УНИВЕРЗИТЕТ У НИШУ

ЕКОНОМСКИ ФАКУЛТЕТ
У НИШУ

Владимир Л. Милићевић

УНАПРЕЂЕЊЕ САВРЕМЕНИХ HELPDESK ПОСЛОВНИХ СИСТЕМА ПРИМЕНОМ НАПРЕДНИХ ИНТЕЛИГЕНТНИХ СОФТВЕРСКИХ АЛАТА

докторска дисертација

Ниш, 2015

УНИВЕРЗИТЕТ У НИШУ

ЕКОНОМСКИ ФАКУЛТЕТ
У НИШУ

Владимир Л. Милићевић

УНАПРЕЂЕЊЕ САВРЕМЕНИХ HELPDESK ПОСЛОВНИХ СИСТЕМА ПРИМЕНОМ НАПРЕДНИХ ИНТЕЛИГЕНТНИХ СОФТВЕРСКИХ АЛАТА

докторска дисертација

Ментор:

Проф. др Славољуб Миловановић, редовни професор

Ниш, 2015

UNIVERSITY OF NIŠ

FACULTY OF ECONOMICS
IN NIŠ

Vladimir L. Milićević

MODERN HELPDESK BUSINESS SYSTEMS IMPROVEMENT BY ADVANCED INTELLIGENT SOFTWARE TOOLS

doctoral dissertation

Mentor:

Ph.D Slavoljub Milovanović, full professor

Niš, 2015

Комисија:

_____

Ментор: Проф. др Славољуб Миловановић, редовни професор Економског факултета у Нишу

_____

Члан: Проф. др Раде Станкић, редовни професор Економског факултета у Београду

_____

Члан: Др Огњен Радовић, доцент Економског факултета у Нишу

Датум одбране:

_____

Посвећено:

Лазару, Петри, Кристини, Лазару, Исидори и Софији

НАУЧНИ ДОПРИНОС ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

У раду се проучава актуелно стање, могућности и ефекти коришћења HelpDesk система у компанијама из различитих области пословања са циљем сагледавања њихових недостатака.

Недостаци су прецизно сагледани при чему рад даје јасне смернице ка њиховом отклањању. Уочено је да актуелни приступи пројектовања и израде оваквих софтверских система имају своје предности и недостатке. Увођењем унифицираног приступа, који обухвата најбоље индивидуалне карактеристике постојећих, омогућен је развој аутоматизованог HelpDesk система, способног да сам проширује властиту експертизу.

Комбиновањем унифицираног аспектног приступа са моделским приступом нулте толеранције добија се иновативни приступ за развој широког спектра експертних система које карактерише највећи могући степен модуларности.

Такође, коришћењем наведених приступа омогућено је проширење и унапређење постојећих експертних система којима је сервисирано пословање бројних компанија из различитих области пословања.

THE SCIENTIFIC CONTRIBUTION OF THE DOCTORAL DISSERTATION

The paper examines the current state and the possibilities and effects of using HelpDesk system in companies operating in different business areas, with the aim to examine its disadvantages. Disadvantages are accurately analyzed and the dissertation gives clear guidance how to eliminate them.

It has been noticed that current design and development approaches have their own advantages and disadvantages. The application of the unified approach that combines the best individual characteristics of the existing approaches enables development of an automated HelpDesk system that is capable of expanding its own expertise.

By combining unified aspect approach with the zero tolerance model driven approach, a new innovative approach to the development of a wide range of expert systems is obtained. This innovative approach is characterized by the highest possible degree of modularity.

The application of the aforementioned approaches has also enabled expansion and improvement of the existing expert systems used for servicing dealings of numerous companies operating in various fields of business.

ИЗЈАВА МЕНТОРА О САГЛАСНОСТИ ЗА ПРЕДАЈУ УРАЂЕНЕ ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Овим изјављујем да сам сагласан да кандидат **ВЛАДИМИР МИЛИЋЕВИЋ** може да преда Реферату за последипломско образовање Факултета урађену докторску дисертацију под називом **УНАПРЕЂЕЊЕ САВРЕМЕНИХ HELPDESK ПОСЛОВНИХ СИСТЕМА ПРИМЕНОМ НАПРЕДНИХ ИНТЕЛИГЕНТНИХ СОФТВЕРСКИХ АЛАТА**, ради организације њене оцене и одбране.

Проф. др Славољуб Миловановић

STATEMENT OF MENTOR'S CONSENT FOR SUBMISSION OF COMPLETED DOCTORAL DISSERTATION

Hereby**,** I declare that I agree that the candidate **VLADIMIR MILIĆEVIĆ**, can submit completed doctoral dissertation to the Officer for postgraduate education of the Faculty under the name of: **MODERN HELPDESK BUSINESS SYSTEMS IMPROVEMENT BY ADVANCED INTELLIGENT SOFTWARE TOOLS** for the purpose of its evaluation and defense.

Prof. Slavoljub Milovanović, Ph.D.

ИЗЈАВА

Под пуном материјалном и моралном одговорношћу изјављујем да је приложена докторска дисертација резултат сопственог научног истраживања и да је коришћена литература на адекватан начин цитирана, без преузимања идеја, резултата и текста других аутора на начин којим се прикрива оригиналност извора. У потпуности преузимам одговорност за спроведено истраживање, анализу, интерпретацију података и закључке.

Владимир Милићевић

У Нишу, 02.07.'15. године

S T A T E M E N T

With due material and moral responsibility, hereby I declare that the doctoral dissertation is the result of personal scientific research and that the references used are cited adequately without use of ideas, results and texts of other authors in the way that hides the source's originality. I take the full responsibility for conducted research, analysis, data interpretation and conclusions.

Vladimir Milićević

Niš, 02.07.'15.

Унапређење савремених HelpDesk пословних система применом напредних интелигентних софтверских алата

Резиме

У савременом пословном окружењу од кључног значаја је примена савремених ИКТ технологија за сваког учесника на глобалном тржишту. Производи и услуге постају све комплекснији, а самим тим и софтверска решења за подршку пословању компанија. Отуда, савремени алати и технике развоја софтвера захтевају константна проширења са циљем пројектовања и израде софтверских решења која ће у потпуности задовољити актуелне корисничке захтеве, а компанијама донети стратешку предност на тржишту.

Као кључно тржиште, где овакви системи могу у пуној мери доћи до изражаја, дисертација идентификује индустрију осигурања. HelpDesk систем, интегрисан у информациони систем овакве компаније, доприноси у великој мери квалитету производа и услуга, као и повећању степена општег задовољства запослених и корисника услуга. У ту сврху, идентификовани су правци и технике развоја интелигентног HelpDesk система кроз аспектна проширења савремених објектно - оријентисаних технологија алатима који омогућавају динамичку манипулацију новог знања информационим системом. Тестирана су два позната аспектно – оријентисана приступа и комбиновањем њихових доминантних карактеристика креиран је оригиналан унифицирани приступ којим су реализовани зацртани циљеви дисертације.

Као посебно разматрање истиче се и имплементација HelpDesk система у синергији са интелигентним програмима којима је рад оваквог система у потпуности аутоматизован, а експертиза у великој мери померена од човека ка рачунару. Избором оптималног алгоритма претраге базе питања, HelpDesk систем је снабдевен моћним алатом – неуронском мрежом, који је у потпуности омогућио наведне програмске функционалности.

Кључне речи: *HelpDesk, објектна оријентисаност, аспектна оријентисаност, осигурање, алгоритам претраге, неуронска мрежа.*

Научна област: Економија

Ужа научна област: Информатика, информатика и кибернетика у економији

УДК: 004.4’242, 004.891.2

The improvement of Modern HelpDesk business systems by advanced intelligent software tools

Abstract

In today's business environment, the application of modern ICT technologies is crucial for each participant in the global market. Together with the products and services, the software solutions used for supporting businesses are becoming more and more complex. Therefore, modern tools and techniques for software development require constant expansion with the aim of designing and developing software solutions that will fully satisfy current customer requirements and give companies a strategic advantage in the market.

The dissertation identifies the insurance industry as the key market place where these systems can be fully applied. HelpDesk system, integrated into information systems of the insurance companies, contributes to a large extent to quality of products and services and increases the level of general satisfaction of employees and service users. For this purpose, tools and techniques for developing intelligent HelpDesk system have been identified through expansion of modern object-oriented technologies by tools that allow the dynamic manipulation of the new knowledge by information system. Two well-known aspect-oriented approaches have been tested and their dominant characteristics have been combined, thus creating an original unified approach that helped to achieve the intended goals of the dissertation.

In the dissertation, the implementation of the HelpDesk system in synergy with intelligent programs has been especially considered. The work of such a system is fully automated and expertise is greatly shifted from a man to a computer. By choosing the optimal algorithm for searching a database of questions, HelpDesk system is equipped with a powerful tool – neural network, which fully enables the aforesaid software functionalities.

Key words: *HelpDesk, object oriented, aspect oriented, insurance, searching algorithm, neural network.*

Scientific field: Economics

Special topics: Informatics, Informatics and Cybernetics in Economics

UDC: 004.4’242, 004.891.2

Садржај

1. Увод

Савремено пословно окружење карактеришу динамичке промене које свакодневно утичу на пословање сваког пословног субјекта. За лакше суочавање са изазовима оваквог окружења, компаније су принуђене да у свом пословању свакодневно и активно користе савремене информационо-комуникационе технологије (ИКТ). Последња декада доноси велике промене у начину креирања и употребе пословног софтвера. По бројним ауторима *пословни софтвер је основа успеха у конкурентном окружењу глобалне економије.*[1]

У савременој индустрији софтвера, пројектовање софтверског решења, а касније његов дизајн и имплементација, све више инсистирају на употреби интелигентних рачунарских техника. Примену наведених техника могуће је срести код решења за претрагу великих база података (Big Data)[2], предвиђања кретања на великим тржиштима[3], па све до препуштања одређених управљачких улога оваквим системима[4]. По мишљењу групе аутора: *Пословни менаџери и експерти свакодневно се сусрећу са кључним изазовом. Од њих се очекује брже и квалитетније доношење одлука за подршку савременом концепту пословања – урадити више за краћи временски период. У борби са наведеним изазовима од кључног значаја је примена интелигентних софтверских решења као подршка властитим пословним операцијама.*[5] Савремени информациони системи користе све више интелигентну подршку за бројне пословне процесе као што су: [6] [7]

- аутоматизација хардвера;
- планирање производње;
- планирање и креирање пословних процеса;
- редуковање комплексности система;
- управљање финансијским токовима;

[1] Ballou M. C. 2008. *Improving Software Quality to Drive Business Agility*, IDC – White Paper
[2] Curry S, Kirda E, Schwartz E, Stuart W, Yoran A. 2013. *Big Data Fuels Intelligence-Driven Security*, RSA Security Brief, January 2013
[3] dobney.com. http://www.dobney.com/market_intelligence.htm (приступљено 12.02. 2015. у 10:20)
[4] Sousa K, Effy O. 2014. *Management Information Systems*, Cengage Learning
[5] Simur J, Schulte W. R, Hill B. J, Jones T. 2012. *Magic Quadrant for Intelligent Business Process Management Suites*, Gartner 2012.
[6] utwente.nl. www.home.math.utwente.nl/~omnerenjcw/153084/sheetsApp.pdf (приступљено 12.02. 2015. у 10:44)
[7] comarch.com. www.comarch.com/files_eu/file_1585/ComarchAltum.pdf (приступљено 12.02. 2015. у 10:48)

- управљање залихама;
- контрола опреме и инвентара;
- управљање ефикасном комуникацијом између запослених, итд.

Од посебног значаја за овај рад јесте уочавање могућности ангажовања интелигентних софтверских алата и техника за подршку комуникацији између службеника компаније и менаџмента различитих нивоа, као и између корисника услуга и службеника. Правовремена и брза информација у савременом пословању је од кључног значаја за његов квалитет. По ауторима Дарден и Бриц, *компонента информационог система која омогућава размену информација по шеми питање-одговор, између експерата и осталих корисника система, назива се HelpDesk*.

Савремени HelpDesk системи углавном функционишу тако што се постављено питање, на страници за упит, прослеђује одговарајућем експерту на разматрање и одговарање. Од тренутка постављања питања до добијања одговора може проћи значајно времена, а то за последицу може имати застој у обављању посла што је у савременим условима пословања недопустиво.

Приликом пројектовања напредних софтверских решења све чешће се постављају задаци који имају заједнички именитељ – пројектовани и имплементирани софтвер мора да омогући минимизацију грешака у пословању проузрокованих људским фактором и да при томе обезбеди тренутно пласирање прецизних информација обезбеђујући стратешку предност на тржишту компанији која га уводи у властито пословање. Водећи се наведеним тезама, у раду ће бити фокус на анализи могућих праваца унапређења савремених HelpDesk система из области индустрије осигурања и давању властитог решења за аутоматизацију њиховог функционисања, ослањајући се на интелигентне технике претраге. Компанијама из области осигурања недостају овакви системи који би омогућили препознавање образаца понашања запослених и корисника, а самим тим и ефикасно управљање без потребе за интервенцијом менаџмента различитих нивоа. На овај начин, отвара се велико поље изучавања интелигентних рачунарских система који могу самостално да препознају нове чињенице, креирају на основу њих ново знање које аутоматски, без интервенције развојног софтверског тима, могу да угради у властиту функционалност. Отуда је неопходно иновирати приступ пројектовању и кодирању будућег софтверског решења које ће бити

резултат анализе изложене у наредним поглављима овог рада. Од посебног значаја је потрага за новим алатима и техникама којима ће бити омогућено динамичко управљање знањем које се корисницима савремених HelpDesk система најчешће презентује у форми *FAQ (Frequently Asked Questions)* библиотека. Омогућавањем динамичке контроле знања, HelpDesk систем ће бити оспособљен да тренутно одговара на постављена питања без интервенције човека – експерта. Такође, обезбеђивањем шаблона – класа правила добијених на основу постојеће експертизе, база знања ће имати могућност аутоматског проширивања новим специјализованим правилима – објектима правила. На овај начин биће структуриран интелигентни рачунарски систем способан да самостално учи.

1.1. Предмет и циљеви истраживања

Анализом савремених HelpDesk решења, као и актуелних објектно-оријентисаних софтверских алата, могуће је доћи до следећег закључка***:*** база знања оваквог експертног система, одакле би HelpDesk систем проналазио одговоре на питања корисника, не може бити креирана по стандардном *if – then - else* сценарију, јер је он извршив на нивоу имплементације софтверског решења и не може резултовати аутоматском уградњом новог знања у постојеће софтверско решење. Дакле, правила морају имати форму класа тј. објеката који су у веома лабавој вези са класама постојећег софтверског решења, да би било могуће размишљати о њиховој динамичкој имплементацији у постојеће софтверско решење.

Као могући правац, у којем треба тражити решење, намеће се млада техника пројектовања и програмирања позната као аспектно-оријентисани развој софтвера (AOSD – Aspect Oriented Software Development) при чему су од посебног значаја радови следећих група аутора:

1. Аутори окупљени око Грегора Кишалеса са Ксерокс (енг. Херох) института у САД;
2. Аутори окупљени око Ивана Киселъева, APP Group USA.
3. Аутори окупљени око професора Маје Д'Хондт и Вивиен Јонкерс са Универзитета Вриje у Бриселу, са посебним акцентом на рад аутора Мариа Аугустина Сибран.

Истраживањем и анализом радова поменутих група аутора, могуће је добити полазну тачку за процес развоја планираног софтверског решења. Прве две групе аутора

експериментисали су са аспектно-оријентисаним алатом познатим као AspectJ. Програмски код који се добија применом овог алата скоро је идентичан JAVA коду, изузетно се брзо извршава и повезује, али није имао могућност динамичког везивања за постојећу апликацију. Детаљна анализа овог алата могућа је кроз истраживање радова следећих аутора: В. Б. Гризволд, Е. Хилсдејл, Ј. Хагунин, М. Керстен, Ц. Палм, Г. Кишалес и осталих са Ксерокс института у САД[8] [9] [10], затим Ц. Хунт[11] и И. Кисељев[12]. Аутори у својим радовима представљају нову софтверску форму *савет* која представља иновативан одговор базе знања на упит главног управљачког дела софтверског решења.

Аутори са Универзитета Врије[13] [14] [15] [16] [17]су експериментисали са кодом који је познат као JasCo. Његове карактеристике су: спорије повезивање и извршавање, али могућност динамичког везивања за постојеће софтверско решење по сценаријима: *before, around* и *after*, а који се односе на време примене правила које одговара одговору на постављено питање HelpDesk систему.

Уочено је да сваки од наведених приступа има своје предности и недостатке. Будући да се базирају на логици отвореног кода, додатним развојем могуће је комбиновати најбоље појединачне карактеристике и добити идеалан алат за кодирање објеката правила. Тако је као основ за креирање класа и објеката правила у HelpDesk бази знања могуће искористити AspectJ, проширен могућностима динамичке примене по наведеним сценаријима, а то би представљало оригинално решење из области аспектно-оријентисаног развоја софтвера.

---

[8] Kiczales G, Lamping J, Mendhekar A, Maeda A, Videira Lopes C, Loingtier J. M, Irwin J. 2007. *Aspect-oriented Programming*, Xerox Palo Alto Research Center - 2007.
[9] Hilsdale E, Hugunin J. 2004. *Advice Weaving in AspectJ*, 3rd International Conference on Aspect-Oriented Software Development (AOSD). April 2004.
[10] Kiczales G, Hilsdale E, Hugunin J, Mik Kersten, Palm J, Griswold V. B. 2001 . *An Overview of AspectJ*. In *Proceedings of the 15th European Conference on Object-Oriented Programming (ECOOP)*. Springer, 2001.
[11] Hunt J. 2006. Aspect oriented programming with Java, The Register.
[12] Kiselev I. 2008.Aspect-oriented programming with AspectJ ,Sams.
[13] Suvee D, Jonckers V, Vanderperren W. 2004. Supporting JAsCo AOP, ENTCS vol. 107.
[14] Suvee D, Vanderperen W, Jonckers V. 2003. JAsCo: An Aspect-Oriented Aproach forComponent Based Software, AOSD Boston.
[15] Cibrán M. A, Suveé D, D'Hondt M, Vanderperren W, Jonckers V. 2004. Integrating Rules with Object-Oriented Software Applications using Aspect-Oriented Programming, Proceedings of ASSE'04, Argentine Conference on Computer Science and Operational Research, Córdoba, Argentina, 2004.
[16] Cibrán M. A, D'Hondt M, Vanderperren W. 2003. Aspect-Oriented Programming for Connecting Business Rules. Proceedings of BIS International Conference, Colorado Springs, USA, June 2003.
[17] Cibran M. A. 2007. Connecting High Level Business Rules With Object Oriented Applications. Vrije Universiteit Brussel

Добијени приступ може да добије и званичан назив *AspectJ+*. Такође, од посебног значаја је давање одговора на питање: „Да ли је AOSD засебна и независна парадигма развоја софвтера или представља проширење познатих објектно-оријентисаних техника развоја софтвера?“ Одговор на ово питање није дала ниједна од наведених група аутора.

Након избора оптималног алата за креирање базе знања HelpDesk система, неопходно је дефинисати иновативан приступ за развој информационог система који би био способан да на динамички начин управља властитим знањем. Иновативност се огледа у немогућности савремених објектно-оријентисаних алата да се самостално изборе са наведеном проблематиком, те је неопходно користити извесна проширења попут аспектно-оријентисаних алата. У наведеном светлу, могући пут до жељеног софтверског решења могао би ићи следећем правцу:

1. Разматрања аутора Марије Аугустине Сибран биће уважена и проширена увођењем новог доменског језика. На високом нивоу, нивоу домена, биће пројектована и кодирана правила HelpDesk базе знања новим језиком домена. Превођењем доменског кода у имплементацију, добиће се програмски код највећег могућег степена модуларности, а то за резултат даје могућност динамичке интеграције правила, огромну брзину извршавања софтверског решења и најефикасније отклањање уочених грешака током тестирања;
2. За претрагу по објектима базе знања и откривање новог знања биће програмирана монотона неуронска мрежа и оптималан тренинг алгоритам. О овој проблематици могуће је направити детаљну анализу на основу увида у радове аутора: Р. Рохас[18], М. Цилимкович[19], и други, те на основу уочених недостатака њихових приступа, као и недостатака уочених током двогодишње експлоатације другог интелигентног софтверског решења[20], могуће је наћи квалитетније решење са оптималним тренинг алгоритмом.
3. Биће развијена софтверска компонента *NDL/Generator* која ће новооткривено знање кодирати дескриптивним доменским језиком;

[18] Rojas R. 1996. *The BackPrpagation Algorithm*, Neural Networks, Springer Verlag Berlin - 1996
[19] Cilimkovic M. 2008. *Neural Networks and Back Propagation Algorithm*, Institute of Technology Blanchardstown
[20] Милићевић В. 2012. *Допринос развоју интелигентног рачунарског система базираног на приступу пословних правила*, ФАМС

4. Биће развијена софтверска компонента *Translator/AspectJ+* која ће горе описани код превести у објекте правила и проследити на динамичко уграђивање у базу знања.

Посебан акценат у овом раду је на анализи бројне литературе, књига, веб презентација и пројектне документације, за изналажење оптималног алгоритма за тренинг механизма претраге заснованог на монотоној математичкој функцији којом ће бити одређени тежински фактори чворова у скривеном слоју мреже. Затим, користећи статистичке тестове (по средњој квадратној грешци) биће изабрана идеална архитектура механизма претраге која се огледа у оптималном броју чворова у скривеном слоју неуронске мреже. Од посебног значаја је примена изабраног алгоритма претраге у комбинацији са принципом препознавања природних језика (прилагођеним за нови доменски језик правила), Зифовим законом и методом резервисаних речи. Такође, у саму архитектуру неуронске мреже биће уграђени софтверски детаљи познати као *семафори*[21] [22] који ће имати задатак да синхронизују процесе механизма претраге чиме ће бити обезбеђено елиминисање недостатака уочених на двогодишњој експлоатацији познатог софтверског решења[20] на великом броју итерација претраге.

Сагласно апострофираним теоријским ставовима и студијама, **предмет истраживања** у докторској дисертацији је класа хибридних интелигентних информационих система способних да аутоматски препознају ново знање из релевантног узорка (скупа питања), да га кодирају и на динамички начин угради у већ постојећу базу знања (скуп могућих одговора).

Сходно формулисаном премету, **циљ истраживања** је био трагање за одговором да ли је могуће обезбедити технике пројектовања и програмирања за развој информационог система који из доступног домена аутоматски препознаје нова пословна правила и њихове везе, кодира их и аутоматски, на најфлексибилнији могући начин, повезује са властитим већ функционалним делом за управљање знањем.

Посебан задатак, који је обављен у дисетацији, јесте давање одговора на постављене хипотезе:

[21] Zelenski J. 2008. *Thread and Semaphore Examples*, Stanford, CS107
[22] ufsc.br. http://www.inf.ufsc.br/~bosco/ensino/ine5645/Semaphore-Monitor.pdf (приступљено 19.02.2015. у 8:30)

1. Могуће је креирати оперативан информациони систем који из доступног домена аутоматски препознаје нова пословна правила и њихове конекције, кодира их, и аутоматски, на најфлексибилнији могући начин, повезује са властитом компонентом за управљање знањем.
2. Комбиновањем доминантних карактеристика репрезентативних аспектно-оријентисаних приступа могуће је развити унапређен алат за решавање проблема повезивања правила са језгром система.
3. Пројектовање засновано на најновијем *MDSD* моделском приступу обезбеђује исказивање извршивих правила високог нивоа и одговарајућих конекција.
4. Ефикасно повезивање и највећи степен модуларности могуће је обезбедити кроз енкапсулацију правила и конекција у JAVA објекте правила и *AspectJ*+ аспекте конекција.
5. Алгоритмом претраге уназад, унапређеним применом семафора, и методом резервисаних речи могуће је тренирати неуронску мрежу за аутоматско препознавање пословних правила из релевантног узорка.

Одговорима на постављене хипотезе посебно се баве поглавља која се односе на увод у проблематику, идентификовање технологија за постизање постављених циљева, примену изабраних технологија за израду прототипа којим се презентују резултати истраживања, као и извлачење релевантних закључака.

На самом почетку рада дискутовано је о значају савремених HelpDesk система у електронском пословању. Овде је јасно истакнута савремена дефиниција HelpDesk система са правцима у којима се њихова примена креће. Затим, HelpDesk системи из домена електронске трговине презентовани су кроз захтеве корисника и најзначајније представнике који су развијени од стране највећих произвођача софтвера попут компаније *Oracle*. У даљем излагању, акценат је стављен на решења за подршку пословању у домену јавне управе са посебним освртом на решење *teleNetwork*. На самом крају овог поглавља акцентовани су HelpDesk системи који се примењују у савременој индустрији осигурања. Након уочавања корисничких захтева демонстрирана су два савремена решења из ове области:

- *LBi HR HelpDesk* и
- *IBM Claims Fraud Solution.*

Наведена софтверска решења су била кључна за развој идеје о интелигентном софтверском HelpDesk решењу које је предмет истраживања овог рада.

У наставку, рад се бави примерима имплементације и могућим правцима унапређења постојећих HelpDesk система у Србији и земљама које су апострофиране као развијена информатичка друштва. У ту сврху је прво елаборирaн HelpDesk систем Пореске управе Републике Србије са свим уоченим предностима и недостацима. Даље, рад се бави још неким напредним HelpDesk решењима, из развијених информатичких друштава. Као представници оваквих система истакнути су системи:

- *KronoDesk* и
- *SysAid*.

Након истицања недостатака постојећих система, у раду су истакнуте технологије чијом је применом могуће отклонити уочене недостатке и постићи оптимално решење на пољу аутоматизованог HelpDesk система.

