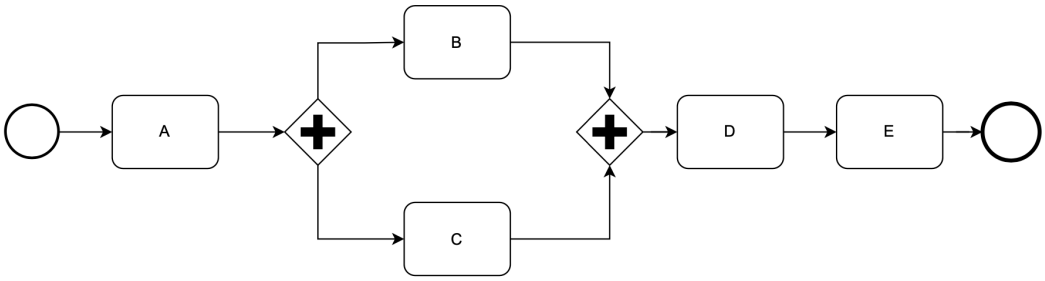
Dinamičke adaptacije su promene u strukturi procesa koji se trenutno izvršava, odnosno promene u instanci trenutno izvršavanog procesa. Alati koji su razvijani da podrže ove procese nisu i još uvek ne daju adekvatan odgovor na dinamičke adaptacije koje još uvek nisu podržane u komercijalnim sistemima upravljanja radnih tokova (eng. *workflow management systems*). Kao takva ova situacija je ograničavajući faktor za dalji razvoj i isporuku aplikacija organizacijama koje koriste ove sisteme jer često, kao što je pomenuto, novonastale situacije nisu predviđene, a samim tim i obrađene tokom dizajna procesa.

Prevazilaženje ovih situacija često se svodi na to da korisnici sistema, moraju da obave nešto što nije deo njihovog posla ili čak pristupaju delovima sistema kojim tradicionalno i hijerarhijski ne bi smeli, kako bi tekući proces čiji uslovi izvršavanja su sada promenjeni na neki način snabdeli ili dopunili informacijama koje su značajne za nastavak njegovog izvršavanja. Na taj način bi odstupili od redosleda izvršavanja aktivnosti u procesu i njegovih željenih ishoda, a kao rezultat bi dobili vrednost koja se uglavnom naknadno mora korigovati.

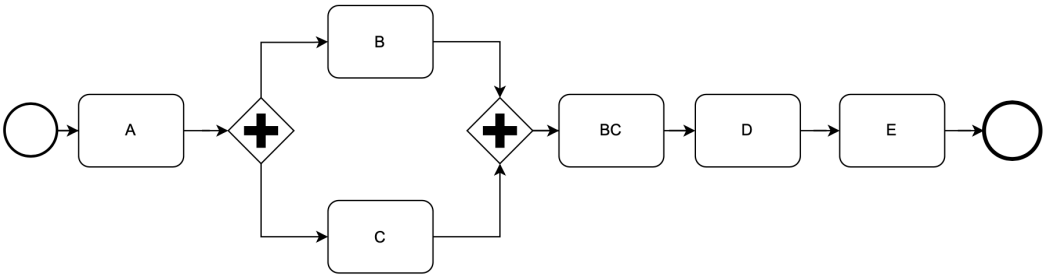
Jedan primer radnog toka nad kojim je potrebno izvršiti dinamičku adaptaciju prikazan je na slici 2.14.

Slika 2.14 ilustruje pojednostavljen model nekog radnog toka, sa pet aktivnosti i paralelnim grananjem. Ukoliko je željena adaptacija dodavanje nove aktivnosti BC, kao što je to prikazano na slici 2.15, adaptaciju je moguće izvršiti nad tekućim instancama radnog toka samo u određenim situacijama.

Ukoliko je radni tok došao do aktivnosti D, onda adaptaciju nije moguće izvršiti,

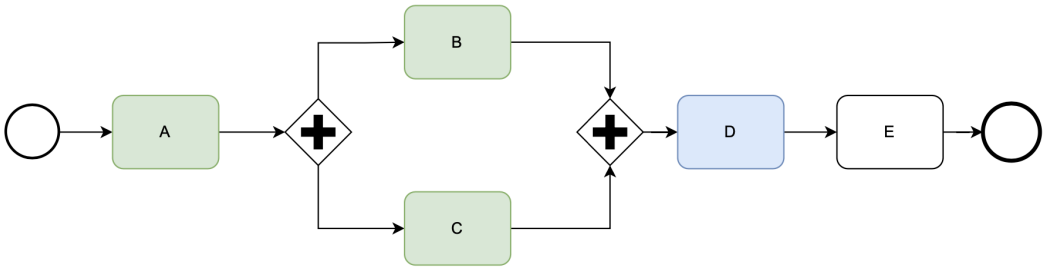


Slika 2.14: Uprošćen primer radnog toka za adaptaciju



Slika 2.15: Uprošćen primer radnog toka za adaptaciju, željena adaptacija radnog toka

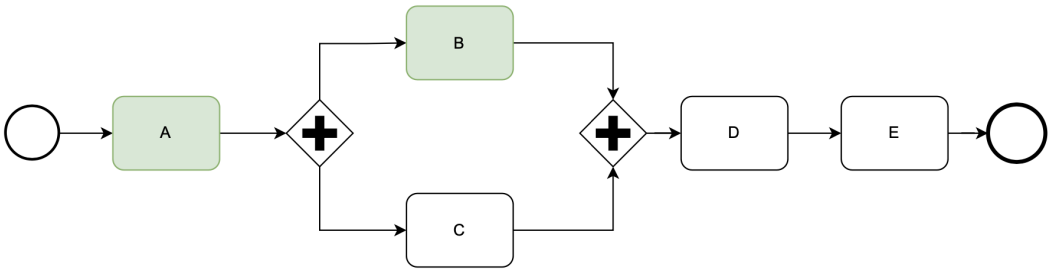
odnosno data instanca radnog toka nije adaptabilna, kao što je to prikazano na slici 2.16. Na slici zelenom bojom su obeležene izvršene aktivnosti, dok je plavom bojom obeležena tekuća aktivnost. odnosno instanca i je adaptabilna na model s' ako i samo ako postoji nastavak i, tako da i odgovara s' [22].



Slika 2.16: Primer radnog toka za adaptaciju, nedozvoljena adaptacija

Korektna adaptacija je moguća ukoliko je instanca radnog toka u stanju kao na slici 2.17, gde su izvršene aktivnosti A i B, odnosno nije došlo do izvršavanja aktivnosti C i do završetka paralelnog grananja, te je moguće dodati novu aktivnost BC neposredno

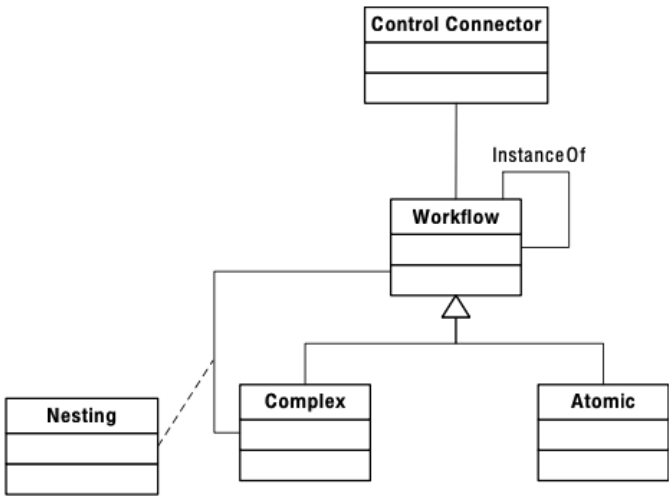
nakon završetka paralelnog grananja.



Slika 2.17: Primer radnog toka za adaptaciju, dozvoljena adaptacija

Dalje istraživanje dinamičkih adaptacija odnosno promena u strukturi procesa tokom njegovog izvršavanje se ispituju i u radovima koji su deo ove sekcije, kao i u narednim poglavljima ove disertacije.

Za razmatranje dinamičkih adaptacija moguće je krenuti i od konceptualnog dizajna poput *Weske*-a i autora u [22] gde su dinamičke adaptacije opisane putem meta-modela koji je razvijen pomoću objektno orijentisanih tehnika koje opisuju relevantne entitete i strukture softverskog sistema i prikazan je na slici 2.18.



Slika 2.18: Meta-model fleksibilnog sistema za upravljanje poslovnim procesima [22]

Meta-model prikazan na slici 2.18 prikazuje *Workflow* klasu kao centralnu klasu ovog metamodela, odnosno kompozitna je i može da sadrži objekte radnog toka koji su ili modeli radnog toka ili instance radnog toka, što se utvrđuje na osnovu rekurzivne veze *instanceOf* koja je povezana sa *Workflow* klasom. Asocijativna klasa *Nesting* opisuje

hijerarhijsku strukturu u ovom modelu, koja daje vezu između kompleksnog radnog toka i radnog toka, koji opet može biti kompleksan ili jednostavan (atomički), gde se modeli i instance radnog toka razlikuju, odnosno identifikuju putem stanja objekata koji učestvuju u radnom toku.

Različiti sistemi za upravljanje poslovnim procesima u kompanijama, imaju različite karakteristike. Iako su poslovni procesi specifični i zavise od više faktora, moguće je naći neke zajedničke osobine i pravila ponašanja koja su slična za ove sisteme. *Aalst* [52] sistematično pristupa zahtevima radnog toka, na osnovu kojih se predlažu šabloni radnog toka (eng. workflow patterns) kao skup najboljih praksi za rešavanje određenih situacija u poslovnim procesima, ali i kao baza za poređenje stepena razvoja komercijalno dostupnih sistema za upravljanje poslovnim procesima. Šabloni radnog toka se u poslovnim zahtevima izražavaju kroz imperativne izraze stila radnog toka, odnosno različite konstrukcije elemenata radnog toka kreiraju šablone. Šabloni radnog toka se primenjuju u modernim procesno-orijentisanim sistemima, odnosno alati koji služe za upravljanje procesima u okviru sistema se oslanjaju na date šablone, koje je moguće prilagoditi zahtevima u poslovnim sistemima kompanija.

Ferreira u [53] razrađuje mogućnost kreiranja alata za upravljanje radnim tokovima (eng. *worfklow engine*) za višekratnu upotrebu. Odnosno, autori navode kao jedan od problema sa kojima se oblast susreće je da se određena funkcionalnost implementira svaki put iznova sa kreiranjem novih sistema za upravljanje radnim tokovima. Fokus je na kreiranju jezgra radnog toka (eng. *workflow kernel*) koje sadrži zajedničku funkcionalnost uočenu u sistemima za upravljanje radnim tokovima, koju je moguće koristiti u više različitih scenarija. Njihovo rešenje se zasniva na *Petrijevim* mrežama.

Whittingham se u radu [54] pozabavio pitanjem podrške u radnim tokovima, odnosno adaptabilnim sistemima koji služe za upravljanje poslovnim procesima i predstavio je pristup koji je nazvao *OpenWater*. Koristeći njegov pristup učesnici radnog toka mogu sami da definišu i izvršavaju procese, odnosno odgovornost u okviru poslovnog procesa je pre distribuirana nego kontrolisana na centralizovan način, odnosno artefakti radnog toka se prenose, kad jedan učesnik završi sa svojim zadatkom, prelaze na drugog učesnika koji treba da završi njegov zadatak na osnovu odluka donesenih od strane prethodnih učesnika. Zbog toga autor *OpenWater* naziva alat za podršku radnim tokovima, a ne sistem za upravljanje radnim tokovima, jer podrška podvlači da tok nije diktiran, nego je ostavljen učesnicima poslovnog procesa, odnosno ljudima koji obavljaju posao, kad god je to moguće [54]. *Narendra* u [55] koristi ovaj pristup, odnosno neke od ideja sa željom da ih integriše u alat koji je prethodno razvio [56]. Glavni cilj njegovog istraživanja je da pokaže kako se podrška radnog toka (eng. *workflow support*) može kombinovati sa adaptabilnim upravljanjem u cilju kreiranja fleksibilnog sistema za podršku i upravljanje radnim tokovima - kompromis između podrške i upravljanja radnim tokovima.

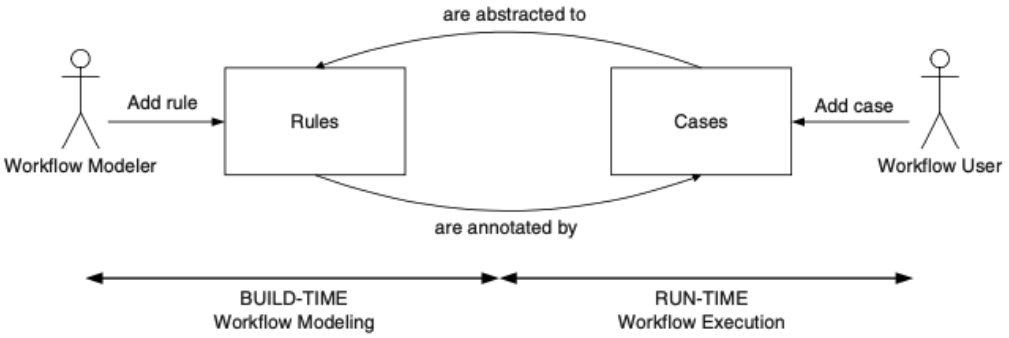
Motivacija za kreiranje *CAKE* (eng. *Collaborative Agent-Based Knowledge Support for Empirical and Knowledge-Intense Processes*) sistema koncipirana je na osnovu

potreba za čestom promenom zahteva u različitim aplikacijama poput vatrogasnih službi, agilnog softverskog inženjerstva, medicine i drugih, te je kreiran kao domenski-nezavisan sistem ciljan kao podrška empirijskim procesima i procesima sa intenzivnim znanjem (eng. *Knowledge-Intensive Processes*) [57]. KIP (*Knowledge-Intensive Processes*) su procesi čije izvršavanje je u mnogome zavisi od ljudi koji donose odluke na osnovu sopstvenog domenskog znanja i velikog iskustva (eng. *knowledge workers*), ovi procesi između ostalog omogućavaju idetnifikovanje postojećeg, kreiranje novog i deljenje znanja kroz različite strategije [58]. Empirijski procesi zahtevaju pomno praćenje i kontrolu jer su uglavnom nepredvidivi, neponovljivi i podložni su promenama tokom njihovog izvršavanja [59]. Da bi se izborio sa čestim promenama i zahtevima promene, *CAKE* zahteva dosta fleksibilnosti, za koju konvencionalni softver ne poseduje dovoljno mogućnosti, jer uglavnom zahteva predefinisane putanje izvršavanja [57]. *CAKE* prevazilazi ove probleme korišćenjem agilnih radnih tokova (eng. *agile workflows*), nudeći izbor između detaljnog modelovanja pojedinačniih zadataka i mogućnosti kasnog odlučivanja. U ovom drugom slučaju, modelovanje je na višem nivou apstrakcije, gde se specificira da određena aktivnost postoji, ali omogućava da se tokom izvršavanja definiše specifičan način obavljanja zadatka. Ova vrsta radnih tokova je podržana putem CBR-a (eng. *Case Based Reasoning*) [60].

Još jedna arhitektura za adaptabilne radne tokove koja se oslanja na *CBR*, odnosno *CCBR* (eng. Conversational Case-Based Reasoning) koji predstavlja proširenje *CBR*-a u kojoj je korisnik aktivno uključen u proces zaključivanja [60]. *CBRFlow* proširuje izvršavanje radnog toka sa *CCBR* kako bi prilagodio predefinisani model na promenljive okolnosti i da bi *WFMS*-u pružio mogućnost učenja [61]. U sklopu *CBRFlow*-a poslovna pravila su anotirana tokom izvršavanja radnog modela sa informacijama specifičnim za kontekst a kojia su potrebna *CCBR* podsistemu koji se koristi u okviru ovog alata. Kad ponovno iskorišćenje postojećih slučaja postane često, onda se oni ručno refaktorišu u pravila koja podstiču automatsko izvršavanje. Što se češće ovaj postupak ponavlja postižu se bolji rezultati. Na osnovu opisanog zaključuje se da je glavna ideja integrisanje *CCBR*-a i WFMS-a da se podrže modifikacije predefinisanog modela radnog toka putem inkrementalne sposobnosti učenja, kao što je to prikazano na slici 2.19[1].

Dumas se u [1] bavi dalje bavi procesno-svesnim informacionim sistemima (eng. *Process Aware Information Systems*) i načinima njihove primene. Dato je pojašnjenje o tranziciji sa do tada najzastupljenijeg podacima-upravljanog (eng. Data-driven) pristupa, gde je modelovanje podataka bila inicijalna aktivnost za kreiranje informacionog sistema i fokus informacionih tehnologija primarno na skladištenju, čitanju i prezentovanju podataka, odnosno informacija. Kada je došlo do problema u primeni ovog pristupa, fokus se prebacio sa podacima-upravljanog na procesom-upravljan pristup. Suština je automatizaciji procesa ili dela procesa softverskim sistemom, koji prosleđuje informacije od jednog učesnika ka drugom, zbog obavljanja

[1]Slika preuzeta iz rada [61]



Slika 2.19: Adaptabilni pristup koji koristi CBR Flow

neke akcije, usklađeno prema vremenskim i logičkim zavisnostima koje su postavljene u modelu procesa.

Prelazak na PAIS donosi brojne prednosti:

- Korišćenjem eksplicitnih modela omogućeno je jasno sredstvo komunikacije između ljudi različitih profila u sklopu kompanije. Komunikacija osoba koji učestvuju u poslovnom sistemu je od velike važnosti, prethodno ta komunikacija nije bila dobra između tehničkih i ne-tehničkih lica(npr. menadžera i poslovnih analitičara sa jedne strane i programera i administratora sistema sa druge strane).

- PAIS su upravljani modelima, što znači da je menjanje poslovnih procesa moguće bez ponovnog implementiranja sistema ili njegovih delova. Odnosno, ako je informacioni sistem upravljan procesnim modelima, onda je potrebno menjati samo modele da bi se podržale promene postojećih ili novih poslovnih procesa [62].

- Korišćenjem eksplicitne reprezentacije procesa koji su podržani od strane organizacije, dozvoljava njihovo automatsko izvršavanje [12, 63, 64].

- Eksplicitna reprezentacija procesa omogućava podršku menadžmenta na nivou dizajna/redizajna tj. eksplicitni procesni modeli podržavaju napore prilikom dizajna/redizajna (eng. *design effort*) [65].

- Eksplicitna reprezentacija procesa takođe omogućuje podršku menadžmenta na nivou kontrole, odnosno praćenje procesa tokom njegovog izvršavanja pruža razne korisne informacije o procesu koje se mogu iskoristiti za razna poboljšanja (npr. bolja kontrola procesa, uočavanje uskih grla u procesu).

Autori u [1] obrazlažu da u suštinu postoje dva načina da se kreiraju PAIS

- Kreiranje specifičnog sistema za podršku procesima

- Konfigurisanje generičkog sistema

Prilikom kreiranja specifičnog sistema za podršku procesima, kompanija kreira svoj sistem od početka i kontroliše sve aspekte razvoja tog sistema, u cilju podrške sopstvenih procesa. Kreiranje ovih sistema odlikuje se različitim stepenima kompleksnosti u zavisnosti od zahteva kompanije. Ti zahtevi mogu biti jednostavni od kreiranja programske biblioteke koja treba da se ugradi u neko rešenje i koja bi pružala postojećoj aplikaciji dovoljno informacija o postojećim procesima, pa do nekih kompleksnih sistema odnosno platformi koje pružaju mogućnosti od definisanja procesa pa sve do njihovog izvršavanja i nadgledanja. Ovaj pristup kreiran je samo sa jednim ciljem, a to je da se integriše u postojeće sisteme kompanije i da se prilagodi specifičnosti njihovih procesa, međutim pored prednosti koji nosi jedan takav sistem, njegovo kreiranje je jako skupo, jer kao što je pomenuto radi se od samog početka, istovremeno rezultujući sistem ukoliko nije dobro sagledan i dizajniran može postati trom i neefikasan kako procesi postaju zahtevniji i sofisticiraniji, odnosno neće se skalirati. Generički sistemi za podršku poslovnim procesima su najčešće WFMS, koji ne poseduju informacije o strukturi i procesima neke konkretne kompanije, odnosno nisu zaista PAIS niti su razvijeni od strane kompanija koji koriste PAIS. Korišćenje WFMS zahteva njihovu konfiguraciju koja bi odgovarala potrebama kompanije i koja se izvršava od strane sistema.

Puccini u svojoj disertaciji [14] nudi jedno od rešenja za ograničenja koja nastaju pojavljivanjem novih zahteva koji se ne mogu rešiti postojećim alatima i jezicima iz razloga što zavise od nekih novih koncepata ili što koriste strukture drugih jezika koje nisu podržane. Tradicionalno, to je rešavano promenom ili modifikacijom jezika radnih-tokova, ili čak kreiranjem novih. U svojoj disertaciji predlaže uklanjanje pomenutih ograničenja tako što nudi platformu koja služi kao osnov za proširive jezike. Platforma se zasniva na notaciji otvorenih objekata (eng. *open objects*), kao osnovnim gradivnim elementima. Otvoreni objekti imaju ulogu u predloženoj platformi, slično kao što klase imaju u UML-u [66]. Autor navodi da klase i objekti dele mnogo zajedničkih stvari, tako da se otvoreni objekti mogu smatrati ekstenzijama klasa. Srž ekstenzije je u tome da se stanje koje je obično enkapsulirano u objektu, eksternalizuje u mašini prelaza stanja (eng. *state machine*).

Dumas [2] se bavi sa BPM (eng. *BPM - Business Process Management*) i navodi da je BPM "umetnost i nauka kako se posao obavlja u organizaciji da bi se osigurali konzistentni ishodi i da se iskoristi svaka prilika za poboljšanje". U knjizi se detaljno obrađuju razni aspekti kao i zahtevi u sistemima poslovnih procesa, kao što su modelovanje procesa, kvalitativna i kvantitativna analiza procesa, redizajn i nadgledanje procesa, kao i procesno-svesni informacioni sistemi (eng. PAIS). *Dumas* se dalje naslanja na BPMS (eng. BPMS – Business Process Management Systems), odnosno sisteme koji podržavaju sve korake u poslovnim sistemima od dizajna i analize, pa do izvršavanja do nadgledanja poslovnih procesa. Da bi temeljno

razumeli BPMS, potrebno je detaljno izučiti BPMN (eng. BPMN – Business Process Model and Notation) standard. *Chinosi* [67] daje detaljniji uvod u BPMN standard, trenutno aktuelna verzija BPMN je 2.0 koja je pretrpela nekoliko revizija. *Aegesen* [68] rezimira BPMN 2.0 i daje neke evaluacije koje se odnose na analizu i dizajn poslovnih procesa. Razvoj alata zasnovanog na BPMN standardu, se uglavnom zasniva na referentnoj arhitekturi propisanoj od strane WfMC (eng. WfMC – *Worfklow Management Coalition*).

YAWL [69] (eng. *Yet Another Workflow Language*) je kreiran sa ciljem da pruži odgovore na pitanje da li je moguće u jezik za radne tokove ugraditi sveobuhvatnu podršku za šablone, pri tome da jezik ostane relativno jednostavan. Odlučeno je da glavni oslonac za YAWL budu *Petrijeve* mreže, jer je autor u [31] predočio zašto su *Petrijeve* mreže pogodne za specifikaciju radnih tokova. Mnogi šabloni radnih tokova mogu direktno izraziti kroz *Petrijeve* mreže [70], međutim pošto se ispostavilo da je neke šablone teže realizovati na predloženi način, to je dovelo do nekih specifičnih konstrukcija u YAWL-u. CPN (eng. *Colored Petri Nets*) formalizam [33], odnosno formalizam "Obojenih Petrijevih mreža" je iskorišćen kao nova formalna semantika za novi YAWL (eng. *newYAWL*), jer originalna semantika koja je specifirana kao sistem tranzicija nije pružala podršku za najnovije šablone kontrole toka, kao i šablone podataka i resursa [71]. Na osnovu opisanog YAWL ima kriterijume dizajna koji su usvojeni tokom razvojnog procesa i koji za cilj imaju da osiguraju podršku poslovnim procesima sa širokom primenom. Kriterijumi su sledeći:

- Pogodnost - YAWL je nastao sa idejom da podrži što više šablona radnih tokova, a pri tome iz šablona se crpe i uočavaju pogodne konstrukcije procesa, tako da što više šablona podržava, pogodniji je za opštu specifikaciju radnih tokova.
- Formalnost - Jezik poseduje formalne temelje, iako radni tokovi mogu biti veoma kompleksni po svojoj prirodi, izvršavanje tih tokova ne bi trebalo biti proizvoljno.
- Razumljivost - Domenski stručnjaci bez velikih poteškoća trebali bi da razumeju i da koriste rešenje, što znači da mora da bude intuitivno bez obzira ako domenski stručnjak nije vešt u IT oblasti.
- Usvojivost - Radni tokovi moraju biti izvršivi da bi se mogli usvojiti u neko radno okruženje, međutim treba da budu nezavisni od tehnologije, odnosno ne bi trebalo da budu vezani za određeno okruženje.
- Minimalnost - Kao što su autori naveli, jezik bi trebao da bude minimalan koliko god je to moguće, iz razloga kako bi jezik bio lakši za razumevanje, ali i za objašnjavanje.

YAWL je zasnovan na nabrojanim kriterijumima, iako sve kriterijume nije moguće ispoštovati do kraja bitno je konvergirati ka svakom od njih, u sklopu pomenutog YAWL

je pored ADEPT-a [72] bio jedan od prvih jezika koji je fleksibilan po dizajnu. Odnosno alternativni putevi izvršenja su navedeni eksplicitno u procesu tokom njegovog dizajna.

jBPM [73] je kreiran sa glavnom idejom da mora da podržava deklarativnu specifikaciju stanja radnog toka i programsku logiku. Takođe jBPM je dizajniran tako da omogućava jednostavno ugrađivanje osnovne mašine prelaza stanja, kako bi Java programerima bilo jednostavno da uključe jBPM u njihove projekte (jBPM je napravljen da se izvršava na Java platformi), kao i da omogućava podršku za najkompleksnije šablone radnog toka. Pomenuto rešenje svoju primenu nalazi u tri scenarija:

- U sklopu poslovnih aplikacija, kao aplikativna komponenta - U poslovnim aplikacijama zasnovanim na Java platformi, korišćenje rešenja se svodi na uključivanje programske biblioteke i korišćenje njenih funkcionalnosti
- U isporučivanju procesno baziranih aplikacija - jBPM se može uključiti u već gotove ERP (eng. *Enterprise Resource Planning*) proizvode poput SAP-a i Oracle. Aplikacija može dodatno da dozvoli pristup jBPM alatu za izvršavanje procesa određenim korisnicima ERP prozivoda, čime bi dozvolila dodatno proširenje i prilagođavanje.
- Kao komponenta u poslovnoj arhitekturi - jBPM je moguće isporučiti kao odvojenu komponentu u sklopu IT infrastrukture, gde bi jBPM upravljao poslovnim procesima postojećeg rešenja i nad postojećom bazom.

Međutim, iako to nije njegova prvobitna namena jBPM se dotakao i fleksibilnosti u poslovnim procesima. jBPM ima ugrađenu osnovnu podršku za *ad-hoc* modifikaciju tekućih instanci procesa, međutim samo za jednostavne šablone adaptacije [73].

Declare [74] je deklarativni sistem za upravljanje poslovnim procesima, ideja sa kojom je nastao je da pokaže da deklarativnim pristupom upravljanju radnih tokova je lakše napraviti balans između fleksibilnosti i podrške koje pruža rešenje. Dok tradicionalni pristupi koriste proceduralne procesne modele za eksplicitno specificiranje procedure za izvršavanje radnog toka, deklarativni pristup je zasnovan na ograničenjima, odnosno sve je moguće dok god nije eksplicitno zabranjeno [74]. Dakle, ovi modeli zasnovani na ograničenjima implicitno daju specifikaciju, odnosno tok izvršenja putem nametnutih ograničenja, odnosno dozvoljeno je sve što ne krši ograničenja.

Declare je proširiv, odnosno može se konfigurisati da pruža podršku za razne deklarativne jezike, dok dolazi sa ugrađenom podrškom za DecSerFlow [75] i ConDec [76]. Sastoji se iz tri glavne komponente dizajnera (eng. Designer), radnog okvira (eng. *Framework*) i radne liste (eng. *Worklist*). Dizajner komponenta se koristi za tri zadatka, da dizajnira procesne modele, da kreira šablone ograničenja i da verifikuje te modele. Radni okvir izvršava instance procesnog modela, takođe je zadužen za manipulaciju

i *ad-hoc* izmene tekućih instanci procesa. Svaki korisnik koristi svoju radnu listu da bi pristupio tekućim instancama procesa, iako radni okvir upravlja izvršenjem svih instanci, preko radne liste korisnik pristupa samo pripadajućim instancama, a korisnik takođe može obavljati aktivnosti u aktivnim instancama pripadajuće radne liste.

Declare se razlikuje od većine tradicionalnih WfMS po tome što koristi skup proizvoljnih šablona ograničenja, za razliku od ostalih koji koriste već utvrđeni skup konstrukcija koje služe za definisanje zavisnosti između aktivnosti u procesu kao što su sekvence, petlje i drugi. Šabloni koji se koriste su definisani u dizajner komponenti, gde svaki šablon ima jedinstveno ime, grafičku reprezentaciju i formalnu specifikaciju semantike u smislu LTL (eng. *Linear Temporal Logic*) [77, 78]. Međutim, zbog pomenutog načina funkcionisanja alat ne obezbeđuje okruženje za robusno izvršavanja procesa i za omogućavanje integracije sa aplikacijama, gde dodatno ne podržava specifikaciju zavisnosti podataka između aktivnosti niti provere korektnosti toka podataka [79].

Alaska Simulator je još jedan deklarativni sistem za upravljanje poslovnim procesima, za razliku od Declare alata, Alaska Simulator pruža korisnicima način za kreiranje grubog plana koji se kasnije inkrementalno validira. Alat je kreiran sa ciljem da odgovori na pitanje da li su korisnici kadri da prilagode određeni plan kako bi izvršili poslovni proces pri korišćenju deklarativnog pristupa [80]. Ta specifičnost ga u velikom razdvaja od Declare alata, koji ne pruža tu podršku i na kome se rezultati ne bi mogli replicirati. Iako autori smatraju da su prednosti deklarativnih alata mnogostruke, podvlače da prednosti i mane deklarativne paradigme za modelovanje poslovnih procesa još uvek nisu u potpunosti shvaćene. Najveća nepoznanica je način na koji bi korisnici razumeli i kontrolisali okruženje sa dobijenom fleksibilnošću u kompleksnom sistemu sa velikim brojem procesa i sa velikim brojem ograničenja.