У даљем излагању, од посебног значаја за рад је избор технологија и алата за развој интелигентног HelpDesk система. У овом делу поређена су два приступа: JasCo и AspectJ. За оба приступа тестиране су перформансе одзива правила и трајања процеса извршавања правила и наведене могућности примене за остваривање циљева рада. Приказано је да један приступ обезбеђује квалитетније перформансе (AspectJ), док је другим приступом (JasCo) омогућено динамичко уграђивање нових објеката правила у већ постојеће софтверско решење. Комбиновањем најбољих индивидуалних карактеристика наведених приступа креиран је оригиналан приступ под називом AspectJ+. Овај приступ је такође тестиран на исти начин као и претходна два приступа уз поређење перформанси у односу на JasCo приступ и преузимање његових особина које су у вези са динамичким повезивањем новог знања и оперативног софтверског решења.

У наставку, у раду је описана анализа модела домена по MDSD приступу. Све идентификоване класе правила, из којих се касније креирају објекти, као и одговарајуће везе са делом софтверског решења за управљање знањем, кодиране су језиком домена који је

веома близак говорном језику. На овај начин омогућено је активно учешће експертима различитих профила у развоју софтвера, а да не морају да буду упознати са алатима и техникама програмирања. Овај језик је означен као NDL и представља специјализацију језика високог нивоа за примену у кодирању знања и његову дистрибуцију кроз HelpDesk системе. Као такав, такође, представља оригиналан начин примене језика високог нивоа у области HelpDesk система. Овај део рада представља основу развоја софтверске компоненте NDL/Generator која има задатак да из узорка, скупа питања прослеђених путем HelpDesk система, користећи принцип резервисаних речи и Зифов закон, кодира откривено знање NDL језиком и проследи га на превођење у код који одговара програмском језику са одговарајућим аспектним проширењима.

У следећем поглављу рада акценат је на креирању специфичних софтверских трансформација којима се NDL код преводи директно у JAVA код проширен AspectJ+ функционалностима. Овде су посебно тестиране перформансе трансформација за превођење веза објеката правила у JasCo и AspectJ+ аспекте веза. Тест је показао да је нови AspectJ+ приступ доминантан по питању стабилности и брзине извршавања софтверских трансформација. Овим поглављем постављен је основ развоја софтверске компоненте из класе транслатора којом се директно преводи ново знање у извршив програмски код.

У кључном делу рада приказано је софтверско решење у форми интелигентног HelpDesk система за подршку компанијама из индустрије осигурања. У овом поглављу обједињена су теоријска разматрања и тестови из претходних поглавља са савременим развојним технологијама и алатима. У посебним деловима овог поглавља говори се о:

- појединачним компонентама софтверског решења,
- аспектно-оријентисаној бази знања тестираној на конкретним примерима, подсистемима за препознавање и аутоматску интеграцију новог знања,
- избору алгоритма претраге и тренингу неуронске мреже,
- архитектури софтверског решења за управљање знањем и, на самом крају,
- имплементацији и презентацији HelpDesk компоненте посматраног информационог система.

Као додатак, дискутовано је о безбедности овако конципираног софтверског решења.

У завршним поглављима овог рада истакнути су релевантни закључци, коришћена литература и биографски податаци кандидата, као и додаци који прецизније показују природу развијеног софтверског решења.

Гледано са економског аспекта, у дисертацији је истакнута методологија развоја софтверског решења које ће пословне задатке обављати ефикасније и ефективније уз могућност динамичког прилагођавања променама које диктира тржиште.

2. HelpDesk системи у електронском пословању

Гледано из перспективе савремених великих компанија, HelpDesk је алат који у великој мери представља потпору обављању свакодневних активности. Откако су овакви системи постали редовне компоненте моћних корпорацијских софтвера, заједно са њиховом еволуцијом, мењала се и њихова дефиниција.

По аутору Маргарет Роуз (Margaret Rose) за савремену компанију *HelpDesk представља место на којем корисник информационих технологија може позивом добити помоћ везану за решавање конкретног проблема*.[23] За велики број компанија HelpDesk представља систем који се ослања на једног оператера са обезбеђеном телефонском везом који, на основу опште усвојених процедура, даје одговоре корисницима у вези са свакодневним пословним активностима. Ти одговори су најчешће у директној вези са законском регулативом, актима компаније или пак са проблематиком која се често понавља. За велике компаније, које улажу значајна средства у информационо-комуникациону подршку, HelpDesk представља групу експерата која користи специјализован софтвер за праћење статуса иницијализованог проблема, а затим, применом напредних софтверских алата, анализира тај проблем и управља његовим коначним решењем.

На самом почетку, HelpDesk је осмишљен као централизована подршка службеницима унутар великих компанија путем телефонске везе. Отуда потиче и први назив за овакав тип подршке – кол (енг. call) центар. Развојем Интернета, философија размене информација кроз центар за подршку у великој мери је промењена. Отуда је могуће наћи бројне називе за овакав вид подршке корисницима. Уместо назива HelpDesk често се може чути ServiceDesk, центар за рачунарску подршку, информатички центар за одговоре, информативни центар, центар за техничку подршку итд.

У савременим околностима, комплетна подршка пословању великих мултинационалних компанија подржана је HelpDesk компонентама њихових моћних корпорацијских система. Отуда, примена оваквог система, за сервирање правовремених и прецизних информација корисницима, захтева изузетно моћну информатичко –

[23] techtarget.com. http://searchcrm.techtarget.com/definition/help-desk (приступљено 06.10. 2014. у 22:50)

телекомуникациону подршку. За последицу имамо експерте који, да би успешно одговорили пословним задацима, морају да пролазе кроз сталан процес учења и обуке.

У домаћим оквирима, HelpDesk тренинзи скоро да и не постоје. Углавном се своде на теме у оквиру специјализованих предмета на високошколским установама или на обуку унутар компаније која користи систем за подршку. У земљама развијеног света, стандард за обуку HelpDesk експерата поставила је организација под називом *Service & Support Professionals Association*.[24] Оснивач и председник ове организације је Фил Вергис (Phil Verghis) који је у својим радовима[25] дефинисао оквире по којима највеће светске компаније управљају системима за подршку. На Универзитету Харвард, његов уџбеник[25] налази се на листи учила као *строго* препоручљив, окарактерисан као *најинтелигентнији приручник за управљање подршком пословној организацији* од стране организације Association of Support Professionals.[26] По овом аутору, *квалитет HelpDesk система је у директној вези са квалитетом базе најчешће постављених питања (FAQ – Frequently Asked Question)*.[27] База најчешће постављених питања, изграђена на основу свеобухватне и стручне експертизе, представља најбољу базу за властито унапређење и проширење.

Отуда се као циљ, до којег ће се доћи у наредним излагањима дисертације, поставља примена овако дефинисане базе најчешће постављених питања као основе за доношење одлука и даље учење интелигентног HelpDesk система чији развој представља коначни циљ дисертације. Коначно, кроз процес учења долази и до ширења базе најчешће постављених питања.

Будући да су савремени HelpDesk системи у потпуности Интернет усмерени, њихова примена раширена је на све сегменте електронског пословања. Отуда, корисници услуга и запослени, путем HelpDesk система, могу добити правовремене и прецизне информације из домена електронске трговине, јавне управе, електронских сервиса осигуравајућих компанија, електронског банкарства, електронског маркетинга итд.

[24] http://www.verghisgroup.com/ (приступљено 06.10. 2014. у 23:50)
[25] Verghis P. 2013. *The Ultimate Customer Support Executive*, Silicon Press, ISBN 092930634
[26] asponline.com. http://www.asponline.com/essentials (приступљено 07.10. 2014. у 00:14)
[27] philverghis.com. http://www.philverghis.com/helpdesk.html (приступљено 07.10. 2014. у 00:20)

2.1. Примена HelpDesk система у електронској трговини

За сваког трговца је од великог значаја да буде присутан када потрошач разгледа производе и доноси одлуку о коначној куповини. Када се ради о електронској трговини потребно је узети у обзир додатне елементе. Прво, потрошач мора да добије на време производ који је поручио. Затим, битну ставку представља начин плаћања нарученог производа, евентуална рекламација производа, замена производа новим, повраћај новца због неисправног производа или несагласног са нарученим итд. Отуда је неопходно следеће: *Потрошачу треба показати да вам је стало. Задовољан потрошач ће се поново вратити. То значи да вам је потребан перфектан скуп алата помоћу којег ћете успешно одговорити на свако потрошачко питање*.[28]

2.1.1. Електронска трговина и захтеви корисника

Савремени HelpDesk софтвери настоје да прошире приступ управљању односима са клијентима (Customer relationship management - CRM) тако што се фокусирају на места окупљања клијената омогућујући компанијама лојалне и профитабилне клијенте. Сервисирањем богатог и правовременог искуства кроз вишеканалну инфраструктуру, савремени HelpDesk системи креирани су као подршка ефективнијој продаји и сервирању информација од значаја за потрошаче и запослене. Кроз интеграцију са осталим компонентама корпорацијског софтвера омогућава се богато и динамично продајно искуство којим се богатство и обрт компаније из домена електронске трговине вишеструко увећава.

Свакодневним освежавањем знања, везаног за подршку корисницима и запосленима, компанијама из области електронске трговине омогућен је брз одговор на динамичке промене тржишта. Инвестирањем у CRM, електронска трговина добија нови, проширени облик, централизоване администрације за одређивање цена, промовисање производа и услуга, утврђивање подобности, компатибилности и конфигурације производа и услуга итд. У маркетиншком смислу, све промене које се једном имплементирају системски се преносе и користе на нивоу целе корпорације.

[28] freshdesk.com. http:// www.freshdesk.com/industries/help-desk-software-retail-ecommerce (приступљено 07.10. 2014. у 01:10)

Посебан задатак HelpDesk компоненте јесте да омогући компанији брзо постављање нових стандарда електронске продаје, доносећи иновације по најнижој могућој цени, омогућујући запосленима веће фокусирање на клијенте.

Као једно од најбољих софтверских решења из HelpDesk области, а за подршку електронској трговини, јесте решење компаније Oracle - *Siebel E-Commerce*.[29]

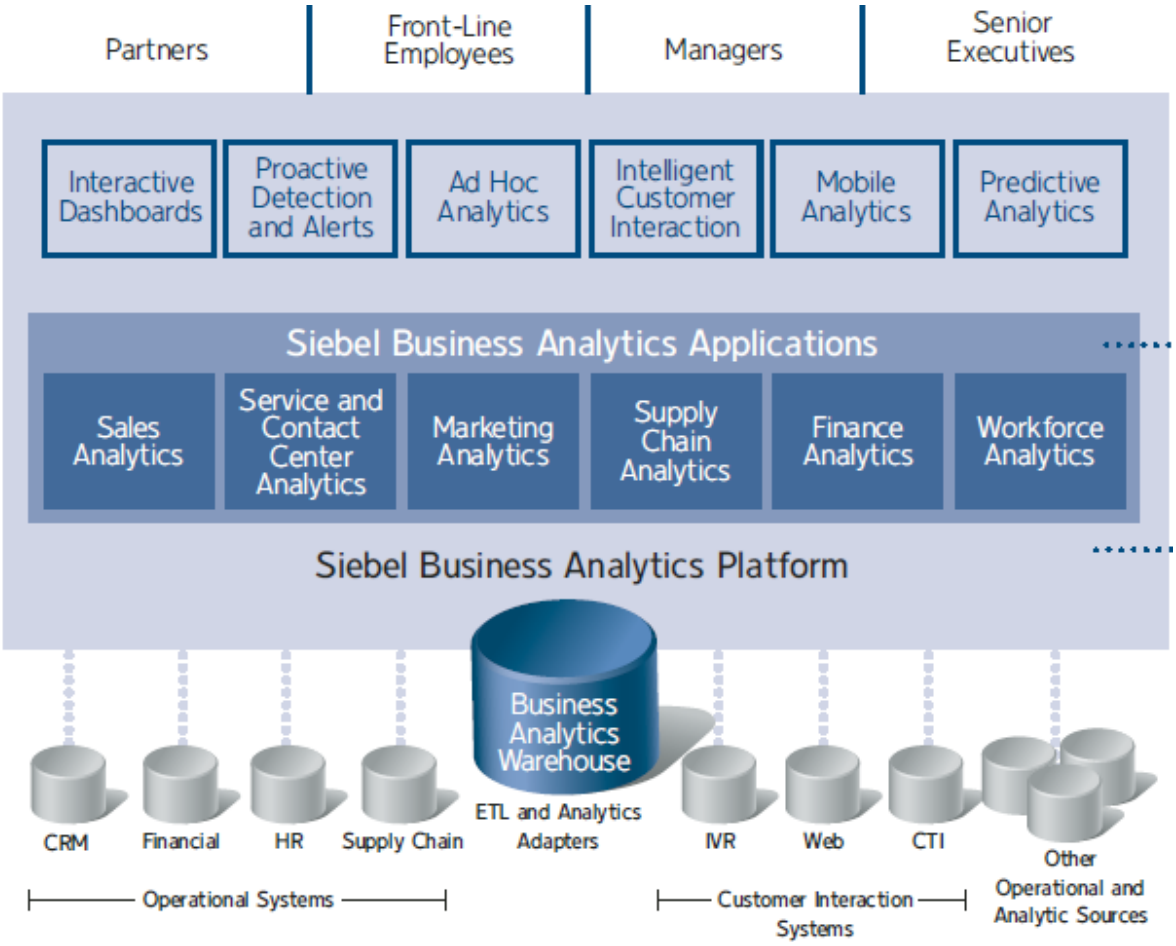
2.1.2. Софтверско решење Oracle - *Siebel E-Commerce*

По тврдњама компаније Oracle, *Siebel E-Commerce* (Слика 2.1) представља комплетно решење по принципу *commerce-in-a-box*.[30] Наведено софтверско решење изграђено је као комплексно и свеобухватно решење за подршку приступу управљању односима са клијентима (Слика 2.1). У самом фокусу софтвера су жеље и намере клијената и свака интеракција корисника са софтвером је усмерена ка коначном задовољству клијената. Знање на којем се базира рад софтверског решења дизајнирано је на основу корисничких искустава тако да је свако добијање корисних информација и доношење коначне одлуке о куповини производа или услуга доступно кориснику једним кликом миша.

Захваљујући огромној бази одговора на корисничка питања, Орацловим решењем проналажење траженог и адекватног производа представља веома једноставан процес. Кроз једну интеракцију са HelpDesk компонентом корисник се аутоматски упућује на део електронског каталога који садржи све податке о траженом производу. Користећи кључне речи и динамичку претрагу, потрошач веома једноставно дефинише филтер по којем се навигацијом кроз каталог долази до производа и услуга који су од интереса за потрошача. Посебан механизам резоновања, који је имплементиран у ово софтверско решење, назива се *Oracle Real-Time Decisions*. Користећи затворене петље, у предикативној аналитици, механизам резоновања омогућава сталне предлоге релевантних понуда које су базирани на корисничким профилима, претходним куповинама и акцијама претраге у реалном времену. Уколико понуда производа или услуга није у потпуности задовољавајућа, потрошач може

[29] oracle.com. http://www.oracle.com/us/products/applications/siebel/051165.pdf. (приступљено 21.10. 2014. у 17:10)

веома једноставно да истражује сродне промоције и да сам изабере производ који му није примарно понуђен.



Слика 2.1. Организација Oracle Siebel E-Commerce система (извор www.siebel.com[30])

Посебно, за сложеније производе и промоције, софтвер омогућава конфигурисање специфичних потреба потрошача на веома једноставан начин. Било да се ради о куповини новог производа или промени плана претходно дефинисане куповине, избор конфигурације могуће је обавити кроз интуитивни интерфејс скројен по захтевима крајњег корисника.

Такође, за различите кориснике, странице на којима се приступа садржају HelpDesk компоненте могу имати различито прилагођен садржај захваљујући вишеканалном управљању веб садржајем. Када систем препозна пословно правило које одговара захтеву

[30] www.siebel.com. https://s3.amazonaws.com/rmc_docs/siebel_business_analytics_brochure.pdf. (приступљено 21.10. 2014. у 19:30)

корисника, страница се попуњава садржајем диктираним од стране правила. Такође, подешавање понашања на страници, попут приказивања каталога, ценовника и опција избора производа и услуга, одређено је одговарајућим правилима базе знања.

Посебно јако оруђе овог система за подршку електронској трговини представљају алати за убрзавање одлука о куповини без напуштања матичне странице. Управљање знањем ослања се на претраге које обављају потрошачи у реалном времену омогућујући увид у целокупну базу знања из које HelpDesk компаније црпи одговоре. Од посебног значаја је могућност додавања богатог садржаја било којег система за динамичко управљање садржајем (CMS - Content Management System) у електронске каталоге.

Даље, као посебан вид подршке потрошачима омогућена је комуникација са агентима продаје кроз различите канале истовремено. Слично као *ћаскањем* на друштвеним мрежама, овим системом је омогућено директно обраћање потрошача агентима продаје и добијање допунских информација о производима, услугама, ценама, начинима плаћања, попустима и сл.

На крају, наведено решење компаније Oracle има одличну основу за резоновање на основу историје претраге и посебних корисничких захтева похрањених као правила у бази знања. Међутим, овакво решење још увек није способно самостално да учи и користећи динамичке алате да аутоматски интегрише ново знање у базу знања. Управо у овом правцу ће дисертација дати смернице ка могућем унапређењу оваквог система.

2.2. Примена HelpDesk система у јавној управи

Електронска управа представља примену савремених информационо телекомуникационих технологија у циљу обезбеђивања погоднијег приступа информацијама и услугама грађанима, организацијама и службеницима, уз повећање квалитета услуга и охрабривање грађана ка узимању већег учешћа у демократским процесима и институцијама. Електронска управа омогућава велике погодности попут

већег квалитета и ниже цене услуга јавне управе као и квалитетнију комуникацију између грађана и управе.[31]

2.2.1. Јавна управа и захтеви корисника

Грађани често захтевају идентичан квалитет услуга од управе као што их пружају друге пословне организације. У том светлу, неопходно је развити квалитетан систем који ће омогућити континуирану комуникацију између грађана и јавне управе и на основу искустава грађана довести до унапређења квалитета услуга јавне управе. Кроз комуникацију грађана са управом, на свако питање које грађани поставе мора уследити брз и прецизан одговор. Само у том случају задовољство грађанина, као крајњег корисника, биће на жељеним нивоу, а позитивно искуство ће бити пренето на друге будуће кориснике електронских услуга јавне управе.

Са друге стране, пословни субјекти који комуницирају са јавном управом такође захтевају бројне електронске услуге. Брз и квалитетан одговор на питања која су у вези са сарадњом пословних субјеката и управе је у директној вези са квалитетом пословања посматраних пословних субјеката као и њиховим задовољством на пружене услуге система електронске управе.

На крају, разни делови јавне управе захтевају квалитетну међусобну комуникацију са циљем подизања квалитета услуга које електронским путем пружају грађанима и пословним организацијама. У ту сврху, информациони систем мора да омогући брзо и квалитетно сервисирање информацијама из домена законске регулативе службеницима задуженим за рад са грађанима и пословним субјектима.

Кроз аутоматизацију процеса којих извршавају службеници, унапређењем процеса доношења одлука, комуникације и имплементације одлука, као и увођењем нових метода достављања услуга крајњим корисницима, долази се до бројних бенефита које носи увођење система електронске управе: [31]

[31] Agraval A, Mittal M, Rastogi L. 2002. *Enabling e-Governance Integrated Citizen relationship Management Framework – the Indian Perspective*, Deloitte & Touche (2002) Budget 2002 Report, India.

- јавна управа је јефтинија;
- јавна управа обави више посла;
- посао се брже обавља;
- јавна управа је иновативнија.

На основу наведеног могуће је приметити да је квалитетна HelpDesk компонента оваквог система, која омогућава сервисирање корисника одговорима на постављена питања, од веома великог значаја за експанзију електронских услуга система јавне управе. HelpDesk системи као компоненте информационих система електронске управе увелико постоје. Од посебног значаја за овај рад јесте сагледавање тренутног стања најквалитетнијих HelpDesk система електронске управе и поређење са тренутним стањем у Србији. Након тога биће дато виђење у којем правцу може да се креће унапређење наведених система.

2.2.2. Решење teleNetwork[32]

Компанија teleNetwork је водећи светски провајдер техничке подршке услугама и HelpDesk решењима компанијама лидерима у области телекомуникација, управљања услугама и технолошким консалтингом. Као компанија која је у приватном власништву, са седиштем у САД, teleNetwork нуди решења заснована на резултатима који омогућавају клијентима остваривање њихових стратешких циљева.[32] HelpDesk решење ове компаније имплементирано је као подршка услугама електронске управе у бројним градовима и областима САД, Аустралије, Канаде и остатка развијеног света.

Компанија teleNetwork има 7.000 запослених који пружају електронске услуге за 2,2 милиона корисника из 25 земаља развијеног света. [32] Пре увођења напредног софтверског решења за подршку корисницима различитих нивоа, компанија teleNetwork је зацртала следеће циљеве које би софтверско решење требало да оствари:

[32] www.telenetwork.com. https://www.telenetwork.com/insights/articles/case_studies/Internet_Help_Desk_Performance_Improvement.pdf (приступљено 29.10. 2014. у 11:30)

- повећање задовољства корисника кроз унапређење нивоа и стандарда услуга;
- смањење броја приговора корисника кроз проширење јаког корисничког искуства сваком наредном итерацијом;
- смањење трошкова корисничке подршке смањењем броја позива корисницима и увођењем стратегије за унапређење перформанси система;
- унапређење оперативне ефикасности која би требало да резултује већим задовољством корисника.

Подацима, које истиче компанија teleNetwork, обједињеним са свих локација на којима су њена софтверска решења имплементирана, заиста су оправдана улагања у овакав систем:[33]

- анкетирањем корисника дошло се до показатеља да је задовољство корисника, након коришћења е-услуга сервисираних teleNetwork решењем, повећано за 90%;
- број приговора корисника смањен је за 2%;
- трошкови корисничке подршке смањени су за 20%;
- просечно чекање на одговор на постављено питање HelpDesk компоненти система износи око 2 минута;
- одустајање корисника од оваквог типа услуга смањено је за 7%.

Од посебног значаја за дисертацију јесте време одговора на питања корисника које упошљавањем посебних интелигентних софтверских решења може додатно да се унапреди. Дисертација ће се посебно бавити овом проблематиком у наредним излагањима.

Решење компаније teleNetwork имплементирано је у управама скоро свих савезних САД држава, са главним центром у Остину, Тексас. Компанија је након освајања матичног тржишта проширила деловање на Канаду, Аустралију, Европу и још неке земље остатка развијеног света. Највећу експанзију употреба HelpDesk система компаније teleNetwork бележи у САД државама на атлантској обали. Пораст профита јавних управа ових земаља

[33] www.telenetwork.com (приступљено 02.11. 2014. у 11:30)

износила је 18% у првом кварталу, прве године, након увођења овог система.[34] У најближем окружењу HelpDesk решење компаније teleNetwork налази се у употреби као компонента система електронске управе италијанских регија Умбрија и Тоскана. Регија Умбрија, након увођења teleNetwork решења у своје пословање, бележи уштеду од око 50 милиона Евра на свим нивоима.[35]

2.3. Примена HelpDesk система у индустрији осигурања

2.3.1. Индустрија осигурања и захтеви корисника

Примена информационо – комуникационих технологија у индустрији осигурања је у великој експанзији и свакодневно добија неки нови облик. Савремена софтверска решења могу се јавити у форми:[36]

- управљања односима са корисницима;
- управљања ефективношћу продаје производа и услуга;
- омогућавања тренутног приступа информацијама корисницима;
- обезбеђивања хитног реаговања на корисничке захтеве;
- обезбеђивања нових информација корисницима у вези са производима и услугама;
- прикупљања информација у вези задовољства корисника производима и услугама;
- аутоматизованог извештавања и мониторинга перформанси и продуктивности пословања.

Наведене тезе требало би проширити и могућностима електронског комуницирања унутар саме компаније јер се као корисници оваквих система јављају и службеници осигуравајуће компаније који су задужени за продају и сервисирање производа и услуга

[34] www.bnamericas.com . www.bnamericas.com/news/telecommunications/Atlantic_Tele-Network_Q3_net_income_up_18* (приступљено 02.11. 2014. у 12:01)
[35] www.pubblicaamministrazione.net. www.pubblicaamministrazione.net/e-government/news/929/p1/telenetwork-il-progetto-per-il-telelavoro.html (приступљено 02.11. 2014. у 11:55)
[36] www .my.safaribooksonline.com . www .my.safaribooksonline.com/book/management/9788131759844/case-study-on-reliance-life-insurance-use-of-information-system-as-a-strategic/c08-sec1-011 (приступљено 02.11. 2014. у 15:10)

крајњим корисницима. Развој одговарајућег HelpDesk система, где би запослени и корисници услуга, добијали савете од експерата различитог нивоа, могао би у великој мери да унапреди пословање осигуравајуће компаније. Добијањем прецизних и правовремених информација у вези са корисничким захтевима, задовољство запослених било би на вишем нивоу, а самим тим и ефективност и ефикасност њиховог рада. Међутим, брзина одговора на постављено питање зависи од тога када ће експерт, којем је питање упућено, реаговати. Отуда, намеће се потреба за размишљањем и истраживањем на пољу развоја софтверских решења која ће експертизу, у највећој могућој мери, померити од човека ка рачунару. Управо је то један од примарних циљева дисертације.

2.3.2. Софтверска HelpDesk решења из области индустрије осигурања

Осигуравајуће компаније, у великој мери, успех свог пословања базирају на максималном снижавању трошкова пословања. Отуда, оправдање за инвестирање у софтвер снабдевен HelpDesk подршком мора да се тражи на пољу снижавања укупних трошкова пословања. Будући да се ради о изузетно профитабилној области производа и услуга, највећи учесници на тржишту развоја софтвера увелико нуде властита решења за подршку индустрији осигурања.

Једно од најквалитетнијих решења на тржишту је производ компаније Dell. Наведено софтверско решење имплементирано је за сервисирање пословања осигуравајуће компаније Hickox (Велика Британија). Од почетка примене овог система, до данас, Hickox је остварио огромне бенефите. Осигуравајућа компанија је за 75% подигла ефективност и ефикасност сопственог пословања у првих 9 месеци примене HelpDesk решења компаније Dell.[37]

Такође, изузетно решење представља производ компаније LBi Software познато под називом *LBi HR HelpDesk*. Приликом развоја овог софтверског решења, компанија LBi Software је интензивно ослушкивала захтеве и потребе корисника. Кроз снажну повратну комуникацију са корисницима развијено је софтверско решење које је потом имплементирано за сервисирање пословања највећих северноамеричких компанија из

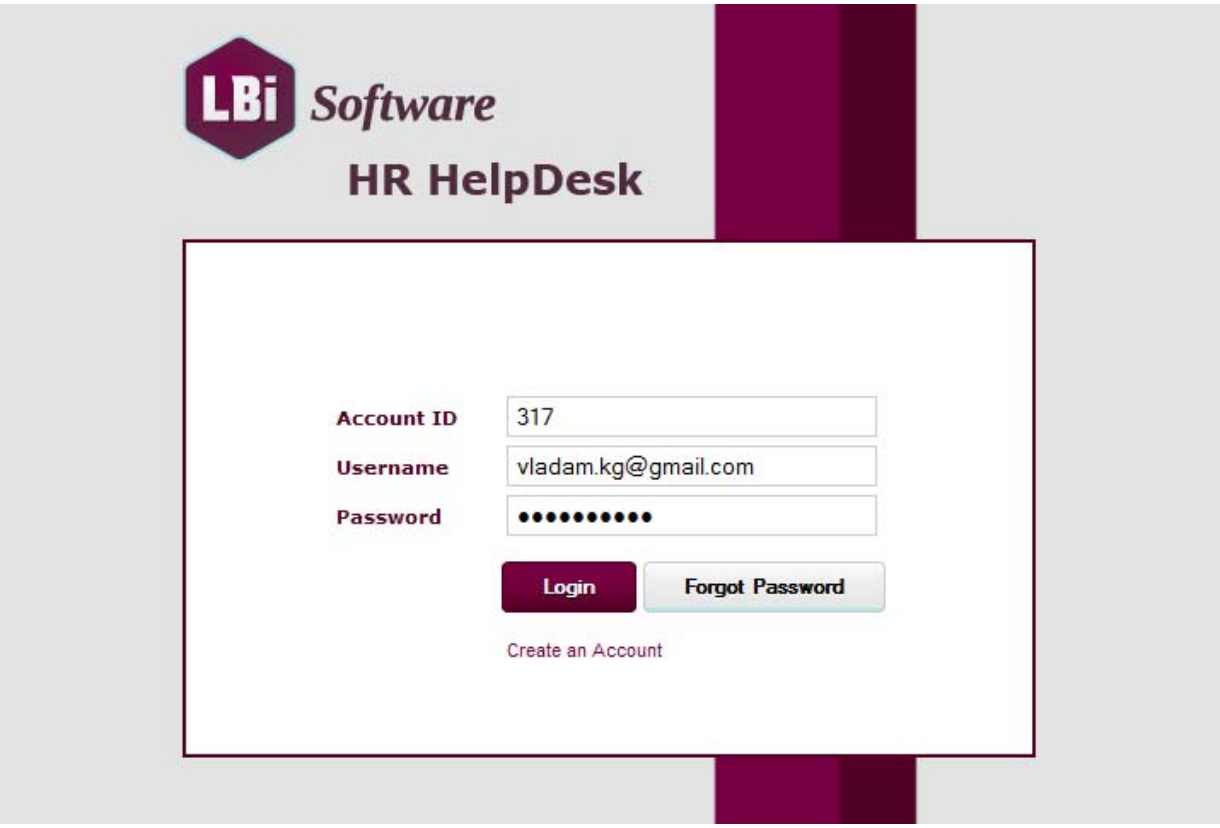
[37] www.quest.com. www.quest.com/documents/insurance-company-saves-75-percent-of-a-service-desk-fte-within-nine-months-of-launch-casestudy-26716.pdf (приступљено 02.11. 2014. у 15:50)

области осигурања. Редукцијом кључног проблема – времена одговора и повећањем броја успешних одговора службеницима и корисницима, софтвер је побољшао перформансе запослених (CSR - Corporate Social Responsibility) и повећао њихово задовољство условима рада, а самим тим задовољство корисника услуга уз повећање лојалности осигуравајућој компанији.[38] Компанија LBi Software посебно је поносна на лакоћу којом корисници приступају и користе систем. Након иницијалног пријављивања (Слика 2.2) улази се у веома прегледан и свеобухватан кориснички интерфејс (Слика 2.3). Од посебног значаја за дисертацију је могућност наведеног софтверског решења да обезбеди менаџерима управљање на основу растуће базе знања коју чине стандардни проблеми и њихова решења.

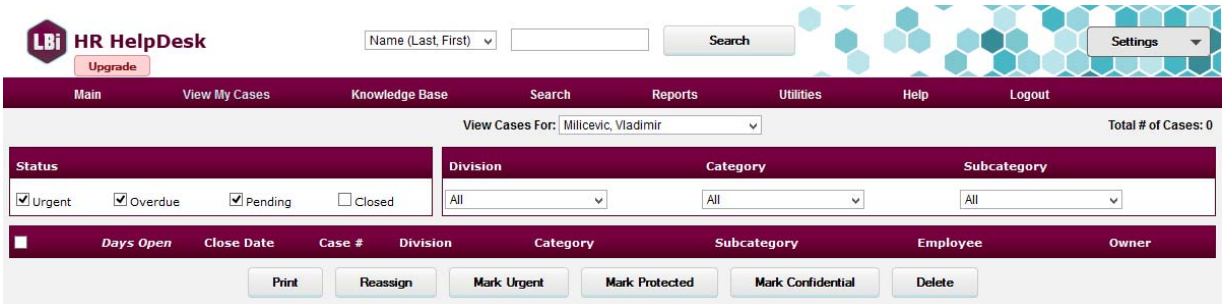
Област истраживања на коју се ослања дисертација јесте, такође, могућност препознавања лажних изјава у одштетним захтевима корисника осигурања и сервирање правовремене и прецизне информације службенику компаније који обрађује захтев за одштетом путем HelpDesk система. На овом пољу, изузетно је значајно решење компаније IBM, под називом *IBM Claims Fraud Solution*, које осигуравајућим компанијама нуди алате за детектовање и елиминисање одштетних захтева у случају давања лажних изјава. *Ово софтверско решење је способно да прати и детектује лажну изјаву осигураника у свим фазама животног циклуса праћене изјаве* (Слика 2.4).[39]

У прошлости, где системи за превенцију и борбу против оваквих типова изјава нису били на адекватном нивоу, осигуравајуће компаније су имале велике губитке надокнађујући штету која је документована лажном изјавом. Развијене и богате земље издвајају огромна средства за превенцију и борбу против наведене појаве. У 2010. Конгрес Сједињених Америчких Држава одобрио је буџет од 1.7 милијарди USD за борбу против превенције превара у систему здравственог осигурања.[40]

[38] www.lbisoftware.com . www.lbisoftware.com/casestudies/LBi-HR%20HelpDesk-Case%20Study.pdf (приступљено 02.11. 2014. у 16:10)
[39] www.ibm.com. www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=SP&infotype=PM&appname=SWGE_IM_IM_USEN&htmlfid=IMS14396USEN&attachment=IMS14396USEN.PDF (приступљено 02.11. 2014. у 16:40)
[40] Torpey D. (2011). Fraud in insurance on rise – survey 2010-2011, Ernst & Young 2011

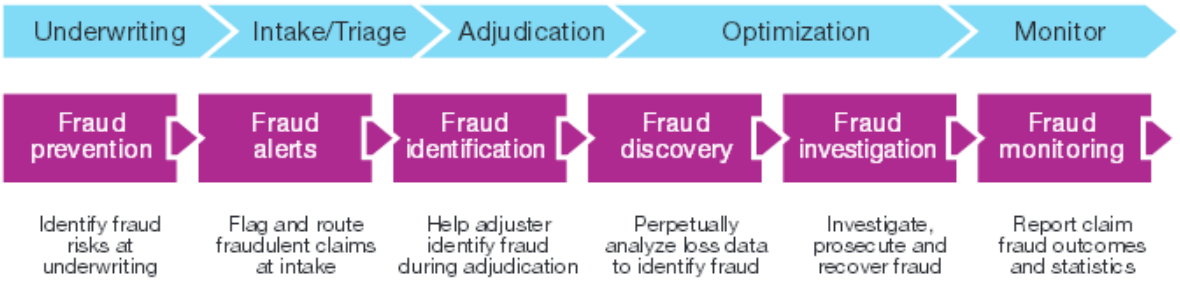


Слика 2.2. *LBi HR HelpDesk* екран за пријављивање (извор www.lbihrhelpdesk.com [41])



Слика 2.3. *LBi HR HelpDesk* почетни екран (извор www.lbihrhelpdesk.com[42])

[41] www.lbihrhelpdesk.com. www.lbihrhelpdesk.com/HRHelpDesk/login.do (приступљено 03.11. 2014. у 10:10)



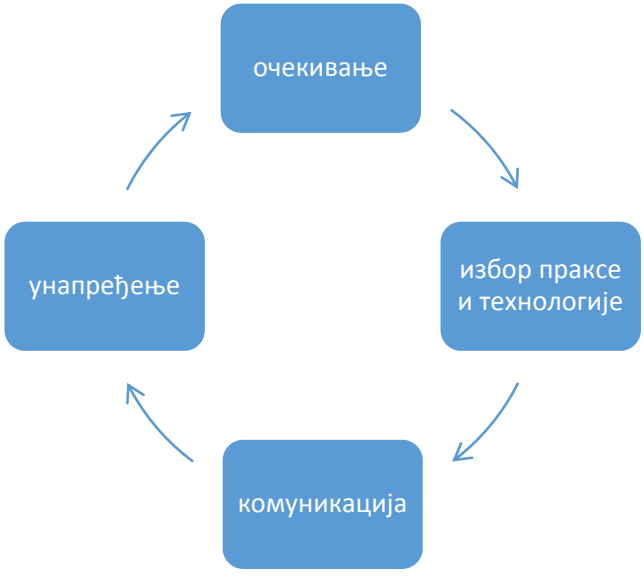
Слика 2.4. Логика решења *IBM Claims Fraud Solution* (извор www.ibm.com [40])

У светлу наведених чињеница, развој напредних рачунарских система који ће бити у могућности да у потпуности елиминишу појаву лажних изјава у одштетним захтевима клијената биће од приоритетног значаја за водеће компаније из области осигурања. Постоје бројни приступи за трагање ка могућем решењу. Овај рад је предложио приступ развоја хибридног рачунарског система који користи интелигентну компоненту за претрагу у комбинацији са софтверским развојем који омогућава динамичко управљање базама знања. За запослене у осигуравајућој компанији, овакав систем, путем HelpDesk компоненте, тренутно ће одговорити на питање: *Да ли је одштетни захтев клијента сумњив или не?*

Такође, овакав систем мора да има и способност учења и динамичког прилагођавања на све новонастале околности везане за пословање компаније чије пословање сервисира.

3. Примери имплементације и могућност унапређења HelpDesk система

У савременом пословном окружењу HelpDesk системи имају значајну улогу у подизању квалитета комуникације између експерата, са једне стране, и клијент корисника са друге стране. Клијент корисници, било да су то запослени у компанији или крајњи корисници производа и услуга, својим захтевима директно утичу на квалитет и функционалност самог HelpDesk система. Данас готово да и не постоји област индустрије и услуга у којима није могуће наћи имплементацију HelpDesk система. По аутору Џону Санденбергу: *Све што HelpDesk мора да уради јесте обезбеђивање ефективне комуникације која води ка експедитивном решавању проблема у складу са корисничким очекивањима.*[43] По организацији Forrester Research[44], креирање квалитетног HelpDesk система лежи у познатом циклусу успеха: управљање очекивањем, праћење проверених лидера - избор најбоље праксе и технологије, креирање нове културе комуникације и менаџмента и унапређењу пословања (слика 3.1).



Слика 3.1. Циклус успеха HelpDesk система

[43] www.kineticdata.com. http://www.kineticdata.com/news/media-coverage/5-steps-to-a-better-service-desk.html (приступљено 25.11.2014. у 16:30)
[44] https://www.forrester.com (приступљено 25.11.2014. у 16:45)

Водећи се наведеним задацима водеће компаније из индустрије софтвера нуде различита HelpDesk решења за подршку компанијама из различитих домена индустрије и услуга. Циљ овог дела рада је приказивање репрезентативних решења имплементираних у Србији и развијеном делу света, поређење таквих решења и идентификовање могућих праваца унапређења постојећих HelpDesk система.

3.1. Имплементација HelpDesk система у Републици Србији - пример Пореске управе Републике Србије

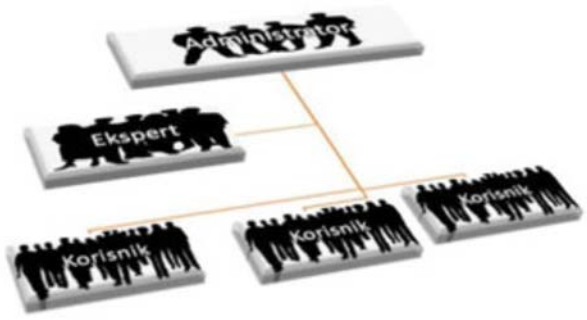
HelpDesk систем Пореске управе Републике Србије (ПУ) је међу првима уведен у некој од јавних организација у Србији, и као такав, идејом је предњачио у односу на друге постојеће системе, како у Пореској управи, тако и у осталим органима који припадају државној управи. Био је новина у начину комуникације између запослених јер до тада сва решења за различите проблеме, морала су да буду да тражена од виших организационих јединица ПУ путем телефона или писаних захтева упућених преко поште.

У Пореској управи ради око 6.000 радника. Већина радника у свом свакодневном раду користи рачунар. Информациони систем Пореске управе је доста комплексан, развијан на различитим оперативним системима (Windows и Unix), као и различитим програмским платформама (FoxPro, Clipper, Visual Basic 6.0, MS Access, ASP.NET, php). Запослени у свом раду користе MS Word пишући различите записнике и решења, MS Excel за попуњавање различитих табела, MS Outlook за примање и слање електронске поште као и различите веб претраживаче.[45]

У оваквом информационом систему, који није обједињен и развијан на једној системској и програмској платформи, запослени у ПУ често наилазе на проблеме који отежавају коришћење самог информационог система. У циљу бржег пријављивања и отклањања истих проблема, постављена је веб апликација HelpDesk, која представља механизам питање - одговор, помоћу којег се постављају питања и на њих, у што краћем временском периоду, добијају одговори.

[45] Беговић Б, Атанасијевић С, Дулић С, **Милићевић В**. *Примена HELPDESK-а у функцији повећања ефикасности пословања у државној управи*, (Копаоник 2014, YU INFO 2014 – зборник радова, 23-28).

HelpDesk је веб апликација инсталирана на Apache серверу у централи ПУ са адресом http://10.1.6.12/helpdesk/. Ово је *OpenSource* веб апликација, чија инфраструктура треба да омогући својим корисницима добијање подршке за проблеме на које наилазе, на такав начин да се за исте постављају питања или приступа бази знања или бази најчешће постављених питања (FAQ). Све врсте питања која се односе на HelpDesk (тј. проблеме везане за апликације које се користе у ПУ, питања везана за рачунарску мрежу, па чак и питања везана за систем и законе који се примењују у ПУ , итд) могу бити постављена од стране корисника, а адекватан одговор враћен од тима за подршку. HelpDesk - експерт, који је део тима за подршку, ће се побринути за упутства у вези насталог проблема, као и контроле везане за ефикасност предложеног упутства за решење проблема (слика 3.2).



Слика 3.2. Хијерархија HelpDesk система Пореске управе Републике Србије (извор: Беговић Б, Атанасијевић С, Дулић С, Милићевић В.[45])

3.1.1. Механизам питања – одговори

Корисник пријављује проблем или поставља питање попуњавањем захтева на веб апликацији HelpDesk система Пореске управе и тај захтев ће имати јединствену бројчану вредност или референцу. Захтев у том тренутку има статус *Отворен* и задржава статус док не буде додељен неком експертском тиму на решавање. Приликом попуњавања захтева, проблем треба класификовати унутар експертске групе, као и нивоа на којем ће проблем бити решаван. Оваква класификација проблема није коначна и може се променити на

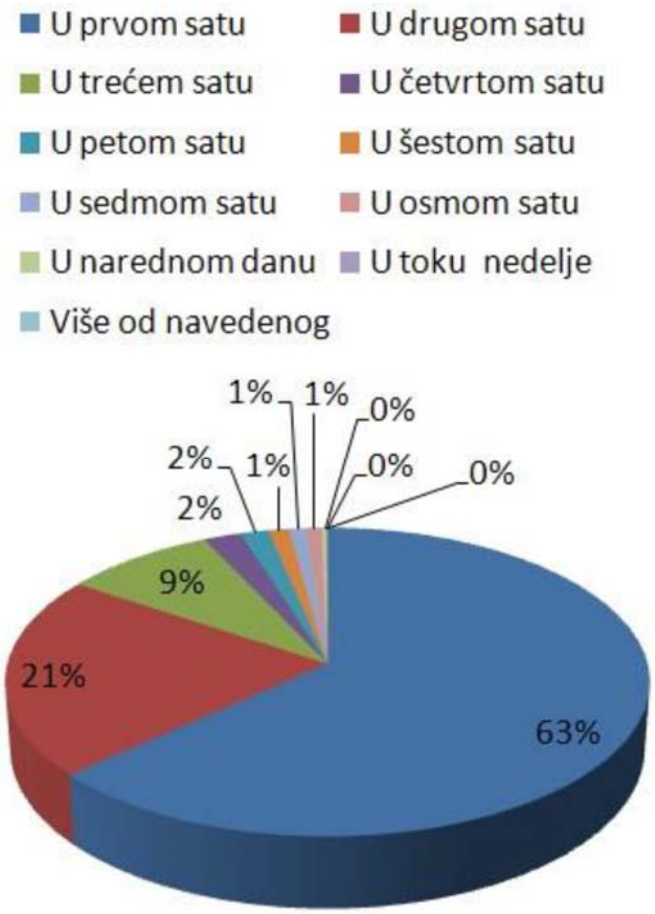
експертском нивоу, али би требало да помогне у бржем решавању проблема, слањем истог конкретном експертском тиму, који ће се побринути за решење текућег проблема.

Експерти који одговарају на њима упућен захтев, могу извршити додатну класификацију на друге експертске групе или променити ниво решавања проблема, ако сматрају да првобитна класификација од стране корисника или експерта није правилно урађена. У случају да на исти захтев може да одговори неколико експерта, који припадају истој експертској групи, он се додељује експерту који има мањи број додељених захтева, а који имају статус *Решава се*. Ипак, захтев се може распоредити, у било ком тренутку, било ком експерту из експертске групе независно од броја додељених захтева, без обзира на њихов статус.

Када експерт одговори на добијени захтев, преводи га у статус *Решен*, и тиме покреће механизам којим ће корисник који је пријавио тај проблем бити обавештен да је добио одговор и могуће решење изложеног проблема. Експерт, уз поштовање одређених критеријума (колико је од значаја за остале кориснике постављени проблем, његова учесталост, број захтева упућених за исти проблем итд...) уписује решење за изложени проблем у базу знања, како би био доступан за све кориснике.

3.1.2. HelpDesk база знања

У овом случају, база знања је база текстуалних података која садржи класиране проблеме пријављене од стране корисника са датим одговарајућим решењима. Треба напоменути да то није софтверска компонента класе информационих система познатих као експертни системи у којој је похрањена експертиза у форми пословних правила која управљају процесом доношења одлука. База знања конкретног HelpDesk система заправо представља скуп текстуалних података са одговорима на најчешће постављена питања. Као таква она не даје могућност систему да самостално доноси одлуке већ је експертиза у великој мери ослоњена на људски фактор. У поређењу са савременим HelpDesk системима имплементираним у компанијама и организацијама развијеног света, овај систем поприлично заостаје када се ради о брзини реаговања на постављена питања (Графикон 3.1).



Графикон 3.1. Статистика брзине реаговања HelpDesk система ПУ (извор: Беговић Б, Атанасијевић С, Дулић С, Милићевић В.[45])

Примена овако структурираног и организованог HelpDesk система донела је значајно повећање ефикасности рада: већ се у првом сату решавају се две трећине захтева, у прва два сата 85%, а у прва 3 сата 94%. На овај начин издвојени су најкомплекснији захтеви, за чије разрешење треба укључити спољне сараднике или за чије се разрешење захтева посебна опрема. На овај начин, овако филтрирани захтеви постају битни полазни елементи за израду плана ризика и даљег побољшања ефикасности, поузданости и одрживости пословног система.

Међутим, могуће је приметити да за пословне субјекте, чији је квалитет пословања у директној вези са временом реаговања на постављена питања, овакав систем не даје

довољно добре перформансе и као таквог га је могуће у значајној мери унапредити. Од посебног значаја за овај рад је немогућност самог система да самостално учи на основу постојећих шаблона правила у бази знања, а као такав није способан ни да аутоматски проширује властито знање додавањем нових правила у базу знања. Отуда информациони систем Пореске управе Републике Србије не може да се сврста у класу интелигентних рачунарских система чијом применом би се у великој мери повећала брзина реаговања на постављена питања уз померање експертизе од човека ка рачунару. Управо структурирање, изградња и имплементација оваквог система, са интелигентном HelpDesk компонентом, представља примарни циљ овог рада.

3.2. Имплементација HelpDesk система у развијеним земљама

3.2.1. Презентација система *KronoDesk*

KronoDesk је моћан HelpDesk и представља производ компаније Inflectra чије је седиште у Силвер Спрингу, Мериленд, Сједињене Америчке Државе. Компанија делује на глобалном нивоу и њене услуге користе бројне компаније и организације из САД, Европе, Јужне Африке, Тајланда, Аустралије, итд.[46]

KronoDesk данас представља синоним за напредно решење из HelpDesk области. На његовом одржавању и унапређењу свакодневно ради велики број инжењера и експерата из разних области, при чему се посебан акценат ставља на константно проширивање корисничког искуства значајног за функционисање самог система.

Компанија Inflectra је приликом изградње и имлементације наведеног софтвеског решења за стратешког партнера изабрала софтверског гиганта Microsoft са којим заједно наступа приликом презентације *KronoDesk* решења и освајања нових тржишта. Отуда, *KronoDesk* је у потпуности развијен на технологијама компаније Microsoft и за функционисање захтева следеће технологије:[46]

- .NET окружење верзија 4.0;
- MS SQL Server;

[46] www.inflectra.com. https://www.inflectra.com/Company/In-The-News.aspx (приступљено 26. 11. 2014. у 9:20)

- Internet Explorer;
- MS IIS;
- ASP.NET 4.0

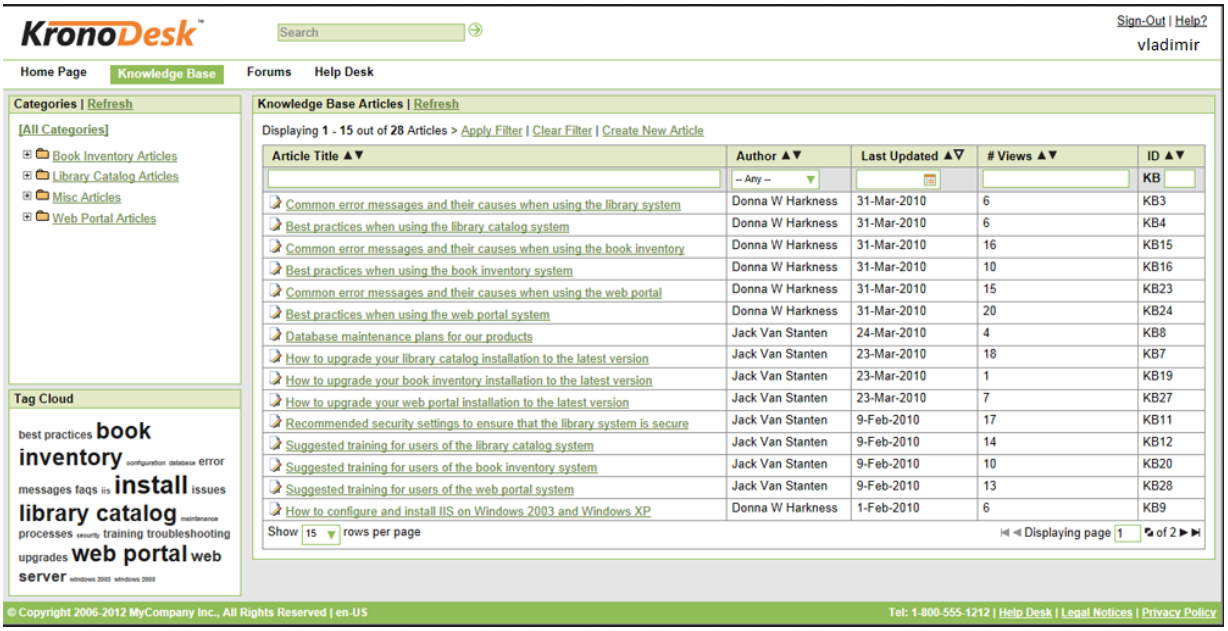
KronoDesk клијент се преузима, након куповине лиценце, директно са сајта компаније, лако се инсталира и има изузетно висок степен прилагођавања интегрисаних сервиса захтевима и навикама корисника. Компанија дозвољава бесплатно коришћење софтвера у трајању од 30 дана, а након тога омогућава нови период од 30 дана након којег, у случају незадовољства софтверским решењем, корисницима враћа уложени новац.