Adaptivni *workflow engine* ima zadatak da pruži podršku za procese, kao što to rade i tradicionalni *workflow* sistemi, ali na takav način da omoguće da sistem može da se izbori sa određenom vrstom promena. Ove promene su različite, od onih najjednostavnijih gde se menja redosled dva zadatka, pa sve do onih koji zahtevaju redizajn skoro celog poslovnog procesa. Promene koje ovaj doktorat obrađuje nazivamo *dinamičkim promenama*, koje moraju da odgovore na pitanje "šta uraditi sa tekućim instancama radnog toka, kad se promeni definicija radnog toka"?

Trenutno najaktuelniji komercijalni alati, ne pružaju veliku podršku za upravljanje promenama, dok većina alata koja postoji i pruža podršku za upravljanje promenama dolazi iz akademske zajednice.

ADEPT je alat koji ima ugrađenu podršku za dinamičke promene i opisan je u [72]. Alat je prvobitno nastao za podršku procesima zdravstvene zaštite. *ADEPT* omogućava odstupanje od modelovane šeme procesa, dodavanjem novih koraka, preskakanje postojećih, povratak na prethodne i slično. Međutim, izostala je podrška za propagaciju izmena na tekuće instance procesa, prilikom promene procesne šeme. Sledeći korak je bio razvijanje *ADEPT2* [19] alata.*ADEPT2* zasniva svoje promene na šablonima na

visokom nivou i omogućuje operacije izmene koje se mogu iskoristiti za strukturalne promene, odnosno dodavanje fragmenta procesa, brisanje fragmenta procesa, kao i premeštanje fragmenta procesa. Međutim iz više različitih razloga i malog interesa kompanija za datu tehnologiju, ADEPT2 nije zaživeo, nego je integrisan u alat za industrijske potrebe sa nazivom *AristaFlow BPM Suite* [18].

FLOWer je komercijalni WFMs koji je zasnovan na rukovanju slučajevima (eng. *case handling)*. Rukovanje slučajevima je paradigma za podršku fleksibilnim poslovnim procesima sa fokusom na aspekt podataka, odnosno bazirana je na podacima kao tipičnim proizvodima procesa [81]. Za razliku od upravljanja radnim tokom koje koristi predefinisane procesne kontrolne strukture koje određuju šta treba da se uradi u radnom toku, rukovanje slučajevima se fokusira na to šta može biti urađeno da se postigne poslovni cilj. U okviru alata distribucija posla je razdvojena od autorizacije, tako da je moguće podržati akcije kao što su preskakanje, ili ponovno izvršavanje aktivnosti u procesu.

Iz prethodno navedenog, zaključuje se da su adaptabilni poslovni procesi, odnosno fleksibilno upravljanje radnim tokovima nešto što uzima i skreće dosta pažnje na sebe, a u cilju rešavanja problema koje donosi moderno poslovanje i tržište koje treba da odgovori na novonastale zahteve, kao što je to prethodno opisano. Pomenuto je da dinamička adaptabilnost nije bila tema, niti deo razmišljanja u poslovnim procesima, pa i alatima koji su nastali za podršku poslovnim procesima nedostaje pomenuta funkcionalnost. U okviru ovog poglavlja opisani su neki pristupi i predložena rešenja koja bi trebalo da daju odgovor na pitanje, kako je u tekućem radnom toku u sklopu nekog sistema za upravljanje istim, moguće izvršiti dinamičke adaptacije njegovih tekućih instanci.

U okviru ove disertacije dat je jedan predlog, rešenja problema dinamičke adaptacije u interpretaciono zasnovanom pristupu, odnosno pristupu u kome nema striktne separacije između vremena kreiranja i izvršavanja procesa (eng. *compile time and runtime*), odnosno u okviru živog okruženja (kao što je programski jezik i okruženje Pharo), jer se instance tekućeg radnog toka kontrolišu interpretiranjem njegovog modela, te je moguće na osnovu skupa određenih pravila vršiti adaptacije nad nekom instancom radnog toka, i zbog interpretaciono zasnovanog pristupa te promene se odmah odražavaju na tekuće instance radnog toka.

Poglavlje 3

Dizajn i arhitektura sistema

U ovom poglavlju definisani su arhitektura, dizajn i implementacija NewWave platforme i njenog proširivog jezgra, odnosno *Workflow Engine*-a. Postavljena arhitektura je glavni oslonac i podrška za manipulacije radnim tokovima koje su opisane u sledećem poglavlju.

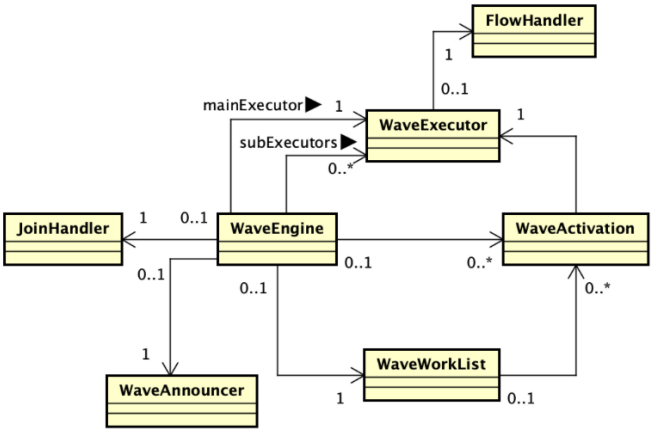
3.1 Arhitektura

Glavni deo NewWave-a nazvan je *Workflow Engine Core* ili jednostavno *Core (Jezgro)*. *Core* je osnovni gradivni deo bez koga NewWave ne može da funkcioniše. Na svom najelementarnijem nivou *Core* omogućava upravljanje bazičnim radnim tokovima, koji mogu automatizovati zadatke u okviru jednog sistema. *Workflow Engine Core* takođe podržava razne šablone kao što su *xor, split, and, join* kao i razne vrste događaja *start, end, boundary* koji se mogu primeniti na bazičnim radnim tokovima. Pojednostavljen prikaz arhitekture *Core* dela dat je na slici 3.1 i inicijalno je predstavljen u [82].

Core je nezavisan i može da funkcioniše samostalno, za razliku od svih ostalih delova koji zahtevaju *core* deo. Jedini izuzetak je *web* aplikacija, koja je kreirana tako da može da radi nezavisno od *core* dela i koja se može iskoristiti za kreiranje predefinisanih konfiguracija aplikacije i o kojoj će više reči biti kasnije.

NewWave je kreiran sa idejom da bude proširiv. Odnosno elemente sistema je moguće menjati, dodavati i uklanjati po potrebi, a sve u cilju boljoj podršci za izvršavanje procesa, odnosno za dobijanje generalizovanog, ali istovremeno i adaptabilnog alata. Proširivost u okviru *NewWave* dolazi u nekoliko različitih manifestacija, odnosno koriste se komponente, dodaci (eng. *plugins*) i različite konfiguracije, tako da:

- komponente - delovi sistema koji imaju jasno određene zadatke i koji obezbeđuju slabu vezu zavisnosti između delova tog sistema, time omogućujući da budu



Slika 3.1: Pojednostavljena arhitekura *Workflow Engine Core* dela

lako zamenljivi. Komponente predstavljaju sredstvo za eliminisanje složenosti softverskog rešenja.

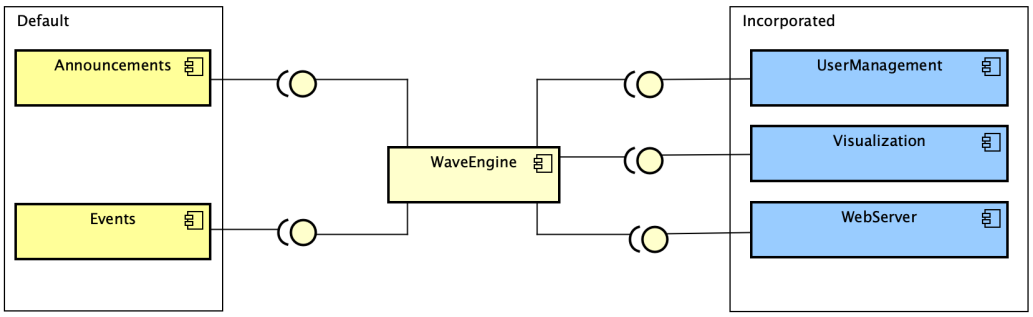
- dodatke (plugin) - pružaju mogućnost korišćenja gotovih biblioteka i aplikacija. Jedan dodatak može da se sastoji iz više komponenti.

- konfiguracije - doprinose da NewWave može da se izvršava u različitim režimima rada sa različitim dodacima i pod različitim uslovima.

3.1.1 WaveEngine komponenta

WaveEngine je jedna od glavnih komponenti i predstavlja *engine (izvršno okruženje)* ovog rešenja. *Engine* kontroliše i organizuje izvršavanje radnog toka, sa svim informacijama dostupnim u vremenu izvršavanja *(runtime)* koje se koriste da se postigne efikasna kontrola samog izvršavanja, kao što su radne liste, aktivacije, kontrola više izvršivača (*egzekutora*), dodataka i slično. *WaveEngine* može da izvršava više radnih tokova, gde svaki radni tok ima jednu glavnu izvršnu komponentu (*WaveExecutor*), koja je odgovorna za njegovo izvršavanje. U slučaju potrebe, u toku rada, moguće je kreirati više izvršnih komponenti, npr. kada se u radnom toku izvrši *ParallelSplit* (paralelno grananje), u tom trenutku se kreira dodatna izvršna komponenta koja je podređena glavnoj (onoj koja ga je kreirala). Naknadno kreirane izvršivne komponente postoje sve dok za njih postoji potreba u okviru procesa, odnosno do spajanja grana u okviru radnog toka (dok se ne izvrši *ParallelJoin*).

Funkcionalnosti NewWave-a je moguće proširiti korišćenjem raznih dodataka. Svaki dodatak može da ima više komponenti, odnosno delova koji imaju jasno određene



Slika 3.2: Komponente u okviru WaveEngine dela

zadatke. Komponente koje NewWave koristi kao zavisnosti u najvećem broju povezane su na *WaveEngine*, odnosno glavnu komponentu. Na slici 3.2 prikazane su neke od bitnijih komponenti koje koristi NewWave. Podeljene su na dva dela *Default i Incorporated*, odnosno podrazumevane i ugrađene. Podrazumevane komponente su one koje dolaze uz *Core* deo i nisu predviđene da se menjaju od strane korisnika, odnosno moguće ih je zameniti ali pošto imaju veliku ulogu u tome kako se ponaša *NewWave*, njihova izmena bi se značajno odrazila na arhitekturu ovog rešenja, što je opet dozvoljeno ukoliko neki razvojni tim želi da kopira *NewWave* u repozitorijum kojim oni upravljaju (eng. *fork*) i da eksperimentišu sa željenim promenama, u trenutku pisanja ove disertacije postoji sedam *fork*-ova *NewWave* rešenja. U podrazumevane komponente spadaju:

- Announcements - Obaveštenja na nivou čitavog okruženja, NewWave ih koristi da upravlja događajima.
- Events - Događaji imaju zadatak da obaveste različite delove sistema o dešavanjima unutar NewWave-a.

Ugrađene komponente su komponente koje ne spadaju u *Core* deo, one dolaze uz različite konfiguracije NewWave-a o kojima će biti više reči kasnije. Moguće ih je koristiti proizvoljno u zavisnosti od konfiguracije i moguće ih je zameniti sa odgovarajućim alternativnim komponentama. Dodatke koji predstavljaju gotove biblioteke, moguće je prilagoditi za specifično ponašanje koje koristi NewWave, tako da se ti dodaci ponašaju kao komponente sistema (npr. u implementaciji dodatak za vizualizaciju koristi *Roassal* biblioteku, koji je prilagođen za prikazivanje radnih tokova u okviru NewWave-a, WebServer koristi Teapot za isporučivanje zadataka, dok se Magritte koristi za generisanje formi). U ugrađene komponente spadaju:

- UserManagement - komponenta koja pruža podrazumevanu korisničku konfiguraciju i rukovanje korisnicima i korisničkim ulogama.

- Visualization - komponenta koja omogućava vizualizaciju, pored vizualizacije koristi se i za inspekciju radnih tokova.
- WebServer - za isporučivanje zadataka i preuzimanje odgovora putem HTTP-a.

WaveEngine je takođe odgovoran za manipulaciju procesima. Slanjem poruke `addProcess:name:` podržano je dodavanje procesa, `startProcess:` ima zadatak da pokrene proces, dok `stopProcess:` zaustavlja proces. Slanjem neke od ovih poruka se aktivira i određeni element sistema koji je za njega zadužen, odnosno *WaveEngine* delegira odgovornost ka odgovarajućoj komponenti, npr. `startProcess:` poziva odgovarajući *WaveExecutor* koji započinje sa izvršavanjem datog procesa.

WaveEngine može po potrebi, u zavisnosti od količine definisanih procesa koje treba da izvrši da kreira proizvoljan broj *WaveExecutor*-a. Svaki pokrenuti proces sadrži jedan glavni *WaveExecutor*, za koje je je odgovoran *WaveEngine*, dok svaki od tih *WaveExecutor*-a mogu imati svoje podređene izvršavače za koje su oni nadređeni.

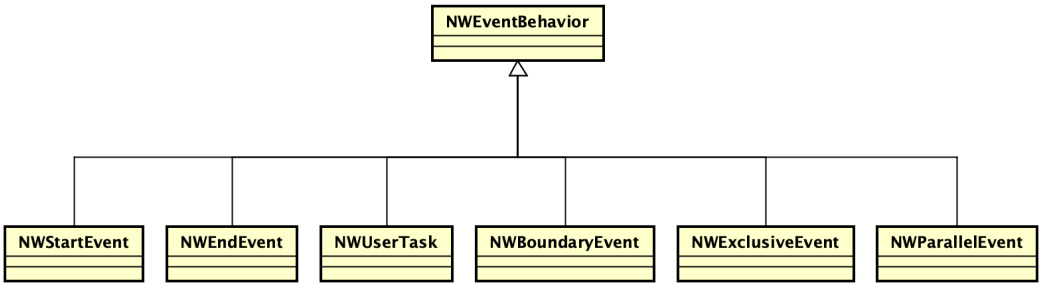
3.1.2 WaveExecutor komponenta

WaveExecutor ima zadatak da pruži odgovarajuće okruženje za izvršavanje elemenata u okviru radnog toka. Da bi element mogao da se izvrši, odnosno da bi *WaveExecutor* dobio element za izvršavanje, on se oslanja na usluge koje pruža rukovalac radnog toka *FlowHandler*. *FlowHandler* upravlja tokom izvršavanja radnog toka i sadrži informacije o elementima koji se izvršavaju. *FlowHandler* sadrži niz sekvenci koje se formiraju prilikom kreiranja radnog toka. Svaka sekvenca sadrži dva čvora, početni i krajnji na osnovu kojih *FlowHandler* zna da prolazi kroz radni tok. Nakon što završi sa trenutnim elementom, *FlowHandler* određuje, iz definicije radnog toka, sledeći element koji treba da se izvrši. Međutim, pošto *NewWave* radi u interpreter modu, *FlowHandler* nema i ne zna unapred sve predefinisane putanje u okviru radnog toka, nego se one formiraju u zavisnosti od toga kako se elementi izvršavaju, odnosno kako se prolazi kroz radni tok. Takođe, čuva informacije o izvršenim elementima i o redosledu izvršavanja. Ova istorija izvršavanja (eng. *execution history*) je bitna za naknadno praćenje koraka izvršavanja u radnom toku.

Izvršavanje radnog toka počinje slanjem poruke `startEngine` ka *WaveEngine* komponenti 3.1. Data poruka spada u poruke koje označavaju događaje u okviru sistema, konkretno *start* događaj. Ova poruka aktivira *WaveExecutor* koji započinje izvršavanje prethodno definisanog radnog toka. Na osnovu poslate poruke, dati model počinje da se interpretira i kreira se njegova instanca, koja se izvršava u okviru alata. Pošto je i model na osnovu kojeg je kreirana instanca učitan u okruženje i radnu memoriju, svaka instanca u okviru *NewWave*-a zna na osnovu kog modela je kreirana, što će biti posebno korisno kasnije, tokom vršenja adaptacija, jer se na osnovu tražene promene nad modelom vrši propagacija nad njegovim instancama i proverava se mogućnost za izvršavanje tražene promene.

Izvršavanje radnog toka može biti zabeleženo (radi prikupljanja dodatnih informacija) i propraćeno od početka do kraja. Tokom izvršavanja korisnik je u interakciji sa sistemom u zavisnosti od vrste zadatka koji se izvršava i na taj način utiče na tok izvršavanja u zavisnosti od ponuđenih i odabranih opcija u okviru radnog toka.

U zavisnosti od čvora koji se trenutno izvršava, *WaveExecutor* zna kako da izvrši trenutni zadatak na osnovu informacije koja je pridružena uz čvor. Informacija je kratkog tekstualnog opisa, npr. *početak, kraj, zadatak, grananje i sl.* Na osnovu te informacije *WaveExecutor* pristupa sistemu koji opisuje ponašanje za izvršavanje različitih vrsta čvorova, prikazano na slici 3.3.



Slika 3.3: Komponente u okviru WaveEngine dela

Svaka od prikazanih klasa zna da odgovori na poruku `performExecution:executor:` gde je prvi parametar koji se prosleđuje čvor koji se trenutno izvršava, a drugi *WaveExecutor* na kome se čvor izvršava. Nakon izvršavanja čvor se dodaje u listu izvršenih čvorova a njegova aktivacija menja u stanje završeno, nakon čega *WaveExecutor* proverava da li postoji sledeći čvor koji treba da se izvrši. Postupak se ponavlja do kraja radnog toka.

3.1.3 Aktivacije u NewWave-u

Aktivacije su jako bitan deo *NewWave* platforme, odnosno njenog dela za upravljanje procesima, konkretnije *Wave Engine*-a. Svaki čvor (zadatak ili gejt) u okviru ovog alata ima odgovarajuću povezanu aktivaciju. Aktivacija ukazuje na stanje čvora. Svaki čvor može imati više stanja, ali u jednom trenutku može biti opisan najviše jednim stanjem, i to su: aktivan, neaktivan, završen ili nije uspeo da se završi. Inicijalno su sve aktivacije u neaktivnom stanju, odnosno nisu aktivirane. Aktivacija postaje aktivna, odnosno aktivira se, kad *WaveExecutor* pokuša da izvrši dati čvor. U tom

trenutku, čvor je aktiviran i informacija o tome se beleži u *WaveEngine*-u. Aktivaciju kreira *WaveExecutor* prilikom prvog pokušaja da izvrši dati čvor. Svaka aktivacija ima i informaciju o čvoru i izvršavaocu koji ga je kreirao.

Nakon izvršavanja, aktivacija menja stanje u završeno i ažurira tu informaciju u *WaveEngine*-u, sa svim dodatnim informacijama koje se zahtevaju. Ove informacije se koriste za dobijanje informacija o čvorovima u *WaveEngine*-a, na osnovu kojih se može proveriti u kom su stanju, da li su aktivni ili se još nisu aktivirali, pored toga moguće je dobiti informacija o instanci izvršivača *WaveExecutor* koji je izvršava neki čvor i slično.

Ovaj pristup dozvoljava detaljan uvid u sve izvršavače i čvorove koji su prošli kroz sistem, jer u slučaju da je neki čvor završio u stanju u kome nije očekivano, moguće je napraviti introspekciju i detaljne analize koje mogu značiti za poboljšanje rada alata.

3.1.4 Obaveštenja kao mehanizam komunikacije

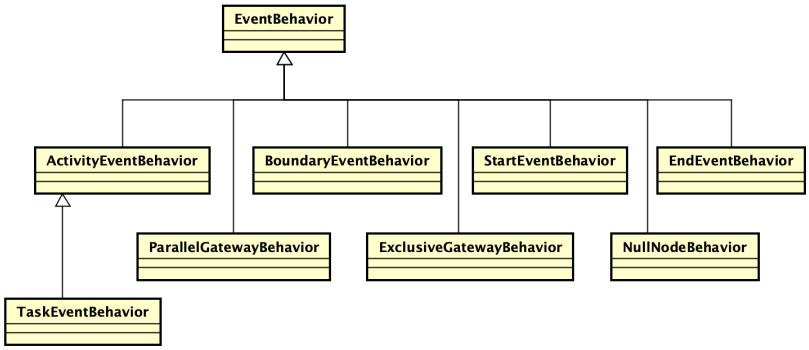
NewWave koristi obaveštenja kako bi upravljao događajima u okviru alata, pa i okruženja u kome se izvršava. Obaveštenja se zasnivaju na *Announcement* radnom okviru (eng. *framework*). *Announcement* radni okvir je zasnovan na *Observer* šablonu [83] i pruža njegovu generičku implementaciju. Korišćenje ovog radnog okvira pojednostavljuje upravljanje sistemom za obaveštenja u okviru NewWave-a i pruža mogućnosti prilagođavanja za izvršavanje različitih tipova događaja.

Primera radi, može se posmatrati sinhronizacija, gde više grana konvergira u jednu. Da bi izvršavanje radnog toka bilo moguće, nakon tačke spajanja, sve prethodne paralelne grane trebaju da se završe. Svaki put, kada grana dostigne tačku sinhronizacije, objavi (eng. *announce*) se *JoinEvent* (obaveštenje o spajanju). To obaveštenje "slušaju" zainteresovani delovi sistema, odnosno oni delovi koji treba da odreaguju i na nastave sa izvršavanjem nakon spajanja, te se to obaveštenje koristi da se proveri da li je sinhronizacija završena.

Announcement radni okvir dozvoljava menjanje i prilagođavanje, tako što se *Announcement* klasa može naslediti i prilagoditi specifičnim potrebama projekta u okviru koga se koristi, a opet zbog veze nasleđivanja ostaje veza za komuniciranje sa ostatkom sistema.

JoinEventAnnouncer ima dve promenljive (varijable) instance *parameter* i *executor*. Prva promenljiva - *parameter* predstavlja sekvencu koja vodi do tačke spajanja, a druga promenljiva - *executor* je instanca klase *WaveExecutor* koja radi izvršavanje. Pošto radni okvir dozvoljava da se obaveštenja prosleđuju kao objekti, ovo ponašanje se koristi za obaveštavanje dela *WaveEngine*-a koji je zadužen za sinhronizaciju sa objektima koji se izvršavaju. NewWave pruža podršku za različite vrste događaja i dozvoljava ugrađivanje novih; češće korišćeni prikazani su na slici 3.4.

Bitno je napomenuti, da se sva obaveštenja koja pošalje NewWave mogu registrovati u svakom delu okruženja u kome se izvršava, identično neki drugi element iz okruženja može poslati obaveštenje za koje se NewWave može registrovati da sluša. Ovo

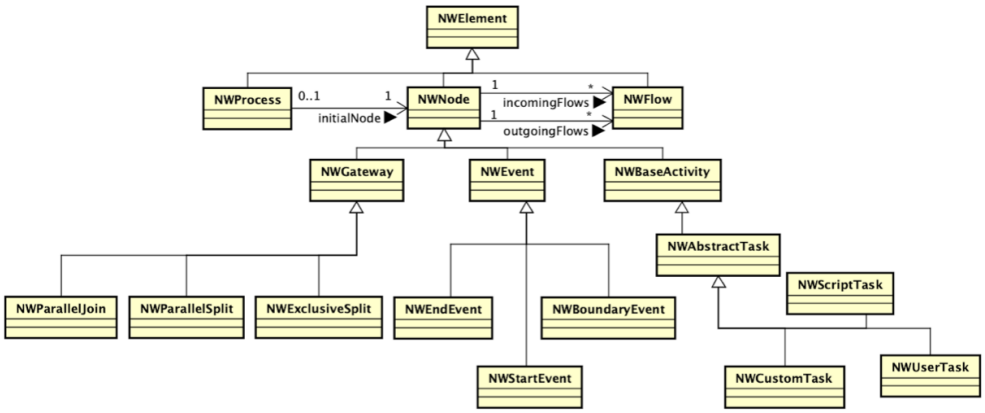


Slika 3.4: Podržana obaveštenja

ponašanje omogućava slabu povezanost (eng. *loose coupling*) [84] između komponenti sistema, tačnije omogućava da neka komponenta ima malo ili nimalo znanja o drugim komponentama sistema, čime se povećava fleksibilnost samog rešenja i omogućava da se komponente dodaju ili menjaju, bez velikog uticaja na postojeće rešenje.

3.1.5 Gradivni blokovi

Specifikacija radnog toka u okviru NewWave oslanja se na njegove gradivne blokove, odnosno elemente koji su kreirani tako da odgovaraju BPMN-u [8]. Gradivni blokovi predstavljaju odgovarajuće elemente BPMN specifikacije (BPMN Proces odgovara elementu *NWProcess*, BPMN StartEvent odgovara elementu *NWStartEvent* i tako dalje). Pojednostavljena hijerarhija ovih elemenata prikazana je na slici 3.4.



Slika 3.5: Gradivni elementi NewWave Core dela

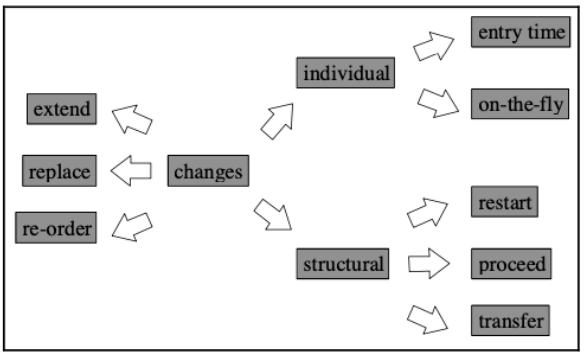
Prikazani elementi učestvuju u formiranju radnog toka, odnosno koriste se za opisivanje modela radnog toka. Kompozicijom ovih elemenata, dobijamo model radnog toka koga je moguće instancirati i pokrenuti njegovo izvršavanje. Već je pomenuto da *WaveEngine* radi kao interpreter, te da bi se uklopili u postojeće zahteve, ovi elementi takođe imaju dinamičku prirodu. Odnosno oni su deo dinamičkog okruženja, te im je po potrebi moguće pristupiti i izvršiti potencijalne korekcije nad njima tokom samog rada alata, odnosno tokom izvršavanja procesa. *NWElement* je glavni gradivni element jednog radnog toka, svi ostali elementi se nasleđuju iz njega. *NWProcess* opisuje jedan proces radnog toka i između ostalog u sebi sadrži inicijalni čvor `initialNode` koji pomoću *FlowHandler*-a propagira kroz elemente procesa. Jedan čvor *NWNode* može da ima više ulaznih (`incomingFlows`) i izlaznih (`outgoingFlows`) tokova koji pripadaju procesu. Svaki tok *NWFlow* ima dva čvora, početni i krajnji, i oni zajedno čine jednu sekvencu ili tok. *NWGateway* predstavljaju kapije koje služe za formiranje alternativnih tokova izvršavanja jednog radnog toka. Kapije mogu biti ekskluzivne (eng. *XOR-split*) gde se odabira jedna od više ponuđenih grana i izvršava se sekvencijalno ili paralelne (eng. *AND-split*) gde se izvršavaju u paraleli sve ponuđene grane. *NWEvent* opisuju događaje koji se mogu desiti u okviru NewWave-a i oni su detaljno opisani u prethodnoj sekciji. Zadaci koji su podržani nasleđuju *NWAbstractTask* klasu i predstavlja deo aktivnosti sistema, odnosno ona nasleđuje *NWBaseActivity*. Trenutno su podržana tri vrste zadataka *NWScriptTask, NWUserTask i NWCustomTask*. *NWScriptTask* opisuje sve zadatke koji su automatski, odnosno zadatke koji podržavaju proizvoljnu programsku logiku za automatizovanje nekih zadataka. *NWUserTask* opisuje zadatke koji su namenjeni za korisničko izvršavanje, najčešće u obliku popunjavanja informacija kroz korisničke forme. *NWCustomTask* opisuje sve zadatke koji ne spadaju u prethodno pomenute, odnosno za omogućava definisanje kompleksnih zadataka. Omogućava ugradnju i interakciju sa različitim elementima sistema i bibliotekama kako bi na odgovarajući način prikupio tražene podatke. Povezivanje sa nativnom grafičkom bibliotekom u Pharo okruženju u cilju izvršavanja nekog zadatka je jedan primer kompleksnog zadatka.

3.2 Fleksibilnost u podršci workflow

Kao što je prikazano u prethodnim poglavljima NewWave je kreiran da bude fleksibilan, kako bi odgovorio na potrebe dinamičkih zahteva koji su postali sve češći u eri savremenog poslovanja. NewWave donosi nove ideje i koncepte koje realizuje korišćenjem dinamičkog programskog jezika sa živim okruženjem i dinamičkih promena.

Postoji nekoliko vrsta izmena koje su identifikovane, i prikazane su na slici 3.6. Ove promene opisane su u [85] i dele se na:

- individualne - *ad-hoc* adaptacija radnog toka, gde izmena utiče na jedan ili niz ograničenih zadataka, koje se dešavaju dok zadatak još nije u sistemu. Druga je



Slika 3.6: Klasifikacije promene (slika preuzeta iz [85])

"on-the-fly" adaptacija, odnosno adaptacija u letu dok je zadatak u sistemu.

- strukturalne - evolutivne promene radnog procesa.

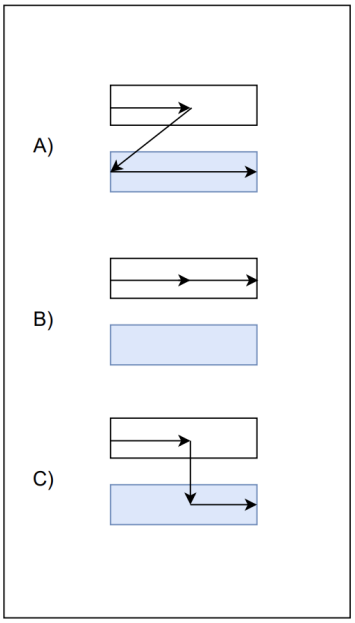
NewWave i podržavajući radni okvir bez problema mogu da se izbore sa individualnim promenama. Mnogo kompleksniji slučajevi dešavaju se u strukturalnim promenama. Na slici 3.6 sa leve strane prikazana su tri različita načina kojim se može promeniti radni tok.

- extend - Definicija procesa je proširena (npr. dodavanjem novih zadataka)
- replace - Zadaci su zamenjeni drugim zadacima
- re-order - Redosled zadataka je promenjen

Slika 3.7 opisuje tri moguće alternative za upravljanje postojećim slučajevima u sistemu kad se promeni definicija radnog toka, takođe date u [85].

Promene koje su obeležene sa *A)*, predstavljaju ponovno pokretanje (eng. *restart*) i prilagođavanje vrše tako što, restartuju započeto izvršavanje radnog toka i kreću iz početka sa novom verzijom radnog toka, dok promene obeležene sa *B)*, opisuju nastavljanje (eng. *proceed*) i one nastavljaju izvršavanje po zadatom radnom toku dok se ne završe, tek prvo naredno izvršavanje nakon izvršene promene biće izvršeno po novoj verziji radnog toka. Ove vrste promena su podržane velikim brojem komercijalnih alata i nisu predmet ovog istraživanja.

Promene koje su obeležene sa *C)* opisuju prebacivanje (eng. *transfer*) su promene koje opisuju dinamičku promenu i koje treba da adaptiraju pokrenute instance na novu verziju radnog toka. Dinamičke promene su posebno interesantne jer je potrebno proveriti da li su svi zahtevi koji su neophodni za adaptaciju ispunjeni. Radni okvir u okviru opisanog rešenja opisuje na koji način primenjuje *extend, replace i re-order* strukturalne promene koje se prenose na tekuću instancu procesa.



Slika 3.7: Upravljanje promenama

Da bi se neki proces uspešno adaptirao, potrebno je ispuniti niz kriterijuma koji su navedeni u nastavku:

- Workflow engine radi u interpreter modu;
- Na osnovu zahteva za adaptaciju moguće je naći mesto gde se pokušava adaptacija;
- Stanje čvora ispred tačke adaptacije mora biti u nekom od stanja *neaktivan, aktivan, pauziran*, a nikako *otkazan* ili *završen*;
- Adaptacija ne poništava stanje sledećeg čvora;
- Pokušava se primena adaptabilnosti nad odabranim trenutno pokrenutim instacama procesnog modela.

NewWave radi u intepreter modu, što znači da je model procesa učitan u radnoj memoriji na osnovu koga se pokreće instance procesa. Ovo ima nekoliko benefita od kojih je najveći taj da je relativno jednostavno (sa tehničke strane) naći model procesa na osnovu njegove instance i izvršiti poređenje stanja modela sa instancom.

Stanje čvora ispred tačke adaptacije govori da li je uopšte dozvoljeno pokušati adaptaciju. Ukoliko je čvor aktiviran, a nije završen moguće je pokušati adaptaciju pod uslovom da je proces "zaključan", odnosno da nije moguće izvršavati proces dok se postupak adaptacije ne završi.

Na ovaj kriterijum se direktno odnosni sledeći, koji se odnosi na to da adaptacija ne poništava stanje sledećeg čvora. Ovo bio kriterijum za izlaznu stranu novododatog čvora, gde adaptacija sigurno ne menja ulazne uslove sledećeg čvora, odnosno nakon adaptacije ulazni podaci sledećeg čvora su zadovoljeni.

Poslednja tačka prethodnog listinga ima niz koraka koji se moraju ispuniti da bi se proces adaptacije smatrao uspešnim. U procesu adaptacije razmatramo skup postojećih zadataka (eng. *tasks*) T_p i skup zadataka koji se dodaju T_d, skup svih zadataka je $T = T_p \cup T_d$. Takođe od značaja za proces adaptacije biće skup postojećih gejtova (eng. *gates*) G_p kao i skup gejtova koji se dodaju G_d. Skup svih gejtova je $G = G_p \cup G_d$, gde $\forall g_i \in G$ postoji prethodni i sledeći čvor.

N - skup svih čvorova (eng. *nodes*) u procesu
$T_p = \{t_1, ..., t_{np}\}$ - skup svih postojećih zadataka
$T_d = \{t_1, ..., t_{nd}\}$ - skup svih zadataka koji se dodaju
$T = T_p \cup T_d$ - skup svih zadataka
$G_p = \{g_1, ..., g_{np}\}$ - skup svih postojećih gejtova
$G_d = \{g_1, ..., g_{nd}\}$ - skup svih gejtova koji se dodaju
$G = G_p \cup G_d$ - skup svih gejtova

Skup I je skup svih izmena. Izmene se mogu odnositi na proširenje, zamenu ili zamenu redosleda čvorova, odnosno mogu se odnositi na gejtove ili na zadatke. Skup izmena je I je definisan sa:

$$I = \{i_j \in T_d \vee i_j \in G_d \big| j = 1, ...n_u - 1\},$$

gde indeks j predstavlja mesto gde se može desiti izmena, a n_u ukupan broj čvorova u procesu u trenutku izmene. Kako se svaki od čvorova u toku procesa može naći u jednom stanja N - *neaktivan* (eng. *non-active*),A - *aktivan* (eng. *active*), P - *pauziran* (eng. *paused*), C - *otkazan* (eng. *canceled*),F - *završen* (eng. *finished*), možemo definisati funkciju f koja skup svih postojećih čvorova preslikava u jedno od stanja.

$$f \colon N_p \mapsto S, \text{ gde je } S = \{N, A, P, C, F\}$$

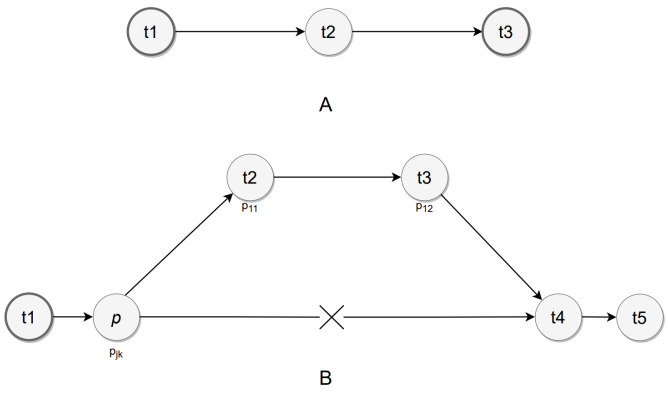
Izmena i_j se može odigrati pod uslovom da je $f(n_{pj}) \in \{N, A, P\}$.Uz poštovanje datih polaznih pretpostavki dalje opisujemo proširenje radnog toka, zamenu elemenata u radnom toku, ili zamenu redosleda izvršavanja.

Proširenja predstavljaju dodavanje jednog ili više čvorova na mestu izmene.

$$N' = N_p \cup \{i_{jk,j=1,...n_u,k=1,...m} \in I\} \text{ gde važi da } (\forall n_i, i > j)(i = i + k)$$

U zadatom opisu N' je skup svih čvorova nakon proširenja, N_p skup svih čvorova pre proširenja, i_{jk} je proširenje, j je mesto proširenja, a k je broj zadataka sa kojim

se vrši proširenje, $m = t_{nd} + g_{nd}$, n_i je i-ti čvor u procesu. Slika 3.8 ilustruje primer proširenja, gde je A stanje pre proširenja, dok je B stanje nakon proširenja.



Slika 3.8: Primer proširenja

Kod zamena, E predstavlja skup izmena koje se odnose na zamenu jednog ili više zadataka, a $E \subset I$. Zamene su opisane sa:

$$E = \{e_{jk,j=1,\dots n-1,k=1,\dots m} \in I \big| (\forall e_{jk})(\exists t_{j+k})(t_{j+k} \mapsto e_{jk})\}$$

Slično kao i kod proširenja, j je indeks izmene, a k je broj zamena. Za svaku zamenu mora da postoji odgovarajući element iz skupa izmena sa kojim se vrši zamena. Slika 3.9 prikazuje primer zamene zadataka, gde je A stanje pre zamene, dok je B stanje nakon zamene.

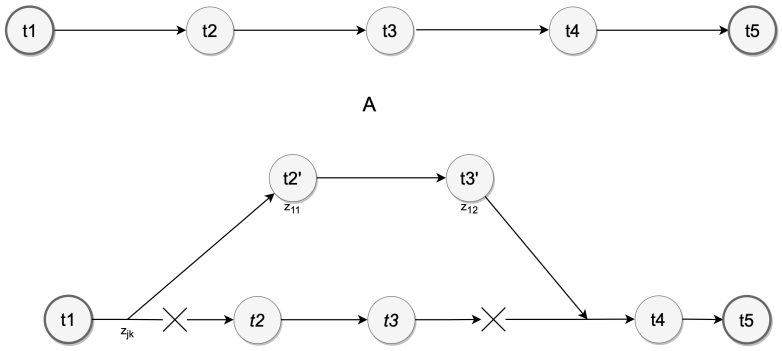
R predstavlja skup zamene redosleda, koje se odnose na zamenu redosleda naredna dva izvršavanja, gde $R \subset I$. Zamena redosleda izvršavanja je opisana sa:

$$R = \{r_{j,j=1\dots n-2} \in I | (\forall r_j)(\exists (t_{j+1} \wedge t_{j+2}))(t_{j+1} \mapsto t_{j+2} \wedge t_{j+2} \mapsto t_{j+1})\}$$

U opisu zamene redosleda, r je zamena koja treba da se izvrši, a j je mesto izvršavanja zamene. Za svaku zamenu redosleda moraju da postoje naredna dva elementa kojima se vrši zamena.

3.2.1 Primenjivanje adaptacija

Primenjivanje adaptacija se vrši nad postojećim modelom koji je pokrenut u okviru sistema. Rezultat adaptacije treba da predstavlja promenjen model radnog toka sa



Slika 3.9: Primer zamene zadataka

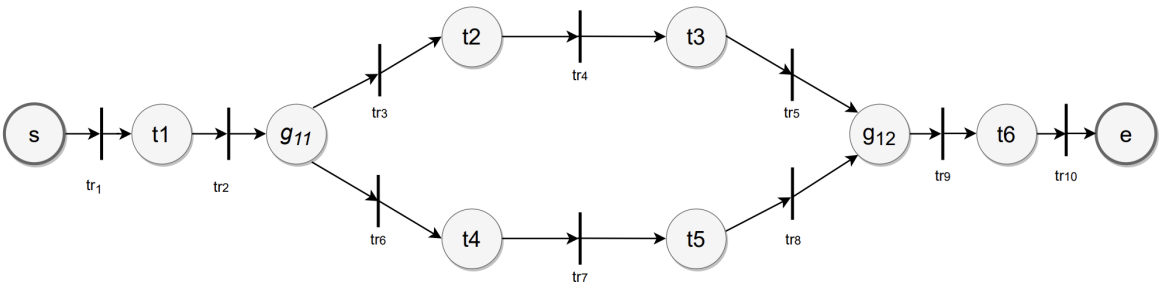
odraženim promenama na tekućim instancama datog modela, nad kojima je ta izmena moguća.

Primena adaptacije ilustrovana je na primeru koji je prikazan na slici 3.10. Na slici se nalazi deset čvorova, od toga šest zadataka *t1...t6*, jedan gejt koji se sastoji iz dva čvora g_{11}, g_{12}, kao i dva posebna čvora *s i e*, koji opisuju početak i kraj radnog toka respektivno. Sve čvorove možemo da opišemo skupom N, gde je:

N - skup svih čvorova

U primeru sa slike 3.10, skup N sastojao bi se iz sledećih čvorova:

$$N = \{s, t1, g_{11}, t2, t3, t4, t5, g_{12}, t6, e\}$$



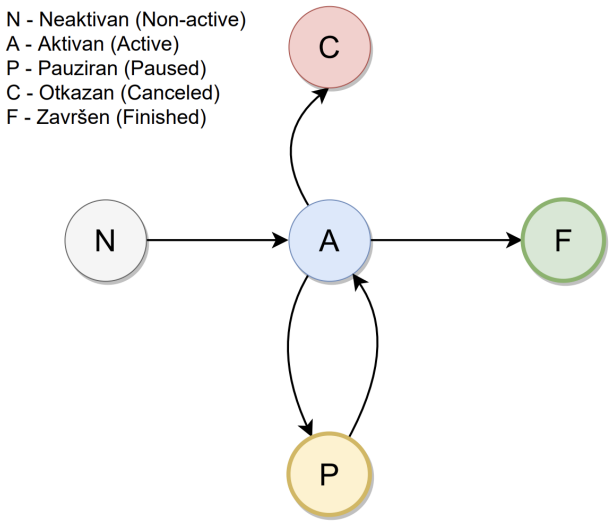
Slika 3.10: Primer radnog toka za proširivanje

Svi čvorovi imaju jednu ulaznu i jednu izlaznu granu, odnosno tranziciju obeleženu sa tr_x. Izuzetak su početni i krajnji čvor, gde početni čvor nema ulaznu granu, a krajnji

čvor nema izlaznu granu. Svaka grana ima izvorišni i odredišni čvor koji se nalaze na početku, odnosno na kraju tranzicije.

Da bi imao tačan uvid u stanja instanci radnog toka *NewWave* vodi evidenciju o stanju čvorova, tranzicija, zadataka i drugih elemenata radnog toka koji između ostalog učestvuju u procesu adaptacije. *NewWave* vodi evidenciju o svojim čvorovima u takozvanoj listi otvorenih čvorova (eng. *open node list*) koja je u *NewWave* platformi predstavljena sa *WaveWorkList* elementom.

Čvorovi imaju životni ciklus kojim su opisani i koji zavisi od stanja samog čvora. Čvor može biti *neaktivan, aktivan, završen,prekinut,pauziran*. Svaki čvor je inicijalno *neaktivan*, on ostaje u tom stanju sve dok *NewWave* ne pokuša da ga izvrši, kad se njegovo stanje menja u *aktivan* i za njega se veže *WaveActivation* element koji ima zadatak da upravlja datim čvorom. Dijagram prelaza stanja čvora u okviru *NewWave* prikazan je na slici 3.11.



Slika 3.11: Dijagram prelaza stanja čvora

Aktivacije opisujemo skupom A gde je:

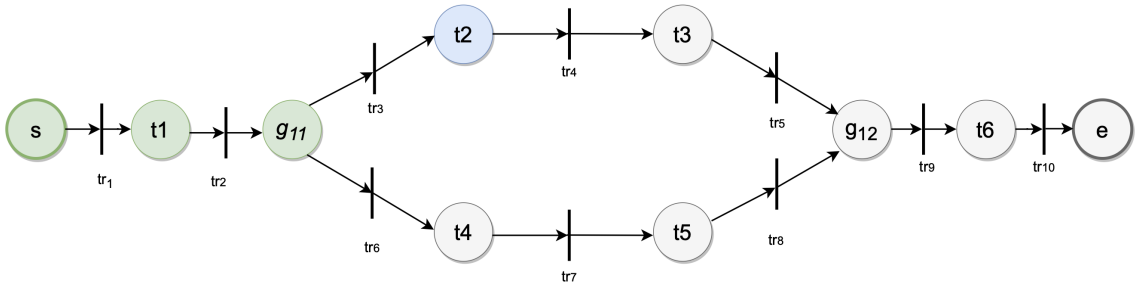
$$A \subset N$$

U skup A ulaze samo aktivirani čvorovi. Nakon modifikacije slike 3.10, prikazano je novo stanje gde su aktivni čvorovi *s, t1,* g_{11}*, t2*, tako da je skup A :

$$A = \{s, t1, g_{11}, t2\}$$

Slika 3.12, prikazuje novo stanje radnog toka sa stanjima čvorova, gde su čvorovi *s,t1,*g_{11} završeni, a čvor *t2* je aktivan. Završeni čvorovi označeni su zelenom bojom, plavi su aktivni, dok su sivom bojom označeni neaktivni čvorovi.

Potrebno je napomenuti da je *FlowHandler* element (deo *NewWave* platforme), zadužen za upravljanje radnim tokom, odnosno navigaciju u radnom toku, koji prati tranzicije i obeležava čvorove za izvršavanje. Odnosno za dati primer, bile bi izvršene tranzicije tr_1, tr_2, tr_3, gde tranzicija tr_1 povezuje čvorove *s i t1*, tranzicija tr_2 povezuje čvorove *t1 i* g_{11} i na kraju tranzicija tr_3 povezuje čvorove g_{11} *i t2*.



Slika 3.12: Primer radnog toka za proširivanje, sa završenim čvorovima

3.2.2 Primena proširivanja

Na osnovu definisanih pravila u ovoj sekciji, definišemo skup izmena za proširivanje, tako da:

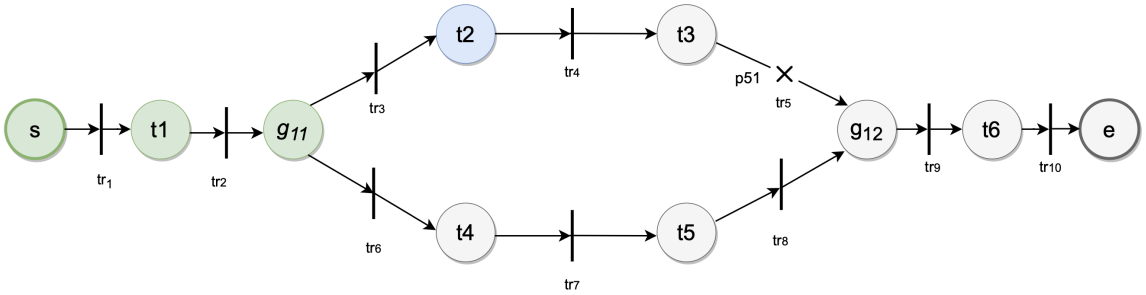
$$I = \{t_n\}$$

Dakle, skup izmena sadrži jedan zadatak kojim treba da se proširi postojeći radni tok, obeležen sa t_n. Pošto je definisan skup izmena, neophodno je definisati skup proširenja, po pravilima navedenim u ovoj sekciji. Skup P je dat sa:

$$P = \{p_{51}\}$$

Pošto je data jedna izmena, imamo jedno proširenje p_{51}, prvi indeks predstavlja mesto proširenja, dok drugi indeks predstavlja broj izmena kojim se proširuje, proširenje je prikazano na slici 3.13.

Željeno mesto proširenja p_{51}, obeleženo je sa x, na tranziciji tr_5. Prvi korak primene adaptacije je provera da li može da se izvrši proširenje po kriterijumima datim u ovoj sekciji. Ukoliko su svi kriterijumi zadovoljeni *NewWave* nastavlja sa adaptacijom, a



Slika 3.13: Proširivanje radnog toka

pošto je mesto proširenja na tranziciji tr_5, a zadatak *t3* još uvek nije aktiviran, moguće je nastaviti sa adaptacijom.

Drugi korak adaptacije je provera korišćenja vrednosti data-objekata koji su vezani za zadatke, jer neki zadaci mogu da zavise od data-objekata iz prethodnih zadataka. Kao što je pomenuto ranije, data-objekti su dostupni na nivou instance procesa, i na osnovu toga *NewWave* može da proveri njihovo korišćenje u zadacima. Međutim pošto ovo proširenje ne menja postojeće zadatke, nego samo dodaje novi i ova provera prolazi tako da se nastavlja dalje sa adaptacijom.

Treći korak je prekidanje tranzicije i dodavanje novog čvora, tj. izmene na mestu p_{51}. *NewWave* prekida tranziciju tr_5 i izvršava proceduru dodavanja čvora i kreiranja novih tranzicija. Odnosno izlazna grana *t3* čvora se povezuje sa t_n čvorom, odnosno biće njegova ulazna grana. Novokreiranu vezu između čvorova $t3$ i t_n nazvaćemo tranzicijom tr'_5. Pored pomenute, uspostavlja se i tranzicija između čvorova t_n i g_{12} koju nazivamo tr''_5. Novokreirani radni tok prikazan je na slici 3.14. Pošto je skup izmena I sadržao jednu izmenu, koja je uspešno primenjena, može se reći da su svi uslovi prethodno zadovoljeni i da je adaptacija uspešno primenjena.

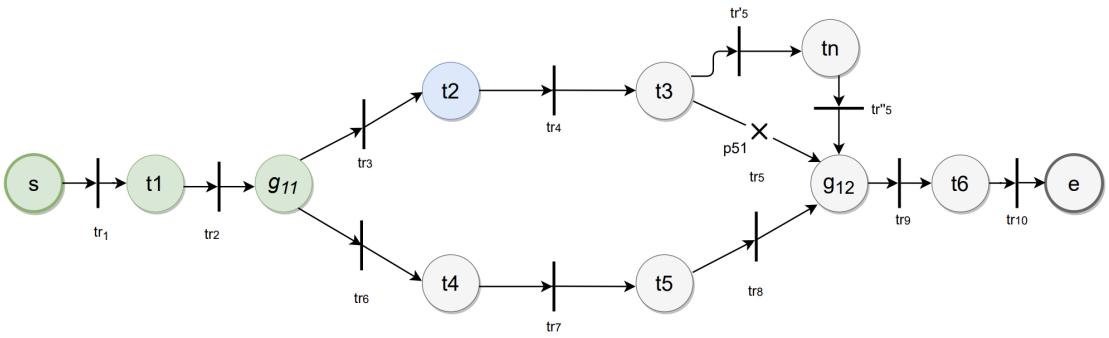
NewWave nakon završene adaptacije, vraća obaveštenje o uspešnosti primene željenog proširenja, nad datom instancom procesa. Postupak se ponavlja sve dok se postupak ne završi za sve postojeće instance datog radnog toka.

3.2.3 Primenjivanje adaptacije zamene

Na osnovu pravila definisanih u ovoj sekciji, definišemo skup zadataka, namenjen za izmenu koji je dat sa:

$$I = \{t2', t3'\}$$

Dakle, u ovom primeru tražene su dve zamene, koje bi menjale dve zadatka *t2 i t3*, što dalje implicira da nam trebaju dve zamene:



Slika 3.14: Radni tok nakon proširenja

$$e = \{e_{31}, e_{42}\}$$

Ako koristimo radni tok dat na slici 3.12, skup aktivacija će biti isti kao i u podsekciji koja opisuje proširenja, odnosno:

$$A = \{s, t1, g_{11}, t2\}$$

Novokreirani radni tok bi izgledao kao što je to prikazano na slici 3.15. Međutim, na osnovu pravila utvrđenih na početku ove sekcije, postupak zamene u ovom slučaju nije moguće izvršiti, jer je čvor *t2* aktiviran, a jedan od čvorova koji je naveden za zamenu je upravo taj čvor. *NewWave* zbog toga već u prvom koraku odustaje od postupaka adaptacije zamenama i vraća informaciju o neuspehu prilagođavanja instance koja je ilustrovana ovim primerom. Postupak zamene čvorova bi se odvijao po proceduri koja je opisana u podsekciji koja opisuje proširenja.

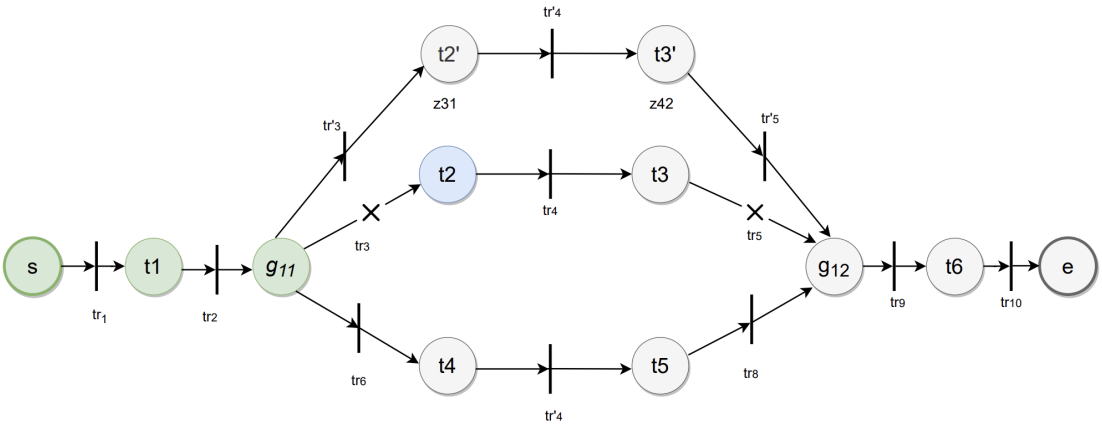
3.2.4 Primena zamene redosleda zadataka

Na osnovu pravila datih u ovoj sekciji, definišemo skup zadataka, namenjen za zamenu redosleda koji je dat sa:

$$I = \{t2, t3\}$$

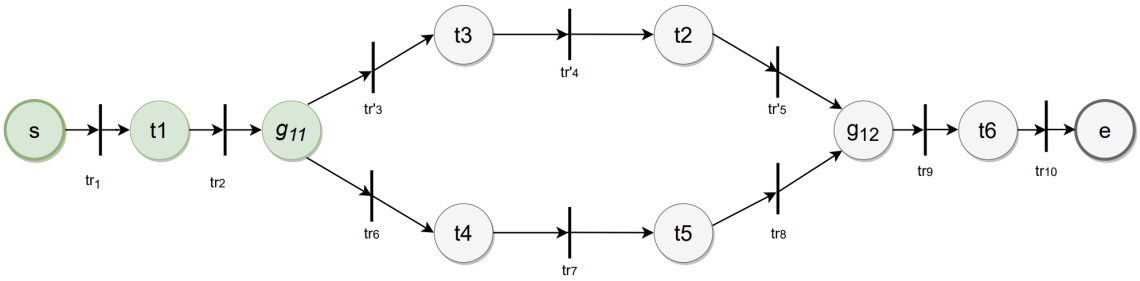
Koristimo radni tok dat na slici 3.12, pošto je ovo menjanje postojećih zadataka u okviru radnog toka, navedeni zadaci koje želimo da promenimo moraju postojati u radnom toku. Nakon toga definišemo željenu zamenu:

$$R = \{r_3\}$$



Slika 3.15: Željeni izgled radnog toka nakon zamene zadataka

Za menjanje redosleda naredna dva izvršavanja potreban je indeks mesta na kome se zamena vrši. Prateći pravila data za zamenu redosleda zadataka u ovoj sekciji, ako je *3* indeks zamene (koji odgovara tranziciji tr_3), odnosno zadaci koji se razmatraju za zamenu su *t2 i t3*. Željeni radni tok bi izgledao kao na slici 3.16.



Slika 3.16: Željeni izgled radnog toka nakon zamene redosleda zadataka

Neophodno je prevezati izlaznu granu čvora g_{11} sa ulaznom granom čvora *t3*, kao i izlaznu granu čvora *t3* na ulaznu granu čvora *t2* i na kraju izlaznu granu čvora *t2* na ulaznu granu čvora g_{12}. Što bi dovelo do novih tranzicija koje su nazvane tr'_3, tr'_4, tr'_5 respektivno.

Međutim, prateći pravila koja su data za zamenu redosleda, nije moguće izvršiti datu adaptaciju. Jer bi to značilo pomeranje čvora koji je već aktiviran, tako da bi *NewWave* ovde vratio informaciju da adaptacija nije izvršena.

3.2.5 Verifikacija i završetak adaptacija

Da bismo proverili korektnost adaptacija opisanih u prethodnom koraku, vršimo njihovu verifikaciju. Verifikacija dobijenih rezultata vrši se koristeći dve metode iz *Petrijevih* mreža, a to su *živost* (eng. *liveness*) i *dostupnost* (eng. *reachability*).

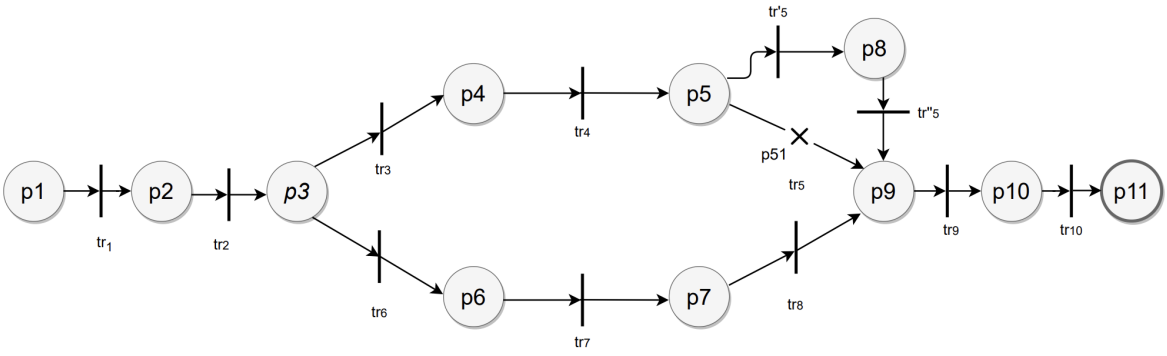
Prvo se razmatra postupak koji treba da pokaže da li je dobijena mreža živa, odnosno da li je izbegnuto tzv. *mrtvo zaključavanje* (eng. deadlock). Mrtvo zaključavanje u suštini označava da bez obzira do koje tačke se došlo u mreži, ne može se nastaviti njeno izvršavanje, odnosno tranzicija (ili skup tranzicija) koja ne može da se izvrši. Živost garantuje mrežu bez mrtvog zaključavanja, odnosno mogućnost daljih tranzicija sa tačke do koje je došla.

Petrijeva mreža može biti L0-živa, L1-živa, L2-živa, L3-živa ili L4-živa, odnosno koncept koji se odnosi na odustvo mrtvog zaključavanja (eng. *deadlock*) u radnom toku.U ovom radu u smislu poslovnih procesno-orijentisanih informacionih sistema, potrebno je da je svaka tranzicija u mreži minimalno L1-živa, odnosno da postoji neka sekvenca izvršavanja (okidanja) iz x_0, gde je x_0 inicijalno stanje, tako da tranzicija može da se izvrši (okine) bar jednom. Mreže opisane adaptacijama u prethodnim podsekcijama, nakon izvršenja adaptacija zadovoljavaju uslov da je svaka tranzicija u njima bar L1-živa, zato što se mogu okinuti barem jednom u nekoj sekvenci okidanja, tako da se zbog toga može reći da su te mreže žive.

Pored živosti mreže, razmatra se i dostupnost elemenata mreže, odnosno formira se stablo dostupnosti (eng. *reachability tree*) koje služi za ispitivanje dostupnosti mesta i mogućnosti izvršavanja tranzicija u *Petrijevoj* mreži. Odnosno za datu Petrijevu mrežu (N, M_0) sa inicijalnom oznakom M_0 može se dobiti onoliko novih oznaka koliko ima i omogućenih tranzicija, tako da iz svake nove oznake može doći do još oznaka, gde se ponavljanjem ovog postupka dobija struktura tipa stabla gde čvorovi predstavljaju oznake dobijene iz korena M_0 i njegovih naslednika, pri tome svaki luk predstavlja okidanja tranzicije iz jedne oznake ka drugoj [29]. Na osnovu navedenog formira se *reachability tree* za primer koji je dat u ovoj sekciji, konkretno za slučaj primenu proširivanja radnog toka, prikazan na slici 3.14.