Само софтверско решење је изузетно оптимизовано, управо захваљујући тесној вези са компанијом Microsoft, тако да ради веома елегантно, неоптерећујући рачунарске платформе чак и из нижег ценовног ранга. Дакле, ради се о софтверском решењу које се годинама уназад налази на званичној листи најбољих софтверских решења из из HelpDesk области.[47]

KronoDesk база знања

Од посебног значаја за овај рад јесте начин чувања експертизе у бази знања и њеног коришћења у механизму питање - одговор. *KronoDesk* решење поседује базу знања која, захваљујући великом обиму коришћења софтвера, свакодневно се проширује омогућујући корисницима решење за велики број питања из разних области увидом у библиотеку најчешће постављених питања. Приступ бази знања је омогућен путем инсталираног клијент софтвера који корисника упућује на веома прецизно истакнуте, подељене и организоване категорије питање-одговори (слика 3.3).

[47] www.capterra.com. http://www.capterra.com/help-desk-software/ (приступљено 26.11. 2014. у 9:55)



Слика 3.3. Приступ KronoDesk бази знања (извор: www.inflectra.com[46])

Компанија Inflectra посебну пажњу ставља на задовољство корисника *KronoDesk* система тако да сваког месеца, а на основу новог знања ускладиштеног у бази, електронским и штампаним путем обавештава кориснике о новим проблемима и решењима из области значајних за кориснике. Везано за ову проблематику, софтвер користи интелигентне технике, неуронске мреже, којима повезује проблеме и њихова решења са базом података корисника.[48] На овакав начин, корисник добија огромну уштеду у времену када је у питању прибављање одговора на постављено питање тако да његов посао може да се одвија експедитивно, без великих интервала чекања на реакцију система или експерта.

Међутим, систем *KronoDesk* и даље пуни базу знања решењима која захтевају значајну интервенцију експерата. Иако систем користи интелигентне технике, он није у потпуности спреман за већи степен аутоматизације система питање – одговор. За разлику од претходно наведеног случаја, база знања овде представља софтверску компоненту која има веома елегантну и прегледну репрезентацију у форми текстуалних извештаја приказаних клијент *KronoDesk* апликацијом. Правила у бази су организована по стандардном *if – then - else* сценарију и као таква нису погодна за аутоматску

[48] www.inflectra.com.http://www.inflectra.com/KronoDesk/Documentation.aspx (приступљено 26.11.2014. у 10:20)

имплементацију у постојеће софтверско решење. Такође и технологија пројектовања и програмирања овог софтверског решења не дозвољава динамичко имплементирање новооткривеног знања. Отуда, у наредним излагањима посебан акценат ће бити стављен на идентификовање техника и алата за пројектовање и изградњу напредног решења из HelpDesk области способног да се динамички суочи са новим знањем.

3.2.2. Презентација система *SysAid*

Компанија *SysAid* описује своје софтверско решење као HelpDesk систем који функционише на јединственој платформи и ефикасно управља свим ИТ задацима. Ради се о моћном управљачком алату са напредним аутоматизованим инструментима.[49] Као такво, ово софтверско решење је веома значајно за анализу која је претходила изради овог рада.

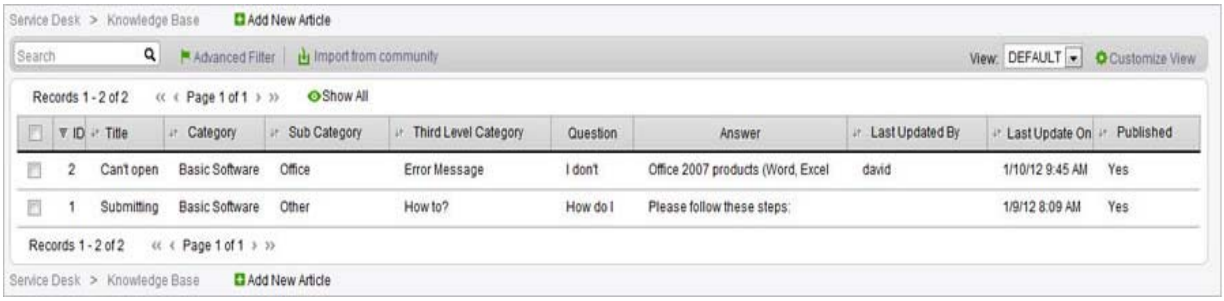
Седиште компаније *SysAid* се налази у Израелу, за тржишта Европе и Азије, и у САД за тамошње тржиште. Као и претходно решење у потпуности је изграђено и оптимизовано за коришћење на технологијама компаније Microsoft. Такође, већ годинама се налази на званичној листи најбољих HelpDesk решења[47] са посебним акцентом на интелигентну компоненту.

SysAid база знања

SysAid платформа функционише на веома специфичан начин, база знања се налази на серверу у Израелу и директно се пуни са два чворишта – Њу Јорк и Тел Авив. Интелигентна компонента, неуронска мрежа, готову експертизу преузима, тестира и, ако се ради о новом знању, задужени администратор добија обавештење да је ново правило неопходно уградити у базу знања. Неуронска мрежа ради синхронизовано са преводилачким софтвером тако да сва правила тј. решења на постављене проблеме истовремено су доступна на три језика: енглеском, хебрејском и арапском. Садржају базе знања веома се лако приступа помоћу веб апликације инсталиране на *SysAid* веб серверу (слика 3.4). Садржај је организован по

[49] www.sysaid.com. https://www.sysaid.com/features (приступљено 26.11.2014. у 10:55)

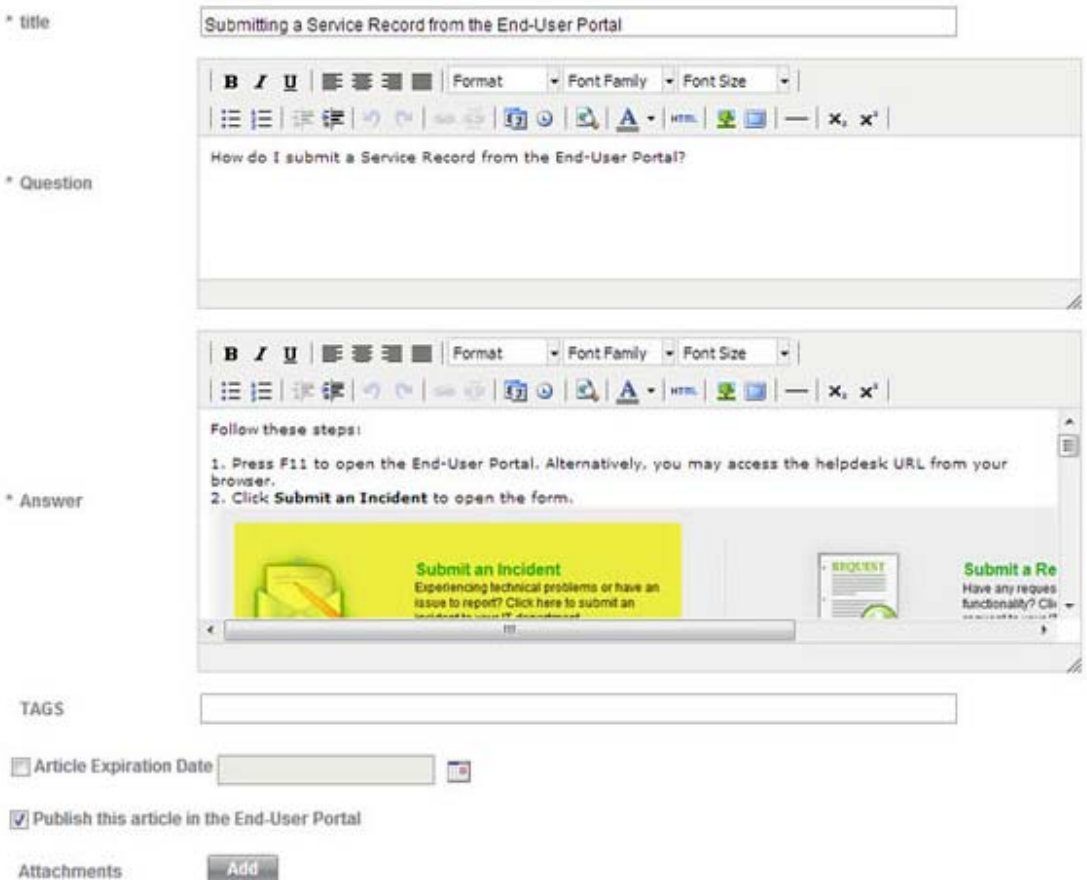
категоријама тако да корисник, пре постављеног питања, лако може да провери да ли је решење за његов проблем већ присутно у бази знања.



Слика 3.4 – *SysAid* механизам питање-одговор (извор: www.sysaid.com[49])

Такође, сам процес пријављивања проблема је веома једноставан путем странице која је веома прилагодљива потребама и навикама корисника (Слика 3.5).

База знања *SysAid* решења, због свог дистрибуираног пуњења и великог тржишта које систем покрива, изузетно је велика и под контролом великог броја администратора – експерата различитих профила. Својим корисницима нуди велику брзину одзива на постављена питања, а то резултује повећаним задовољством самих корисника. Међутим, од посебног значаја за овај рад је чињеница да и поред великог степена коришћења интелигентних алата, овај софтвер нема способност аутоматског интегрисања новог знања. Интелигентна компонента је посредник између два серверска чворишта и функционише тако да новооткривено знање, обележено аутоматским нумеричким идентификатором, препознаје тек ако је одговарајући експерт дати проблем обележио као *решено*. Компонента примећује да такав идентификатор не постоји у бази знања, претражује категорију проблема и, када је пронађе, упућује сигнал администратору да креира правило и да га имплементира у систем под одговарајућом категоријом.



Слика 3.5. Страница за постављање питања (извор: www.sysaid.com[49])

За унапређење оваквог система, дајући му могућност аутоматског препознавања, кодирања и имплементирања правила у властито софтверско језгро, неопходно је увести другачије технике пројектовања и програмирања. Иако су *KronoDesk* и *SysAid* водећа HelpDesk решења, изграђени су технологијама које онемогућавају динамичко повезивање новог знања и постојећег софтверског решења. Да би овај проблем био превазиђен, потребно је ангажовати алате и технике који припадају тзв. *OpenSource* сегменту. Од посебног значаја је организација базе знања. Стандардни *if – then - else* сценарио није добар, карактеристичан је за необјектни приступ и са много гранања доводи до *замршеног* (енг. crosscutting) програмског кода. Идеја рада је да коришћењем извесних проширења за програмски језик JAVA, правила и њихове везе са језгром апликације буду енкапсулирани и да имају форму специфичних објеката. Само софтвер највећег могућег степена модуларности, са изузетно

лабавим везама између модула, може да има успеха на пољу аутоматског имплементирања нових објеката. Отуда ће у даљем излагању бити говора о техникама и алатима за постизање наведених захтева.

3.3. Идентификовање праваца унапређења HelpDesk система

3.3.1. Аспектно-оријентисан развој софтвера

У протеклих 50 година програмски језици, а самим тим и методологије развоја софтвера, пролазили су кроз бројне еволутивне циклусе. Савремени објектно-оријентисани трендови настали су као тенденција да се софтверске компоненте моделирају као објекти из реалног окружења уз могућност њихове вишеструке поновне употребе.[50] Тако је објектно-оријентисана парадигма развоја софтвера постала доминантан начин софтверског размишљања од краја деведесетих година прошлог века до данас.

Управо поновна употребљивост софтверских модула је кључна предност објектно - оријентисаног начина развоја софтвера у односу на структурни. Међутим, искуства из праксе показују још нешто. И поред очигледних предности, објектна - оријентисаност не управља овим концептом како је било очекивано и софтверска решења постају све већа, а њихов развој и одржавање све компликованији. Посебно, објектно - оријентисана парадигма развоја софтвера још увек се није у потпуности изборила са неким наслеђеним проблемима, посебно у области развоја експертних система:[51]

- Код за повезивање пословних правила из базе знања са управљачком класом је комплексан и замршен (лит. пресечни код - crosscutting code);
- Извршива правила су ниског нивоа, тј. нивоа имплементације;
- Веза између правила и управљачких класа су круте;
- Извршиве везе су, такође, ниског нивоа;

[50] Highly T. J, Lack M, Meyers P. 2013. *A Critical Analysis Of A New Programming Paradigm* , Programming Languages – CS655 Semester Project
[51] Cibran M. A. 2007. *Connecting High Level Business Rules With Object Oriented Applications.* Vrije Universiteit Brussel

- Није могуће аутоматско управљање пословним правилима на основу постојећих техника објектно-оријентисаног развоја софтвера.

Отуда се пред софтверске развојне тимове поставља нови задатак – **креирање софтвера са највећим степеном модуларности и минималним везама између софтверских модула.** Очигледно је да постојећи објектно-оријентисани алати нису били у стању да се изборе самостално са наведеним изазовима па је било неопходно осмислити нову парадигму развоја софтвера. Револуционаран искорак ка новом софтверском размишљању дала је група аутора са Института Ксерокс из САД, предвођена професором Грегором Кишалесом. У својим радовима[52], с краја деведесетих година прошлог века, аутори уводе решења која се базирају на примени технике пројектовања и програмирања софтвера познатој као **аспектно-оријентисано програмирање.** Аутори су представили први аспектно-оријентисани приступ у савременој литератури познат као AspectJ.

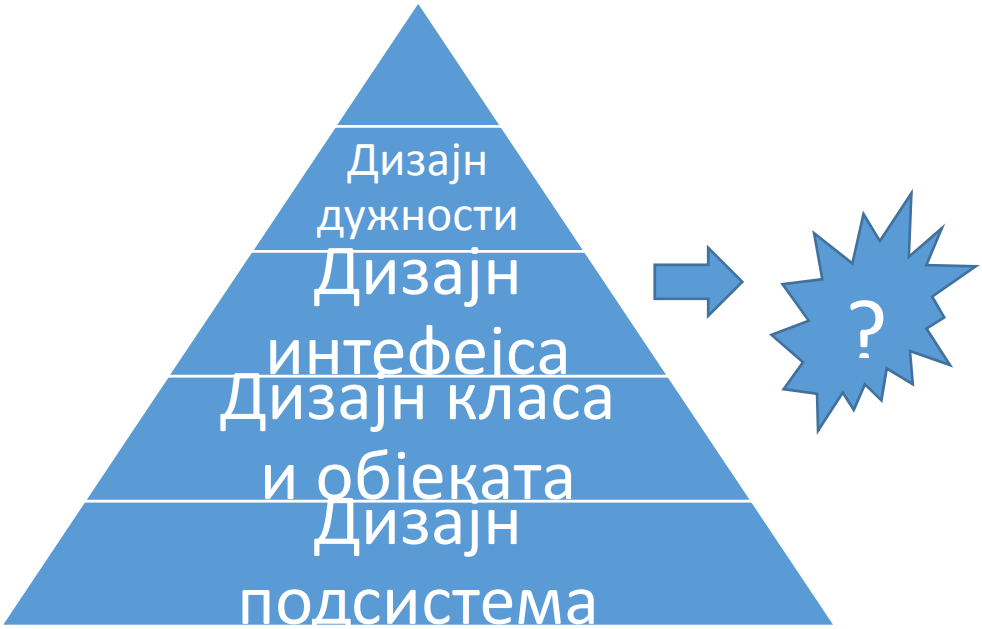
У наредним годинама долази до појаве нових аспектно-оријентисаних приступа и алата: AspectWerkz и JBoss. Од посебног значаја је нов приступ развијен с почетка двадесетпрвог века кроз истраживања и радове групе научника[53] са Врије Универзитета у Бриселу, окупљених око професора Вивијен Јонкерс и Маје Д' Хонд.[54] [55] [56] [57] Веома значајна за овај рад јесте студија[51] аутора Марије Сибран која је прва указала на могућност динамичког управљања пословним правилима применом аспектно-оријентисаних техника. Наведени приступ данас у литератури је познат као JasCo.

Током последњих година, аспектно-оријентисани програмери су остали јасно подељени између AspectJ и JasCo приступа. Приступ JBoss је у изворном облику напуштен

[52] Kiczales G., Lamping J., Mendhekar A., et al. 1999. *Aspect-Oriented Programming*, Xerox PARC, Palo Alto, CA. 1997.
[53] JasCo Community. http://ssel.vub.ac.be/jasco/community.html (приступљено 26.11.2014. у 12:55)
[54] Suvee D, Jonckers V, Vanderperren W. 2004. *Supporting JAsCo AOP*, ENTCS vol. 107.
[55] Suvee D, Vanderperen W, Jonckers V. 2003. JAsCo: An Aspect-Oriented Aproach forComponent Based Software, AOSD Boston.
[56] Cibrán M. A, Suveé D, D'Hondt M, Vanderperren W, Jonckers V. 2004. Integrating Rules with Object-Oriented Software Applications using Aspect-Oriented Programming, Proceedings of ASSE'04, Argentine Conference on Computer Science and Operational Research, Córdoba, Argentina, 2004.
[57] Cibrán M. A, D'Hondt M, Vanderperren W. 2003. Aspect-Oriented Programming for Connecting Business Rules. Proceedings of BIS International Conference, Colorado Springs, USA, June 2003.

и данас представља једну од AspectJ дистрибуција за JAVA развојни алат Eclipse.[58] Остали приступи нису имали озбиљнију примену.

Посебно је значајно напоменути да оба приступа имају властитих предности и недостатака. AspectJ је у масовнијој употреби, раније је *прележао дечје болести* што је довело до стабилнијих софтверских решења која се брже извршавају за разлику од софтверских решења у чијој изградњи је коришћен JasCo. С друге стране, JasCo има способност динамичког управљања знањем, а то је од пресудног значаја за циљеве овог рада. Отуда, као допринос унапређењу аспектно-оријентисаног начина развоја софтвера, у овом раду је инсистирано на комбиновању доминантних карактеристика оба приступа са циљем добијања унифицираног приступа који ће дати најбоље резултате у светлу постављених циљева. Као посебан задатак рада је давање одговора на следеће питање: *Да ли је аспектна-оријентисаност независна парадигма развоја софтвера или проширење постојеће објектно-оријентисане парадигме?* Другим речима, да ли ће врх познате објектно-оријентисане пирамиде (Слика 3.6) коначно добити име или не?



Слика 3.6. Пирамида објектно-оријентисаног дизајна

[58] https://**eclipse**.org/ (приступљено 27.11.2014. у 12:26)

3.3.2. Објекти правила

Идеја аспектно-оријентисаног развоја софтвера јесте да кроз кораке анализе, дизајна и програмирања, имплеметира софтверско решење са степеном модуларности који није могуће постићи актуелним објектно-оријентисаним техникама. То подразумева проширење објектно-оријентисаног приступа на базу података и на базу знања. Користећи алат *Hibernate ORM (Object-Ralation Mapping – Пресликавање објекти-релације)*[59], посебном нотацијом подаци из табела базе података добијају објектни облик и на ефективнији и ефикаснији начин бивају искоришћени од главног управљачког дела софтверског решења.

Што се тиче базе знања, проблем је још комплекснији. База знања представља софтврску компоненту са строго структурним обликом. Будући да је изграђена од великог броја *if – then - else* блокова и скокова у главни део програма по шаблону *goto*, она условљава обимно понављање програмског кода на различитим местима, а то је познато у литератури као пресечни код (crosscutting).[51] [60] Пресечни код чини софтвер споријим, тежим за одржавање и отклањање грешака, а такође онемогућава реализовање циљева овог рада који су у вези са аутоматским укључивањем нових правила у базу знања. По аспектно-оријентисаном начину пројектовања софтвера, нову компоненту је могуће укључити у активну софтверску апликацију без њених измена искључиво у случају када компонента представља одвојену и независну целину, повезану лабавим безама са управљачким класама апликације. Отуда овај приступ предлаже сасвим другачију организацију базе знања. Све њене компоненте морају да буду енкапсулиране у засебне модуле, и то:

- Пословна правила у објекте правила;
- Везе правила са главним делом апликације у специфичне објекте – аспекте.

На овај начин сви делови софтверског решења биће објектно - оријентисани, а то старим објектно-оријентисаним техникама није било могуће. Дакле, овај рад предлаже приступ по којем ће главни део софтверског решења бити изграђен на JAVA базираним објектно - оријентисаним алатима и техникама, уз примену Hibernate ORM нотације за

[59] hibernate.org. http://hibernate.org/orm/ (приступљено 27.11. 2014. у 11:08)
[60] Schwanninger C, Wuchner E, Kircher M. 2012.Encapsulating Crosscutting Concerns in System Software, Siemens AG

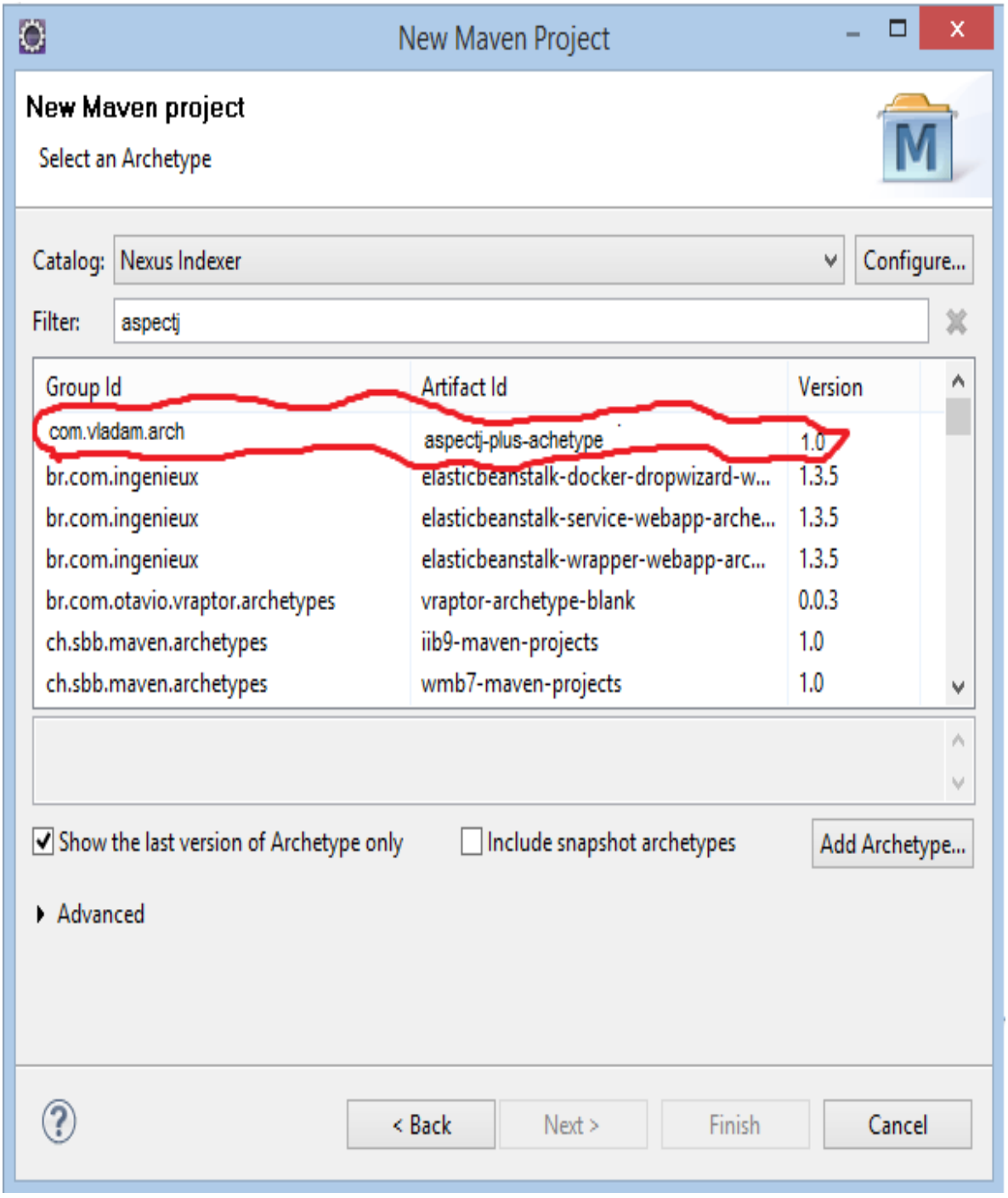
управљање базом података, а база знања ће бити пројектована и реализована применом адекватног аспектно-оријентисаног проширења.

Посебно у радовима[61] [62] JasCo аспектни приступ је прихваћен као репрезентативан приступ и на њему су базирани анализа, дизајн и кодирање базе знања. Међутим, примећено је да уколико се на погодан начин модификују аспекти веза AspectJ приступа могуће је добити нов приступ који комбинује најбоље елементе оба приступа. Управо ће то у неком од наредних излагања у овом раду бити презентовано, а то ће представља оригиналан аспектно-оријентисани приступ. Модификација аспеката организована је у посебан пакет који је назван *AspectJ+* и који је као *.jar* архивска датотека похрањен у веб JAVA репозиторијум који је могуће неограничено користити развојним алатом Eclipse (слика 3.7).

Када Eclipse укључи AspectJ+ пакет, превлачењем из репозиторијума, путем Интернета, аутоматски се генерише *pom.xml* (код 3.1) датотека која омогућава управљање и коришћење класа из овог пакета. Након креирања наведене датотеке, програмер може несметано да користи класе садржане у овом пакету, њихове атрибуте и методе, као и да креира објекте изведене из класа AspectJ+ пакета.