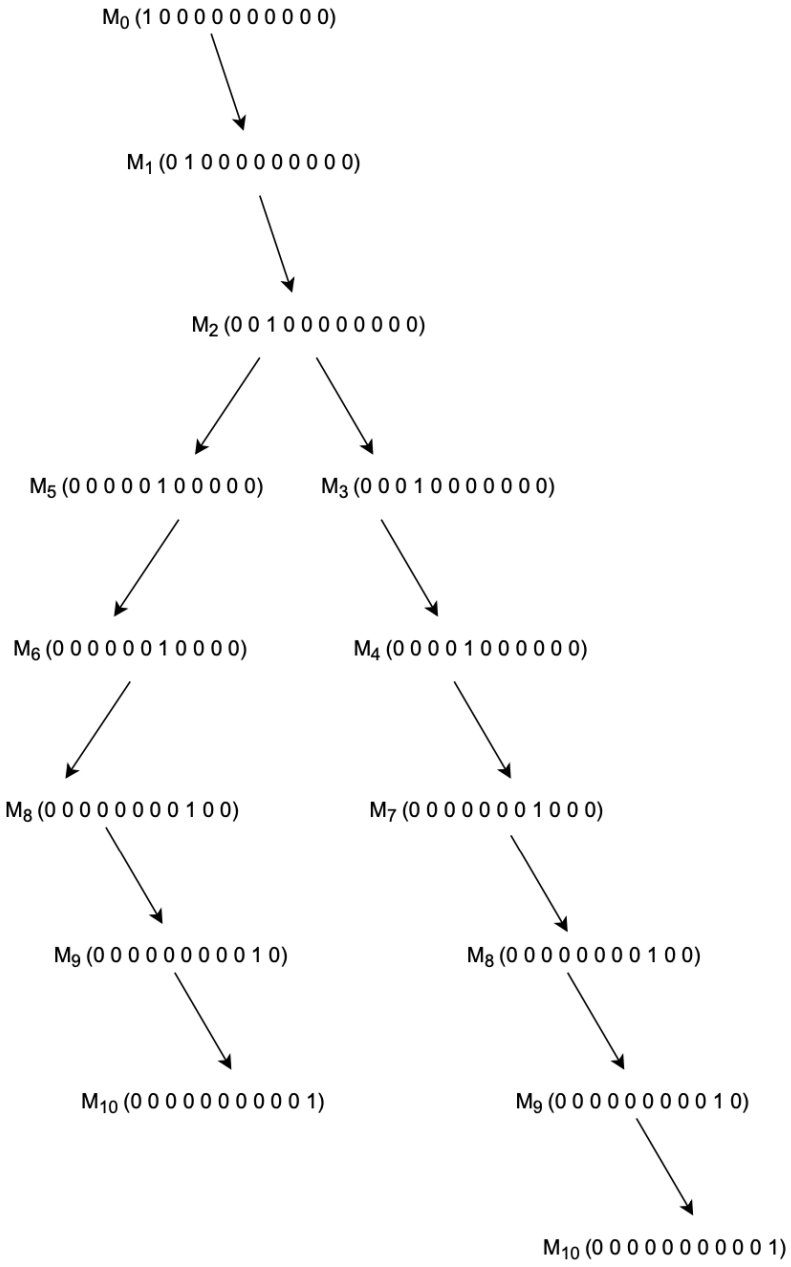
Inicijalna oznaka od koje se formira stablo je M_0. Iz inicijalne oznake moguće je dobiti onoliko "novih" oznaka, koliko ima i omogućenih tranzicija, konkretno 10 u datom primeru. Iz svake nove oznake može dostići još oznaka, ovaj postupak rezultuje reprezentacijom stabla datih oznaka. *Petrijeva* mreža na osnovu koje se formira stablo dostupnosti, prilagođena za lakše razumevanje prikazana je na slici 3.17, ova *Petrijeva* mreža se sastoji iz mesta p i tranzicija tr.

Stablo dostupnosti, kao što se vidi na slicii 3.18, je prilično jednostavno. Mreža za ovaj primer je ograničena, pošto sadrži sve moguće dostupne oznake, zbog toga kažemo i da je stablo ograničeno,odnosno da neće rasti beskonačno. Da je mreža neograničena i stablo bi bilo neograničeno, pa bi onda bilo formirano stablo pokrivenosti (eng. *coverability tree*), odnosno stablo prikazano na ograničen način, za neograničenu mrežu.



Slika 3.17: Izgled Petrijeve mreže sa mestima i tranzicijama

Nakon izvršenja svake od ovih adaptacija, *NewWave* ima tačne informacije nad koliko instanci procesa je uspeo, a nad koliko nije uspeo da izvršiti zadate promene. Te informacije se koriste za prikaz izveštaja korisniku. Ono što je bitno napomenuti, za sve instance procesa koje nisu adaptirane *NewWave* dozvoljava da se izvrše po verziji modela za koju su pokrenute. Svaka naredna instanca koja se pokrene, biva kreirana po novoj verziji modela koja je prethodno definisana.



Slika 3.18: Izgled stabla dostupnosti

Poglavlje 4

Integracija i jedna implementacija predložene arhitekture

Većina danas dostupnih platformi razvijana je u Javi, iako postoje i druge manje poznate koje su razvijane u drugim programskim jezicima. Kako Java predviđa kompajliranje, samim tim nije najpogodnija za probleme koji po prirodi traže interpretativni režim rada. Kako je adaptabilnost, koja se postiže stalnim ponovnim konsultovanjem promenljivog procesnog modela, jedna od najbitnijih ciljeva arhitekture, nametnulo se kao logičan izbor da se odabere jezik koji je dovoljno ekspresivan i pruža mogućnost da se tokom izvršavanja analiziraju i po potrebi menjaju model, ali i kod, bez dodatnog kompajliranja, odnosno rad u interpretativnom režimu. Iako se ova predložena arhitektura mogla implementirati u Javi, ili u nekim drugim interpretiranim programski jezicima, kao što su Javascript i Python, kao platforma za razvoj odabran je Pharo, zbog svoje inherentne osobine da je dinamički osmišljen, te da su radno i izvršno okruženje spojeni u jedno,što pojednostavljuje implementaciju komponenti za koje se očekuje promenljivost. NewWave platforma je zasnovana na Pharo okruženju, pa kao i mnogi projekti koji su nastali iz Pharo zajednice, tako i NewWave ima za cilj da u potpunosti iskoristi ovaj mali, ali moćan programski jezik i njegovo okruženje. Ovo poglavlje se fokusira na napredne elemente Pharo okruženja, koji su korišćeni da bi se kreirao NewWave. Nisu pokriveni svi aspekti Pharo jezika jer prevazilaze potrebe ove disertacije. Jedan posebno interesantan aspekat ovog jezika je mogućnost kreiranja alata, odnosno delova softverskih rešenja i radnih okvira koji rade na interpretaciono zasnovanom pristupu. Pharo okruženje je dinamičko sa neposrednom povratnom informacijom za sve što se desi u okviru sistema, što je iskorišćeno kao podstrek za realizaciju radnog okvira i ideja koje su implementirane kako bi se došlo do željenog rezultata ovog istraživanja.

4.1 Okruženje i objektni model

Pharo je moderan, objektno-orijentisan dinamički programski jezik, koji je nastao na osnovu Smalltalk-a [86]. Pharo teži ka jednostavnosti, tako da jezik kao takav ima jednostavnu sintaksu koju je u potpunosti moguće prikazati i na razglednici.

Pharo okruženje se sastoji iz četiri glavne komponente.

- *VM* (virtualna mašina) - predstavlja izvršno okruženje Pharo-a, odnosno preuzima Pharo bajt kod (*byte code*) koji se generiše svaki put kad se izvrši neka promena u sistemu, konvertuje ga na mašinski kod i zatim ga izvršava. VM je jedina komponenta koja se razlikuje za svaki operativni sistem.

- *Sources file* (datoteka sa izvornim kodom) - Datoteka koja sadrži izvorni kod za delova Pharo-a koji nisu podložni čestim izmenama. Ekstenzija ove datoteke je `.sources`.

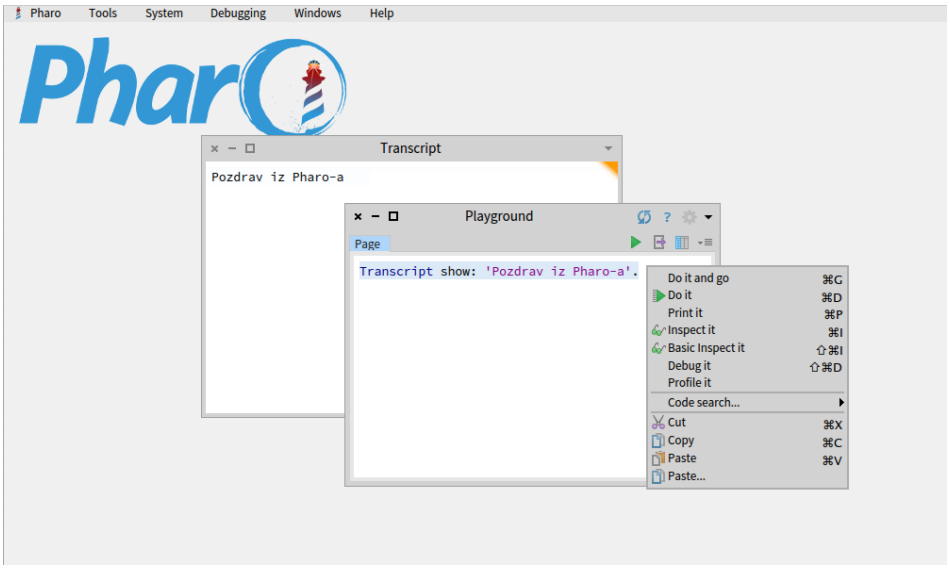
- *Changes file* (datoteka sa modifikovanim kodom) - Datoteka koja sadrži zabeleške svih modifikacija izvornog koda. Ova datoteka se može iskoristiti u slučaju "pucanja" tokom rada programa, da se oporave prethodno napravljene promene. Ova datoteka je uvek uparena sa datotekom slike (eng. *image*) i njena ekstenzija je `.changes`.

- *Image file* (datoteka koja predstavlja sliku) - Ova datoteka predstavlja zamrznutu sliku (*snapshot*) pokrenutog Pharo sistema. U datoteci se nalaze sačuvani svi objekti i njihovo stanje, kao i stanje sistema. Ova datoteka je portabilna, odnosno moguće je preneti na različite operativne sisteme.

Navedene datoteke zajedno čine Pharo okruženje, koje omogućavaju pokretanje, menjanje, prenošenje i izvršavanje Pharo-a. Nakon pokretanja, Pharo je spreman za izvršavanje i nije potrebno nikakvo dodatno podešavanje. Na slici 4.1 prikazano je Pharo okruženje, sa primerom izvršavanja izraza, koji ima zadatak prikaže pozdravnu poruku u Transcriptu (Objekat koji služi kao sistemska konzola za pravljenje beleški).

Objektni model Pharo-a je zasnovan na nizu pravila koja se primenjuju uniformno i sistematski bez izuzetka. Pravila su sledeća:

1. sve je objekat,
2. svaki objekat je instanca klase,
3. svaka klasa ima super-klasu (nad-klasu),
4. sve ključne operacije se realizuju slanjem poruka,
5. pretraživanje metoda prati lanac nasleđivanja,



Slika 4.1: Pharo okruženje

6. klase su takođe objekti i poštuju potpuno ista pravila [87].

   Neka od ovih pravila zahtevaju dodatna objašnjenja, te su posebno obrađena u nastavku ove sekcije.

4.1.1 Sve je objekat

U Pharo-u sve je objekat, pa čak i celobrojne vrednosti (*Integer*), koji su u drugim programskim jezicima, obično primitivni tipovi. Objekat `5` je drugačiji od objekta `25 factorial`, jer je `5` instanca klase *SmallInteger*, dok je `25 factorial` instanca klase *LargePositiveInteger*. Pošto su i jedan i drugi objekat polimorfni, odnosno znaju kako da odgovore na isti skup poruka, ni jedan ni drugi objekat to ne mora da zna.

Klase su objekti takođe i oni su objekti prvog reda, odnosno entiteti koji se mogu dinamički kreirati, uništiti, proslediti funkciji, mogu se vratiti kao vrednost, može im se poslati poruka i imaju sva prava koje imaju promenljive u Pharo-u, što dalje implicira da je Pharo zaista reflektivni programski jezik, što omogućava veliku izražajnu moć njegovim korisnicima da misle u domenu problema koji rešavaju, a ne da su ograničeni programskim jezikom koji koriste.

4.1.2 Svaki objekat je instanca klase

Moguće je proveriti kojoj klasi neka instanca pripada, tako što se objektu pošalje poruka `class`. Listing 4.1 prikazuje kojim klasama pripadaju objekti `5` i `25 factorial`.

```
1  5 class
2  >>> SmallInteger
3
4  25 factorial class
5  >>> LargePositiveInteger
```

Listing 4.1: Objekti i pripadajuće klase.

Klasa predstavlja entitet koji je deo sistema, takođe definiše strukturu svojih instanci putem promenljivih vrednosti instance - *instace variables (polja)*. Ponašanje instance se definiše putem njenih metoda, koje imaju jedinstven naziv u okviru klase.

Klase su takođe instance klase. Klase čije su instance klase, nazivamo meta-klasama. Kad korisnik kreira klasu, sistem za njega automatski pravi odgovarajuću meta-klasu. Prateći pravilo da klasa definiše strukturu svojih instanci i ponašanje putem metoda, tako i meta-klasa definiše ponašanje svojih klasa, koje su njene instance.

4.1.3 Svaka klasa ima super-klasu (nad-klasu)

Pharo prati princip jednostrukog nasleđivanja. Jednostruko nasleđivanje označava da svaka klasa u Pharo-u nasleđuje svoje ponašanje od nad-klase. Koren hijerarhije nasleđivanja u Pharo-u je klasa *ProtoObject*, za razliku od većine programskih jezika gde je klasa u u korenu hijerarhije *Object*. *ProtoObject* sadrži minimalni skup poruka na koji objekti moraju da odgovore. Međutim *Object* je taj koji je namenjen za nasleđivanje, jer definiše dodatni skup poruka koje ostali objekti razumeju i na koje znaju da odgovore.

Pharo omogućava lako proveravanje nad-klase neke klase slanjem poruke `superclass` datoj klasi. Nad-klase za *SmallInteger, LargePositiveInteger i Object* su date na listingu 4.2.

```
1  SmallInteger superclass.
2  >>> Integer
3
4  LargePositiveInteger superclass.
5  >>> LargeInteger
6
7  Object superclass.
8  >>> ProtoObject
```

Listing 4.2: Objekti i pripadajuće nad-klase.

4.1.4 Sve se dešava slanjem poruka

Slanje poruka predstavlja jednu od bitnijih razlika Pharo-a u odnosu na tradicionalne programske jezike kao što su Java, C#, C++. Odnosno pozivanje određene metode

se vrši na osnovu njenog imena, gde pozivalac statički na osnovu imena, bira koju će metodu izvršiti, gde u tom slučaju nema nikakve dinamičnosti i pretraživanja metoda. Pharo ne koristi koncept pozivanja metoda, nego slanja poruka. Slanje poruka označava da je odgovornost na primaocu poruke da odabere metodu za izvršavanje, umesto pozivaoca. Ovo za posledicu ima to da možemo istu poruku poslati ka više primaoca, međutim da je odgovornost na njima šta će i na koji način će da odgovore.

Prethodni primeri dati u listinzima 4.1 i 4.2 ilustruju slanje poruka. Listing 4.1 prikazuje slanje poruke `class` objektu 5, dok listing 4.2 prikazuje slanje poruke `superclass` klasi *SmallInteger*. Pošto su i klase objekti, kao što je navedeno i poštuju potpuno ista pravila, tako je i njima moguće poslati poruku.

4.1.5 Pretraživanje metoda prati lanac nasleđivanja

Recimo da objektu `5` pošaljemo poruku `asString`. Metoda `asString` se ne nalazi u klasi *SmallInteger*, tako da ona nije odmah pronađena. Pretraga se nastavlja u njenoj nad-klasi i tako dalje sve dok se ne pronađe. Pošto se metoda `asString` nalazi u klasi *Object* pretraga se tu zaustavlja, kao što je to ilustrovano na slici 4.2.

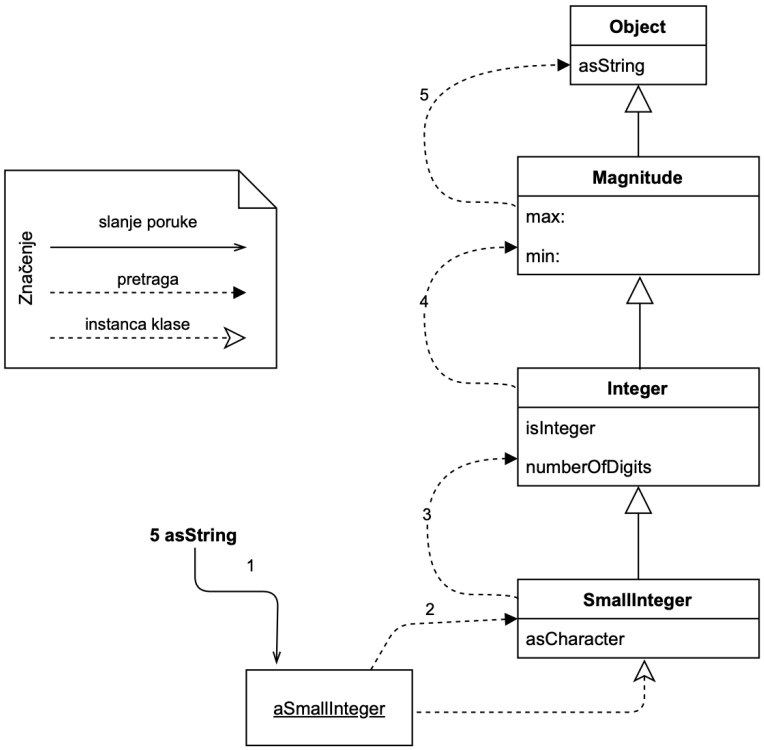
4.2 Kreiranje klasa, objekata i njihova inspekcija

Kreiranje klase se vrši u pregledaču u Pharo okruženju, odnosno u okviru alata koji se zove *Calypso*, koji služi i za navigaciju kroz Pharo okruženje. Kreiranje klase se svodi na slanje poruke klasi Object koja ima zadatak da na osnovu primljenih vrednosti kreira novu klasu, kao što je to prikazano na slici 4.3.

Na slici 4.3, prikazane su i dve promenljive vrednosti koje pripadaju instanci, odnosno varijable instance (polja), a to su `ulica i broj`. Polja u Pharo-u su privatna samo za samu instancu, što znači da im se može pristupiti iz samo te instance. Pristup poljima iz drugih instance moguć je samo indirektno preko metoda. Sve metode u Pharo-u javne i virtualne, odnosno pretražuju se dinamički putem lanca nasleđivanja.

Kreiranje instance klase *Adresa* se postiže slanjem poruke `new` klasi *Adresa*, odnosno `adr := Adresa new`, gde je `adr` privremena promenljiva koja sadrži instancu novo kreirane adrese, a operator dodele je predstavljen se `:=`.

Kao posledicu toga što je i klasa objekat, Pharo ima metode i polja koja pripadaju klasi, odnosno *class side* metode i polja. U prethodnom pasusu su pominjani polja i metode koja pripadaju instanci, odnosno nalaze se na strani instance. Metode i polja koje se nalaze na strani klase mogu biti veoma korisni, jer se tu može naći programska logika i vrednosti koji nisu vezani ni za jednu konkretnu instancu, a takođe mogu i da odgovore na poruke koje im se pošalju, mehanizam koji ne postoji u Javi, C#, C++ i sličnim programskim jezicima. Metode za inicijalizaciju instance se često pišu sa strane klase, jer na jednostavan način omogućavaju slanje poruka koje imaju sve potrebne



Slika 4.2: Traženje metode prati lanac nasleđivanja

```
1  Object subclass: #Adresa
2      instanceVariableNames: 'ulica broj'
3      classVariableNames: ''
4      package: 'primer'
```

Slika 4.3: Primer kreiranja klase Adresa

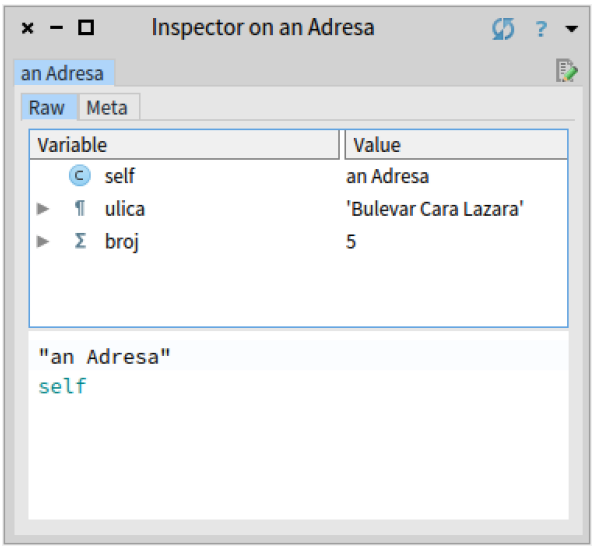
vrednosti da se kreira instanca. Primer inicijalizacije klase *Adresa* sa strane klase dat je na listingu 4.3.

```
1  ulica: novaUlica broj: noviBroj
2      ^ self new
3          ulica: novaUlica;
4          broj: noviBroj;
5          yourself
```

Listing 4.3: Inicijalizacija slanjem poruke metodi koja se nalazi na strani klase.

Nakon izvršene izmene, ulicu je moguće kreirati na mnogo jednostavniji i čitljiviji način, slanjem poruke njenoj metodi koja se nalazi na strani klase, (npr. `Adresa ulica: 'Bulevar Cara Lazara' broj: 5`).

Pošto je Pharo živo programsko okruženje, postojeću *Adresu* je moguće menjati tokom rada, kao i pratiti vrednosti njenih polja. Izmena se može izvršiti putem alata koji se zove *Inspector*, koji nam omogućava da putem grafičkog korisničkog interfejsa pristupimo svim elementima objekta, kao što je prikazano na slici 4.4.



Slika 4.4: Inspekcija instance objekta jedne adrese

Izmena vrednosti polja adrese moguće je uraditi putem inspektora, nakon čije intervencije vrednosti objekta su izmenjeni u celom sistemu. Inspektor je jedan od korisnijih alata u okviru Pharo okruženja, koga je takođe po potrebi moguće proširiti i prilagoditi specifičnim korisničkim potrebama i predstavlja direktan prozor u Pharo okruženje dozvoljavanjem pristupa svakom objektu koji u okviru ovog sistema postoji.

4.2.1 Integracija okruženja sa predloženim rešenjem

Pharo okruženje poseduje ekosistem koji ima veliki broj biblioteka, a koje su prilikom razmatranja odgovarale potrebama neophodnim za realizaciju određenih delova ovog projekta. Kontrolisanje svakog aspekta ovog rešenja kroz jedinstven interfejs Pharo-a, doprinosi tome da nam nije neophodan set različitih alata koji često nisu kompatibilni, time ograničavajući mogućnosti kreiranog softverskog rešenja. Naprotiv, korišćenjem jednog jedinstvenog sistema je moguće pomeriti granice koje tradicionalni *workflow engine*-i pružaju, što se i razmatra u okviru ove disertacije.

Neophodnu adaptabilnost procesa je tako moguće obezbediti koristeći inherentna svojstva samog okruženja, jer su instance procesa takođe predstavljene kao Pharo objekti, te je moguće obezbediti povratnu informaciju o svakoj modifikaciji (manipulaciji) iz bilo koje instance dobijene na osnovu modela pojedinog procesa (ako je takva izmena svrsishodna, kada korisnik ima takva prava i pod definisanim uslovima). Ovo podrazumeva da se na nivou workflow engine-a razreše problemi propagacije izmena. Još jedna prednost korišćenja jedinstvenog ekosistema jeste mogućnost definisanja entiteta direktno u Pharo okruženju, bez potrebe za korišćenjem drugih alata i problemima interoperabilnosti koji se pojavljuju. Tradicionalno, workflow engine-i podržavaju ugrađivanje objekata u radni tok, međutim to nije nativna podrška, nego se obično vrši kreiranjem nekog skript jezika koji konkretan engine podržava.

Pošto su instance procesa takođe predstavljene kao Pharo objekti, moguće je tokom izvršavanja vršiti manipulaciju i nad elementima procesa - čime se dobija mogućnost menjanja same specifikacije procesa tokom njegovog izvršavanja. NewWave je kreiran tako da iskoristi prethodno opisanu dinamičnost Pharo okruženja. Na taj način operacije koje je neophodno obaviti prilikom dinamičke adaptacije procesa su znatno pojednostavljene, i samo usvajanje izmene u pokrenuti proces brže nego kod klasičnih rešenja koja podrazumevaju izmenu modela i njegovo ponovno prevođenje u izvršni proces.

Zbog različitih konfiguracija koje NewWave poseduje, u zavisnosti od potreba, objekti kojima su opisani podaci koji učestvuju u radnom toku se mogu transformisati u različite formate, kao što su JSON (JavaScript Object Notation) [88] i STON (Smalltalk Object Notation) [89]. Transformisani objekti putem HTTP-a (HyperText Transfer Protocol) [90] mogu se proslediti na računarski sistem koji takođe izvršava NewWave ili klijentsku veb aplikaciju i tamo se ponovo rekreirati u Pharo objekte. Rekreiranje se vrši zbog potreba jednostavnije manipulacije i upravljanja objektima. Naravno, STON format je pogodniji ukoliko udaljeni računarski sistem takođe koristi Pharo, međutim ostavljena je mogućnost da se za potrebe klijentske aplikacije koristi i JSON, te se onda sa klijentske strane može naći bilo koja savremena aplikacija koja može da interpretira ovu notaciju.

4.3 Jedna implementacija predloženog rešenja

Arhitektura opisana u prethodnom poglavlju predstavlja specifikaciju koju treba konkretizovati implementacionim delom. Za implementaciju ovog rešenja razmatrane su mnoge opcije (većina prethodno pomenutih alata i rešenja je realizovana uz oslonac na Javu), međutim za implementaciju ovog rešenja odabran je Pharo. Pharo pripada grupi dinamičkih jezika, odnosno jezicima koji tokom svog rada (eng. *runtime*) mogu da izvrše razne operacije koje statički programski jezici izvršavaju tokom kompajliranja. Pharo je odabran zbog svojstva živog okruženja i trenutnih povratnih informacija koje pruža, odnosno ne postoji razlika između vremena kompajliranja i vremena izvršavanja, kao što je to opisano ranije. Ovo svojstvo prirodno odgovara i podržava koncept adaptivnosti, odnosno samih izmena u okruženju tokom njegovog rada, odnosno izvršavanja poslovnih procesa. U nastavku ovog poglavlja opisani su neki bitni koncepti iskorišteni za implementaciju ovog rešenja. Naknadno je opisana referentna implementacija određenih elemenata sistema.

4.3.1 Konfiguracija

Pomenuto je da je jedan od ciljeva da predloženo rešenje bude fleksibilno i proširivo, pomenutim konceptima takođe doprinose i mogućnosti alata da radi u različitim konfiguracijama. NewWave koristi dodatke (eng. *plugin*) da isporuči različite konfiguracije projekta. U zavisnosti od potreba, moguće je učitati samo *WaveEngine*, odnosno njegov *Core* deo, dok je npr. u drugoj konfiguraciji potrebno učitati veb modul za izvršavanje određenih zadataka kroz veb aplikaciju i isporučivanje zadataka putem ugrađenog servera u okviru radnog toka i slično. Korišćenje i učitavanje različitih konfiguracija projekata omogućavaju takozvane osnovne linije (eng. *Baseline*). *Baseline* predstavlja skelet ili arhitekturu projekta u smislu strukturalnih zavisnosti između paketa i projekata [91]. Dakle, *Baseline* definiše pakete jednog projekta, njihove međusobne zavisnosti i zavisnosti ka eksternim projektima, kao i pod-grupe koje mogu biti učitane.

Upravljanje konfiguracijom softvera (eng. *Software Configuration Management* - SCM) je jedna od disciplina iz osamdesetih godina prošlog veka, koja je nastala kao odgovor na mnoge neuspehe softverske industrije iz sedamdestih godina prošlog veka [92], a SCM predstavlja disciplinu koja omogućava kontrolisano praćenje i evoluciju softverskog proizvoda [93].

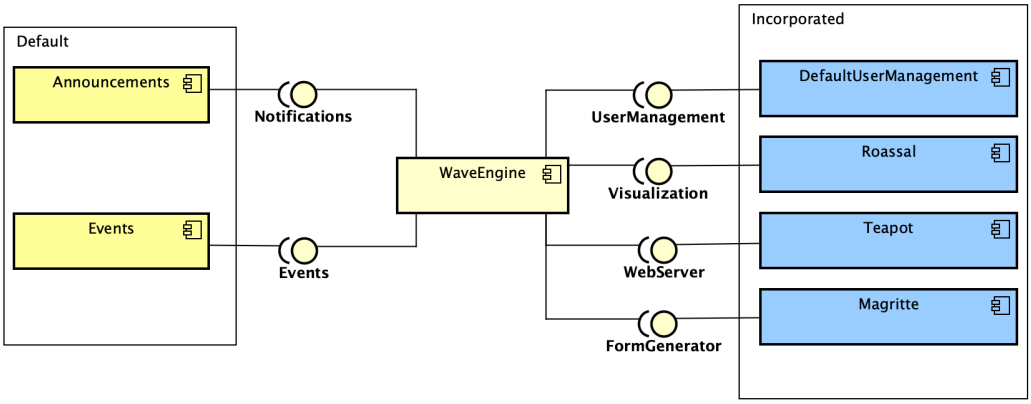
NewWave je rešenje koje je nastalo sa potencijalnim problemima koje nose nepravilne konfiguracije softvera na umu, te je od samog početka upravljanje konfiguracijom uvršteno u razvojni proces ovog rešenja. NewWave podržava više vrsta konfiguracija, u disertaciji navedeno je nekoliko najbitnijih grupa konfiguracije prikazane na listingu 4.4. Uočava se *Core* grupa koja je opisana u sekciji 3.1, pomenuto je da *Core* može funkcionisati nezavisno, zbog čega je omogućeno njegovo samostalno

učitavanje. Drugu grupu predstavlja veb aplikacija, koja je zadužena za preuzimanje zadataka, generisanje formi, kao i slanje odgovora nazad ka serveru. *TeapotServer* grupa sadrži *Core* deo i *Teapot* server koji je u toj konfiguraciji prilagođen za isporučivanje zadataka preko svojih pristupnih tačaka, kao i da prima odgovore koje mu se proslede. Zbog ideje da NewWave ima mogućnost da komunicira sa drugim alatima, podaci koje isporučuje i prima njegova server komponenta prenose se u JSON formatu, o čemu će biti više reči u sledećem poglavlju. *Visualization* grupa služi da omogući vizualizaciju različitih radnih tokova, kao i interakciju putem prikazanih elemenata.

```
1  spec group: 'Core' with: #(NewWave).
2  spec group: 'WebTask' with: #('NewWaveWebApp').
3  spec group: 'TeapotServer' with: #(Core Teapot).
4  spec group: 'Visualization' with: #(Core 'NewWave-Roassal').
```

Listing 4.4: Osnovne grupe za konfiguracije.

Ukoliko se prisetimo slike komponenti 3.2 koja je data u specfikaciji, slika 4.5 predstavlja konkretizaciju u ovom primeru implementacije. Prikazane komponente su:

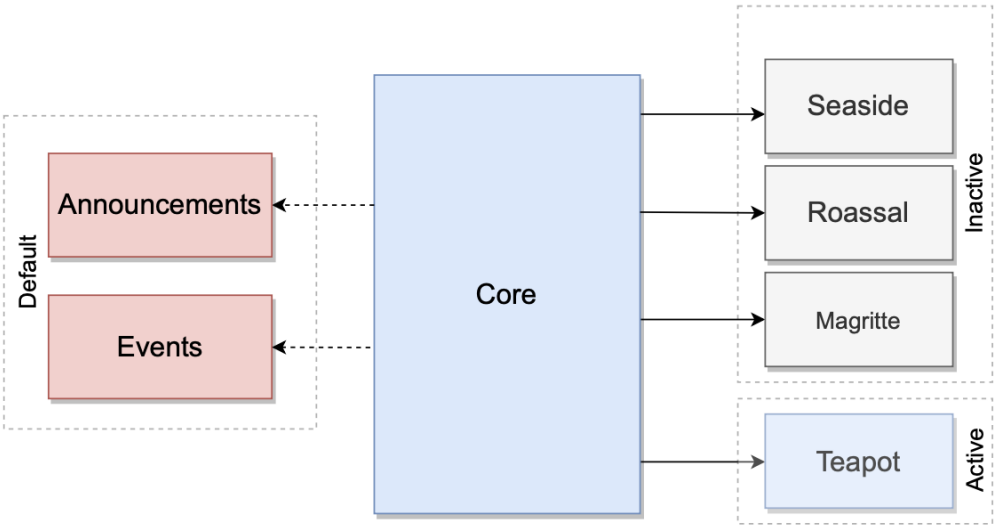


Slika 4.5: Implementacija komponenti u okviru WaveEngine dela

- DefaultUserManagement - Komponenta koja pruža podrazumevanu korisničku konfiguraciju i rukovanje korisnicima i korisničkim ulogama.
- Roassal - Biblioteka koja omogućava vizualizaciju i inspekciju radnih tokova.
- Teapot - Web server za isporučivanje zadataka i preuzimanje odgovora putem HTTP-a.

- Magritte - Za direktnu konverziju data objekata u forme korisničkog interfejsa.

Ukoliko je učitana samo *TeapotServer* grupa, slika 4.6 opisuje stanje NewWave-a u toj konfiguraciji. Kao što je prikazano na slici, aktivan je *Core* deo, sa njegovim podrazumevanim priključcima i aktivan je *Teapot*, dok *Seaside, Roassal, Magritte* nisu aktivni. U zavisnosti od potrebe, određeni elementi mogu se proizvoljno aktivirati.



Slika 4.6: Konfiguracija zavisnosti u NewWave Core delu

Nijedno ozbiljno rešenje se danas koje želi i ima potencijal da se koristi u industriji ne može se razmatrati bez pravilno podešene podrške za konfiguracije, NewWave već nalazi primenu u industriji (o čemu će biti više reči kasnije) te je njegov konfiguracioni sistem kreiran i na taj način, da je ostao otvoren sa podrškom za potencijalne različite konfiguracije koje se mogu ukazati tokom njegove eksploatacije.

4.3.2 Vizualizacija i inspekcija elemenata

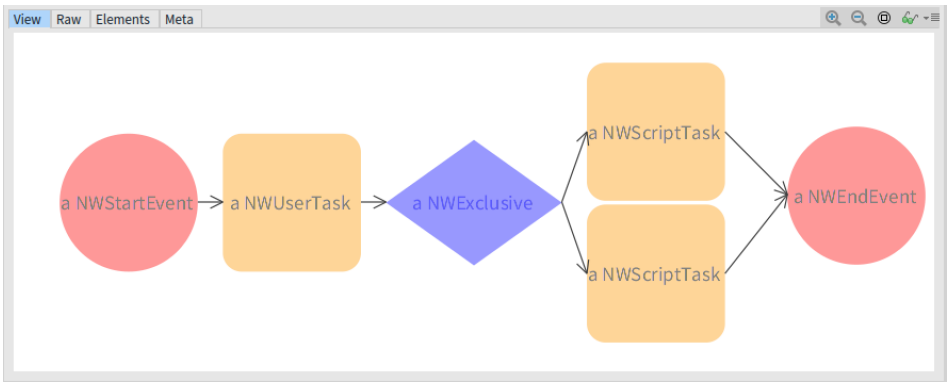
Vizualizacija je bitna stavka u okviru NewWave engine-a. Pošto je moguće menjati elemente i radne tokove tokom rada NewWave-a, vizualizacija olakšava pronalazak i identifikaciju željenih elemenata. Kreiranje jednog jednostavnog radnog toka, koji služi kao primer za ilustraciju vizualizacije dat je na listingu 4.5.

Dodatak za vizualizaciju je duboko integrisan u okviru NewWave-a, te zbog toga da bi vizualizacija radila potrebno mu je proslediti samo listu čvorova, na osnovu kojih

```
1  se := NWStartEvent new.
2  se description: 'StartEvent'.
3
4  t1 := NWUserTask new.
5  t1 description: 'Do you want to send a notification'.
6
7  t2 := NWScriptTask new.
8  t2 description: 'Sending notification...'.
9
10 t3 := NWScriptTask new.
11 t3 description: 'Abandoning...'.
12
13 ee := NWEndEvent new.
14 ee description: 'EndEvent'.
15
16 split1 := NWExclusive new.
17 split1 description: 'Split1'.
18
19 se addOutgoingEdge: t1.
20 t1 addOutgoingEdge: split1.
21
22 split1 addOutgoingEdge: t2 withCondition: [ :x | x = true ].
23 split1 addOutgoingEdge: t3 withCondition: [ :x | x = false ].
24
25 t2 addOutgoingEdge: ee.
26 t3 addOutgoingEdge: ee.
```

Listing 4.5: Primer radnog toka.

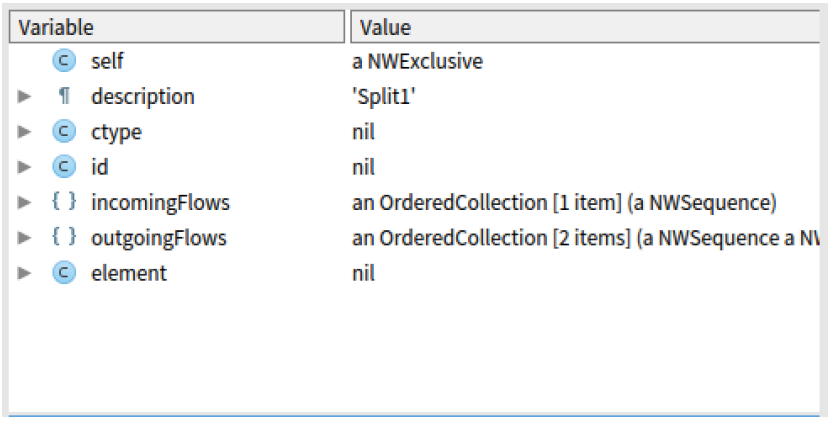
zna da prikaže radni tok. Vizualizacija radnog toka na osnovu listinga 4.5 prikazana je na slici 4.7.



Slika 4.7: Vizualizacija radnog toka

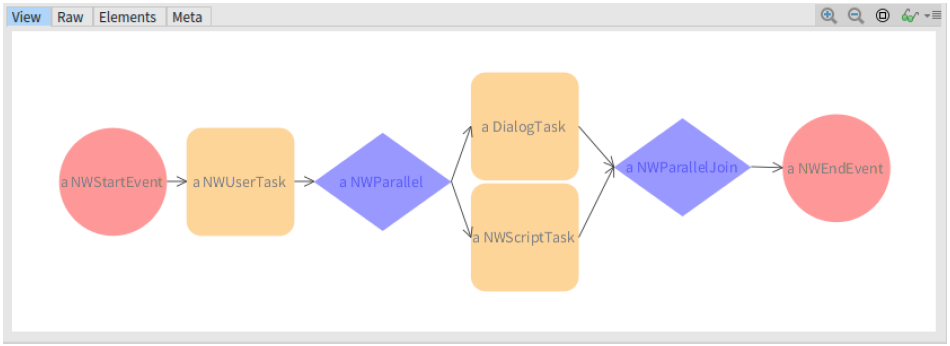
Svaki element koji je prikazan na slici je interaktivan, odnosno povezan je sa pravim objektom na osnovu koga je i kreiran. Zahvaljujući toj vezi, možemo selektovati svakog od njih i dobiti dodatne informacije. Ukoliko je selektovan XOR-split (*a NWExclusive* na slici), sve njegove osobine se mogu pogledati na dijalogu koji je prikazan na slici 4.8.

Na slici se može videti da prikazani XOR-split ima jednu ulaznu granu, dve izlazne, još uvek nema dodeljen id pošto ovaj radni tok nije dodat u proces. Bitno je napomenuti da se sve izmene mogu vršiti i kroz prikazani dijalog i da će biti primenjene svuda.



Slika 4.8: Informacije o XOR-splitu

Na osnovu pomenutog, moguće je kreirati bilo koji radni tok koji koristi elemente koje nudi NewWave, a vizualizator će znati to da prikaže. Ilustracije radi, na slici prikazan je primer koji koristi AND-split i join, odnosno *ParallelSplit i ParallelJoin* jer se ta vrsta primera vizualno znatno razlikuje od XOR-splita. Odnosno vizualizator treba da zna da usmeri strelice nakon splita ka join-u. Sva ostala pravila važe kao i u prethodnom primeru, odnosno moguće je pristupiti, ispitati i modifikovati sve elemente koji su prikazani.



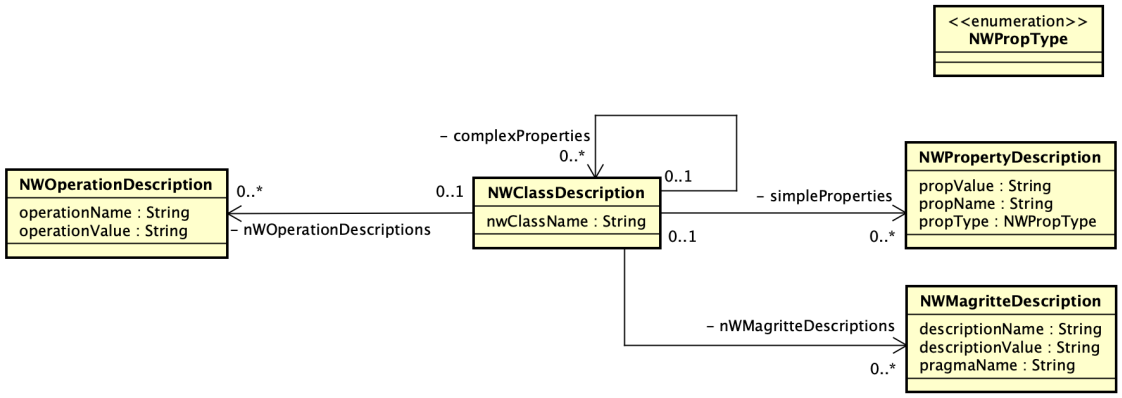
Slika 4.9: Primer Parallel split

4.3.3 Generator objekata

Korišćenje zadataka koji su definisani u okviru alata je jedna od velikih prednosti koje ima NewWave. Zadaci koje NewWave koristi su nativni za okruženje, odnosno predstavljaju stvarne objekte iz Pharo okruženja. Međutim, kad se ti zadaci kao data-objekti prenose u web aplikaciju koja se izvršava na udaljenom Pharo okruženju, ili nekom drugom okruženju koje nije realizovano Pharo tehnologijom (jedna od ideja

jeste da se komunikacija sa samim engine-om bude nezavisna) potrebno je te zadatke isporučiti preko servera koji NewWave poseduje.

Međutim, ideja je da se ne izgubi na prednostima dobijenim korišćenjem nativnih Pharo objekata. Da bi se to postiglo, napravljen je još jedan dodatak koji omogućava dekompoziciju Pharo objekata u oblik koji je pogodan za slanje preko mreže i njihovu rekonstrukciju u prvobitno stanje. Na slici 4.10 prikazan je meta-model koji opisuje data-objekte koji su pridruženi zadacima i koji učestvuju u radnim tokovima.



Slika 4.10: Meta-model

Metamodel opisuje kako treba da izgleda jedan entitet koji opisuje data-objekat u okviru NewWave-a, odnosno njegovog *WaveEngine* dela koji je zadužen za rukovanje tim tipovima objekata. *NWClassDescription* predstavlja entitet koji ima svoj naziv, odnosno *nwClassName*. Svaki entitet može da ima proizvoljno mnogo svojstava koji ga čine i koji su opisani sa opisima - *NWPropertyDescription*. *NWPropertyDescription* predstavlja jednostavna svojstva nazvana *simpleProperties* - koja imaju naziv, vrednost i tip, odnosno *propName, propValue i propType* respektivno. Pri čemu je *propType* enumeracija koja može imate vrednosti kao što su *ByteString, SmallInteger, Date, False, True...*, odnosno vrednosti koje odgovaraju tipovima koji su opisani klasama u Pharo okruženju.

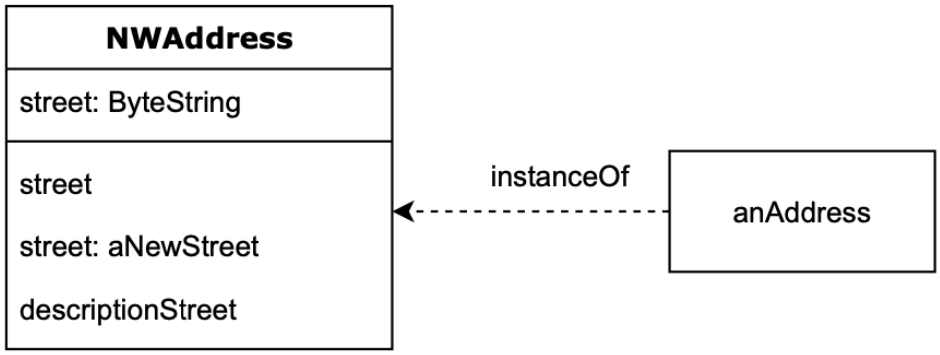
Pored jednostavnih, postoje i kompleksna svojstva nazvana *complexProperties*. Kompleksna svojstva su u suštini entiteti koji su opisani klasom *NWClassDescription* zbog čeka je i prikazana rekurzivna veza na slici 4.10. Odnosno jedan entitet može sadržati više drugih entiteta.

Pored svojstava, opisane su i operacije *NWOperationDescription* i specijalne vrste operacije koje su date sa *NWMagritteDescription*. *NWMagritteDescription* predstavljaju operacije koje su povezane sa *Magritte* radnim okvirom za meta-modelovanje

i služe za automatizaciju postupka generisanja formi. Svaka specijalna operacija ima naziv, vrednost i naziv pragme, odnosno *descriptionName, descriptionValue i pragmaName* respektivno. Naziv (*descriptionName*) služi za davanje naziva operaciji, *descriptionValue* opisuje šta operacija treba da radi, dok *pragmaName* predstavlja naziv pragme, koja se ugrađuje ispod naziva operacije, kako bi signalizirala Magritte radnom okviru da treba da je uključi u postupak generisanja formi.

Na osnovu datog meta-modela, generišu se odgovarajući modeli, koji zavise od radnog toka odnosno od njegovih objekata koji sadrže informacije neophodne za njegovo izvršavanje (data-objekti). Generator kreiran u okviru ovog rešenja, može pored generisanja modela, takođe da reverzno generiše model iz postojećeg data-objekta. Na slici 4.11 prikazan je model koji služi za generisanje data-objekata koji učestvuju u radnom toku.

Generator, kao što je već pomenuto u okviru ove sekcije radi "dvosmerno". Prilikom izvršavanja radnog toka, u postupku spremanja zadataka za slanje veb aplikaciji, vrši dekompoziciju objekata i priprema ih u oblik pogodan za slanje ka veb aplikaciji. Takođe, ukoliko se koristi integrisana veb aplikacija, prosleđeni odgovor od servera koji se dobija u okviru veb aplikacije može se iskoristiti za kompoziciju, odnosno kreiranje objekata kojim će se rukovati direktno u veb aplikaciji (koristi se samo u slučaju integrisane veb aplikacije). Pored navedenog, postoji još jedan scenario, gde se generator može iskoristiti prilikom dobijanja odgovara od veb aplikacije, na serverskoj strani, ukoliko je potrebno izvršiti kreiranje novog objekta.



Slika 4.11: NWAddress model

NWAddress opisuje jednu jednostavnu adresu. Adresa ima svojstvo *street* koje je tipa *ByteString* i tri operacije. Prve dve operacije služe za dobijanje informacija o svojstvu, odnosno *street* i menjanje naziva svojstva, odnosno *street: aNewStreet*, dok je

descriptionStreet specijalna operacija koja služi Magritte radnom okviru za generisanje forme i elemenata forme na osnovu datih svojstava.

4.3.4 Veb aplikacija

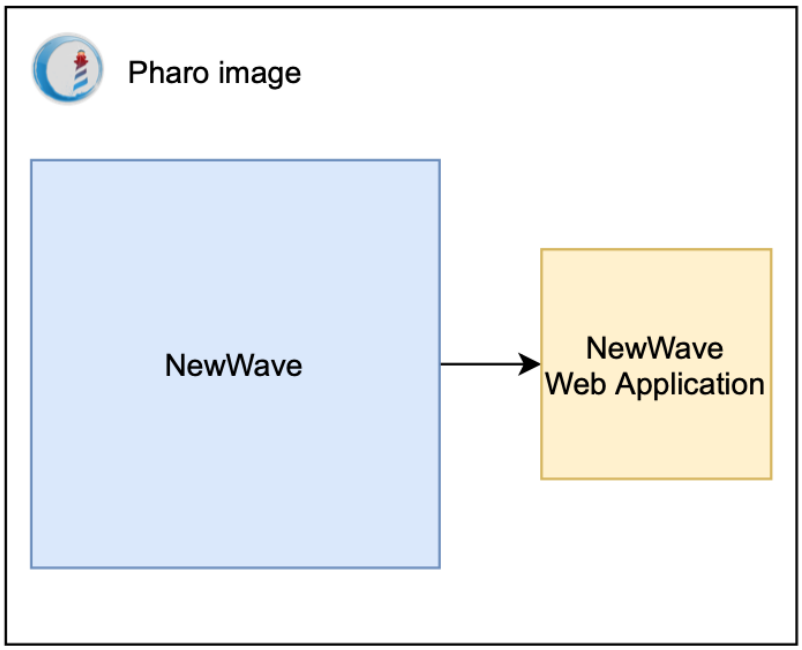
Veb aplikacija koja je kreirana u okviru NewWave-a je celina za sebe. Kreirana je tako, da je njeno funkcionisanje moguće na više načina, odnosno kao zavisnog dodatka u okviru aplikacije, međutim moguće je i njeno korišćenje kao nezavisne celine koja se izvršava u sklopu nekog drugog Pharo okruženja, te je zato i izostavljena sa slike 3.2. Veb aplikacija kreirana je sa ciljem da omogući korisnicima upravljanje radnim tokovima putem veb interfejsa - odnosno da bude dostupna koristeći bilo koji internet pregledač (eng. *browser*). Pošto je odlučeno da ova aplikacija može da se izvršava nezavisno, bilo da se aplikacija nalazi u istom Pharo okruženju kao i NewWave, ili u nezavisnom Pharo okruženju, ova odluka je dodatno uticala na povećanje kompleksnosti i celokupno rešenje ove aplikacije. Jedna od ideja prilikom kreiranja NewWave-a, bila je da se kreira samostalna (*standalone*) aplikacija koja će komunicirati sa NewWave-om preko mreže. To je urađeno da bi veb aplikacija bila nezavisna od ostatka rešenja, odnosno da ne bi postojala sprega ka ostatku sistema koja bi mogla da ima neželjen uticaj na dinamičke karakteristike koje su koncipirane za kreiranje ovog projekta.

Pored toga, benefiti samostalne aplikacije za pristup poslovnim procesima putem interneta su višestruki. Sa jedne strane, ponuđeno rešenje koje dolazi u okviru celog projekta, nije obavezno za komunikaciju sa glavnim delom sistema koji je zadužen za izvršavanje poslovnih procesa. Sa druge strane, u zavisnosti od potreba korisnika aplikacije, moguće je razviti i integrisati novu veb aplikaciju, sa proizvoljnim korisničkim izgledom (eng. *User Interface*) i specifičnim korisničkim iskustvom (eng. *User Experience*) koji mogu biti prilagođeni potrebama i zahtevima korisnika ili kompanije za koju se kreiraju, gde bi komunikacioni sloj za razmenu podatka sa ostatkom rešenja, u najvećem broju slučajeva ostao isti.

Ukoliko se veb aplikacija izvršava lokalno, odnosno u istom Pharo okruženju, na kome se nalazi i ostatak NewWave projekta, onda je ona zavisna od ostatka okruženja i rukovanje podacima je trivijalno jer se nalaze u okviru istog sistema. Ovaj režim izvršavanja ilustruje slika 4.12.

Korišćenje veb aplikacije na udaljenom (eng. *remote*) sistemu je znatno komplikovanije i za implementaciju i za konfiguraciju. U tom režimu, veb aplikacija treba da se izvršava samostalno u okviru Pharo okruženja, odnosno ostatak rešenja se izvršava u drugom Pharo okruženju, te je jedini način za komunikaciju ovako razdvojenih delova aplikacije putem interneta, odnosno korišćenjem HTTP protokola, gde bi veb aplikacija koja se izvršava samostalno u posebnom Pharo okruženju, komunicirala sa serverskim delom NewWave rešenja, koji se pak izvršava u drugom Pharo okruženju, gde slika 4.13 ilustruje opisani scenario.

Ugrađena veb aplikacija je zasnovana na *Seaside* radnom okviru. *Seaside* pruža



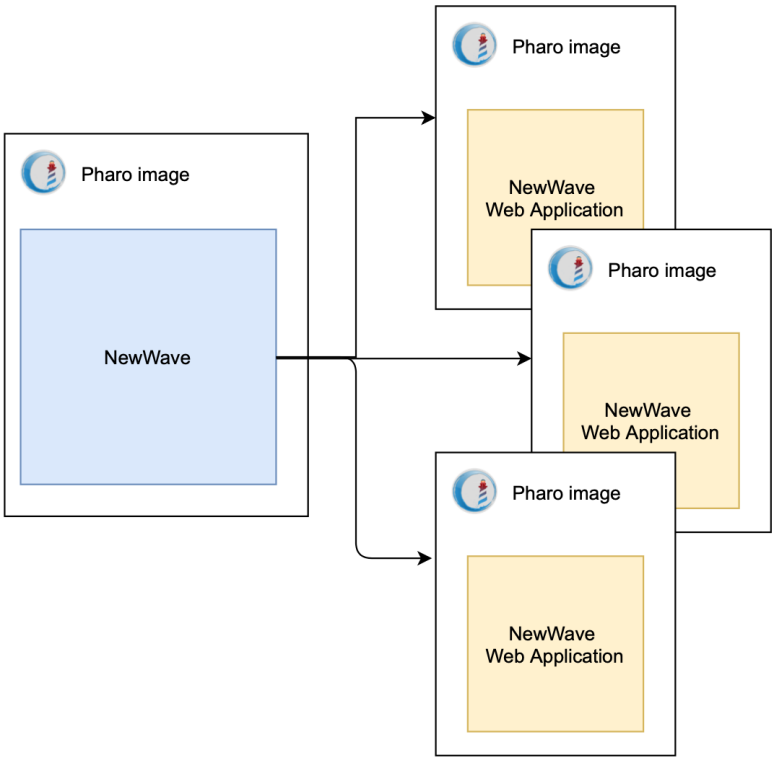
Slika 4.12: Web application - lokalno izvršavanje

slojeviti set apstrakcija preko HTTP-a i HTML-a koji omogućavaju brzo pravljenje interaktivnih web aplikacija. Seaside je integrisan u Pharo okruženje i kao takav koristi sve pogodnosti koje Pharo pruža. Takođe, *Seaside* dozvoljava korišćenje dodataka, te uz pomoć postojećeg dodatka, Seaside je moguće integrisati sa *Magritte* radnim okvirom, koji omogućava automatsko generisanje formi za *Seaside* u okviru veb aplikacije, na osnovu podataka sadržanih u objektima koji su poslati sa servera koji se izvršava u okviru NewWave rešenja.

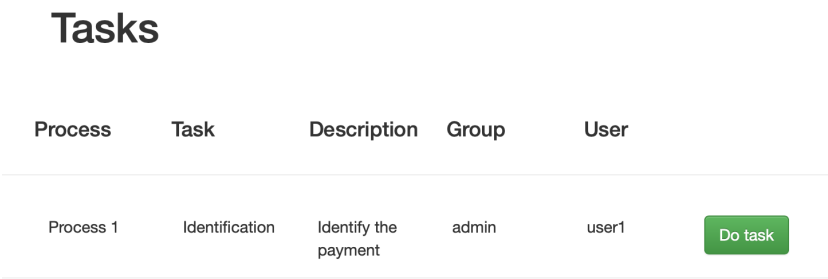
Autorizacija i autentifikacija se vrše na serverskom delu rešenja. Nakon uspešnog prijavljivanja, veb aplikacija šalje relevantne informacije NewWave serveru, koji proverava da li postoje zadaci za datog korisnika, ili za grupu korisnika kojoj on pripada. Ukoliko postoje, pronađeni zadaci se potom šalju veb aplikaciji u JSON/STON formatu. Na osnovu primljenih podataka, veb aplikacija se oslanja na mehanizme opisane u 4.3.3, odnosno na generator objekata koji primljene objekte koji su poslati preko mreže sa NewWave servera, generiše u objekte koji su spremni za korišćenje u okviru veb aplikacije.

Ukoliko postoji zadatak koji je direktno dodeljen korisniku, može mu se direktno pristupiti i izvršiti ga, kao što je to ilustrovano na slici 4.14. Nakon izbora opcije *Do task* korisniku je prikazana generisana forma za konkretan zadatak koji treba da izvrši, a čiji se rezultat popunjavanja prosleđuje nazad ka *NewWave* serveru.

Međutim, ukoliko je zadatak grupni, odnosno nije konkretno dodeljen korisniku, pre



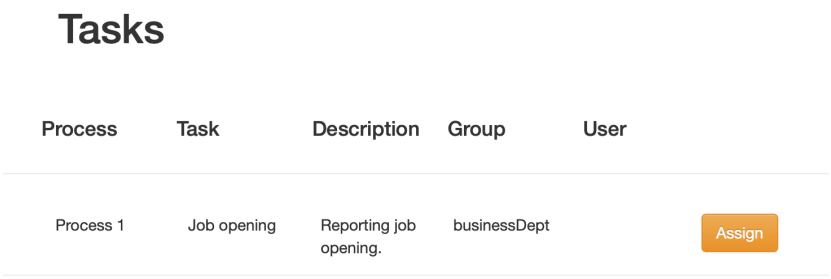
Slika 4.13: Web application - udaljeno izvršavanje



Slika 4.14: Zadatak dodeljen korisniku

nego što se pristupi njegovom izvršavanju, neophodno je prvo preuzeti zadatak, kao što je ilustrovano na slici 4.15.

Zadatak može preuzeti bilo koji korisnik iz grupe korisnika, u ovom slučaju *businessDept*. Slika 4.15 prikazuje i da je polje *User* prazno, odnosno da korisnik



Slika 4.15: Zadatak dodeljen grupi

nije dodeljen. Tek nakon preuzimanja zadatka (*assign*), odnosno njegovog dodeljivanja korisniku, moguće je pristupiti njegovom izvršavanju i nastavku radnog toka.

Ovo poglavlje sadrži relevantne informacije o samom sistemu i kako su odluke koje su donesene uticale na razvijanje ovog rešenja, opisano kroz prizmu koja je bitna za ovu disertaciju. Pored navedenih, postoji još mnogo dosta tehničkih osobina i razloga zašto su određeni delovi sistema osmišljeni i implementirani baš onako kako su osmišljeni i implementirani, međutim opisivanje tih delova prevazilazi potrebe ove disertacije, te se više informacija može pronaći u radu [94].

Poglavlje 5

Postupak i verifikacija adaptabilnosti procesa na NewWave platformi

U ovom poglavlju prikazani su rezultati koji su dobijeni korišćenjem NewWave platforme, odnosno opisan je postupak vršenja adaptacije, kao i radni okvir koji se oslanja na opisano te način na koji se oni integrišu i koriste u okviru predloženog rešenja.

5.1 Sistemi sa velikom dinamikom procesa

Workflow tehnologija je ostavila veliki trag i nepovratno promenila način na koji se kompleksni zadaci obrađuju u modernim organizacijama. Workflow sistemi su uglavnom korišćeni za podršku procesima koji nisu skloni čestim promenama, odnosno statičkim procesima. Sa napretkom tehnologije i sa promenom tržišta sa lokalnog nivoa na globalno i sa promenom načina poslovanja sve više organizacija oslanja se na ovakve sisteme. Ove promene uslovile su veliku dinamičnost u poslovnim procesima organizacija, međutim sistemi koji su bili namenjeni za podržavanje tih istih procesa nisu mogli da odgovore na ovakvu vrstu promene.

Kako je već razmatrano u poglavljima 2.4. i 3.2. adaptabilnost predstavlja sposobnost workflow sistema da se prilagodi određenim promenama kao i da obezbedi podršku procesima kao i standardni workflow sistemi, ali na takav način da sistem zna da se izbori sa određenim promenama. Tipovi mogućih promena su detaljnije analizirani u 3.2. Ugrubo sve promene se mogu podeliti u dve glavne grupe:

- *Izuzetna promena (Exceptional change)* koju karakterišu retki i nepredviđeni događaji tokom izvršavanja procesa i koji zahtevaju da se proces preoblikuje kako bi se uklopio u dati slučaj [72].
- *Evolutivna promena (Evolutionary change)* koja opisuje planiranu migraciju procesa na ažuriranu verziju modela koja implementira nešto novo [95–97]

Da bi ove promene bile moguće, od sistema koji ih koristi zahteva se da poseduje određene sposobnosti. Izuzetne promene potrebno je rešiti brzo i bez mnogo napora, a za evolutivne promene najvažnije je da se svi tekući slučajevi mogu prebaciti na novu definiciju procesa tačno i automatski. [98] Problemi promene u WfMS su i dalje nerešeni, tj. ne postoji opšteprihvaćen način za rešavanje ovakvih problema. Postoji mnogo predloženih pristupa za rešavanje problema promene u WfMS [62, 95, 97, 99] međutim nijedan od njih se nije nametnuo kao opšteprihvaćen i prepoznat način za rešavanje ove vrste problema.