За софтверско решење, које представља стуб истраживања овог рада, наведено правило може да се односи на одобравање попуста клијентима осигуравајуће компаније који задовољавају одређени степен лојалности према производима и услугама компаније или се определе за исте у неком акцијском периоду (Код 3.2). С друге стране, ово апстрактно правило представља родитељску класу за читав низ подкласа – правила којима су регулисани специфични попусти. Кодом 3.3 дефинисана је подкласа настала наслеђивањем наведеног правила, а која се односи на акцијски попуст.

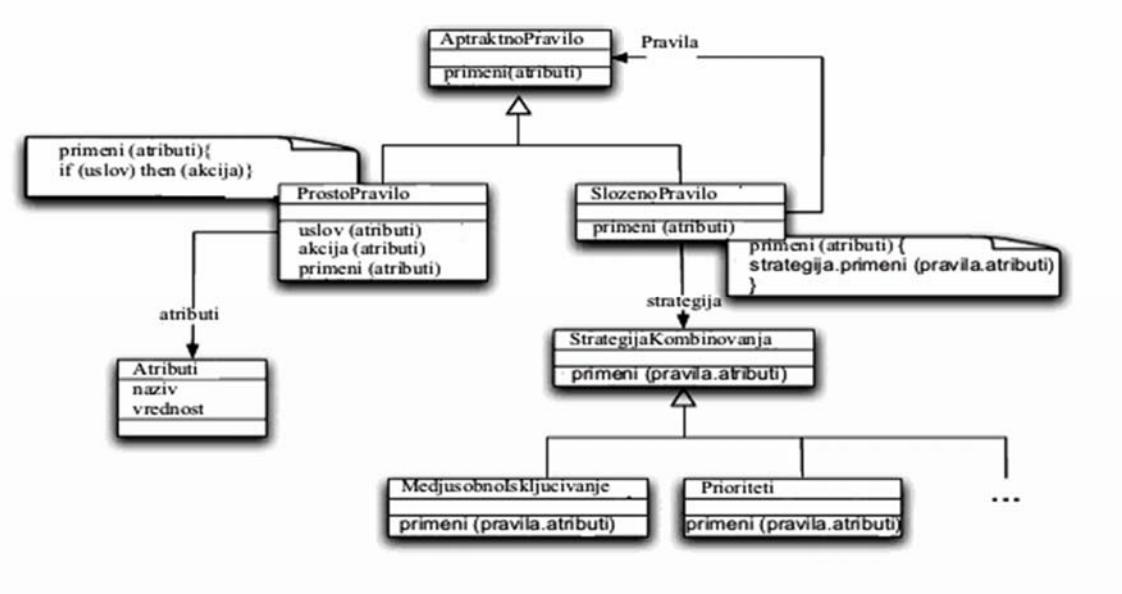
[61] Милићевић В. 2012. *Допринос развоју интелигентног рачунарског система базираног на приступу пословних правила*, ФАМС
[62] Stankić R, Milićević V, Popović M, Savić Z. 2012. Contribution to Intelligent System for Automatic Business Ruless Management Development, TTEM Vol. 7/1, 2012



Слике 3.7 – Избор *AspectJ*+ пакета алатом *Eclipse*

```
<?xml version="1.0" encoding="UTF-8"?>
<lifecycleMappingMetadata>
<pluginExecutions>
   <pluginExecution>
      <pluginExecutionFilter>
         <groupId>com.vladam.arch</groupId>
         <artifactId>aspectj-plus-archetype</artifactId>
         <versionRange>1.0</versionRange>
         <goals>
            <goal>compile</goal>
         </goals>
      </pluginExecutionFilter>
      <action>
         <ignore />
      </action>
   </pluginExecution>
</pluginExecutions>
```

Код 3.1. pom.xml датотека за укључени *AspectJ*+ пакет



Слика 3.8. Класа правило и концепт наслеђивања

```
abstract public class PPPopust{

      protected float procenat;
      abstract public boolean uslov();

      public Float akcija (Float cena) {
           return(new Float(cena*
           procenat/100));
      }

      public Float primeni(Float cena) {

           if (uslov())
             return akcija (cena);
          else return cena;
      }
}
```

Код 3.2. Класа правила

```
public class PPAkcijskiPopust extends PPPopust {


  public AkcijskiPopust () {
     procenat = 10;
  }

private boolean jeAkcija () { ... }
public boolean uslov () {
   return jeAkcija ();
 }
}
```

Код 3.3. Наслеђивање класе правила

3.3.3. Аспекти конекција правила са језгром пословне софтверске апликације

Кључни проблеми које аспектно-оријентисани развој софтвера настоји да превазиђе не леже у објектима правила већ у аспектима њихових веза са главним делом апликације. Аспекти морају да обезбеде што лабавије повезивање правила и језгра апликације, а да при томе додатно повећају степен модуларности софтверског решења. Из наведеног разлога, избор најбољег аспектно-оријентисаног приступа, у овом раду, ослања се пре свега на могућност кодирања и управљања аспектима веза.

Посебно је вођено рачуна о тачкама у извршењу програма, у којима долази до прекида (интерапта – енг. interrupt) да би се извршила логика неког пословног правила. Та тачка у извршењу програма назива се *тачка прекида*, а инструкција која је резултат примене правила назива се *савет*. Отуда се често каже да се објекат правила ангажује по сценарију тачка прекида – савет[51 61]. Скуп објеката правила који могу бити примењени на исту тачку прекида назива се *унија*. Концепт тачке прекида и избор сценарија по којем се у њој прекида програм, а да би се имплементирало правио, представља кључни алат за постизање циљева рада. Апликација, у тачки прекида, може бити прекинута по следећим сценаријима:

- Прекид пре (before) тачке прекида (код 3.4);
- Прекид у (around) тачки прекида (код 3.5);
- Прекид после (after) тачке прекида (код 3.6).

```
aspect DPrimenaPopusta{
        pointcut kalkulacijaCene():izvrsi (float Proizvod.uzmiCenu()), before
                                        kalkulacijaCene();
        {// pokretanje primene pravila...}
}
```

Код 3.4. Прекид пре (before) тачке прекида

```
aspect DPrimenaPopusta {
        pointcut kalkulacijaCene():izvrsi (float Proizvod.uzmiCenu()), around
                                        kalkulacijaCene();
        {// pokretanje primene pravila...}
}
```

Код 3.5. Прекид у (around) тачки прекида

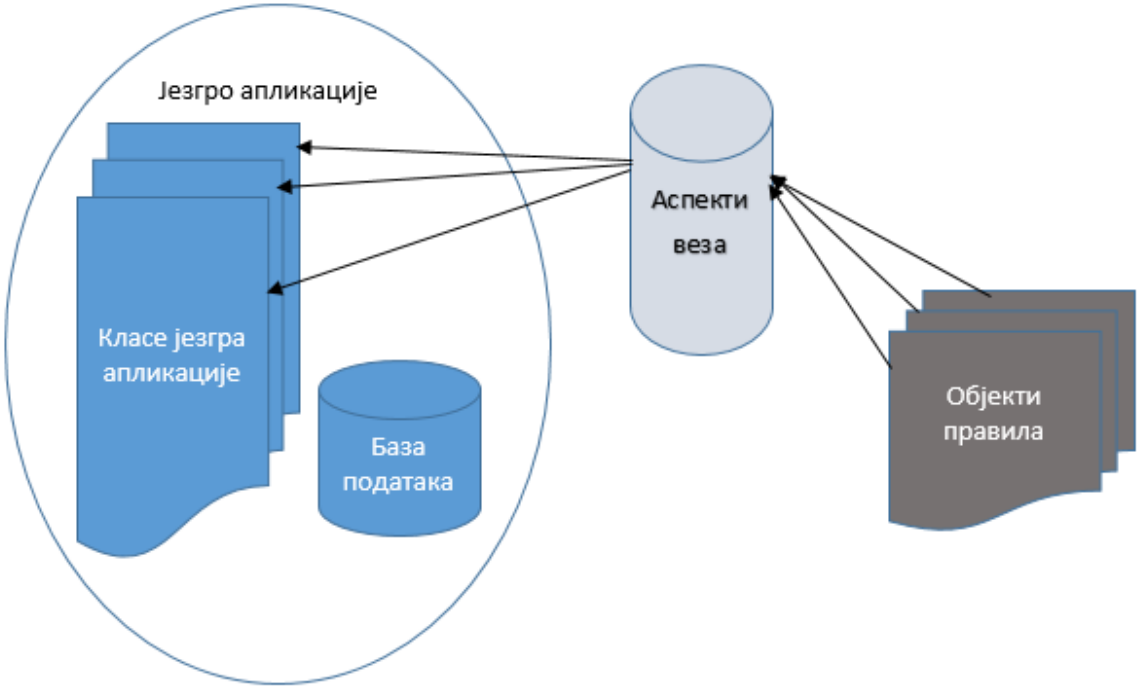
```
aspect DPrimenaPopusta {
        pointcut kalkulacijaCene():izvrsi (float Proizvod.uzmiCenu()), after
                                        kalkulacijaCene();
        {// pokretanje primene pravila...}
}
```

Код 3.6. Прекид после (after) тачке прекида

Посебно, у раду је предложено да називи објеката правила почињу увек са *PP* (од пословно правило), а називи аспеката веза са *D* (од догађај – који узрокује примену правила). Такође, могуће је приметити да су аспекти веза организовани као блокови наредби и да представљају специфичну објектну репрезентацију. На овај начин обезбеђена је потпуна модуларност софтверског решења и минимална веза објеката са остатком софтверског решења у тачки прекида. Приказани аспекти веза написани су AspectJ нотацијом и не дозвољавају аутоматско уграђивање новооткривеног правила у решење, већ би било неопходно поново, применом JAVA преводиоца, превести програм да би правило било оперативно. Међутим, овакви аспекти се брже и стабилније извршавају од аспеката динамичког JasCo приступа. Па ће, управо, следеће поглавље поредити решења оба приступа са циљем обједињавања појединачних доминантних карактеристика у нови приступ.

Дакле, објекти базе знања повезани су лабавим везама, у тачки прекида, помоћу аспеката конекције са класама и објектима управљачког дела софтверског решења (слика 3.9). На тај начин елиминисан је пресечни код (означен још и као *тиранија доминантне*

декомпозиције[63]), дуплирање кода је елиминисано у потпуности, степен модуларности и поновне употребе класа је на највећем могућем нивоу и процес отклањања грешака никада није био лакши будући да се за истом трага унутар једног модула.



Слика 3.9. Комуникација објеката правила са управљачким делом апликације

[63] D'Hondt M, D'Hondt T. 2002. The Tirany of Dominant Model Decomposition, Vrije Univesitat Brussel

4. Технологије и алати за развој интелигентног HelpDesk система

У овом поглављу приказана су два репрезентативна аспектно-оријентисана приступа. Посебан акценат ће бити стављен на аспекте веза правила са централним делом софтверског решења, на брзину њиховог извршавања, као и на могућност директног динамичког повезивања пословних правила током оперативности постојећег софтверског решења. Поред, брзине извршавања, критеријум за оцену квалитета аспеката веза биће и брзина одзива на *окидачки* догађај. Тестови ће бити вршени на узорку од педесет правила, за сваки од посматраних аспектно-оријентисаних приступа. Резултати ће бити приказани табеларно и графички, а након тога ће бити извучени и одговарајући закључци.

Из економске перспективе, оптимални приступ је значајан јер доводи до софтверског решења којим се пословни процеси извршавају брже, квалитетније и без отказивања система.

4.1. Анализа решења базираних на JasCo нотацији

Почеци примене овог приступа датирају од 2002. године објављивањем студије[64] о *Тиранији доминантне декомпозиције* аутора са Врије Унивезитета у Бриселу која први пут истиче да објектно-оријентисани алати не успевају у потпуности да се изборе са наслеђеним проблемима структурног приступа развоју софтвера. Убрзо од професора и истраживача оформљен је пројектни тим који ради на увођењу приступа пројектовању софтвера који би отклонио уочене недостатке савременог објектно-оријентисаног приступа. Већ после годину рада аутори Суве, Вандерпен и Јонкерс, на међународној конференцији у Бостону, посвећеној аспектно-оријентисаном начину развоја софтвера, представљају нови приступ – JasCo.[65] У наредном периоду ређају се радови[66] [67] којима су детаљно презентоване могућности овог приступа на различитим пољима и проблемима. Учешће у овим радовима,

[64] D'Hondt M, D'Hondt T. 2002. The Tirany of Dominant Model Decomposition, Vrije Univesitat Brussel
[65] Suvee D, Vanderperen W, Jonckers V. 2003. JAsCo: An Aspect-Oriented Aproach forComponent Based Software, AOSD Boston.
[66] Suvee D, Jonckers V, Vanderperren W. 2004. Supporting JAsCo AOP, ENTCS vol. 107.
[67] Cibrán M. A, Suveé D, D'Hondt M, Vanderperren W, Jonckers V. 2004. Integrating Rules with Object-Oriented Software Applications using Aspect-Oriented Programming, Proceedings of ASSE'04, Argentine Conference on Computer Science and Operational Research, Córdoba, Argentina, 2004.

поред наведених аутора, активно узима и аутор Мариа Агустина Сибран која је поставила основ за динамичко повезивање правила са централним делом софтверског решења. Овај аутор тврди: *Изградњом аспектних шаблона на највишем нивоу апстракције – нивоу домена, на основу којих настају извршива правила и везе на нивоу имплементације, обезбеђује се механизам који омогућава директно интегрисање правила у постојеће језгро апликације, без потребе за накнадним превођењем програма и без измена на осталим деловима софтверског решења*.[68] Рад овог аутора учврстио је JasCo на месту најнапредније технике за развој софтвера, поготово из домена веб оријентисаних система за пружање електронских услуга. Посебан значај, за овај рад, има *модел трансформација*[68] на основу којег се правила и везе правила са делом софтверског решења за управљање знањем, креирани језиком високог нивоа и веома блиском говорном језику, преводе у извршиве објекте правила и везе кодиране JasCo нотацијом. Ово је веома значајно зато што у креирању новог знања и његовој имплементацији у постојеће софтверско решење активно учешће могу да узму експерти различитих профила, а да не морају да буду упућени у програмирање. Прилагођавањем трансформација приступу који комбинује најбоље особине JasCo и AspectJ приступа, у овом рад је образложено, у неком од наредних излагања, како се од текстуалног узорка, који одговара HelpDesk питању, креира правило високог нивоа од којег се применом трансформација добија извршиво правило и одговарајући аспект везе.

4.1.1. JasCo објекти правила и аспекти веза

Софтверско решење, које представља основ за истраживање и постизање циљева овог рада, намењено је подршци управљању у компанијама из области осигурања. HelpDesk овог система већ поседује одређени број пословних правила унутар базе знања. У првом случају биће тестиран сет од педесет правила исказаних JasCo нотацијом. У наставку, биће тестиран идентичан скуп правила, али исказаних AspectJ нотацијом. На самом крају, правила ће бити кодирана предложеним комбинованим приступом уз извлачење конкретних закључака.

[68] Cibran M. A. 2007. Connecting High Level Business Rules With Object Oriented Applications. Vrije Universiteit Brussel

Нека је дата следећа класа (Код 4.1) за дефинисање објеката правила који се односе на лојалност корисника производа и услуга осигуравајуће компаније:

```
class PPLojalniKorisnik extends PPPopust {
      public boolean usloviPopusta (Korisnik korisnik) {
           return LojalniKorisnik.is LojalniKorisnik(korisnik);
            }
 }
```

Код 4.1. Пословно правило *PPLojalniKorisnik*

На основу шаблона *PPPopust* (Код 3.1) наслеђивањем је креирана класа *PPLojalniKorisnik* која, даље, представља шаблон за креирање великог броја објеката правила за исказивање корисничке лојалности према компанији. Логичка функција *usloviPopusta()* проверава да ли корисник испуњава услове лојалности на основу којих му се одобравају различите погодности и попусти приликом куповине производа и услуга. Из овог примера могуће је видети да код класа и објеката правила у потпуности одговара коду програмског језика JAVA. Специфичности кода приступа виде се тек код кодирања аспеката веза (код 4.2). У JasCo нотацији, да би правило успешно било реализовано у тачки прекида, кључну улогу врши метода *хватач* (енг. hook). Ова метода чека реакцију аспекта везе на прекид у извршењу тока управљачког дела софтверске апликације. Чим вредност прекида добије логичку вредност *тачно*, *хватач* се аутоматски активира, везује правило за тачку прекида и реализује га у зависности од сценарија: *before, around* или *after*.

```
class LojalniKorisnik {
  public boolean proveriLojalniKorisnikUslovi (Korisnik korisnik) {
          if (!LojalniKorisnik.isLojalniKorisnik (korisnik)) {
                int Proizvodi = korisnik.getNalog().getKupljeniProizvodi();
               return Proizvodi > 5;
           } else return false;
}
```

```
hook LojalniKorisnikHook {
    LojalniKorisnikHook(Float method(Korisnik korisnik)) {
            execute(method);
    }
    isApplicable() {
   return proveriLojalniKorisnikUslovi (korisnik);
    }
    after() {
             LojalniKorisnik.addLojalniKorisnik (korisnik);
        }
 }
}
```

Код 4.2. Аспект везе пословног правила *PPLojalniKorisnik*

4.1.2. Идентификовање предности и недостатака JasCo приступа

Аспектом, приказаним Кодом 4.2, остварује се веза по сценарију *после* где се кориснику одобрава попуст уколико је задовољио услове лојалности – куповину више од пет производа осигуравајуће компаније. Методом *isApplicable()* активира се правило у тачки прекида и извршава се тек након потпуног заустављања управљачког дела софтверског решења, а то је одређено методом *after()*. Управо ове две методе реализују динамичке карактеристике JasCo приступа. JasCo апсекти веза ослањају се на апстрактне модуле – класе, које на нивоу имплементације добијају конкретне вредности својих атрибута и метода. Управо због високог нивоа апстракције модула који учествују у репрезентацији знања, динамичко интегрисање правила је могуће током извршавања софтверског решења, а без потребе за поновним превођењем. Међутим, аспектне JasCo и JAVA нити (софтверски процеси) показују тенденцију *сударања*, па због нешто слабијег степена синхронизације спорије се извршавају од комбинације аспектних AspectJ и JAVA нити

Перформансе решења изграђеног применом JasCo приступа

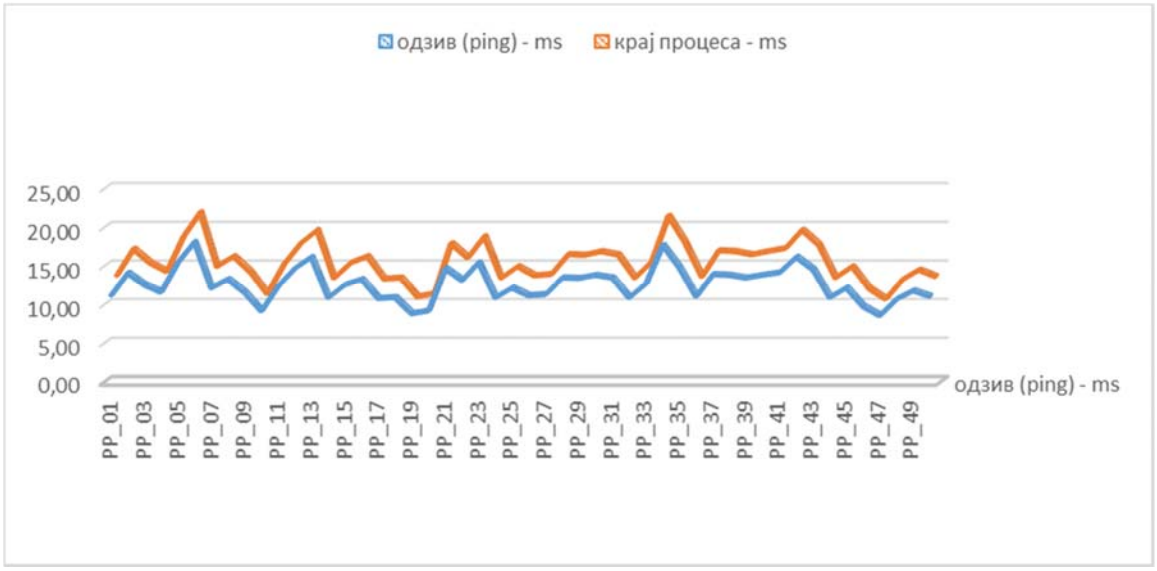
. У овом делу рада, из разлога разумљивијег образлагања логике објеката правила и аспеката, коришћени су строго персонализовани називи за правила и аспекте. У реалној имплементацији, у бази знања свако правило поседује генерички назив који почиње са *PP* и наставља се додавањем аутоматског нумеричког идентификатора (по принципу autoNumber()). Следећом табелом обухваћен је скуп од педесет JasCo правила од којих је свако покренуто једанпут, по одговарајућем сценарију, при чему су мерене перформансе које се односе на брзине реаговања на позив и извршавање. Ове перформансе су у директној вези са стабилношћу и брзином извршавања целокупног софтверског решења. Њихове лошије вредности указују на повећан број процесних конфликата и машинских инструкција, добијених превођењем софтвера, неопходних да се логика правила реализује језгром софтверског решења. Тест је покренут обраћањем серверу на којем је подигнут HelpDesk са клијента повезаног на сервер локалном мрежом. Резултат тест је приказан Графиконом 4.1.

Табела 4.1. Тест над JasCo базом знања

База знања	одзив (ping) - ms	крај процеса - ms
PP_01	11,33	13,37
PP_02	14,27	16,84
PP_03	12,74	15,03
PP_04	11,79	13,91
PP_05	15,66	18,48
PP_06	18,3	21,59
PP_07	12,33	14,55
PP_08	13,45	15,87
PP_09	11,75	13,87
PP_10	9,33	11,01
PP_11	12,58	14,84
PP_12	14,87	17,55
PP_13	16,33	19,27
PP_14	11,05	13,04
PP_15	12,74	15,03

PP_16	13,44	15,86
PP_17	10,96	12,93
PP_18	11,08	13,07
PP_19	8,99	10,61
PP_20	9,36	11,04
PP_21	14,88	17,56
PP_22	13,3	15,69
PP_23	15,66	18,48
PP_24	11,06	13,05
PP_25	12,33	14,55
PP_26	11,33	13,37
PP_27	11,48	13,55
PP_28	13,66	16,12
PP_29	13,58	16,02
PP_30	13,99	16,51
PP_31	13,63	16,08
PP_32	11,06	13,05
PP_33	12,88	15,20
PP_34	17,9	21,12
PP_35	14,99	17,69
PP_36	11,22	13,24
PP_37	14,08	16,61
PP_38	13,99	16,51
PP_39	13,64	16,10
PP_40	14,01	16,53
PP_41	14,33	16,91
PP_42	16,34	19,28
PP_43	14,74	17,39
PP_44	11,09	13,09
PP_45	12,33	14,55
PP_46	9,93	11,72
PP_47	8,75	10,33

PP_48	10,89	12,85
PP_49	11,96	14,11
PP_50	11,23	13,25



Графикон 4.1. Резултати тестирања перформанси базе знања изграђене JasCo приступом

Забележени резултати показују да је просечно време одзива правила (реакције аспекта везе) 12,85ms, док је просечно трајање процеса реализације правила 15,17 ms. Такође, очувана је пропорционалност између посматраних параметара за било која два правила, а то је могуће приметити на претходном графикону.

4.2. Анализа решења базираних на AspectJ нотацији

AspectJ приступ данас представља најшире коришћен аспектно-оријентисани приступ и означен је као *родитељ* савремених напредних техника програмирања и пројектовања. Овај приступ је осмишљен и уоквирен у Ксерокс (Xerox PARC) институту од стране групе истраживача предвођене професором Грегором Кишалесом. Први радови[69], у

[69] Kiczales G., Lamping J., Mendhekar A., et al. 1999. *Aspect-Oriented Programming*, Xerox PARC, Palo Alto, CA. 1997.

којима је овај приступ представљен, датирају још с краја деведесетих година прошлог века. Приступ је унапређен кроз две итерације о чему сведоче студије из 2002.[70] и 2007.[71] године.

Овај, најстарији аспектно-оријентисани приступ, посебну популарност је стекао када је означен као револуционарни заокрет у односу на традиционални објектно-оријентисани приступ. Као такав брзо је почео да буде изучаван у оквиру бројних напредних курсева и студијских програма.[72] [73] [74] Наведено има за последицу укључивање великог броја истраживача у развој и унапређење овог приступа тако да не постоји ниједан савремени JAVA развојни алат који нема интегрисане додатке за рад са AspectJ нотацијом. Посебно, данас најтраженији JAVA развојни алат Eclipse може да користи различите дистрибуције AspectJ архива. Бројна побољшања, пре свега у домену перформанси софтвера, јављају се интензивно од 2008. године са студијом коју је објавио професор Иван Кисељев.[75] Управо тада је AspectJ добио коначну дефиницију.

4.2.1. AspectJ објекти правила и аспекти веза

У овом делу рада акценат је поново стављен на програмски код 4.1. и на класу којом је дефинисана цела фамилија објеката правила за дефинисање лојалности корисника производа и услуга осигуравајуће компаније. Следећи задатак је креирање аспекта, AspectJ нотацијом, којим ће у тачки прекида неки од објеката ове класе бити повезан са управљачким делом софтверског решења. Тражени аспект приказан је програмским кодом 4.3. где се могу одмах уочити разлике између две нотације. JasCo користи екстерну методу *хватач* за повезивање са тачком прекида, а AspectJ дефинише методу унутар модула аспекта. Отуда, JasCo омогућава већи ниво модуларности софтверског решења него што је то случај са AspectJ приступом. Управо је то поље где се AspectJ приступ може додатно унапредити.

---

[70] Kisalesz G, Sung J. 2002. AspectJ , XEROX PARC
[71] Kiczales G, Lamping J, Mendhekar A, Maeda A, Videira Lopes C, Loingtier J. M, Irwin J. 2007. Aspect-oriented Programming, Xerox Palo Alto Research Center - 2007.
[72] www.cambridgecollegesummerschool.co.uk. http://www.cambridgecollegesummerschool.co.uk/spring-school (приступљено 28.11.2014. у 13:04)
[73] www.cs.uu.nl. http://www.cs.uu.nl/docs/vakken/swe/13-swe-aspectj.pdf (приступљено 28.11.2014. у 13:10)
[74] www.ics.uci.edu. http://www.ics.uci.edu/~lopes/(приступљено 28.11.2014. у 13:15)
[75] Kiselev I. 2008.Aspect-oriented programming with AspectJ ,Sams.

```
aspect DLojalniKorisnik {

    pointcut proveriLojalniKorisnikUslovi (Korisnik korisnik);//*
    isApplicable() {
        return proveriLojalniKorisnikUslovi (korisnik);
    }
    after LojalniKorisnik.addLojalniKorisnik (korisnik): {.....}


}
```

Код 4.3. Аспект везе правила и апликације у AspectJ нотацији

4.2.2. Идентификовање предности и недостатака AspectJ приступа

Најбитнија карактеристика решења изграђеног применом AspectJ приступа јесте веома мали број инструкција JAVA бајткода (машинског кода) неопходних да се аспектно-оријентисани модул прикључи и изврши у оквиру постојећег софтверског решења. Такође, процеси који прате извршавање AspectJ кода показују висок степен синхронизације са JAVA нитима, а то за последицу даје изузетно стабилна софтверска решења. Дакле, AspectJ код је мањи, бржи и стабилнији него што је случај са JasCo кодом. Међутим, AspectJ решење комуницира искључиво са објектима (инстукција * из кода 4.3), а не са апстракцијама као што су класе што је био случај са JasCo нотацијом. Овако конципиран AspectJ нема могућност динамичког уграђивања нових модула у тачки додира без исправки на софтверском решењу и накнадног превођења.

Перформансе решења изграђеног применом JasCo приступа

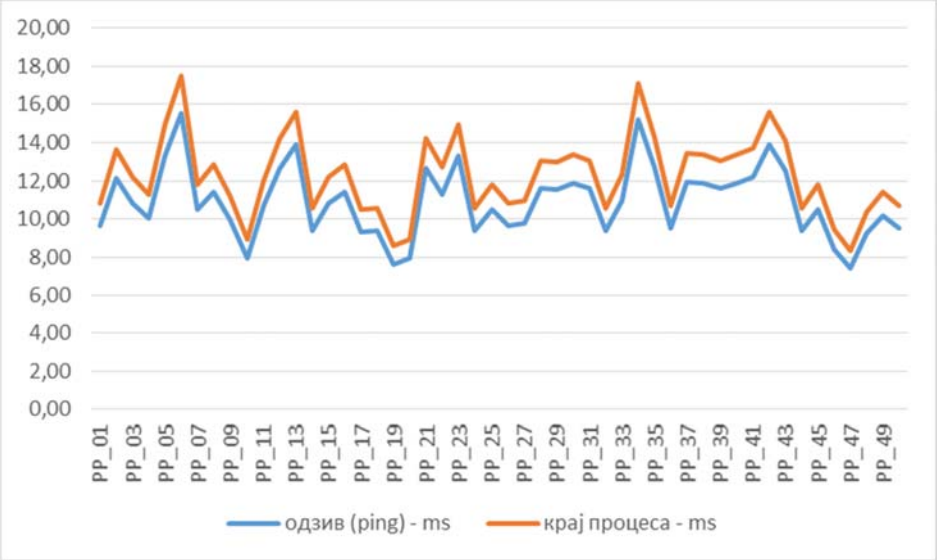
Као што је био случај са претходном нотацијом, и овде је за потребе примера објекту правила додељено име које га идентификује. Приликом креирања AspectJ базе знања, такође је инсистирано на генеричком одређивању назива правила. Правило почиње са PP, а онда по редоследу креирања добија одговарајући нумерички идентификатор. Следећом табелом обухваћен је скуп од педесет AspectJ правила од којих је свако покренуто једанпут, по

одговарајућем сценарију, при чему су мерене перформансе које се односе на брзине реаговања на позив и извршавање. Тест је покренут обраћањем серверу на којем је подигнут HelpDesk са клијента повезаног на сервер локалном мрежом, под идентичним условима као у претходном случају. Скуп правила у потпуности одговара скупу правила из разматрања обележеног под 4.1. Резултат теста је приказан Графиконом 4.2.

Табела 4.2. Тест над AspectJ базом знања

База знања	одзив (ping) - ms	крај процеса - ms
PP_01	9,63	10,83
PP_02	12,13	13,64
PP_03	10,83	12,18
PP_04	10,02	11,27
PP_05	13,31	14,97
PP_06	15,56	17,49
PP_07	10,48	11,79
PP_08	11,43	12,86
PP_09	9,99	11,23
PP_10	7,93	8,92
PP_11	10,69	12,02
PP_12	12,64	14,21
PP_13	13,88	15,61
PP_14	9,39	10,56
PP_15	10,83	12,18
PP_16	11,42	12,85
PP_17	9,32	10,48
PP_18	9,42	10,59
PP_19	7,64	8,59
PP_20	7,96	8,95
PP_21	12,65	14,22
PP_22	11,31	12,71

PP_23	13,31	14,97
PP_24	9,40	10,57
PP_25	10,48	11,79
PP_26	9,63	10,83
PP_27	9,76	10,97
PP_28	11,61	13,06
PP_29	11,54	12,98
PP_30	11,89	13,37
PP_31	11,59	13,03
PP_32	9,40	10,57
PP_33	10,95	12,31
PP_34	15,22	17,11
PP_35	12,74	14,33
PP_36	9,54	10,72
PP_37	11,97	13,46
PP_38	11,89	13,37
PP_39	11,59	13,04
PP_40	11,91	13,39
PP_41	12,18	13,70
PP_42	13,89	15,62
PP_43	12,53	14,09
PP_44	9,43	10,60
PP_45	10,48	11,79
PP_46	8,44	9,49
PP_47	7,44	8,36
PP_48	9,26	10,41
PP_49	10,17	11,43
PP_50	9,55	10,73



Графикон 4.2. Резултати тестирања перформанси базе знања изграђене AspectJ приступом

4.3. Поређење приступа и идентификовање њихових доминантних карактеристика

Забележени резултати показују да је просечно време одзива правила (реакције аспекта везе) 10,92ms, док је просечно трајање процеса реализације правила 12,28 ms. Тест је показао да брзина извршавања AspectJ аспеката веза у просеку за скоро 18% већа него што је случај са брзином извршавања JasCo аспеката веза. С друге стране, цео процес апликације одређеног пословног правила AspectJ приступом трајао је у просеку 12,28ms, а у случају JasCo приступа 15,17ms, што је за скоро 24% квалитетнији резултат.

Тест је потврдио управо оно што је и тврђено током целог овог поглавља: AspectJ приступ даје софтверска решења која се брже и стабилније извршавају, а то је од великог значаја за индустрију пословног софтвера. Квалитет пословних софтверских решења је у директној вези са наведеним атрибутима. Са друге стране, JasCo приступ даје степен модуларности софтверског решења који није могуће постићи ни са једним до сада познатим приступом развоју софтвера. Његово управљање апстрактним класама је алат који омогућава софтверским решењима висок ниво способности самосталног одржавања, а то подразумева укључивање новог знања, аутоматски, без интервенције развојног софтверског

тима. Такође, JasCo приступ омогућава активно учествовање непрограмерских експерата различитих профила у пројекте развоја пословног софтвера.

У светлу наведених тврђења, као један од циљева овог рада постављено је креирање унифицираног приступа који обједињује доминантне карактеристике анализираних приступа. Такав приступ је у раду назван AspectJ+ и он представља резултат оригиналног истраживачког рада. Посебно треба истаћи, да су пакети, класе и интерфејси, који омогућавају динамичко понашање AspectJ приступа, доступни путем Интернета сваком програмеру који користи Eclipse и Maven алате у оквиру главног пакета aspectj-plus-archetype (видети под 3.2.2).

4.3.1. Комбиновање најбољих карактеристика анализираних приступа

Задатак овог дела рада јесте да прикаже перформансе софтверског решења изграђеног применом иновативног аспектно оријентисаног приступа. AspectJ+ је конципиран на следећи начин:

- Класе објеката правила писани су основним JAVA кодом;
- Аспекти веза кодирани су AspectJ нотацијом издвајањем инструкције *pointcut* из аспекта, остављајући јој могућност комуникације са класама правила;
- Креиране су нове софтверске трансформације, којима се правила високог нивоа преводе аутоматски у JAVA код, а аспекти веза високог нивоа у AspectJ+ код.

Објекат правила приказан је кодом 4.1, а аспект везе приказан је у горе наведеном светлу кодом 4.4.

```
class LojalniKorisnik {
  Static pointcut proveriLojalniKorisnikUslovi (Korisnik korisnik) {
        if (!LojalniKorisnik.isLojalniKorisnik (korisnik)) {
              int Proizvodi = korisnik.getNalog().getKupljeniProizvodi();
             return Proizvodi > 5;
         } else return false;
}
```

```
aspect DLojalniKorisnik extends LojalniKorisnik {
    isApplicable() {
        return proveriLojalniKorisnikUslovi (korisnik);
    }
    after LojalniKorisnik.addLojalniKorisnik (korisnik) {.....}

}
```

Код 4.4. Аспект везе пословног правила *PPLojalniKorisnik* AspectJ+ нотацији

На овај начин AspectJ аспект конекције добија једну нову димензију која се огледа у могућности управљања шаблонима тј. апстрактним класама. Тиме је омогућено динамичко креирање неограниченог броја објеката изведених на основу шаблонске класе. Такође, аспект наслеђује класу која сакрива пресечну тачку. Међутим, тачка прекида је везана модификатором Static, а то значи да је класног типа и да не може бити пренесена наслеђивањем на потклассе и објекте. То додатно растерећује извршавање аспекта, који једино мора да води рачуна о методи *isApplicable()*. Уколико је примећен жељени прекид, овом методом се омогућава тренутно упошљавање правила, по неком од познатих сценарија.

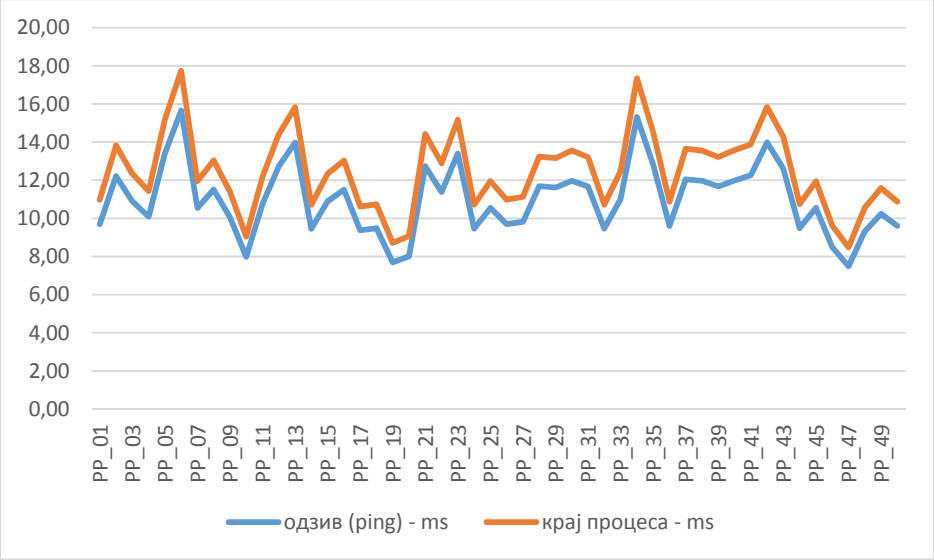
4.3.2. Перформансе решења изграђеног применом AspectJ+ приступа

У наредном тест примеру, табелом је приказан скуп од педесет правила, идентичних правилима из претходна два тест примера. Правила су кодирана програмским језиком JAVA, а одговарајући аспекти веза правила са управљачким делом софтверског решења изграђени су применом иновативног AspectJ+ приступа. Тестирање је обављено применом идентичних рачунарских ресурса као у случајевима претходна два тест примера. Табелом 4.3. приказани су резултати за остварене перформансе софтверског решења чија база знања садржи правила организована на начин приказан кодом 4.4.

Табела 4.3. Тест над AspectJ+ базом знања

База знања	одзив (ping) - ms	крај процеса - ms
PP_01	9,70	10,98
PP_02	12,21	13,83
PP_03	10,90	12,35
PP_04	10,09	11,43
PP_05	13,40	15,18
PP_06	15,66	17,74
PP_07	10,55	11,95
PP_08	11,51	13,04
PP_09	10,05	11,39
PP_10	7,98	9,04
PP_11	10,76	12,19
PP_12	12,72	14,41
PP_13	13,97	15,83
PP_14	9,46	10,71
PP_15	10,90	12,35
PP_16	11,50	13,03
PP_17	9,38	10,62
PP_18	9,48	10,74
PP_19	7,69	8,71
PP_20	8,01	9,07
PP_21	12,73	14,42
PP_22	11,38	12,89
PP_23	13,40	15,18
PP_24	9,46	10,72
PP_25	10,55	11,95
PP_26	9,70	10,98
PP_27	9,82	11,13
PP_28	11,69	13,24
PP_29	11,62	13,16

PP_30	11,97	13,56
PP_31	11,66	13,21
PP_32	9,46	10,72
PP_33	11,02	12,48
PP_34	15,32	17,35
PP_35	12,83	14,53
PP_36	9,60	10,87
PP_37	12,05	13,65
PP_38	11,97	13,56
PP_39	11,67	13,22
PP_40	11,99	13,58
PP_41	12,26	13,89
PP_42	13,98	15,84
PP_43	12,61	14,29
PP_44	9,49	10,75
PP_45	10,55	11,95
PP_46	8,50	9,62
PP_47	7,49	8,48
PP_48	9,32	10,55
PP_49	10,23	11,59
PP_50	9,61	10,88



Графикон 4.3. Резултати тестирања перформанси базе знања изграђене AspectJ+ приступом

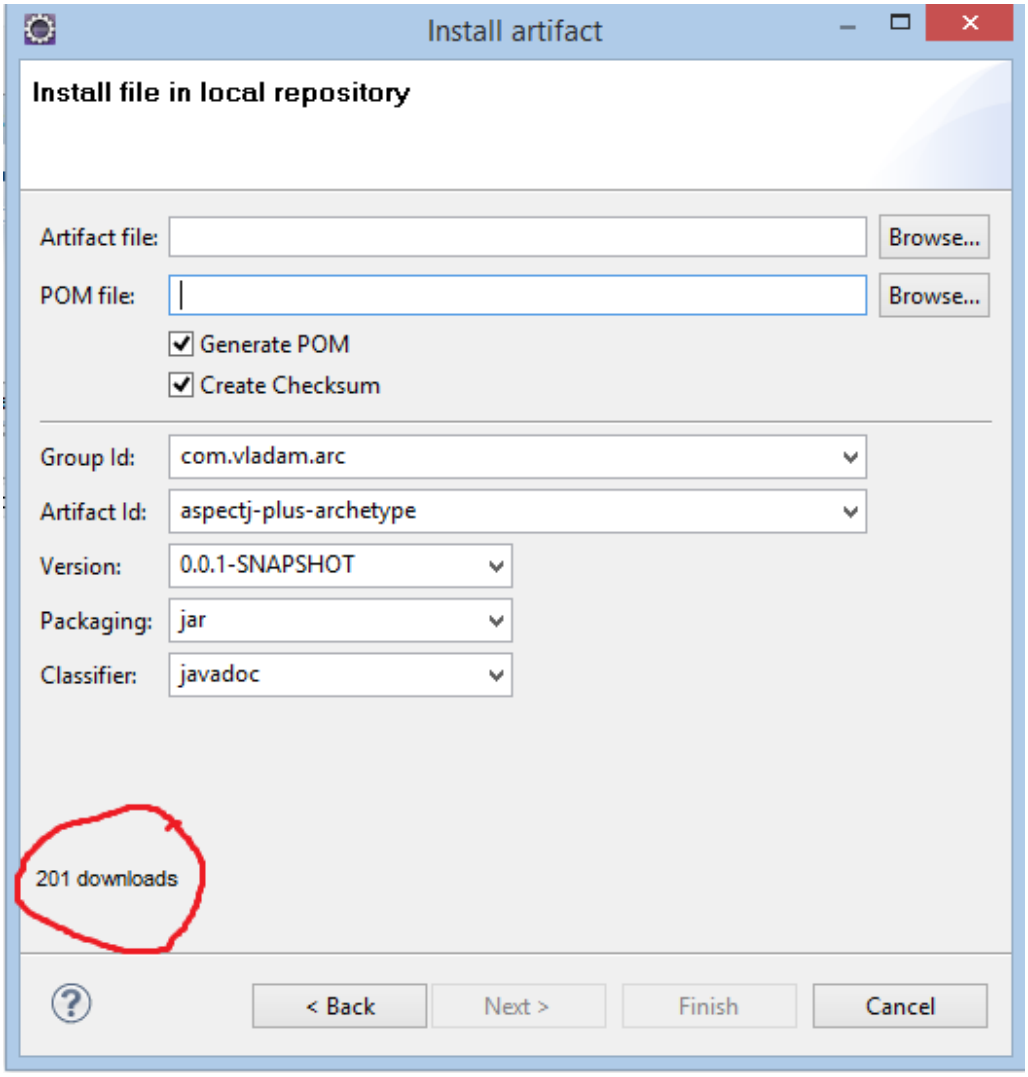
Из теста је могуће извући следеће закључке:

- Просечна реакција правила на позивање износи 11ms. То је за мање од 1% лошије од резултата добијених применом AspectJ аспеката веза, а скоро 17% боље од резултата добијених применом JasCo аспеката веза;
- Просечно трајање процеса примене правила износи 12,46ms. Резултат је за мање од 2% лошији од AspectJ резултата, а за скоро 22% боље од JasCo резултата.

У тренутку писања овог поглавља, AspectJ+ бележи преко двестотине преузимања пакета архива путем Eclipse и Maven (Слика 4.4).

На крају је могуће закључити да су нешто лошије перформансе које је AspectJ+ постигао у односу на AspectJ занемарљиве у односу на бенефите које нуди нови приступ. Као приступ који омогућава степен модуларности софтверских решења еквивалентан са степеном који нуди JasCo, AspectJ+ се намеће као одлично решење за изградњу пословних експертних система високе стабилности и брзине софтверских процеса, способних да аутоматски укључују ново знање у властите базе података

Дакле, комбиновањем доминантних карактеристика репрезентативних аспектно-оријентисаних приступа могуће је развити унапређен алат за решавање проблема повезивања правила са језгром система, а то представља **доказ хипотезе број 2**.



Слика 4.4. Преузимање AspectJ+ пакета алатом *Eclipse*

5. Анализа модела домена и креирање правила језицима високог нивоа – основе софтверске компоненте *NDL/Generator*

Приступ предложен у овом раду полази од приступа софтверској анализи познатог као софтверски дизајн изведен из модела (MDSD – Model Driven Software Development). *MDSD користи специфичне језике домена за креирање модела који показује структуру или понашање софтверског решења на ефикасан и доменски специфичан начин. Модел се, затим, преводи у извршив код применом модела трансформација.*[76]

Овакав начин развоја софтвера је двосмеран и омогућава:

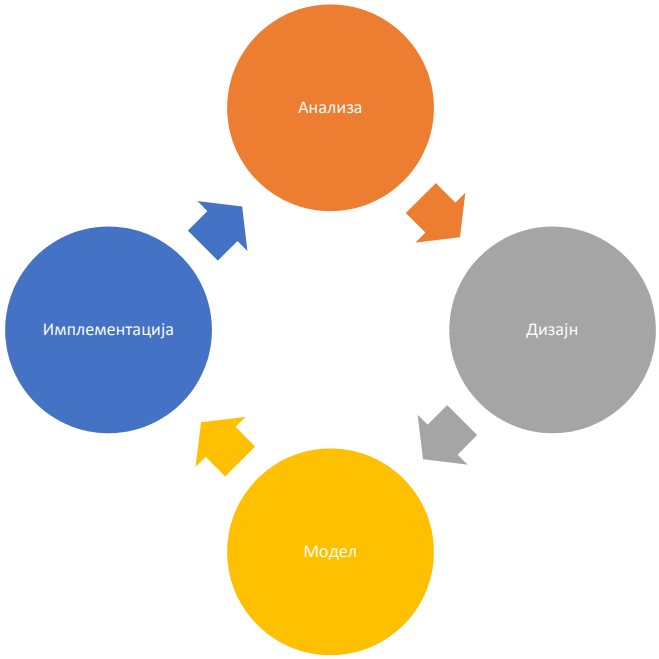
- развој кода на основу модела;
- реинжењеринг модела на основу уочених недостатака или могућности унапређења кода.

Повратна веза омогућава директан утицај софтверских компонената на одговарајуће доменске детаље. Од посебног значаја за овај рад јесте препознавање језика који је близак подскуп говорног језика и никако не може представљати директно обраћање рачунару. Отуда је неопходно прецизно дефинисати начине пресликавања инструкција доменског језика у инструкције програмског језика.

По MDSD приступу, развој софтвера представља један непрекидан циклус сарадње модела и имплементације (Слика 5.1). Могућност враћања имплементационог кода, поново, на ниво апстракције омогућава да експерти, који много боље разумеју пословање него програмери, могу да узму активно учешће у изградњи, одржавању и унапређењу софтверских решења. Међутим, MDSD приступ, за разлику од стандардног MDE (Model Driven Engineering [77] [78]) приступа, још увек се развија и захтева све софистицираније алате и технике за подршку новом начину размишљања софтверских инжењера. MDE приступ, поред стандардних објектно-оријентисаних техника, користи на флексибилан начин и аспектно-оријентисане алате. Код MDSD приступа нема компромиса:

[76] Stahl T, Völter M, Bettin J, Haase A, Helsen C. 2006. *Model-Driven Software Development: Technology, Engineering, Management*, Wiley
[77] Schmidt D. 2006. Model Driven Engineering, IEEE Computer Society 0018-9162/06
[78] Kent K. 2001. Model Driven Engineering, IT Topics - 2001

- софтвер мора да има максималан степен модуларности;
- везе између модула морају да буду лабаве;
- поновна употребљивост програмираних модула мора да буде максимална;
- свака измена на имплементацији, током тестирања и експлоатације софтверског решења, мора да има утицај на доменске концепте.



Слика 5.1. Циклус сарадње модела и имплементације

Отуда, већ на нивоу дизајна домена, неопходно је раздвојити све класе уочене током процеса софтверске анализе. За софтверско решење, које је представљало основ за истраживање, током софтверске анализе прво су идентификоване следеће класе (иницијална хијерархија класа приказана је Сликом 5.2):

- компанија;
- филијала (физичка филијала, веб филијала);
- корисник (физичко лице, правно лице);
- финансијско пословање;

- сервер;
- клијент, итд.

Након идентификације класа било је неопходно дефинисати минималистички интерфејс преко којег класе система комуницирају. Овде је од посебног значаја флексибилни ниво веза, компоненте BRL/Generator, који садржи трансформације којима се моделски модули директно преводе у класе нивоа имплементације, а сам интерфејс у лабав извршив интерфејс. У конкретном случају, новооткривено знање преводи се директно у објекте правила, а њихове везе у одговарајуће аспекте. У светлу наведеног, а имајући у виду радове који се ослањају на MDE приступ[79] [80], неопходно је било увести нове дефиниције за:

- језик модела домена;
- трансформације које ће превести доменски језик у JAVA и AspectJ+ код.

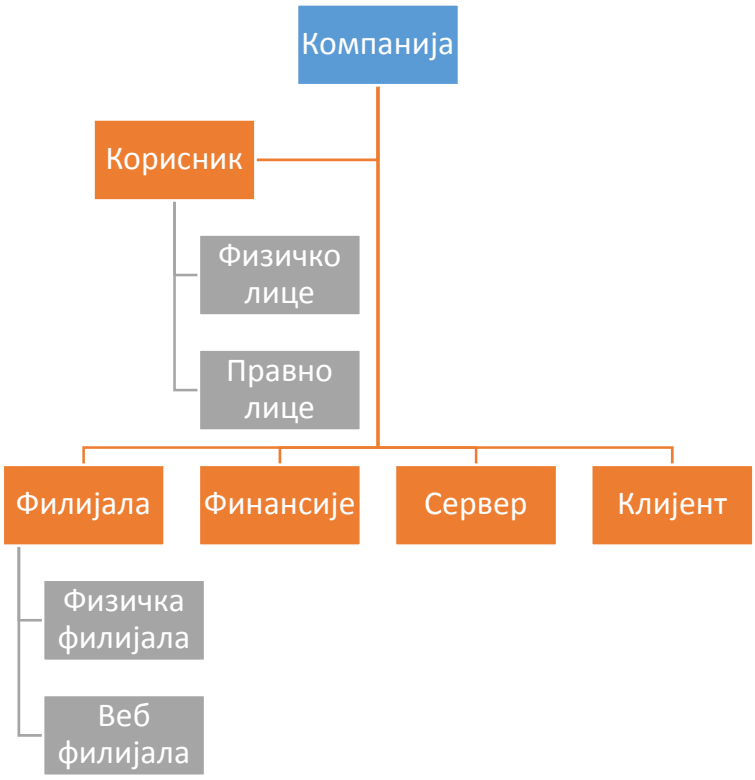
Нове дефиниције представљају полазну тачку за креирање софтверске компоненте, из класе транслатора, Translator/AspectJ+.