5.2 NewWave platforma

Da bi se pristupilo rešavanju problema dinamičnosti nije dovoljno samo prebaciti pažnju i fokusirati se na procese i pristupe koji pružaju fleksibilnost u izvršavanju procesa (o čemu će biti više reči kasnije). Podjednako je važno izvršno okruženje, odnosno ranije opisana uloga *runtime engine*-a koji ima zadatak da izvršava specificirane procese i koji treba da se prilagodi i da odgovori na predložene pristupe za rešavanje problema promene.

Kao što je već pominjano, tradicionalno workflow sistemi su najčešće korišćeni za podršku procesima koji nisu skloni čestim promenama. Nakon što je evidentno nastala potreba za čestim promenama, neophodno je bilo pristupiti i detaljno razmotriti alate koji treba da odgovore na ove nove zahteve, odnosno da na pravi način primene nove paradigme za rešavanje problema adaptivnosti. Sposobnosti pojedinih rešenja da lako prihvate adaptacije procesa su uglavnom definisane ograničenjima programskih jezika i paradigmi na kojima su napisani alati koje iste trebaju da primene.

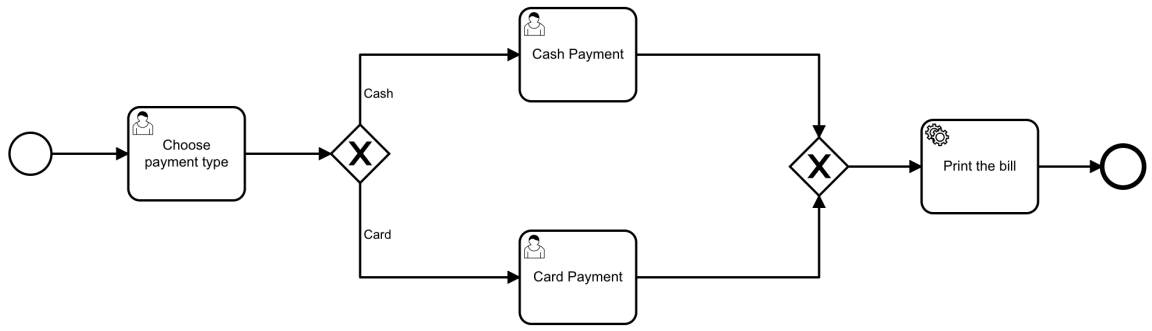
Već su ranije u poglavljima 2 i 4 opisane osobine klasičnih procesno orijentisanih sistema. Tokom godina razvoja većina danas dostupnih sistema, najčešće baziranih na Java programskom jeziku, učinila je ogroman napredak ka usvajanju fleksibilnosti u pogledu izvršavanja procesa, omogućivši tako korišćenje zadataka sa višestukim instanciranjem, kod kojih se dodela zadataka može vršiti i dinamički, na osnovu stanja određenih varijabli u procesu, ili utvrđivanje tačne putanje procesa tek u momentu izvršavanja na osnovu događaja umesto na osnovu predefinisane logike ugrađene u model, kao i mogućnost pokretanja procesa na osnovu eksternih događaja.

Ali fleksibilnost u pogledu promenljivosti definisane strukture procesa nije ostvarena. Problem je u tome što ni alati, kao ni procesi nisu bili projektovani da budu skloni čestim promenama samog modela procesa. Kako su procesi u modernom poslovanju zahtevali sve veću dinamiku i česta prilagođavanja izmenjenim uslovima tržišta, trebalo bi da i alati, odnosno jezici koji se koriste za implementaciju istih, pređu sa statičnih na dinamične. Međutim, nije dovoljno samo preći sa statičnog na dinamički jezik, nego i razmotriti sve mogućnosti koje neki programski jezik pruža, u cilju podrške za pravu fleksibilnost jednog sistema i novih zahteva koji mogu nastati.

U poglavlju 4, predstavljeni su i osnovne karakteristike Pharo programskog jezika, kao i prednosti koje Pharo izvršno okruženje može pružiti u pogledu dinamičke adaptacije, ukoliko se jezgro procesnog sistema implementira u ovakvom okruženju. Ideja "živog" okruženja, koju pruža Pharo, je u dobroj korelaciji sa osobinama koje se žele dobiti od adaptabilnog procesnog okruženja. U narednim odeljcima opisani su detalji implementiracije pojedinih modula, verifikovana je njihova funkcionalnost i prikazano kako je moguće iskoristiti kreirani alat za prilagođavanje na određene paradigme adaptivnog workflow-a.

5.3 Definisanje procesa

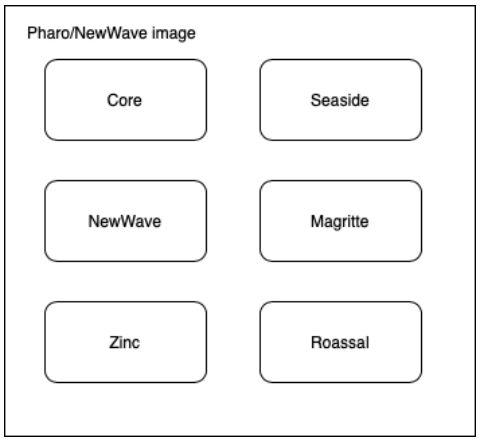
Kreiranje procesa je jedan od osnovnih koraka u svakom WfMS. Iako je kreiranje i inicijalizacija procesa, nešto što je već dobro poznato i ustanovljeno kao procedura, prednosti predložene platforme već na samom početku su uočljive. Kao primer koristi se odabir načina plaćanja u nekoj organizaciji, kao što je prikazano na slici 5.1. Korisnik sistema treba da odabere način plaćanja u okviru informacionog sistema, da odabere karticu ili gotovinu, nakon čega se izdaje račun i završava aktivnost.



Slika 5.1: Primer procesa placanja

Korišćenjem NewWave platforme, celi proces može da se kreira u okviru okruženja u kome treba i da se izvrši. To dalje implicira, da kreiranje elemenata procesa, kao i samog procesa pripada istom sistemu kao i NewWave, odnosno da se već u samom startu izbegavaju transformacije sa alata koji koristimo na okruženje koje treba da ga izvrši i da upravlja definisanim procesom.

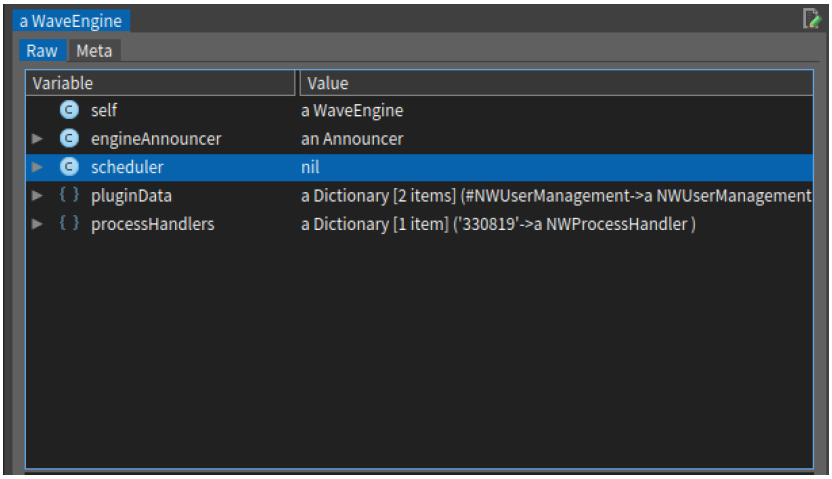
Slika 5.2 prikazuje neke zavisnosti koje koristi NewWave, ono što je bitno naglasiti jeste da su sve biblioteke i aplikacije ovde u okviru jednog sistema, što dalje znači da se pristup navedenim bibliotekama i aplikacijama može vršiti direktno, time znatno



Slika 5.2: Pojednostavljen prikaz Pharo image-a sa NewWave

pojednostavljujući međusobnu komunikaciju i eliminišući potrebu da se prave posebni interfejsi koji bi omogućili da ove biblioteke, odnosno aplikacije komuniciraju.

NewWave, kao što je ranije pomenuto izvršava se u živom okruženju, što znači da je njegova komponenta za upravljanje procesima, spremna da prihvati i odreaguje na izmene tokom rada. Neophodna osobina da bi se zaista postigao adaptivni workflow engine. Na slici 5.3 prikazano je trenutno stanje NewWave engine-a, nakon izvršenja primera koji je dat u sekciji na početku 5.3. Kao što se vidi na slici *engine* ima trenutno aktivan jedan ProcessHandler, koji je zadužen za upravljanje pokrenutim procesom.



Slika 5.3: Prikaz stanja NewWave engine-a, sa akcentom na ProcessHandler

Dakle, NewWave je moguće menjati tokom njegovog rada, svaka promena izvršena nad elementima primenjuje se trenutno, i odražava se u sklopu sistema u kome se izvršava. Ukoliko pogledamo *ProcessHandler*, vidimo koliko je trenutnih procesa

aktivno. Nakon slanja poruke metodi `addProcess:`, zadati proces se trenutno dodaje u listu procesa koji treba da se izvršavaju, i na kraju pozivom metode `startProcess`, pokreće se prethodno dodati proces. Sve navedene naredbe izvršene su tokom rada NewWave alata i nisu zahtevale njegovo zaustavljanje ili pauziranje kako bi se izvršile, čime se postigla bitna stvar a to je živa manipulacija procesa.

5.4 Mogućnost odlučivanja

U poslovnim procesima često je u nekom trenutku, potrebno odabrati između dve ili više opcija na osnovu kojih se menja tok izvršavanja, odnosno bira se između dve ili više alternativnih grana, ovu pojavu zovemo grananje. U primeru koji je dat u sekciji 5.3 prikazan je primer ekskluzivnog (XOR) grananja, gde je potrebno da korisnik odabere opciju *Cash* ili *Card* kako bi workflow engine doneo odluku kojom granom će nastaviti izvršavanje.

Ako se pogleda listing 5.1, primećuje se da `withCondition:` očekuje blok na osnovu koga *engine* odlučuje koju granu da odabere. Blok, kao što je ranije pominjano ima delegirano izvršavanje, odnosno on se izvršava tek u trenutku poziva. U blok, takođe može se ugraditi bilo šta, odnosno proizvoljna programska logika. NewWave koristi mehanizam bloka, kome prosleđuje rezultat izvršavanja zadatka, a koji je opisan promenljivom x u datom primeru. Promenljiva x je upravo rezultat izvršenja nekog zadatka i najčešće je to konkretan objekat čija svojstva i metode možemo iskoristiti.

Još jedna stvar koju NewWave primenjuje jeste prebacivanje kontrole na primaoca poruke, odnosno *paymentType* je poruka koja se šalje primaocu x. Dakle, x treba da odluči šta će da uradi sa primljenom porukom, ako zna da odgovori na nju, vratiće odgovor, ako ne delegiraće je dalje. Ovaj mehanizam dopušta pisanje zahtevnih naredbi na jednostavan način, što u npr. Javi, C++, C# ne bi bilo moguće, jer je potrebno unapred znati sve elemente procesa, kako bi se konstruisala *if* naredba, dok NewWave odlučivanje zaista prepušta onome ko treba da donese odluku, odnosno rezultatu izvršavanja zadatka.

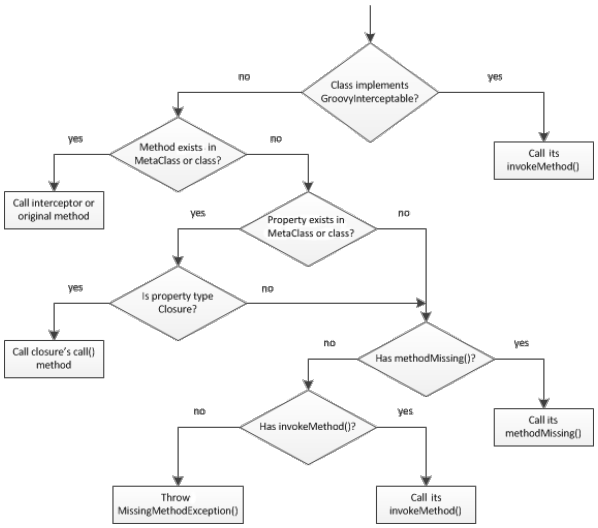
```
1       startEvent addOutgoingEdge: task1.
2       task1 addOutgoingEdge: split1.
3       split1
4           addOutgoingEdge: task2
5           withCondition: [ :x | x paymentType = 'Card' ].
6       split1
7           addOutgoingEdge: task3
8           withCondition: [ :x | x paymentType = 'Cash' ].
```

Listing 5.1: Primer odlučivanja u bloku.

Ovaj mehanizam pruža mogućnost prilagođavanja nekom objektu u toku izvršavanja procesa, gde se odluka o daljem izvršenju, ili poziv neke metode vrši nad nativnim

elementom sistema tj. procesa koji je integrisan u NewWave, a ne pozivom neke skripte koja ima zadatak da se ubaci u neki sistem i pruži ograničenu funkcionalnost koja uglavnom zavisi od integracije datog skript jezika, sa engineom na kome se izvršava.

Poređenja radi, BonitaSoft i Camunda BPM, koriste Groovy kao skript jezik za ugrađivanje dodatne programske logike u procese. Apache Groovy je našao primenu zbog mogućnosti da se integriše u Java aplikacije tokom njihovog izvršavanja. Da bi mogao da se ugradi Groovy koristi "Interception mehanizam" koji prikazan na slici 5.4[1]. Međutim zbog toga što je to različit jezik od jezika na kome su pisani ovi alati (iako je namenjen da se integriše sa prethodnim) pored očigledne krive neophodne za učenje još jednog jezika, skripte koje se najčešće pišu su jednostave iz razloga što se ne može u potpunosti i na jednostavan način, ispratiti, kontrolisati i na kraju debugovati skripta koja se ugrađuje u drugi programski jezik.



Slika 5.4: Groovy interception mechanism

5.5 Ugrađivanje domenskih podataka (data objekata)

Od procesa se često zahteva da skladište neke dodatne informacije koje se koriste u toku njegovog izvršavanja, te dodatne informacije mogu opisivati neki materijalni predmet kao što je mobilni telefon, ali opet mogu opisivati neku informaciju kao što je faktura. NewWave ima ugrađenu podršku za direktnu manipulaciju tim elementima, koje nazivamo data-objektima. Data-objekti, su objekti koji su dostupni dok god je dostupna i instanca procesa. Ukoliko opet pogledamo primer u sekciji 5.3, data

[1]Slika preuzeta sa: https://groovy-lang.org/metaprogramming.html

objekat koji tu postoji jeste *Payment* objekat, koji sadrži informacije o načinu plaćanja transakcije.

NewWave dozvoljava ugrađivanje nativnih data-objekata direktno u okruženje, odnosno ne postoji potreba da se koriste skript jezici ili dodatni alati za dati cilj, što implicira da je taj data-objekat podržan kao i svaka druga klasa iz Pharo okruženja. Da bi neka klasa, odnosno njena instanca služila kao data-objekat potrebno je ugraditi u *NWDataObject* klasu, kako bi *engine* znao da sa njom upravlja. Izgled jednog jednostavnog Payment objekta je dat na listingu 5.2. *NWPaymentMethod* sadrži dva jednostavna polja, *id* i *paymentMethod*. Ukoliko je NewWave engine u režimu rada, gde se njegova veb aplikacija izvršava u okviru istog Pharo *image*-a kao i *engine* , rukovanje ovim objektom postaje trivijalno, jer ne zahteva da se nad njim vrše nikakve transformacije i kao takav može se delegirati u veb aplikaciju.

```
1  Object subclass: #NWPaymentMethod
2      instanceVariableNames: 'id paymentMethod'
3      classVariableNames: ''
4      package: 'NewWaveFieldUserTask-Entities'
```

Listing 5.2: Primer Payment objekta.

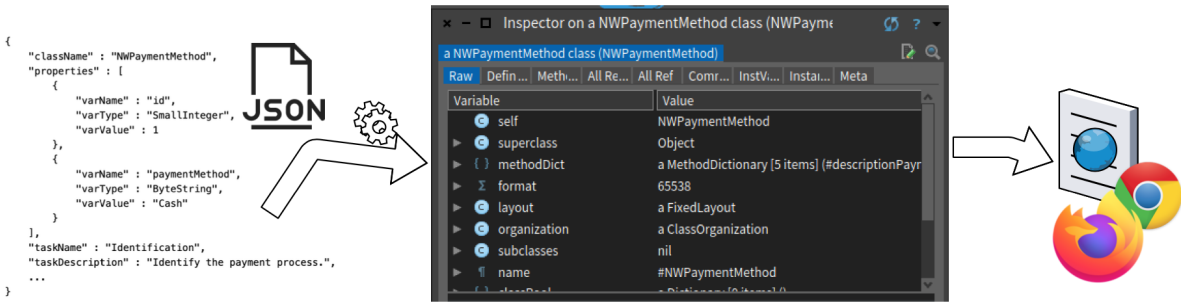
Ukoliko pak, NewWave radi u režimu tako da je web aplikacija van *image*-a u kome se nalazi potrebno je obezbediti da se informacije prenesu do veb aplikacije. Najpogodniji format za prosleđivanje podataka ka veb aplikaciji bio bi STON (Smalltalk Object Notation), za koga postoji velika podrška u Pharo okruženju [89], međutim ideja je da i ovaj deo sistema, odnosno prenos podataka bude fleksibilan, tako da veb aplikacija koja komunicira sa NewWave-om, ne mora biti napisana u Pharou te je stoga odabran JSON (JavaScript Object Notation). JSON je otvoreni standard koji opisuje format datoteke, razumljiv je za čitanje i pisanje od strane ljudi, a jednostavan je za parsiranje i generisanje od strane mašine i kao takav pogodan je za razmenu podataka. NewWave ima podrazumevanu veb aplikaciju, međutim korišćenje JSONa omogućava proizvoljne *plugin*-e za NewWave web aplikativni deo, odnosno kreiranje proizvoljnih veb aplikacija. JSON dobijen na osnovu *NWPaymentMethod* objekta prikazan je na listingu 5.3.

Podrazumevana aplikacija zasnovana je na Seaside frameworku za razvoj veb aplikacija u Pharo okruženju. Međutim, kako bi se nadoknadio nedostatak fleksibilnosti koji je nastao uvođenjem JSON-a za razmenu podataka između *engine*-a i veb aplikacije, kreiran je generator data-objekata koji ima zadatak da tu fleksibilnost vrati tako što iz JSON-a generiše klasu koju instancira i zatim koristi tokom rada aplikacije. Ilustracija opisanog postupka prikazana je na slici 5.5.

Ovo je moguće zbog još jedne jedinstvenosti i prednosti Pharo Smalltalk sistema, a to je da je moguće kreirati klasu, tokom rada samog sistema i instancirati je, te je kao takvu proslediti NewWave veb aplikaciji na korišćenje. Prednost ovog pristupa je

```
{
    "className" : "NWPaymentMethod",
    "properties" : [
        {
            "varName" : "id",
            "varType" : "SmallInteger",
            "varValue" : 1
        },
        {
            "varName" : "paymentMethod",
            "varType" : "ByteString",
            "varValue" : "Cash"
        }
    ],
    "taskName" : "Identification",
    "taskDescription" : "Identify the payment process.",
    ...
}
```

Listing 5.3: Primer JSON-a koji dobijenog od NWPaymentMethod klase.



Slika 5.5: NWPaymentMethod transformacija iz JSON-a u data-objekat u okviru veb aplikacije

u tome što veb aplikacija, ne mora unapred da zna, odnosno ima informacije o bilo kom entitetu koji će se isporučiti od strane *engine*-a, nego će na osnovu primljenih podataka generisati i koristiti potreban entitet. Iako i drugi jezici, kao na primer Java, pružaju mogućnost refleksije i dinamičkog učitavanja klasa, prednost Pharoa je u transparentnsoti i jednostavnosti ovog koncepta.

5.5.1 Transformacije nad objektima

Upravljanje data-objektima nije trivijalno. NewWave se oslanja na mehanizam koji je opisan u 4.3.3. Dati meta-model i odgovarajuća logika koja je opisana predstavlja osnovu ovog rešenja i omogućava potpuno dinamički pristup data-objektima. Pošto

data-objekti nisu deo nekog spoljnog jezika, nego su nativni Pharo objekti, njima se pristupa kao i svim ostalim elementima.

Data-objekte koji čine deo jednog radnog toka moguće je menjati i manipulisati u zavisnosti od potrebe. Jedan objekat može imati više različitih opisa, koji su predstavljeni sa *NWOperationDescription* što znači da se jedan objekat, u zavisnosti od zahteva može predstaviti drugačije određenim korisnicima, odnosno da njegov izgled može biti prilagođen bez menjanja njegovog internog stanja, odnosno atributa. Predloženi pristup, dakle omogućava kreiranje različitih fasada na osnovu zahtevane specifikacije.

NewWave omogućava da se ode i jedan korak dalje time što na osnovu predloženog radnog okvira, ukoliko je to potrebno omogućava parcijalno isporučivanje zadataka, odnosno njegovih delova pri čemu su sve izmene vezane za originalni data-objekat koji deo živog sistema. Delovi jednog data-objekta mogu se isporučiti na više udaljenih Pharo okruženja na kojima je NewWave aplikacija, a traženi odgovor se isporučuje odakle je i poslato, odnosno na Pharo okruženje sa aktiviranim NewWave serverom.

5.5.2 Generisanje izgleda forme

Kreiranje korisničkih interfejsa zavisi od mnogo stvari, NewWave ima podrazumevano ponašanje da na osnovu elemenata koje treba da prikaže generiše korisnički interfejs koji će biti prikazan korisniku u toku rada. Međutim, često je potrebno prilagoditi korisnički interfejs specifičnim potrebama korisnika. Jedan od načina na koji NewWave podržava kreiranje proizvoljnih elemenata forme jeste ugrađivanjem Magritte opisa u data-objekte. Za *NWPaymentMethod* primer je prikazan na listingu 5.4. Na listingu je navedeno da je potrebno generisati select, odnosno *drop-down* element forme, koji ima dve opcije *Cash* i *Card*, da je ovaj opis povezan sa paymentMethod poljem u okviru data-objekta, da labela koja je prikazana uz selektor ima tekst Payment method, i da je prioriteta 40 na formi u okviru koje se prikazuje.

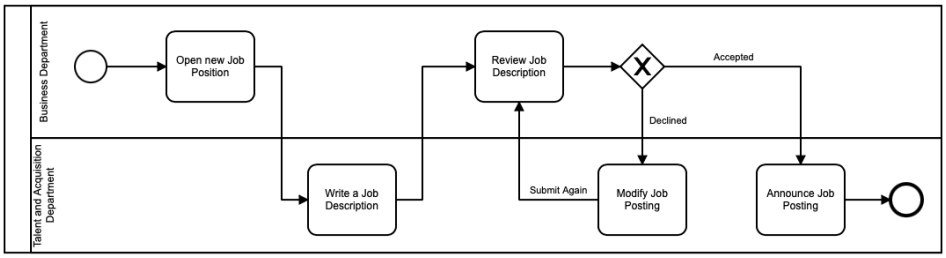
```
1  descriptionPaymentMethod
2
3      ^ MASingleOptionDescription new
4          options: #('Cash' 'Card');
5          reference: MAStringDescription new;
6          accessor: #paymentMethod;
7          label: 'Payment method';
8          priority: 40;
9          yourself
```

Listing 5.4: Primer proizvoljnog opisa elementa forme za NWPaymentMethod objekat.

Ukoliko postoji proizvoljni opis nekog elemenata u data-objektu, NewWave ignoriše podrazumevani i proizvoljni, korisnički kreirani opis.

5.6 Adaptacija radnog toka

Na osnovu prethodno predstavljenog rešenja i opisanih pravila za adaptaciju, prikazan je primer koji treba da ilustruje ponašanje opisane platforme i rešenja. Na slici 5.6 je prikazan model radnog toka koji odgovara poslovnom procesu u okviru kompanije, a koji se odnosi na oglašavanje posla i zapošljavanje. U okviru procesa prikazana su dva sektora, *Business Department* (odeljenje koje oglašava nove pozicije) i *Talent and Acquisition Department* (odeljenje koje je zaduženo za pisanje objava za oglašavanje pozicije i za selekciju). Različiti korisnici i korisničke grupe u okviru NewWave su podržani *DefaultUserManagement* komponentom, koja je opisana ranije.



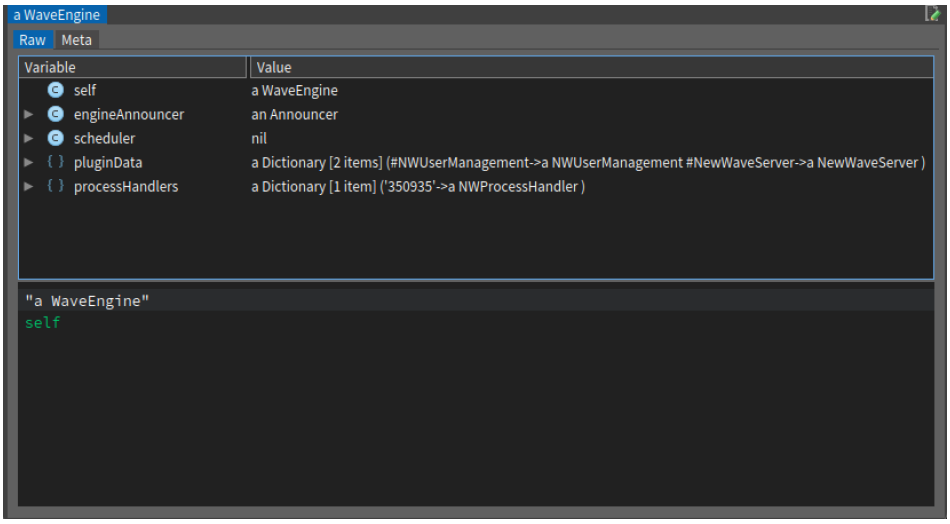
Slika 5.6: Primer poslovnog procesa koji ilustruje oglašavanje za posao

Dati model predstavljen BPM notacijom, transformiše se u model koji je opisan ranije, a koji je podržan od strane NewWave platforme. Postupak transformacije se može izvršiti na više načina, dodavanjem odgovarajuće komponente koja predstavlja parser u komponenti sistem NewWave rešenja, korišćenjem neke od postojećih biblioteka iz Pharo ekosistema za učitavanje BPMN modela i dodavanjem mapera koji mapira dobijene modele na adekvatne NewWave modele, manuelno, korišćenjem specijalizovanih dodataka i slično.

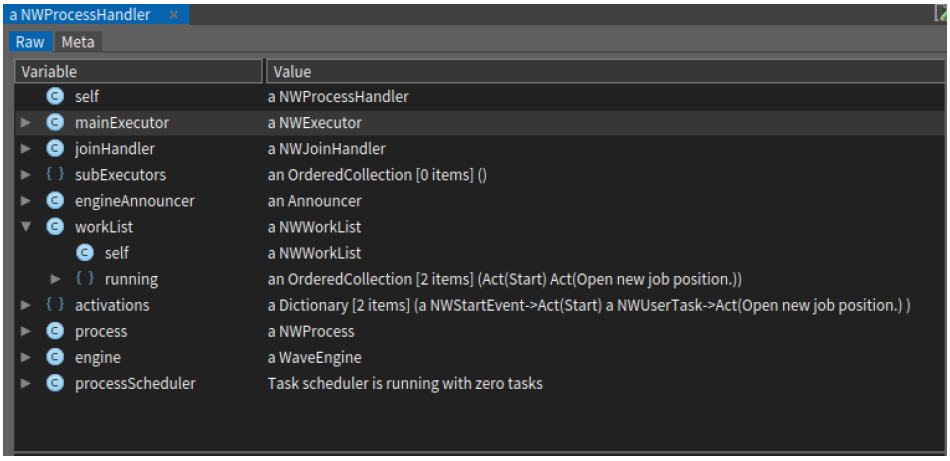
Izvršavanjem datog modela, kreira se njegova instanca i započinje se postupak interpretacije. Prilikom prijavljivanja na sistem, bilo ko iz sektora za oglašavanje nove pozicije (pošto nikome eksplicitno nije dodeljen zadatak) može da preuzme zadatak i da otvori novu poziciju za posao koju će neko iz sektora za pisanje objava i oglašavanje pozicije da preuzme.

Inicijalno stanje *WaveEngine* komponente, prikazno je na slici 5.7, sa učitana dva dodatka (eng. *plugin*) za upravljanje korisničkim grupama i serverom za isporučivanje zadataka i *ProcessHandler* koji je zadužen za upravljanje pokrenutim procesom, čije je stanje nakon pokretanja procesa prikazano na slici 5.8.

Međutim, zbog potrebe da se u proces uključi i odeljene za administraciju *Administration Department* kako bi nakon prihvatanja pozicije za oglašavanje od strane



Slika 5.7: Inicijalno stanje *WaveEngine* komponente

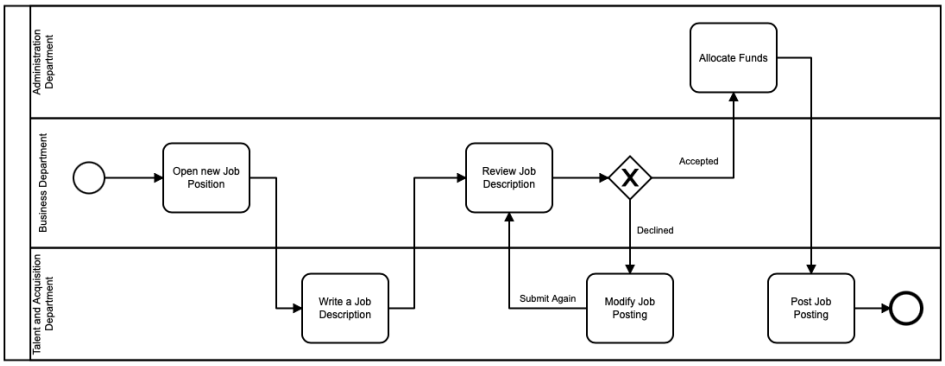


Slika 5.8: Stanje *ProcessHandler*-a nakon pokretanja procesa

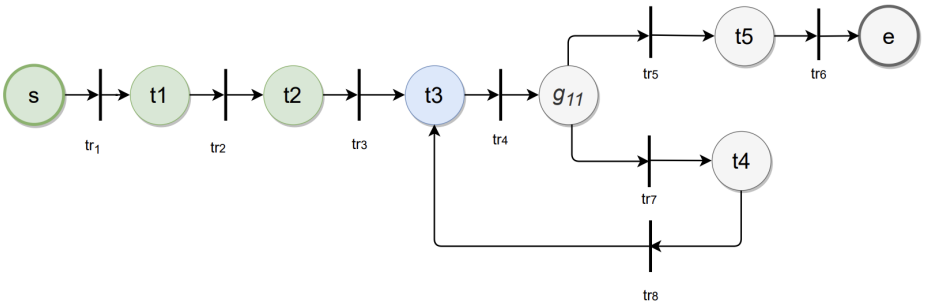
biznisa alocirali predviđena sredstva za tu poziciju, potrebno je izvršiti modifikaciju poslovnog procesa. Novi izgled procesa nakon izvršenih korekcija prikazan je na slici 5.9.