5.1. Нови језик домена

Нови језик домена (NDL – New Domain Language) представља начин исказивања софтверских компонената на нивоу домена конструкцијама веома блиским говорном језику. Уколико се речи, правила и инструкције NDL језика прецизно дефинишу, он може да постане моћно оруђе на основу којег интелигентни рачунарски системи могу да базирају претрагу текстуалних узорака. Од посебног значаја за овај рад јесте дефинисање инструкција NDL језика и програмирање специфичних класа које преводе NDL правила и њихове везе у одговарајући JAVA код са AspectJ+ проширењима. Нови језик домена може да обухвати велики број речи и инструкција. Међутим, овде је од посебног значаја кодирање препознатог знања NDL језиком и његово трансформисање у извршив програмски код. Значај трансформација за превођење језика високог нивоа у извршив програмски код први је истакао аутор Шмит[77], а ауторка М. А. Сибран[80] је детаљно елаборирала трансформације

[79] Милићевић В. 2012. Допринос развоју интелигентног рачунарског система базираног на приступу пословних правила, ФАМНС
[80] Cibran M. A. 2007. Connecting High Level Business Rules With Object Oriented Applications. Vrije Universiteit Brussel

којима се језик високог нивоа преводи у JAVA код са JasCo аспектно-оријентисаним проширењима.



Слика 5.2. Иницијална хијерархија класа

Циљ овог дела рада јесте креирање NDL језика као новог језика високог нивоа и, полазећи од наведеног приступа ауторке M. A. Сибран, дефинисање нових, специфичних трансформација за превођење нове NDL нотације у JAVA код са AspectJ+ проширењима.

Први корак приликом креирања новог језика домена јесте уочавање резервисаних речи за исказивање: назива, догађаја, особина и инструкција:

- PP – почетак назива сваког пословног правила;
- D – догађај;
- OSOBINE – атрибути класа правила;
- KORISTI;

- GDE;
- AKO;
- ONDA;
- POVEŽI;
- PRE;
- POSLE;
- UMESTO…

Резервисаним речима омогућено је креирање доменских ентитета уважавајући у потпуности објектно – оријентисане концепте и принципе. Процес одговоран за креирање модела домена и изоловање доменских ентитета назива се анализа домена.[81] Током анализе домена неопходно је идентификовати следеће елементе на којима почива моделирање:

- класе домена;
- особине (атрибуте) класа домена;
- операције (методе) које објекти класа домена изводе над властитим атрибутима.

Објектно-оријентисани концепти који се примењују на доменске ентитете приликом креирања модела су:

- наслеђивање;
- скривање података (енкапсулација);
- преоптерећење метода;
- полиморфизам.

Класа домена дефинише скуп доменских својстава - скуп операција над доменом, и може имати велики број инстанци (објеката). Атрибути (особине) домена описују опште или карактеристичне особине објеката класе домена, док операције домена репрезентују понашање наведених објеката. Операције могу бити употребљене са циљем извлачења доменског знања из имплементације постојеће апликације, или за исказивање новог речника домена који је неопходно конструисати као резултат еволуције домена.[79]

[81] Милићевић В. 2009. *Објектно-оријентисано пројектовање пословних информационих система*, Економски факултет у Београду.

Сликом 5.3. приказана је објектно-оријентисана репрезентација класе домена, са особинама и операцијама. Особинама и операцијама је, такође, неопходно придружити и модификаторе приступа којима се истиче да ли су елементи класа:

- статички (класни);
- нестатички (објектни);
- јавни (наслеђивањем се преносе у ниже нивое хијерархије класа и приступ им је неограничен);
- приватни (припадају искључиво класи у којој су креирани и не могу да се наслеђују, приступ им је ограничен јавним методама класе у којој су креирани);
- заштићени (преносе се наслеђивањем у ниже нивое хијерархије и приступ им је омогућен искључиво унутар хијерархије класа).

Сликом 5.3. Објектно-оријентисана репрезентација класе

Дакле, у конкретном случају, задатак анализе домена јесте активно укључивање експерата домена у процесу дефинисања правила и њихових веза и поједностављивање ових задатака за софтверске развојне тимове. Отуда, кроз процес анализе домена, преко објектно-оријентисаног дизајна, дошло се до модела домена високог нивоа који се ослања на три тачке:

- доменски ентитети;
- пословна правила која су у директној вези са доменским ентитетима;
- везе правила са класама језгра софтверске апликације по условима које диктирају доменски ентитети.

Наведени приступ може бити илустрован Сликом 5.4.

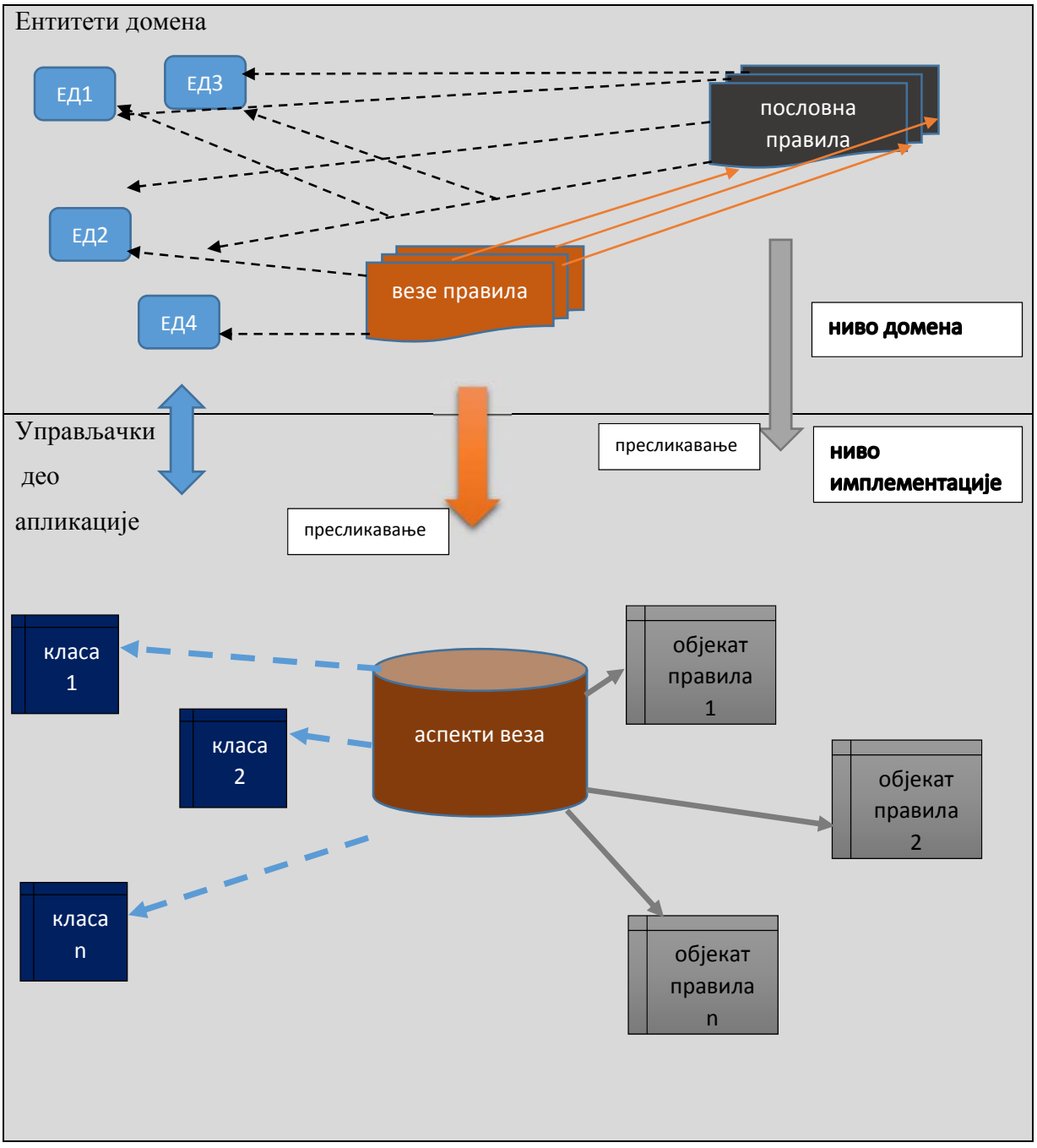
5.1.1. Креирање правила високог нивоа доменским језицима

Приликом анализе и дизајна модела кључни акценат је стављен на креирање речника домена који је у потпуности изграђен од идентификованих доменских ентитета. Доменским ентитетима су исказани: класе језгра софтверског решења и базе знања, атрибути класа и операције које класе изводе над атрибутима, као и аспекти веза између наведених класа.

У овом делу рада посебно је битно идентификовати наведене ентитете за домен индустрије осигурања, а након тога извршити њихово кодирање NDL језиком. Сликом 5.5. приказани су доменски ентитети који се најчешће срећу у индустрији осигурања:

- корисник;
- производ;
- изабрани производ;
- налог;
- компанија итд.

Свакој од наведених класа неопходно је придружити одговарајуће атрибуте и операције (Табела 5.1).



Слика 5.4. Логика MDSD приступа (прилагођено у односу на извор: М. А. Сибран[80])

Табела 5.1. Класе, атрибути и операције ентитета домена индустрије осигурања

А)

корисник

атрибути	методе
• име • старост • ЈМБГ • бројУговора	пријављивање() одјављивање() изборПроизвода() лојалност()

Б)

производ

атрибути	методе
• цена • времеКоришћења	/

В)

налог

атрибути	методе
• вредностПолиса • купљенеПолисе	/

Г)

избор

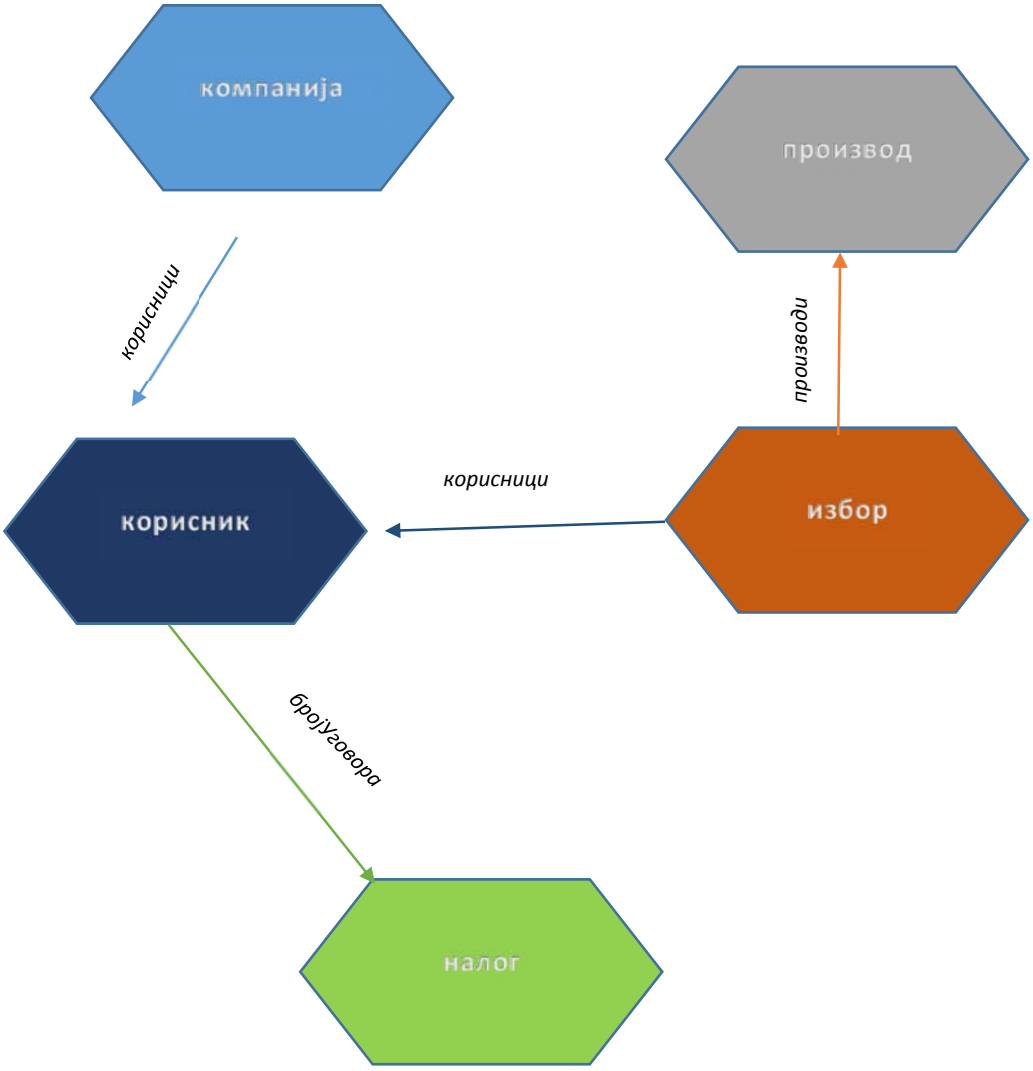
атрибути	методе
• корисник • производи	применіПопуст(корисник)

Д)

компанија

атрибути	методе
• корисници • филијале • финансијскоПословање • …	провераИзбораКорисника(избор) провераКорисника(корисник) одобриПопуст(избор) …

После идентификовања доменских ентитета потребно је дефинисати и начине њихових исказивања на нивоу домена, а у форми конкретног језика високог нивоа. Ауторка М. А. Сибран је предложила метамодел пословних правила и доменских ентитета[80] који је могуће прилагодити и унапредити у светлу домена индустрије осигурања и новог језика домена (Слика 5.6).



Слика 5.5. Графичка репрезентација ентитета из домена индустрије осигурања

Након прилагођавања наведених модела приступу, који предложен о овом раду, може се приступити дефинисању граматике новог језика домена, а у сфери која је у вези са исказивањем пословних правила и њихових веза са класама језгра софтверског решења:

1. **Особине правила**:

OSOBINE [TipPodataka_1][naziv_1], [TipPodataka_2][naziv_2], …, [TipPodataka_N][naziv_N]

Особинама су репрезентоване варијабле правила са одговарајућим типовима података који могу бити: прости, изведени или класни. У случају да се ради о класном типу података, варијабла заправо представља објекат.

2. **Параметри правила:**

KORISTI [nazivKlaseDomena] [nazivParametraPravila]

Овим инструкцијама исказано је проширење правила у контексту у којем ће бити извршени.

3. **Променљиве правила:**

GDE [nazivPromenljive] = [konstanta | promenljiva | izraz]

Наведеним инструкцијама се додају конкретне вредности за променљиве које учествују у имплементацији логике позваног пословног правила.

4. **Акција:**

AKO [logičkiUslov] ONDA [AKCIJA]

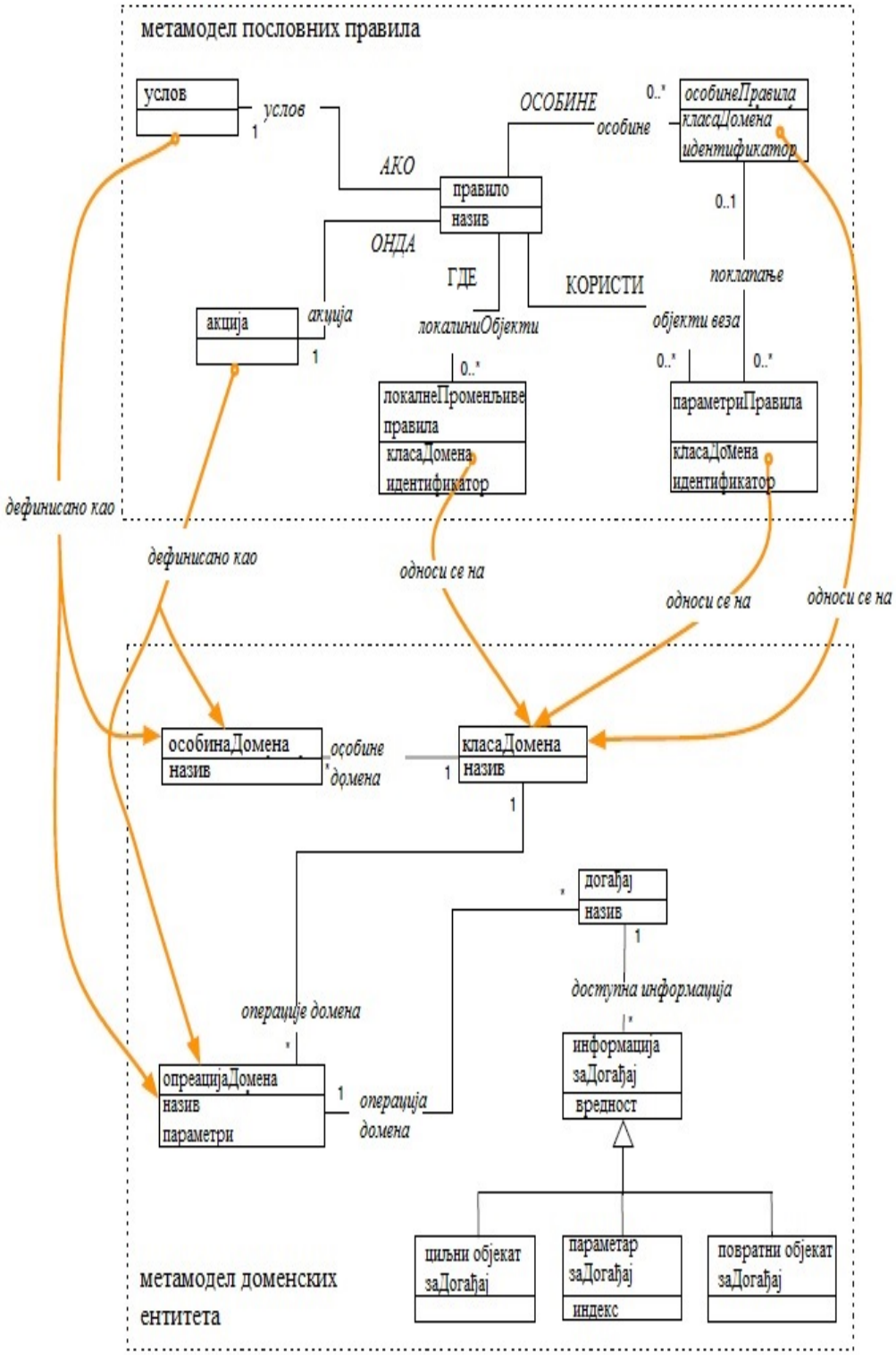
Наведеним инструкцијама исказује се извршавање логике позваног пословног правила у зависности да ли је логички услов испуњен или не.

Комбиновањем наведених инструкција креирају је пословна правила високог нивоа као што је случај у следећем примеру:

```
PP PPPopustNaPolisu

OSOBINE Real vrednostPolise, Real popust

KORISTI korisnik.brojUgovora nalog

GDE ciljniKorisnik = izbor.korisnik

AKO ciljniKorisnik.nalog.vrednostPolise >= vrednost

ONDA izbor.primeniPopust (ciljniKorisnik)
```

Код 5.1. Пословно правило исказано новим језиком домена

Наведеним кодом високог нивоа исказана је реализација пословног правила чијим се превођењем на ниво имплементације добија класа која одређује објекте пословних правила чијим позивањем осигуравајућа компанија одобрава попуст на куповину одређене полисе. Ово правило, као и скуп сличних и сродних правила, налази се у бази знања информационог система *Meridian* тако да путем HelpDesk компоненте службеник осигуравајуће компаније може тренутно да добије одговор на питање: „Да ли да конкретном кориснику одобри попуст на куповину нове полисе“?



Слика 5.6. Метамодели пословних правила и доменских ентитета са везама (прилагођено у односу на извор: М. А. Сибран[80])

5.1.2. Креирање веза правила високог нивоа доменским језицима

Да би аспектима правила био омогућен минимални интерфејс између објеката правила управљачког дела софтверског решења, неопходно је на нивоу домена прецизно дефинисати везе између ентитета домена – изолованих правила и објеката управљачког дела софтверског решења. Овакав интерфејс, на нивоу имплементације, омогућиће следеће:

- ефикасно повезивање наведених типова података;
- брже извршавање софтверских процеса;
- елиминисање пресечног и мултиплицираног кода;
- већу контролу, у смислу одржавања и отклањања грешака, над софтверским решењем.

Везама између пословних правила и објеката дела софтверског решења за управљање знањем, дефинисаним на нивоу домена, описују се детаљи повезивања наведених типова објеката. Од посебног значаја је увођење концепта *догађај* који представља веома прецизно дефинисану тачку у времену извршавања софтвера у којој долази до примене логике позваног пословног правила. Део кода који је задужен за покретање правила, за одређени догађај, назива се *окидач* (енг. trigger). Када се говори о догађајима имплементираним NDL нотацијом, а касније AspectJ+ извршивим кодом, за разлику од помињаних приступа[79] [80] и уважавањем инсистирања MDSD моделирања, овај рад предлаже третирање догађаја као посебних и независних софтверским модула – класа догађаја. Као аргументи конструктора ових класа јављају се аспекти веза одређеног правила преко којих се, у датом тренутку извршавања софтверског решења, дешава прекид и извршавање логике конкретног пословног правила.

Сви језици високог нивоа, па тако и NDL и језици у наведеним приступима, којима се описују везе између правила и објеката управљачког дела софтвера, имају следеће заједничке карактеристике (Слика 5.7):

- одређивање времена примене правила;
- строга примена правила у тачно одређено време примене;
- обезбеђивање неопходне информације правилу;
- иницирање прекида извршавања језгра софтверског решења;

- покретање правила;
- прихватање резултата примене правила;
- наставак извршавања језгра софтвера.

Истицањем неопходних карактеристика пословних правила, ослањајући се на метамоделе пословних правила и доменских ентитета са везама, граматика NDL језика може да буде проширена увођењем инструкција веза правила и догађаја на начин приказан Сликом 5.7.

1. **Повезивање правила:**

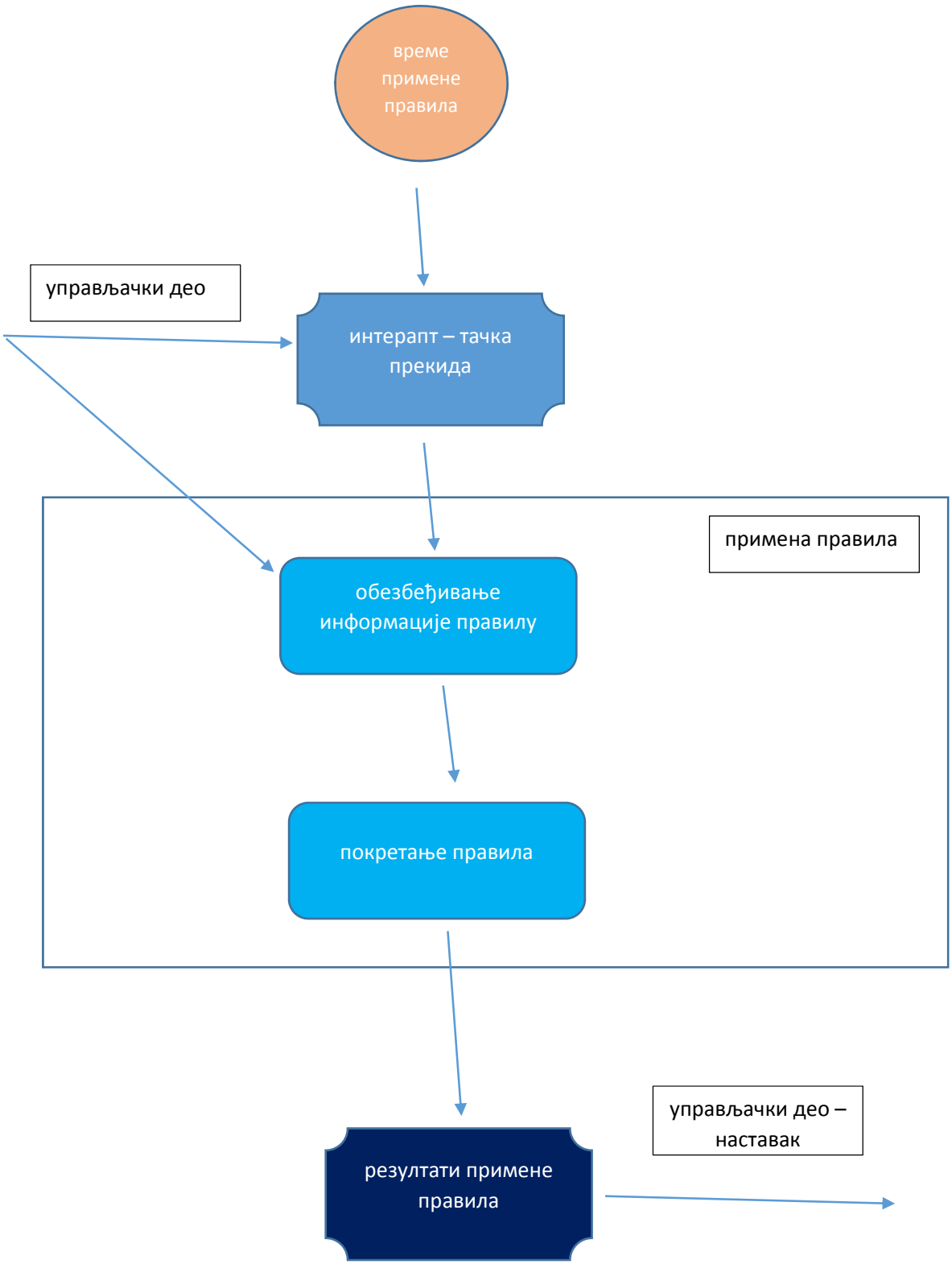
POVEŽI [nazivPravila]

Наведеном инструкцијом се имплементира логика пословног правила у датом тренутку извршавања посматраног софтверског решења.

2. **Особине које се придружују правилима током повезивања**

OSOBINE [vrednost_1], [vrednost_2], …, [vrednost_N]

Наведеним инструкцијама се обезбеђују неопходне информације правилу као што је напоменуто у горњем излагању.