Da bi se proces prilagodio, potrebno je izvršiti prethodno opisanu primenu proširivanja. U datom primeru ilustrujemo dva scenarija, odnosno uspešan i neuspešan ishod. Uspešan ishod se dešava ukoliko stanje čvora koji je predstavljen aktivnošću *Review Job Description* nije završen, odnosno ukoliko je taj čvor još uvek aktivan, odgovarajući dijagram aktivnosti za navedeni scenario prikazana je na slici 5.10.

Prateći notaciju iz prethodnih primera, zelenom bojom su obeleženi čvorovi koji su završeni, a plavom bojom je obeležen trenutno aktivan čvor. Stanje procesa u trenutku kada je čvor *t3* aktivan, u okviru *WaveEngine* komponente prilikom izvršavanja procesa

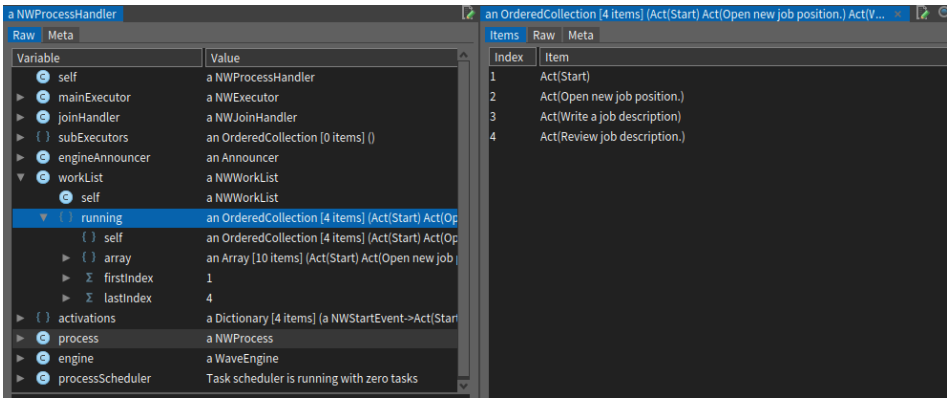


Slika 5.9: Primer tražene izmene u poslovnom procesu koji ilustruje oglašavanje za posao



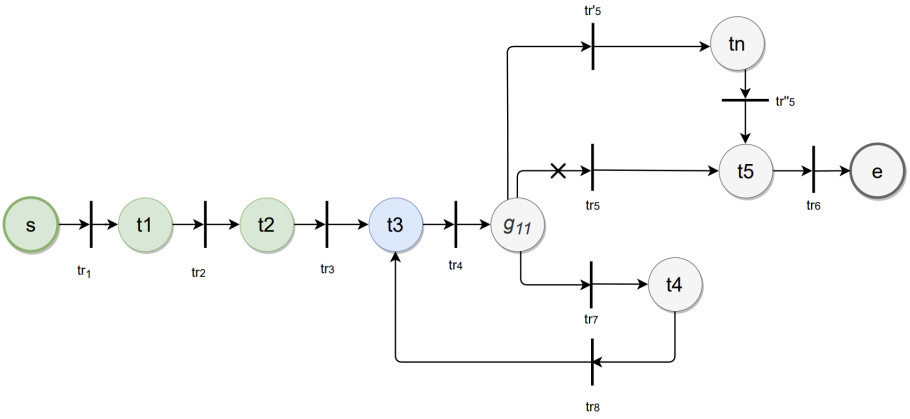
Slika 5.10: Dijagram aktivnosti koji odgovara poslovnom procesu za ilustraciju oglašivanja za posao

prikazano je na slici 5.11.



Slika 5.11: Stanje procesa kada je u čvor *t3* aktivan

U tom slučaju moguće je dodati novi čvor u mrežu i uspešno izvršiti adaptaciju proširenja, kao što je to prikazano na slici 5.12. Novi dodati čvor je *tn*, koji je dobijen tako što je prekinuta veza između čvorova *g11* i *t5*, ukidanjem tranzicije *tr5*. Uvedena je nova tranzija *tr'5* koja povezuje čvorove *g11* i novokreirani čvor *tn*, kao i tranzicija *tr"5*, koja povezuje čvorove *tn* i *t5*. Čvor *t5* odgovara zadatku *Allocate Funds*, koji se nalazi u okviru *Business Department*-a.



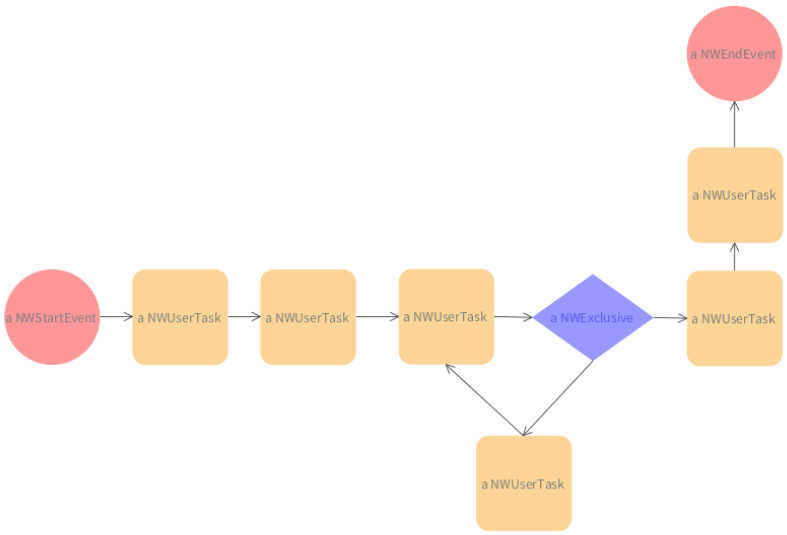
Slika 5.12: Dijagram aktivnosti koji odgovara adaptiranom poslovnom procesu za ilustraciju oglašivanja za posao

Novi izgled poslovnog procesa i njegov prikaz koristeći *NewWave* dodatak za vizualizaciju prikazan je na slici 5.13.

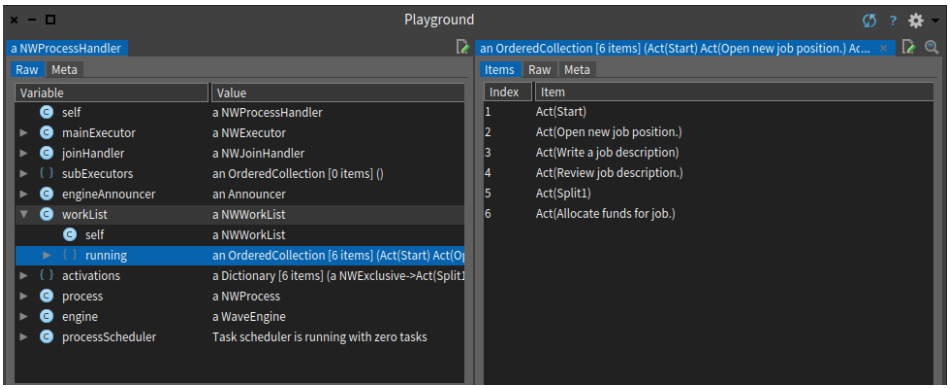
Takođe, promenjeno stanje *ProcessHandler*-a nakon dodavanja novog elementa prikazano je na slici 5.14, gde se vidi da je novi element dodat u radnu listu u okviru procesa.

Nakon izvršenja adaptacije novododati čvor se nalazi učitan u radnoj memoriji tekuće instance procesa, te u zavisnosti od uslova definisanih procesom može se izvršiti kao i svaki drugi čvor u okviru jednog procesa, odnosno tokom interpretiranja *FlowHandler* komponenta će preuzeti čvor za izvršavanje i proslediti ga *WaveExecutor*-u na dalju obradu, nakon čega radni tok nastavlja sa izvršavanjem na uobičajen način. Potrebno je još jednom naglasiti pomenuto u prethodnoj sekciji, a to je da će primenjenu izmenu nad modelom NewWave zabeležiti i inkrementirati njegovu verziju, te će svaka sledeća interpretacija modela započeti po trenutno aktuelnoj verziji modela.

Dosta jednostavnija, ali opet bitna promena je promena koja za rezultat ime neuspešan ishod. Neuspešan ishod se dobija ukoliko je aktivirana aktivnost *Announce Job Posting*, koja je predstavljena čvorom *t5*. U navedenom slučaju radni tok je prošao kroz željeno mesto proširenja i aktivacija je završila sa čvorovima na navedenoj putanji, te u datom slučaju nije moguće prilagoditi radni tok na željeni način.



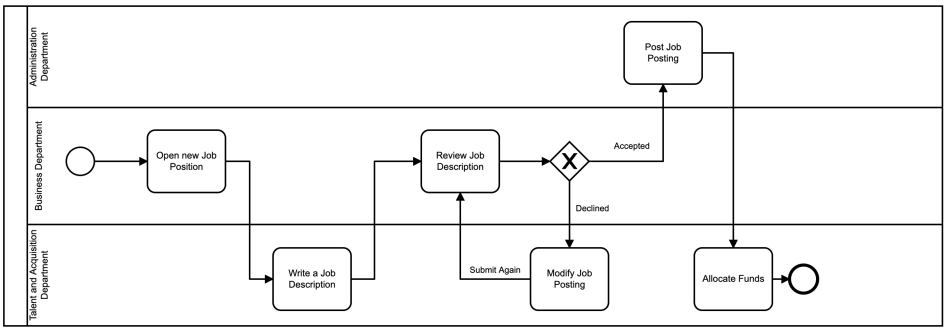
Slika 5.13: Poslovni proces nakon adaptacije



Slika 5.14: Stanje *ProcessHandlera*-a sa novim elementom

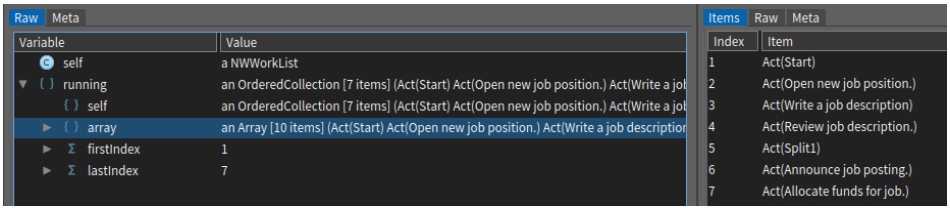
Na ovom primeru takođe moguće je izvršiti i pokazati adaptacije zamene i zamene redosleda zadataka, koje su opisane ranije i čiji je model dat u poglavlju 3. Adaptacija zamene redosleda zadataka, prethodnog primera bi odgovarala primeru koji je prikazan na slici 5.15.

Naslanjajući se na prethodni primer, ilustracije radi, zamenu poslednja dva zadatka moguće je izvršiti po opisanom scenariju, odnosno ukoliko je izvršenje procesa u čvoru *t3*, moguće je izvršiti zamenu redosleda zadataka *Allocate Funds* i *Post Job Posting*. Ta dva zadatka još uvek nisu aktivirana i u ovom datom primeru, dozvoljeno je njihovo



Slika 5.15: Primer aktivnosti zamene redosleda zadataka

rotiranje. Nakon izvršenja zamene redosleda zadataka i daljeg izvršavanja procesa na slici 5.16 prikazani su izvršeni zadaci na indeksima *6* i *7*, sa opisima *Announce job posting* i *Allocate funds for job* respektivno, odnosno adaptacija zamene redosleda zadataka je uspešna.

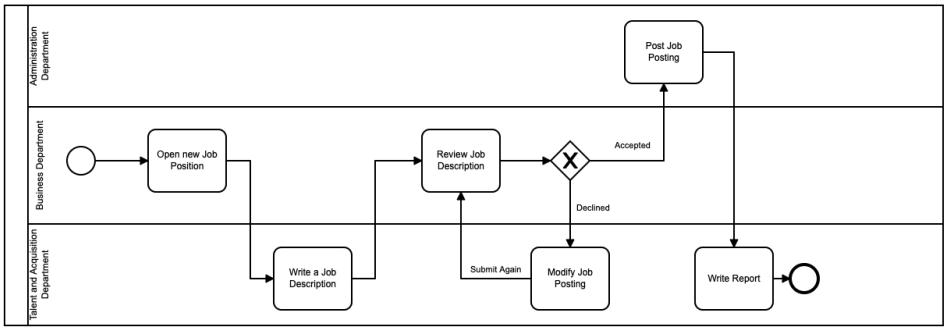


Slika 5.16: Stanje koje opisuje primenu aktivnosti zamene redosleda zadataka

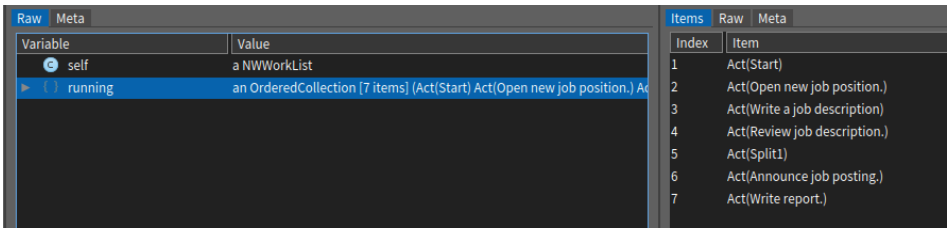
Ukoliko je potrebno zameniti elemente u okviru procesa nekim drugim elementima, odnosno izvršiti operaciju zamene zadataka, postupak je prikazan na slici 5.17. Slika ilustruje traženu izmenu, odnosno zadatak *Allocate Funds* iz prethodnog primera, potrebno je zameniti zadatkom *Write Report*. Adaptacija je moguća pod istim uslovima kao i u prethodnim primerima, odnosno ukoliko je aktivan čvor *t3* adaptacija je moguća. Stanje koje prikazuje izvršenu zamenu zadatka nalazi na slici 5.18. Slika prikazuje dosadašnji tok izvršavanja zadatka u procesu, kao i novi zadatak *Write Report* koji se u radnoj listi izvršavanja nalazi na indeksu broj sedam.

Međutim, ukoliko je pokušana adaptacija nekog slučaja, koji nije podržan opisanim modelom, dolazi se do greške u adaptaciji. Slika 5.19 ilustruje prethodni primer, gde je *Announce job posting* zadatak izvršen (što možemo videti po stanju svojstva *completed* na istoj slici). Pokušaj neke od adaptacija, npr. proširenja zadataka dovodi do greške koja je zabeležena i prikazana korisniku, kao što je to prikazano na slici 5.20.

Ukoliko imamo dva ista procesa, odnosno kreirani su na osnovu istog modela, moguće je izmeniti jednog od njih tako da se prvi proces završi po prvobitnom modelu, dok bi se drugi izvršavao na osnovu novog, izmenjenog modela procesa. Ovaj primer prikazuje kako dva procesa, koja su ista u startu, primenjivanjem adaptacija završavaju



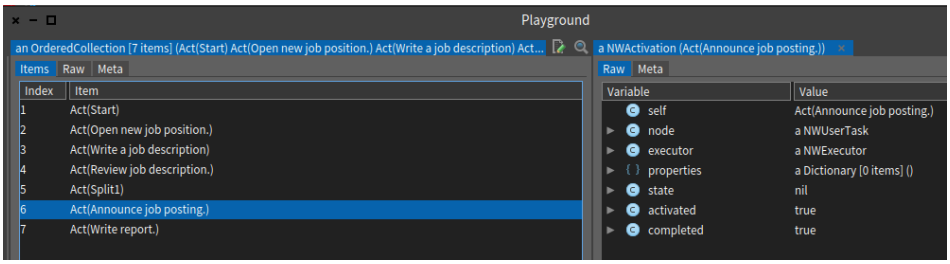
Slika 5.17: Primer aktivnosti zamene zadatka



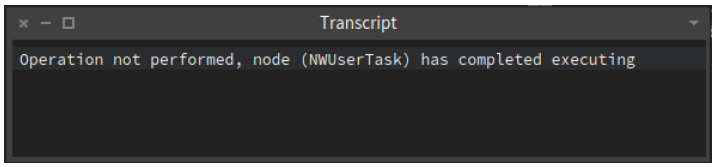
Slika 5.18: Stanje koje opisuje primenu aktivnosti zamene zadatka u procesu

sa različitim modelima procesa. Kao što je već ranije opisano, ukoliko postoji proces koji se trenutno izvršava a koga nije moguće promeniti, on nastavlja svoje izvršavanje bez promena. Slika 5.6 iz ove sekcije, prikazuje primer na osnovu koje su kreirani i pokrenuti procesi, a koji su u okviru okruženja prikazani na slici 5.21. Slika prikazuje *NWProcessHandler* i *id* procesa koji se izvršava a koji su učitani u *WaveEngine*. Kao što se vidi procesima su dodeljeni elementi zaduzeni za njihovo upravljanje i njihove instance su odvojene. Slika 5.9 predstavlja novi model procesa koji je potrebno učitati i pokrenuti. Nakon izmene modela kao što je to prikazano na slici i pokretanja traženih izmena, *WaveEngine* prekо*NWProcessHandler*-a pronalazi željeni proces i menja ga. Proces se menja identično kao što je već opisano u primeru adaptacije proširenja koji je ilustrovan slikom 5.10.

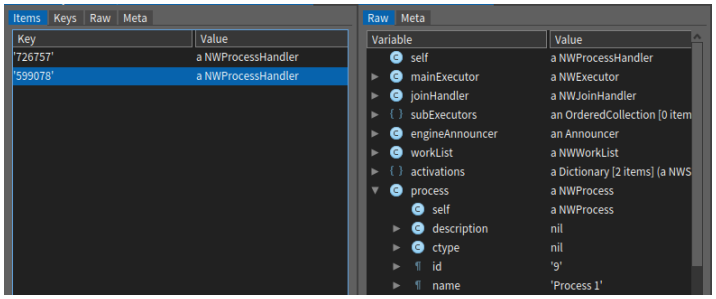
U ovoj sekciji prikazan je jedan mali, ali kompletan ilustrativni postupak kako za zadati model radnog toka izvršiti adaptacije, odnosno prikazani su čvorovi koji oslikavaju elemente iz modela radnog toka kao i tranzicije koje se u njemu nalaze. Prikazan je postupak izvršavanja čvorova u sklopu rešenja i načine kada je moguće, odnosno kada nije moguće izvršiti željenu adaptaciju, držeći se pravila koja su opisana u poglavlju 3. Potrebno je i napomenuti da NewWave skladišti i obrađuje mnogo više informacija nego što su u ovom primeru prikazane, odnosno ovaj primer je zanemario neke aspekte datog rešenja, zbog lakšeg razumevanja adaptacija i kako se one vrše, a čiji opis je primarni cilj navedenog primera.



Slika 5.19: Stanje koje opisuje pogresno stanje prilikom pokušaja adaptacije



Slika 5.20: Greška kao rezultat pogrešnog



Slika 5.21: Inicijalni proces primera oglašavanja za posao

5.7 Analiza predstavljenog rešenja

U ovom poglavlju su pored opisa postupka adaptacije u okviru predloženog rešenja, prikazane i ostale mogućnosti koje *NewWave* pruža. Objašnjeno je i ilustrovano kako je model kojim je opisana arhitektura sistema, doprineo kreiranju jednog fleksibilnog rešenja i kako je odabir implementacione platforme komplementaran u odnosu na dati model. *ADEPT* je kreiran sa sličnom idejom fleksibilnosti na umu i on omogućava odstupanje od prvobitno modelovane šeme procesa dodavanjem novih koraka, preskakanje postojećih i povratak na prethodne, međutim izostala je podrška za propagaciju na tekuće instance procesa, kao i podrška za trenutnom povratnom informacijom i uvidom u stanje sistema, odnosno razlika između vremena kompajliranja i vremena izvršavanja, jer je sistem zasnovan na Javi. *YAWL* se oslanja na Petrijeve mreže i bio je jedan od prvih koji je fleksibilan po dizajnu, kreiran je sa ciljem da podrži što više šablona radnih tokova, odnosno alternativni putevi izvršenja su navedeni

eksplicitno u procesu tokom njegovog dizajniranja, ovaj sistem je takođe zasnovan na Javi.

CAKE koristi agilne radne tokove putem kojih je moguće modelovati pojedinačne zadatke ili upotrebiti kasno odlučivanje, gde se modelovanje vrši na višem nivou apstrakcije, omogućava da se tokom izvršavanja definiše specifičan način obavljanja zadatka (podržano putem CBR-a). *NewWave* koristi data-objekte, koje je moguće iskoristiti za donošenje odluke prilikom izvršavanja radnih tokova, ovi elementi su detaljno objašnjeni u prethodnim poglavljima, jer je za njihovu podršku uveden metamodel, kao i generator koji podržava njihovo kreiranje i rekreiranje u slučaju da se informacije o nekom objektu dobiju u nekim od navdenih formata kao odgovor sa servera , sa kojim je *NewWave* u komunikaciji.

Malo drugačiji sistemi su *Alaska Simulator* i *FLOWer*. *Alaska Simulator* je deklarativni sistem za upravljanje poslovnim procesima, kreiran sa ciljem da odgovori na pitanje da li su korisnici ti koji mogu da izvrše prilagođavanje u cilju izvršavanja poslovnog procesa prilikom korišćenja deklarativnog pristupa, međutim iako je deklarativni pristup ima određene predosti, takođe se naglašavaju i mane istog pristupa u razumevanju i kontrolisanju okruženja dobijenog fleksibilnošću u nekom kompleksnom sistemu, kao što je to opisano ranije. *FLOWer* je zasnovan na rukovanju slučajevima, odnosno paradigmi za podršku fleksibilnom poslovnim procesima sa fokusom na aspekt podataka , koji umesto predefinisanih procesnih struktura fokusira se na to šta može biti urađeno da se ostvari poslovni cilj uz podršku akcija kao što su preskakanje ili ponovno izvršavanje aktivnosti u procesu.

NewWave podržava tri različite adaptacije, koje su analizirane u poglavlju 3.2. a implementacija je ilustrovana u prethodnoj sekciji. Odnosno adaptacije zamene, proširenja i zamene redosleda zadataka. Svaka od navedenih adaptacija, je detaljno opisana i objašnjena u ranijim poglavljima i onda na kraju ilustrovana kroz primere u okviru ovog poglavlja. Proces koji je uspešno izmenjen nekom od navedenih adaptacija razlikuje se od prvobitnog procesa, kao što je to opisano u primeru svakoj pokrenutoj instanci procesa dodeljen je *ProcessHandler* u okviru koga se proces menja ukoliko je to moguće, ukoliko nije, on nastavlja izvršavanje na osnovu modela procesa po kome je i pokrenut. Korišćenjem implementacionog rešenja imamo uvid u proces i njegovo stanje, kao i mogućnost vizualizacije procesa i elemenata koji se izvršavaju u okviru njega.

Na osnovu predstavljenog, opisane su prednosti i mane, kao i različite namene ovih sistema. *NewWave* donosi novi pogled i pristup poslovnim procesima kroz dizajnom podržanu *run-time* modifikaciju poslovnih procesa, kao i njihovu introspekciju. U ovom radu opisan je model sistema koji podržava fleksibilno upravljanje poslovnim procesima, kao i model koji je poslužio kao podrška za kreiranje radnog okvira za adaptacije. U radu je priložena i jedna implementacija predloženog rešenja koja realizuje zadate ciljeve, jer fokus ovog rada je omogućavanje adaptabilnosti, gde su neki drugi ciljevi kao što je perzistiranje, vizuelno kreiranje radnih tokova i drugi, koji se naslanjaju na ovo rešenje

realizovani ili u postupku realizacije o čemu je više navedeno u zaključku.

Zaključak

Predmet istraživanja disertacije pripada oblasti inženjerstvu poslovnih procesa u sklopu poslovnih informacionih sistema.

Poslovni procesi imaju direktan uticaj na privlačnost proizvoda i usluga, korisničko iskustvo, kao i na prihod u slučaju korporacija, samim tim pripadaju oblasti koja je izuzetno dinamična. Međutim, tradicionalni pristup razvoju softvera u kome se dosta vremena troši u diskutovanju i prevođenju poslovnih zahteva u tehničku specifikaciju i dalju implementaciju nije pogodan za ovu oblast zbog njene dinamičnosti.

WFMS (eng. *Workflow Management Systems*) predstavljaju vrstu informacionih sistema koji omogućavaju direktnu vezu između zadatka/posla nekog zaposlenog i računarskog programa, kreiranih sa idejom da korisniku olakša izvršavanje njegovih zadataka. Međutim dizajniranje i implementiranje ovakvih tipova sistema je kompleksan zadatak jer se velika većina oslanja na mogućnost dekompozicije poslovnih funkcija kao sekvence akcija, koje kad se izvršavaju kontrolisanim redosledom usmeravaju sistem ka uspešnom ishodu. Tokom sedamdesetih i osamdesetih godina prošlog veka, modelovanje podataka je bila početna aktivnost za kreiranje informacionog sistema, jer je primarni akcenat bio na podacima-upravljanom pristupu. Fokus informacionih tehnologija je u potpunosti bio posvećen podacima, odnosno skladištenju, čitanju i prezentovanju informacija. Informacioni sistemi koji su zasnovani na WFMS-u, odnosno sistemima koji podržavaju upravljanje poslovnih procesima nazivamo sistemima radnog toka - *workflow* sistemima. Jedan od većih problema je taj što je domen WFMS-a danas u potpunosti fragmentisan, odnosno za svaku svrhu postoje različita rešenja sa određenim specijalizacijama.

Način na koji su sistemi kreirani doveo je do toga da sama dinamika sistema - poslovni procesi koji se izvršavaju u organizacijama, budu zanemareni. Ovo je imalo za posledicu da je poslovna logika, koju informacioni sistem primenjuje transformišući uočene domenske podatke kako bi pratio izvršenje radnih operacija, bila razuđena u aplikacijama informacionih sistema. Ovakva implementacija rezultuje u informacionim sistemima koji dobro i detaljno modeluju domenske podatke, ali malo ili nimalo pažnje posvećuju sagledavanju i modelovanju procesa, i posledično, znatno otežavaju prilagođavanje sistema promenama koje nastaju u domenu samog poslovnog procesa.

Jedna od najznačajnijih aktivnosti u savremenim poslovnim informacionim sistema, odnosno mogućnostima koje WFMS treba da pruže jeste prilagođavanje promenama

u poslovnim procesima. Međutim, na osnovu prethodno izloženog u okviru ove disertacije primećuje se da je primena ove aktivnosti u kontekstu savremenih poslovnih procesno-orijentisanih sistema ograničena. Ograničena je zbog nedostatka konkretne paradigme koa bi propisala način primene ove aktivnosti, kao i ograničenjima načina implementacije procesnim sistemima, koji u najvećem broju slučajeva prave striktnu podelu između procesnog modela - modela procesa i njegovih instanci. Na osnovu navedenog ova teza je imala zadatak da adresira dva ključna istraživačka problema:

- Kako kreirati tehničko rešenje koje će pružati podršku za adaptabilnost procesa u procesno-orijentisanim poslovnim sistemima?
- Kako i u kojoj meri primeniti adatapabilnost na tekućim procesima u procesno-orijentisanim posovnim sistemima?

Na osnovu predstavljenog istraživanja u okviru ove disertacije, dati su odgovori na ova pitanja i izvedeni kao rezultat i zaključak ove disertacije, u okviru koje je kreirano tehničko rešenje (*New Wave* procesna platforma) i radni okvir za razvoj alata za izvršavanje poslovnih procesa koje je generalizovan ali i adaptabilan, kao i dat je postupak adaptacije procesa i podrška na njegovo sprovođenje kroz radni okvir.

Uvodna razmatranja i objašnjenje zašto postoji potreba za jednim ovakvim istraživanjem data je u prvom poglavlju. Prvo poglavlje takođe uvodi i opisuje poslovne procese, njihove modele kao i njihovu ulogu u poslovnim informacionim sistemima. Problemi koji su uočeni, kao i predložena rešenja za njihovo rešavanje su opisani kroz ciljeve disertacije u ovom poglavlju.

Sistematizacija i analiza dostupne literature iz predmete oblasti urađena je u drugom poglavlju. Ovo poglavlje je podeljeno u nekoliko celina. U prvoj celini se analizira literatura iz oblasti poslovnih procesa i *Petrijevih* mreža, gde su definisani koncepti *Petrijeve* mreže i poslovnih procesa. Takođe opisan je i analiziran BPM (eng. *Business Process Management*), kao polazna tačka u upravljanju poslovnim procesima. Druga celina se bavi BPMN (eng. *Business Process Modeling Notation*), odnosno grafičkom notacijom, koja se koristi kao standardni jezik za modelovanje poslovnih procesa, kao i postignućem BPMN-a da smanji jaz u fragmentaciji između postojećih alata za modelovanje poslovnih procesa i notacija. Veliki uspeh BPMN-a je što je opšteprihvaćen kako u industriji tako i u akademiji. Treća celina se bavi interakcijama i adaptacijama u poslovnim procesima, zajedno sa softverskim sistemima koji koordinišu aktivnosti koje su uključene u poslovne procese, odnosno BPMS (eng. *Business Process Management Systems*). Upravljanje radnim tokovima je tradicionalno imperativno i opšteprihaćeno, odnosno sistem je taj kontroliše izvršavanje definisanih poslovnih procesa u skladu sa modelima kojim su ti procesi definisani. Jedna od glavnih stvari u pomenutom pristupu je koncept na kome su oni zasnovani, a to je separacija dve glavne aktivnosti, odnosno vreme kreiranja procesa i vreme izvršavanja procesa, gde je proces kada je jednom pokrenut i čija je instanca kreirana odvojen (eng. *detached*) od modela na kome je

zasnovan i na osnovu koga je kreiran i ne postoji njihova međusobno-zavisna prizma. Prevazilaženje ovog problema značilo bi da su dozvoljene dinamičke adaptacije, odnosno promene u strukturi procesa koji se trenutno izvršava, a te promene predstavljaju nove zahteve na koje proces treba da odgovori, odnosno promene koje nisu očekivane u vremenu kreiranja procesa, a koje su nastale kao odgovor na novonastalu prirodu dinamičkog tržišta. Opisana su različita rešenja i kroz tu prizmu različiti pristupi koji su korišćeni u okviru ovih sistema kako bi dalje približili adaptabilnost u poslovnim procesima trenutno vodećim, tradicionalnim, ali industrijski prihvaćenim sistemima za upravljanje poslovnim procesima.

Treće poglavlje opisuje i daje uvid u dinamička programska okruženja i opisuje širu sliku pogodnosti zašto je ovakvo jedno okruženje odabrano za realizaciju zadatih ciljeva, konkretno kako iskoristiti postojeće okruženje u realizaciji interpretaciono zasnovanog pristupa, odnosno kako je moguće izvršiti dinamičke adaptacije u pristupu u kome nema striktne separacije između vremena kreiranja i izvršavanja procesa. U okviru ovog okruženja instance tekućeg radnog toka kontrolišu se interpretiranjem njegovog modela, te je na osnovu skupa datih pravila moguće vršiti adaptacije nad modelom i instancama radnog toka. Predloženo rešenje je zasnovano na komponentama i komponente mogu da se prilagođavaju traženim zahtevima, kao da se te komponente proizvoljno isključuju i/ili dodaju po potrebi, što je i bila jedna od ideja tokom koncipiranja ovog rešenja. Komponent bazirani razvoj ovog rešenja, omogućio je i olakšao kreiranje različitih konfiguracija koje se mogu isporučiti koristeći ovo rešenje, tačnije moguće je distribuirati samo npr. klijentski deo (korisnički interfejs koji koriste klijenti) konfigurisan u okviru Pharo okruženja, dok bi se konfiguracija ostatka NewWave platforme, nalazila na nekom serverskom okruženju.

Treće poglavlje i radovi opisani u trećem poglavlju, kao i prateća rešenja koja su nastala na osnovu opisanih radova poslužili su kao inspiracija za kreiranje ovog istraživanja. Tokom analiziranja literature primećeni su neki nedostaci kao i slobodan prostor za realizaciju rešenja kao što je NewWave. Inicijalna zamisao NewWave rešenja opisana je i predložena u sklopu rada [94], te i prihvaćena kao originalna softverska publikacija. Disertacija se naslanja i proširuje datu ideju i fokusira se na dinamička okruženja i na adaptabilnost u poslovnim procesima, te benefite koje njihova sinergija može da pruži i šta kroz njihovu pravilnu upotrebu može da se postigne.

Osnovni doprinos disertacije nalazi se u četvrtom i petom poglavlju. U četvrtom poglavlju opisana je arhitektura i formalno tehničko rešenje kao i opis radnog okvira za izvršavanje poslovnih procesa. Doprinos u četvrtom poglavlju je sledeće:

- Data je arhitektura rešenja koja podržava dinamičke promene
- Opisano je dinamičko instanciranje modela i njegovo izvršavanje
- Opisane su komponente sistema i njihova međusobna povezanost
- Podrška za različite konfiguracije sistema

- Podrška za nove komponente koji se mogu dodavati u postojeću konfiguraciju
- Podrška za integraciju elemenata drugih rešenja van postojećeg ekosistema
- Podrška za udaljeno izvršavanje ugrađene veb aplikacije

U petom poglavlju dat je i opisan formalan model za podršku adaptacija u okviru predloženog rešenja, kao i verifikacija predložene platforme obavljena je:

- Kreiranjem radnog toka,
- instanciranjem radnog toka,
- primenjivanjem proširenja,
- primenjivanjem adaptacije zamene i
- primenjivanjem zamene redosleda zadataka.

Opisani postupci i predložena rešenja u četvrtom i petom poglavlju prikazana su kroz ilustrativne primere koji prikazuju ponašanje *NewWave* koristeći prethodno utvrđena i opisana pravila. Tako da su opisani postupci i dati primeri za adaptacije proširenja, zamene, kao i zamene redosleda zadataka. Za date primere prikazani su validni, kao i nevalidni slučajevi, odnosno slučajevi kada je moguće i dozvoljeno izvršiti adaptaciju, te takođe i slučajevi kada nije dozvoljeno i nije moguće izvršiti adaptaciju.

Verifikacija je izvršena na osnovu uputstava datih u petom poglavlju, pridržavanjem pravila i opisa pravila koju platforma podržava. Pored inicijalne zamisli i doprinosa u akademiji predloženo rešenje *NewWave* našlo je primenu i u industriji gde kompanija *ApptiveGrid*[2] je uvrstila razvijeno rešenje u svoj portfolio i radi na njenom daljem unapređenju za svoje potrebe po MIT licenci pod kojom je *NewWave* i kreiran. *ApptiveGrid* se bavi radni tokovima, kao i *Low-Code* rešenjima, odnosno rešenjima koja omogućavaju/pružaju razvojna okruženja pomoću kojih se mogu kreirati različite funkcionalne aplikacije koristeći grafički korisničke interfejse. *NewWave* je otvorio dosta prostora za dalja istraživanja u ovoj oblasti, te je do sada poslužio kao osnova za nekoliko master radova [100–102]. Master radovi su se oslonili na postojeće rešenje, da kroz različite dodatke ostvare svoje potrebe koji su bili ciljevi njihovih radova, integracija je postignuta korišćenjem fleksibilne arhitekture i podrške za dodatke u okviru *NewWave*-a, tako da izloženo predstavlja još jednu validaciju opisanog rešenja.

Kompleksne adaptacije koje predstavljaju kombinaciju prethodno opisanih tipova adaptacije, mogu se ostvariti sekvencijanom primenom pojedinačnih izmena. Zahvaljujući činjenici da se adaptacije obavljaju u režimu suspendovanog izvršavanja, kada se u NewWave engine-u blokira propagacija aktivacija, otvara se mogućnost da

[2] https://en.apptivegrid.de

se veoma kompleksne izmene, koje predstavljaju kombinacije tipova izmena, primene iterativnim ponavljanjem postupka. Kako se ulaskom u suspenziju trenutno stanje procesa može serijalizovati, to ostavlja mogućnost da se, u slučaju da neki od koraka adaptacije da negativan ishod, stanje sistema vrati u početno i nastavi izvršavanje nemdifikovanog modela. Ipak, neophodno je sprovesti dodatne analize kako bi se ovakav režim, sličan transakcionom režimu baza podataka, potvrdio kao validan u većem broju slučajeva i u kompleknsijim adaptacijama.

Pravci daljeg istraživanja i razvoja mogu biti sledeći:

- Distribuirani rad više međusobno povezanih *NewWave* platformi,
- Podrška za menjanje/adaptiranje celih grana poslovnog procesa,
- Dodavanje mogućnosti za rad rešenja u Cloud-u [103],
- Isporučivanje različitih konfiguracija aplikacije koristeći Docker kontejnere [104],
- Integracija sa *Low-Code* rešenjima, u domenu poslovno-orijentisanih procesnih sistema, tako da je NewWave zadužen za izvršavanje i manipulaciju poslovnih procesa.

Bibliografija

[1] M. Dumas, W. M. P. van der Aalst, and A. H. M. ter Hofstede, eds., *Process-Aware Information Systems: Bridging People and Software Through Process Technology*. Wiley, 2005.

[2] J. M. Marlon Dumas, Marcello La Rosa and H. A. Reijers, *Fundamentals of Business Process Management*. Springer-Verlag, 2018.

[3] W. W. Royce, "Managing the development of large software systems," in *Tutorial: Software Engineering Project Management* (R. Thayer, ed.), pp. 118–127, Washington: IEEE Computer Society, 1987. fca.

[4] S. Kent, "Model driven engineering," in *Integrated formal methods*, pp. 286–298, Springer, 2002.

[5] C. Atkinson and T. Kuhne, "Model-driven development: a metamodeling foundation," *IEEE Software*, vol. 20, no. 5, pp. 36–41, 2003.

[6] S. Khoshafian and M. Buckiewicz, *Introduction to Groupware, Workflow and Workgroup Computing*. John Wiley & Sons, 1995.

[7] D. Manolescu, *Micro-workflow: a workflow architecture supporting compositional object-oriented software development*. PhD thesis, University of Urbana-Champaign, 2000.

[8] "Business process model and notation - object management group, version 2.0.2."

[9] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (xml) 1.0 (fifth edition)." W3C Recommendation, 2008.

[10] S. Alvès, A. P. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. G. Ford, Y. Y. Goland, A. Guzar, N. Kartha, C. K. Liu, R. Khalaf, D. König, M. Marin, V. Mehta, S. Thatte, D. Rijn, P. Yendluri, and A. Yiu, "Web services business process execution language version 2.0 (oasis standard)," 2007.

[11] D. C. C. Peixoto, V. A. Batista, A. P. Atayde, E. P. Borges, R. F. Resende, and C. I. P. da Silva e Pádua, "A comparison of bpmn and uml 2 . 0 activity diagrams," 2008.

[12] W. van der Aalst and K. van Hee, *Workflow Management: Models, Methods, and Systems*. The MIT Press Cambridge, Massachusetts London, England, 2002.

[13] D. Hollingsworth, "The workflow reference model. workflow management coalition," *WFMC org*, 1995.

[14] M. S. Puccini, *Executable Models for Extensible Workflow Engines*. PhD thesis, Universidad de los Andes, 2011.

[15] S. Richly, S. Götz, U. Assmann, and S. Schmidt, "Role-based multi-purposeworkflow engine architecture," 2016.

[16] M. Klein, C. Dellarocas, and A. Bernstein, "Introduction to the special issue on adaptive workflow systems," *Computer Supported Cooperative Work*, vol. 9, no. 3-4, pp. 265–267, 2000.

[17] W. Aalst and T. Basten, "Inheritance of workflows: An approach to tackling problems related to change," *Theoretical Computer Science*, vol. 270, pp. 125–203, 01 2002.

[18] P. Dadam and M. Reichert, "The adept project: a decade of research and development for robust and flexible process support," *Computer Science - Research and Development*, vol. 23, no. 2, pp. 81–97, 2009.

[19] M. Reichert, S. Rinderle-Ma, and P. Dadam, *Flexibility in Process-Aware Information Systems*, pp. 115–135. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[20] M. Hammer and J. Champy, *Reengineering the Corporation: A Manifesto for Business Revolution*. Nicholas Brealey, 2001.

[21] T. Davenport, *Process Innovation: Reengineering Work Through Information Technology*. Harvard Business Review Press, 1993.

[22] M. Weske, *Business Process Management - Concepts, Languages, Architectures*. Springer, 2007.

[23] M. Kirchmer, *High Performance through Business Process Management - Strategy Execution in a Digital World*. 03 2017.

[24] A.-W. Scheer, *Business process engineering: reference models for industrial enterprises*. Springer Science & Business Media, 2012.

[25] A.-W. Scheer, *ARIS-business process frameworks*. Springer Science & Business Media, 1999.

[26] M. Kirchmer, *Business Process Oriented Implementation of Standard Software: how to achieve competitive advantage efficiently and effectively: with 107 figures*. Springer Science & Business Media, 1999.

[27] A. Spanyi, *Business Process Management is a Team Sport: Play it to Win!* Anclote Press, 2003.

[28] P. Harmon, *Business Process Change: A Manager's Guide to Improving, Redesigning, and Automating Processes*. The Morgan Kaufmann Series in Data Management Systems Series, Morgan Kaufmann Publishers, 2003.

[29] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[30] C. A. Petri, *Kommunikation mit Automaten*. PhD thesis, Universität Hamburg, 1962.

[31] W. M. P. van der Aalst, "The application of petri nets to workflow management," *Journal of Circuits, Systems, and Computers*, vol. 8, pp. 21–66, 1998.

[32] W. Aalst, van der, *Three good reasons for using a Petri-net-based workflow management system*, pp. 161–182. The Kluwer International Series in Engineering and Computer, Netherlands: Kluwer Academic Publishers, 1998.

[33] K. Jensen, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Coloured Petri Nets, Springer Berlin Heidelberg, 1992.

[34] W. Aalst, K. Hee, A. Ter, N. Sidorova, H. Verbeek, M. Voorhoeve, and M. Wynn, "Soundness of workflow nets: Classification, decidability, and analysis," *Formal Asp. Comput.*, vol. 23, pp. 333–363, 05 2011.

[35] R. David and H. Alla, "On hybrid petri nets," *Discrete Event Dynamic Systems*, vol. 11, no. 1, pp. 9–40, 2001.

[36] J. Cardoso, R. Valette, and D. Dubois, "Fuzzy petri nets: An overview," *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 4866–4871, 1996. 13th World Congress of IFAC, 1996, San Francisco USA, 30 June - 5 July.

[37] M. A. Marsan, "Stochastic petri nets: An elementary introduction," in *Advances in Petri Nets 1989* (G. Rozenberg, ed.), (Berlin, Heidelberg), pp. 1–29, Springer Berlin Heidelberg, 1990.

[38] G. Decker and A. Barros, "Interaction modeling using bpmn," pp. 208–219, 09 2007.

[39] A. Barros, M. Dumas, and A. Ter, "Service interaction patterns.," pp. 302–318, 01 2005.

[40] L. Thom, C. Iochpe, M. Reichert, B. Weber, G. Nascimento, and C. Chiao, "On the support of activity patterns in prowap: Case studies, formal semantics, tool support," 04 2008.

[41] N. Kavantzas, "Web services choreography description language version 1.0, w3c candidate recommendation. technical report, november 2005," 2004.

[42] A. Barros, M. Dumas, and P. Oaks, "A critical overview of the web services choreography description language," 2005.

[43] Z. Xiangpeng, Y. Hongli, and Q. Zongyan, "Towards the formal model and verification of web service choreography description language," in *International Workshop on Web Services and Formal Methods*, pp. 273–287, Springer, 2006.

[44] G. J. Holzmann, "The spin model checker - primer and reference manual," 2003.

[45] J. M. Zaha, A. P. Barros, M. Dumas, and A. H. M. ter Hofstede, "Let's dance: A language for service behavior modeling," in *OTM Conferences*, 2006.

[46] G. Decker, M. Kirov, J. M. Zaha, and M. Dumas, "Maestro for let's dance: An environment for modeling service interactions," in *BPM Demos*, 2006.

[47] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, *et al.*, "Business process execution language for web services," 2003.

[48] F. Leymann, *Web Services Flow Language (WSFL 1.0)*. 05 2001.

[49] S. Thatte, "Xlang: Web services for business process design," *Microsoft Corporation*, vol. 2001, p. 14, 2001.

[50] R. Davis, *ARIS design platform: Advanced process modelling and administration*. 01 2008.

[51] A. Tsai, J. Wang, W. Tepfenhart, and D. Rosea, "Epc workflow model to wifa model conversion," vol. 4, pp. 2758 – 2763, 11 2006.

[52] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distributed and Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003.

[53] D. M. Ferreira and J. Pinto Ferreira, "Developing a reusable workflow engine," *Journal of Systems Architecture*, vol. 50, no. 6, pp. 309 – 324, 2004.

[54] K. Whittingham, *OpenWater White Paper*. IBM Thomas J. Watson Research Division, 1999.

[55] N. C. Narendra, "Flexible support and management of adaptive workflow processes," *Information Systems Frontiers*, vol. 6, no. 3, pp. 247–262, 2004.

[56] N. C. Narendra, "Adaptive workflow management—an integrated approach and system architecture," in *Proceedings of the 2000 ACM symposium on Applied computing-Volume 2*, pp. 858–865, 2000.

[57] A. Freßmann, K. Maximini, R. Maximini, and T. Sauer, "Collaborative agent-based knowledge support for empirical and knowledge-intense processes," in *German Conference on Multiagent System Technologies*, pp. 235–236, Springer, 2005.

[58] B. Estrada-Torres, P. H. P. Richetti, A. Del-Río-Ortega, F. A. Baião, M. Resinas, F. M. Santoro, and A. Ruiz-Cortés, "Measuring performance in knowledge-intensive processes," vol. 19, feb 2019.

[59] K. Schwaber, "Controlled chaos: Living on the edge," *American Programmer*, vol. 9, pp. 10–16, 1996.

[60] D. W. Aha and H. Muñoz-Avila, "Introduction: Interactive case-based reasoning," *Applied Intelligence*, vol. 14, no. 1, p. 7, 2001.

[61] B. Weber, W. Wild, and R. Breu, "Cbrflow: Enabling adaptive workflow management through conversational case-based reasoning," in *Advances in Case-Based Reasoning* (P. Funk and P. A. González Calero, eds.), (Berlin, Heidelberg), pp. 434–448, Springer Berlin Heidelberg, 2004.

[62] "Dealing with workflow change : identification of issues and solutions," *Computer Systems Science and Engineering*, vol. 15, no. 5, pp. 267–276, 2000.

[63] S. Jablonski and C. Bussler, *Workflow Management: Modeling Concepts, Architecture, and Implementation*. 01 1996.

[64] F. Leymann and D. Roller, "Production workflow - concepts and techniques," 01 2001.

[65] N. Gottschalk, "Design and control of workflow processes: Business process management for the service industry," *Design and Control of Workflow Processes*, 2003.

[66] OMG, "OMG Unified Modeling Language (OMG UML), Infrastructure, Version 2.4.1," August 2011.

[67] M. Chinosi and A. Trombetta, "Bpmn: An introduction to the standard.," *Comput. Stand. Interfaces*, vol. 34, no. 1, pp. 124–134, 2012.

[68] G. Aagesen and J. Krogstie, "BPMN 2.0 for modeling business processes," in *Handbook on Business Process Management 1, Introduction, Methods, and Information Systems, 2nd Ed*, pp. 219–250, Springer, 2015.

[69] W. van der Aalst and A. ter Hofstede, "Yawl: yet another workflow language," *Information Systems*, vol. 30, no. 4, pp. 245–275, 2005.

[70] A. Hofstede and W. Aalst, "Workflow patterns: On the expressive power of (petri-net-based) workflow languages.," 2002.

[71] N. Russell and A. ter Hofstede, "Surmounting bpm challenges: the yawl story," *Software-Intensive Cyber-Physical Systems*, vol. 23, no. 2, pp. 67–79, 2009.

[72] M. Reichert and P. Dadam, "Adeptflex - supporting dynamic changes of workflows without losing control," Technical Report UIB-1997-07, University of Ulm, Ulm, April 1997.

[73] J. Koenig, "Jboss jbpm (whitepaper)," 2004.

[74] W. Aalst, van der, M. Pesic, and H. Schonenberg, "Declarative workflows : balancing between flexibility and support," *Computer Science: Research and Development*, vol. 23, no. 2, pp. 99–113, 2009.

[75] W. M. P. van der Aalst and M. Pesic, "Decserflow: Towards a truly declarative service flow language," in *Web Services and Formal Methods* (M. Bravetti, M. Núñez, and G. Zavattaro, eds.), (Berlin, Heidelberg), pp. 1–23, Springer Berlin Heidelberg, 2006.

[76] M. Pesic, H. Schonenberg, and W. M. van der Aalst, "Declare: Full support for loosely-structured processes," in *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, pp. 287–287, 2007.

[77] E. Clarke, E. Peled, O. Grumberg, D. Peled, and EBSCO., *Model Checking*. The Cyber-Physical Systems Series, MIT Press, 1999.

[78] D. Giannakopoulou and K. Havelund, "Automata-based verification of temporal properties on running programs," in *Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001)*, pp. 412–416, 2001.

[79] A. Lanz, M. Reichert, and P. Dadam, "Robust and flexible error handling in the aristaflow bpm suite," in *Information Systems Evolution* (P. Soffer and E. Proper, eds.), (Berlin, Heidelberg), pp. 174–189, Springer Berlin Heidelberg, 2011.

[80] B. Weber, H. A. Reijers, S. Zugal, and W. Wild, "The declarative approach to business process execution: An empirical test," in *Advanced Information Systems Engineering* (P. van Eck, J. Gordijn, and R. Wieringa, eds.), (Berlin, Heidelberg), pp. 470–485, Springer Berlin Heidelberg, 2009.

[81] W. M. van der Aalst, M. Weske, and D. Grünbauer, "Case handling: a new paradigm for business process support," *Data and Knowledge Engineering*, vol. 53, no. 2, pp. 129–162, 2005.

[82] S. Kaplar, M. Zarić, and G. Milosavljević, "Newwave workflow engine," in *International Workshop on Smalltalk Technologies Cologne*, 2019.

[83] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[84] M. Akşit, *Software Architectures and Component Technology*. 01 2002.

[85] W. M. P. van der Aalst, T. Basten, H. M. W. Verbeek, P. A. C. Verkoulen, and M. Voorhoeve, *Adaptive Workflow*, pp. 63–70. Dordrecht: Springer Netherlands, 2000.

[86] A. Goldberg and D. Robson, *Smalltalk 80: the Language and its Implementation*. Reading, Mass.: Addison Wesley, May 1983.

[87] S. Ducasse, D. Zagidulin, N. Hess, D. C. O. written by A. Black, S. Ducasse, O. Nierstrasz, D. P. with D. Cassou, and M. Denker, *Pharo by Example 5*. Square Bracket Associates, 2017.

[88] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč, "Foundations of json schema," in *Proceedings of the 25th International Conference on World Wide Web*, pp. 263–273, 2016.

[89] D. Cassou, S. Ducasse, L. Fabresse, J. Fabry, and S. Van Caekenberghe, *Enterprise Pharo: a Web Perspective*. Square Bracket Associates, 2015.

[90] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol–http/1.1," 1999.

[91] A. Bergel, D. Cassou, S. Ducasse, and J. Laval, *Deep Into Pharo*. Square Bracket Associates, 2013.

[92] E. H. Bersoff, "Elements of software configuration management," *IEEE Transactions on Software Engineering*, no. 1, pp. 79–87, 1984.

[93] J. Estublier, "Software configuration management: A roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, (New York, NY, USA), p. 279–289, Association for Computing Machinery, 2000.

[94] S. Kaplar, M. Zarić, and S. Ducasse, "Newwave: Workflow engine," *Science of Computer Programming*, vol. 203, p. 102581, 2021.

[95] F. Casati, S. Ceri, B. Pernici, and G. Pozzi, "Workflow evolution," *Data and Knowledge Engineering*, vol. 24, no. 3, pp. 211–238, 1998. ER '96.

[96] C. Ellis, K. Keddara, and G. Rozenberg, "Dynamic change within workflow systems," COCS '95, (New York, NY, USA), p. 10–21, Association for Computing Machinery, 1995.

[97] S. Rinderle, M. Reichert, and P. Dadam, "Correctness criteria for dynamic changes in workflow systems: A survey," *Data Knowl. Eng.*, vol. 50, p. 9–34, July 2004.

[98] C. W. Guenther, M. Reichert, and W. M. P. van der Aalst, "Supporting flexible processes with adaptive workflow and case handling," in *2008 IEEE 17th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 229–234, 2008.

[99] B. Weber, S. Rinderle, and M. Reichert, "Change patterns and change support features in process-aware information systems," in *Proceedings of the 19th International Conference on Advanced Information Systems Engineering*, CAiSE'07, (Berlin, Heidelberg), p. 574–588, Springer-Verlag, 2007.

[100] N. Miladinović, "Generisanje formi za poslovne procese koriŠĆenjem magritte razvojnog okvira," Dec 2019.

[101] M. Marković, "Integracija newwave workflow engine-a i openponk alata," Mar 2021.

[102] N. Gavrilović, "Dodela radnih zadataka u poslovnom procesu upotrebom pharo programskog jezika," Mar 2021.

[103] L. Qian, Z. Luo, Y. Du, and L. Guo, "Cloud computing: An overview," in *IEEE international conference on cloud computing*, pp. 626–631, Springer, 2009.

[104] B. B. Rad, H. J. Bhatti, and M. Ahmadi, "An introduction to docker and analysis of its performance," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 17, no. 3, p. 228, 2017.

Indeks slika

Indeks listinga

Biografija

Sebastijan Kaplar rođen je 1990. godine u Bijeljini. Završio je srednju školu u Bijeljini 2009. godine. Fakultet Tehničkih Nauka u Novom Sadu je upisao 2009. godine. Ispunio je sve obaveze i položio je sve ispite predviđene studijskim programom i diplomirao 2013. godine. Iste godine je upisao master akademske studije i nakon završetka svih obaveza na studijskom programu odbranio master rad pod nazivom *Podsistem za administraciju korisničkih prava u okviru KROKI alata* 13.10.2014. godine. Od 2014. godine je student doktorskih akademskih studija na Fakultetu tehničkih nauka. Kao student doktorskih akademskih studija bavi se razvojem softverskih sistema zasnovanim na modelima, poslovnim procesima i tehnologijama radnog toka. Tokom svoje akademske karijere autor je i koautor više naučnih radova predstavljenih na domaćim i međunarodnim konferencijama kao i radova objavljenih u domaćim i međunarodnim časopisima. Učestvovao je na više letnjih škola vezanih za Big Data i matematičko modelovanje. Pored svog akademskog rada bio je aktivan u više projekata vezanih za industriju i učestvovao je u izradi softverskih rešenja koja se danas aktivno koriste u privredi. Tokom rada na svojoj disertaciji boravio je u Francuskom nacionalnom institutu za istraživanje INRIA (Institut national de recherche en sciences et technologies du numérique) u Lilu, Francuska. Rezultati istraživanja ove disertacije publikovani su u dva naučna rada, takođe naučno-istraživački rad na ovoj temi inspirisao je i služio kao podloga za izradu tri master rada. Jedan je od autora knjige *Pharo 9 by Example*.

Овај Образац чини саставни део докторске дисертације, односно докторског уметничког пројекта који се брани на Универзитету у Новом Саду. Попуњен Образац укоричити иза текста докторске дисертације, односно докторског уметничког пројекта.

План третмана података

Назив пројекта/истраживања
Прошириво процесно окружење са подршком за управљање адаптабилним пословним процесима / Extensible Workflow Engine With Support for Adaptable Business Process Management
Назив институције/институција у оквиру којих се спроводи истраживање
а) Факултет техничких наука, Универзитет у Новом Саду
Назив програма у оквиру ког се реализује истраживање
Рачунарство и аутоматика – докторска дисертација
1. Опис података
1.1 Врста студије *Укратко описати тип студије у оквиру које се подаци прикупљају* _____ _____ _____ 1.2 Врсте података а) квантитативни **б) квалитативни** 1.3. Начин прикупљања података а) анкете, упитници, тестови б) клиничке процене, медицински записи, електронски здравствени записи

в) генотипови: навести врсту _____

г) административни подаци: навести врсту _____

д) узорци ткива: навести врсту_____

ђ) снимци, фотографије: навести врсту_____

е) текст, навести врсту: **Актуелна литература у области истраживања**

ж) мапа, навести врсту _____

з) остало: описати _____

1.3 Формат података, употребљене скале, количина података

1.3.1 Употребљени софтвер и формат датотеке:

a) Excel фајл, датотека _____

b) SPSS фајл, датотека _____

c) PDF фајл, датотека _____

d) Текст фајл, датотека _____

e) JPG фајл, датотека _____

f) Остало, датотека _____

1.3.2. Број записа (код квантитативних података)

а) број варијабли _____

б) број мерења (испитаника, процена, снимака и сл.) _____

1.3.3. Поновљена мерења

а) да

б) не

Уколико је одговор да, одговорити на следећа питања:

а) временски размак између поновљених мера је _____

б) варијабле које се више пута мере односе се на _____

в) нове верзије фајлова који садрже поновљена мерења су именоване као _____

Напомене: _____

Да ли формати и софтвер омогућавају дељење и дугорочну валидност података?

а) Да

б) Не

Ако је одговор не, образложити _____

_____

2. Прикупљање података

2.1 Методологија за прикупљање/генерисање података

2.1.1. У оквиру ког истраживачког нацрта су подаци прикупљени?

а) експеримент, навести тип _____

б) корелационо истраживање, навести тип _____

ц) анализа текста, навести тип: **Анализа доступне литературе**

д) остало, навести шта _____

2.1.2 Навести врсте мерних инструмената или стандарде података специфичних за одређену научну дисциплину (ако постоје).

_____

_____

2.2 Квалитет података и стандарди

2.2.1. Третман недостајућих података

а) Да ли матрица садржи недостајуће податке? Да **Не**

Ако је одговор да, одговорити на следећа питања:

а) Колики је број недостајућих података? _____

б) Да ли се кориснику матрице препоручује замена недостајућих података? Да Не

в) Ако је одговор да, навести сугестије за третман замене недостајућих података

_____

2.2.2. На који начин је контролисан квалитет података? Описати

_____

_____

2.2.3. На који начин је извршена контрола уноса података у матрицу?

_____

_____

3. Третман података и пратећа документација

3.1. Третман и чување података

3.1.1. Подаци ће бити депоновани у _____ репозиторијум.

3.1.2. URL адреса _____

3.1.3. DOI _____

3.1.4. Да ли ће подаци бити у отвореном приступу?

а) Да

б) Да, али после ембарга који ће трајати до _____

в) Не

Ако је одговор не, навести разлог _____

3.1.5. Подаци неће бити депоновани у репозиторијум, али ће бити чувани.

Образложење

_____

_____

3.2 Метаподаци и документација података

3.2.1. Који стандард за метаподатке ће бити примењен? _____

3.2.1. Навести метаподатке на основу којих су подаци депоновани у репозиторијум.

_____

_____

Ако је потребно, навести методе које се користе за преузимање података, аналитичке и процедуралне информације, њихово кодирање, детаљне описе варијабли, записа итд.

_____

_____

_____

_____

3.3 Стратегија и стандарди за чување података

3.3.1. До ког периода ће подаци бити чувани у репозиторијуму? _____

3.3.2. Да ли ће подаци бити депоновани под шифром? Да Не

3.3.3. Да ли ће шифра бити доступна одређеном кругу истраживача? Да Не

3.3.4. Да ли се подаци морају уклонити из отвореног приступа после извесног времена?

Да Не

Образложити

_____

_____

4. Безбедност података и заштита поверљивих информација

Овај одељак МОРА бити попуњен ако ваши подаци укључују личне податке који се односе на учеснике у истраживању. За друга истраживања треба такође размотрити заштиту и сигурност података.

4.1 Формални стандарди за сигурност информација/података

Истраживачи који спроводе испитивања с људима морају да се придржавају Закона о заштити података о личности *(https://www.paragraf.rs/propisi/zakon_o_zastiti_podataka_o_licnosti.html)* и одговарајућег институционалног кодекса о академском интегритету.

4.1.2. Да ли је истраживање одобрено од стране етичке комисије? Да **Не**

Ако је одговор Да, навести датум и назив етичке комисије која је одобрила истраживање

_____

4.1.2. Да ли подаци укључују личне податке учесника у истраживању? Да **Не**

Ако је одговор да, наведите на који начин сте осигурали поверљивост и сигурност информација везаних за испитанике:

а) Подаци нису у отвореном приступу

б) Подаци су анонимизирани

ц) Остало, навести шта

_____

_____

5. Доступност података

5.1. Подаци ће бити

а) јавно доступни

б) доступни само уском кругу истраживача у одређеној научној области

ц) затворени

Ако су подаци доступни само уском кругу истраживача, навести под којим условима могу да их користе:

_____

_____

Ако су подаци доступни само уском кругу истраживача, навести на који начин могу приступити подацима:

_____

_____

5.4. Навести лиценцу под којом ће прикупљени подаци бити архивирани.

_____

6. Улоге и одговорност

6.1. Навести име и презиме и мејл адресу власника (аутора) података

Себастијан Каплар, sebastijan.kaplar@gmail.com

_____

6.2. Навести име и презиме и мејл адресу особе која одржава матрицу с подацима

_____

6.3. Навести име и презиме и мејл адресу особе која омогућује приступ подацима другим истраживачима

_____