Слика 5.7. Сценарио повезивања правила са управљачким делом софтверског решења

3. **Класа *Окидач* догађаја**

Наведена класа има задатак да омогући креирање објеката који ће реализовати извршавање инструкција повезивања правила у тачно дефинисаној тачки у којој се дешава прекид управљачког дела софтверског решења. Класа *Окидач* мора да узме у обзир следеће информације:

- активирање везе правила са управљачким делом софтверског решења;
- време примене правила;
- време придруживања неопходне информације правилу.

Активирање везе правила са управљачким делом софтверског решења NDL кодом реализује се на следећи начин:

```
OKIDAČ [POVEŽI [nazivPravila], [vremePrimenePravila], [tačkaPrekida] ]
```

Сценарио по којем се правило повезује са управљачким делом софтверског решења (Слика 5.8) зависи директно од времена примене пословног правила. Време примене може бити:

- *пре* јављања догађаја – није потребно извршавање самог догађаја већ његов почетак представља оријентир за покретање логике пословног правила;
- *после* јављања догађаја – неопходно је да се у потпуности реализује догађај да би дошло до покретања логике пословног правила;
- *уместо* јављања догађаја – реализација правила траје истовремено са реализовањем догађаја, ограничено почетком и крајем извршавања догађаја. Овај сценарио је посебно значајан за подршку имплементацији логике извесног скупа правила у тренутку када се корисник пријави на HelpDesk систем до тренутка његовог одјављивања.

Време примене пословног правила NDL кодом реализује се на следећи начин:

```
OKIDAČ [POVEŽI [nazivPravila], [PRE | POSLE | UMESTO], [tačkaPrekida] ]
```

Последња ставка, на коју догађај мора да обрати пажњу, јесте придруживање неопходне информације правилу. Ова информација је сакривена у клаузули *OSOBINE* која припада вези између правила и језгра софтверског решења. Додавањем клаузуле:

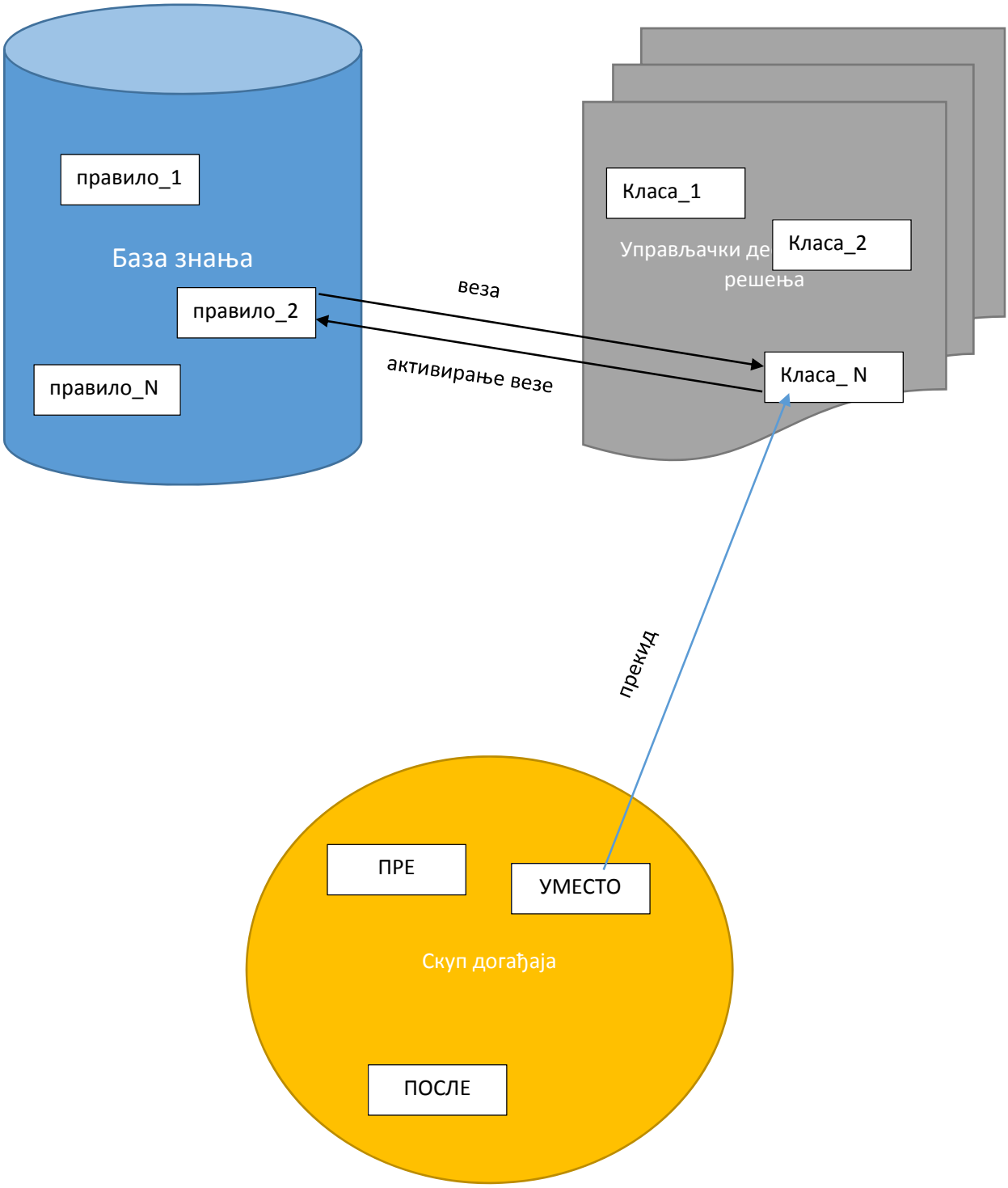
```
POKRENI [OKIDAČ]
```

догађај ставља на располагање пословном правилу неопходну информацију за реализовање његове логике.

Навођењем последње клаузуле заокружена је дефиниција NDL језика када је у питању кодирање пословних правила и њихових веза на нивоу домена. Правило исказано Кодом 5.1, може бити повезано са управљачким класама софтверске апликације на следећи начин:

```
POVEŽI PPPopustNaPolisu

OSOBINE 15000.00 , 10.00

-----------------------------------------------------------------------------------------------------------

OKIDAČ POVEŽI PPPopustNaPolisu, POSLE.ciljniKorisnik.povezan() I PRE.
ciljniKorisnik.odjavljen(), ciljniKorisnik

POKRENI OKIDAČ
```

Код 5.2. Повезивање пословног правила са класом управљачког дела у NDL језику



Слика 5.8. Сценарио повезивања правила и управљачких класа у зависности од догађаја

У овом делу рада показан је приступ моделирању знања који се ослања на моделски базиран приступ са највећим могућим степеном модуларности софтверског решења. Овим приступом је обезбеђено исказивање пословних правила и њихових веза са језгром софтверског решења новим језиком високог нивоа који је означен као NDL језик. У потпуности су приказани елементи језика којима је реализовано:

1. кодирање правила и њихових елемената;
2. кодирање веза са главним делом софтверског решења
3. кодирање догађаја – окидача правила.

На овај начин доказана је Хипотеза број 3.

Даље, инсистирано је да логика сваког правила, већ на нивоу домена, буде изолована у засебан софтверски модул. Превођењем оваквог кода, на ниво имплементације, добија се класа правила која енкапсулира логику правила, дајући бази знања облик скупа објектно - оријентисаних класа уместо документа који обухвата велики број if – then – else инструкција. Такође, везе између правила и објеката језгра софтверског решења сакривене су у засебне модуле. На овај начин, скуп веза, на нивоу имплементације, представља колекцију специфичних класа у AspectJ+ нотацији који се називају аспектима веза. На крају, скуп догађаја окидача је такође организован по објектно – оријентисаном принципу. Узимајући у обзир објектно-оријентисано софтверско решење које чине:

- објектно-оријентисан централни део софтвера за управљање знањем;
- објектно-оријентисана база знања;
- аспектно-оријентисан скуп веза објеката правила и централног дела софтвера;
- објектно-оријентисан скуп догађаја;
- објектно – релациони приступ бази података (Поглавље 7, део 7.1.3),

могуће је закључити да ефикасно повезивање и највећи степен модуларности софтверског решења могуће је обезбедити кроз енкапсулацију правила и конекција у објекте правила и аспекте конекција респективно.

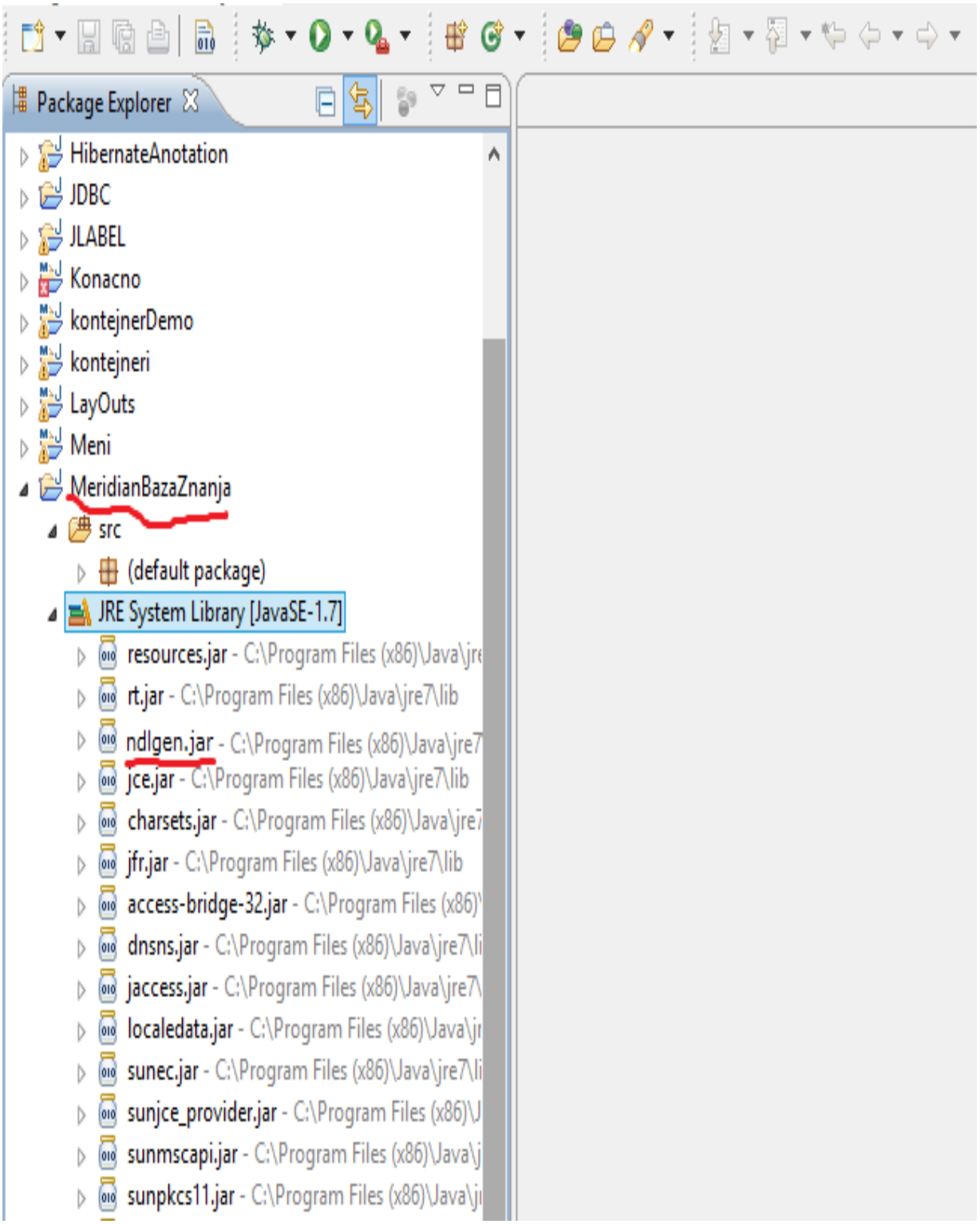
На овај начин доказана је Хипотеза број 4.

У наредним поглављима биће више речи о превођењу NDL синтаксе у JAVA AspectJ+ код. Овде ће још бити демонстрирано преузимање изјаве у текстуалном формату (*.txt, *.doc, *.pdf) од стране софтверског решења и њено обликовање у форми пословног правила високог нивоа. Изоловање правила из текстуалног узорка обавља интелигентна софтверска компонента (Поглавље 7) *NDL/Generator,* која је програмирана у програмском језику JAVA, и која су помоћу развојних алата Eclipse и Maven похрањена у оригиналну библиотеку JAVA архива (***ndlgen.jar***) (Слика 5.9) која је доступна преко Интернета применом наведених развојних алата. Сликом 5.10. приказано је преузимање узорка, који одговара изјави оштећеног након саобраћајне несреће (Поглавље 7, део 7.1.1), која је прослеђена путем HelpDesk система осигуравајуће компаније интелигентној софтверској компоненти NDL/Generator, која је, потом, издвојила ново знање из узорка (Слика 5.11).

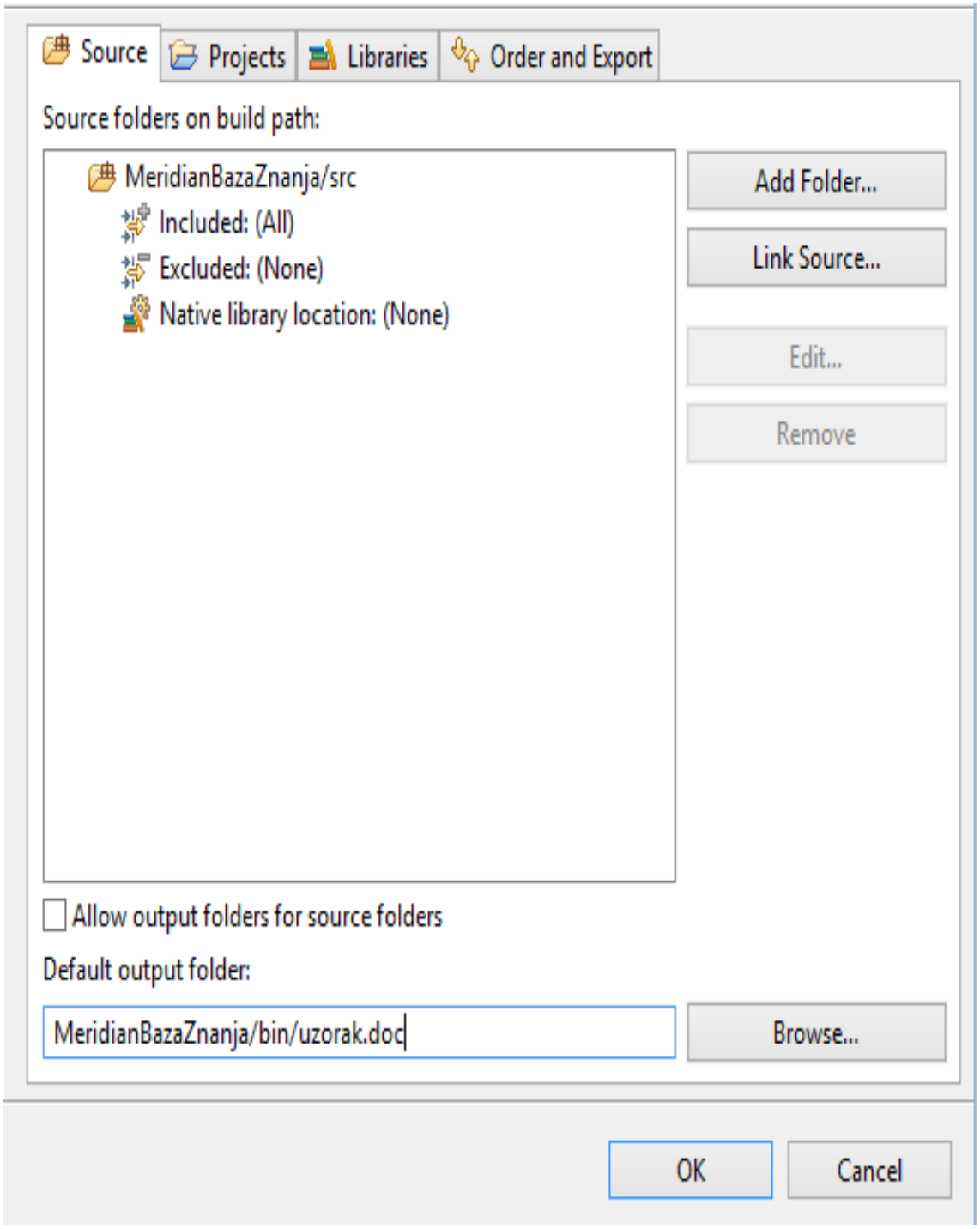
Требало би још напоменути да се преузимање узорка у оригиналном софтверском решењу одиграва аутоматски. Овде је преузимање било кориснички контролисано због омогућавања демонстрације акција које доводе до креирања NDL правила на основу преузетог текстуалног узорка.

Посебно, могућност проширења NDL језика и његове примене на различитим софтверским доменима је велика. За потребе овог рада NDL језик је елабориран као подскуп српског језика, док је за потребе конкретног софтверског решења NDL језик је исказан речима енглеског језика. Комбиновањем NDL језика са напредним преводилачким софтверским решењима, једна дефиниција новог језика домена може да се примени на велики број различитих говорних подручја. NDL језик може да буде комбинован са најквалитетнијим преводилачким софтверским решењима, као што су *Trados*[82] [83] или *OmegaT*[84], применом *принципа преклапања кључних речи*[85] [86] [87]. На овај начин преводилачки софтвер преводи само кључне речи језика (Слика 5.12), а називе променљивих, константи, објеката игнорише (Слика 5.13).

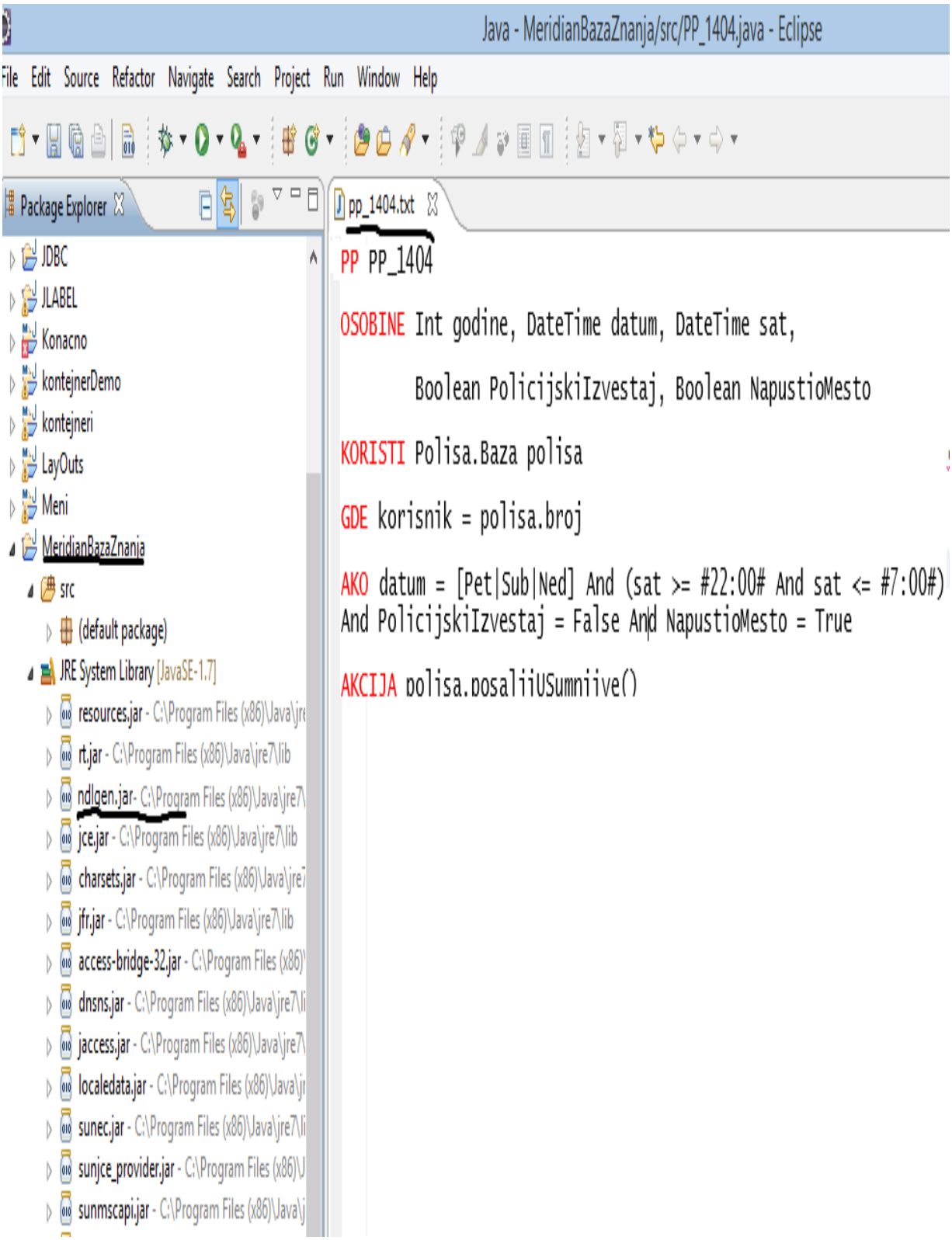
[82] www.trados.com (приступљено 20.01.2015. у 9:30)
[83] www.translationzone.com (приступљено 20.01.2015. у 9:35)
[84] www.omegat.org. http://www.omegat.org/en/dl_overview.php (приступљено 20.01.2015. у 9:40)
[85] www.smallbiztrends.com. http://smallbiztrends.com/2014/08/phrase-exact-keyword-match-adwords.html (приступљено 20.01.2015. у 9:44)
[86] www.wordstream.com (приступљено 20.01.2015. у 9:50)
[87] google.com. https://support.google.com/adwords/answer/6324?hl=en (приступљено 20.01.2015. у 9:55)



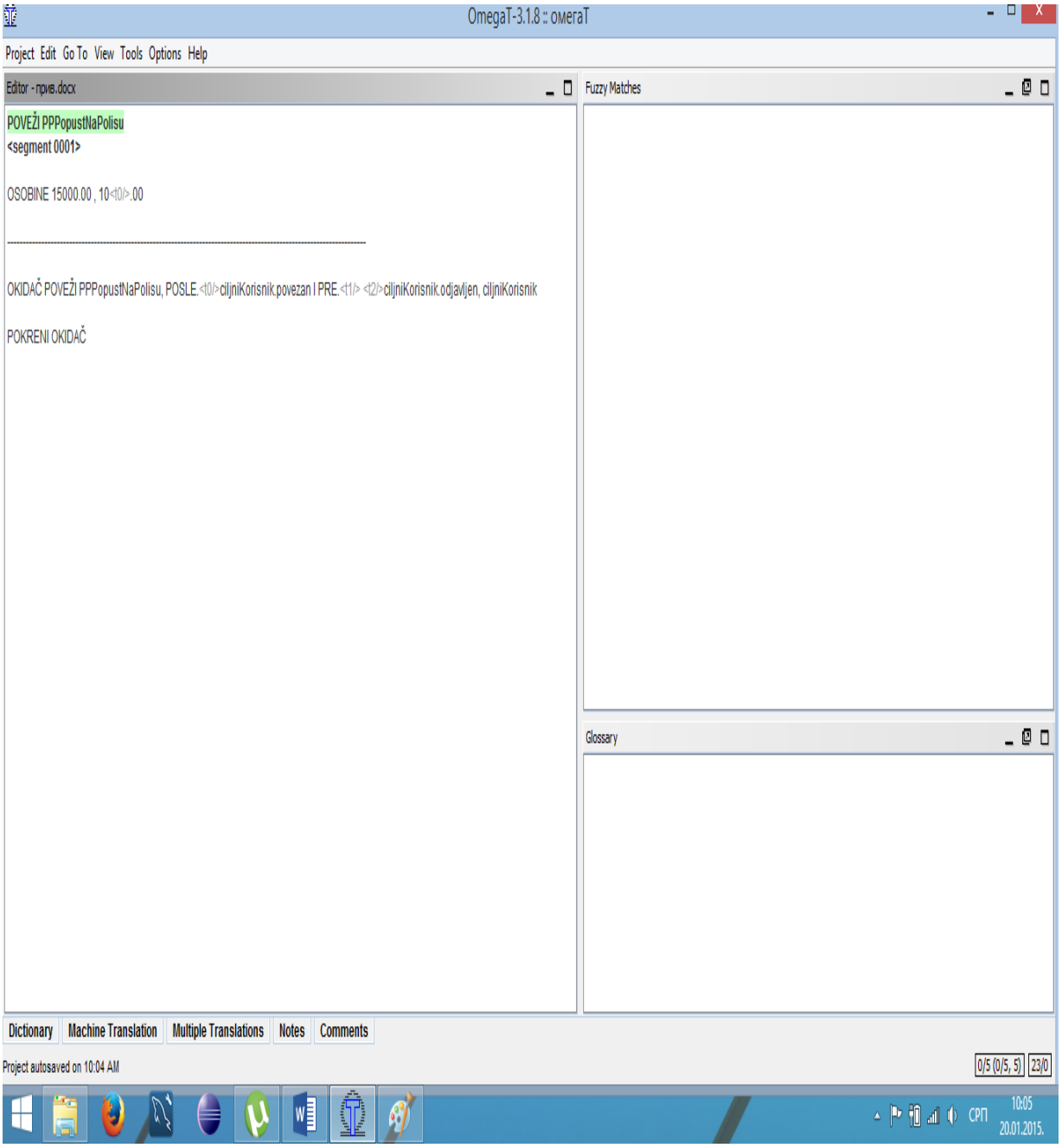
Слике 5.9. JAVA архива софтверске компоненте *NDL/Generator*



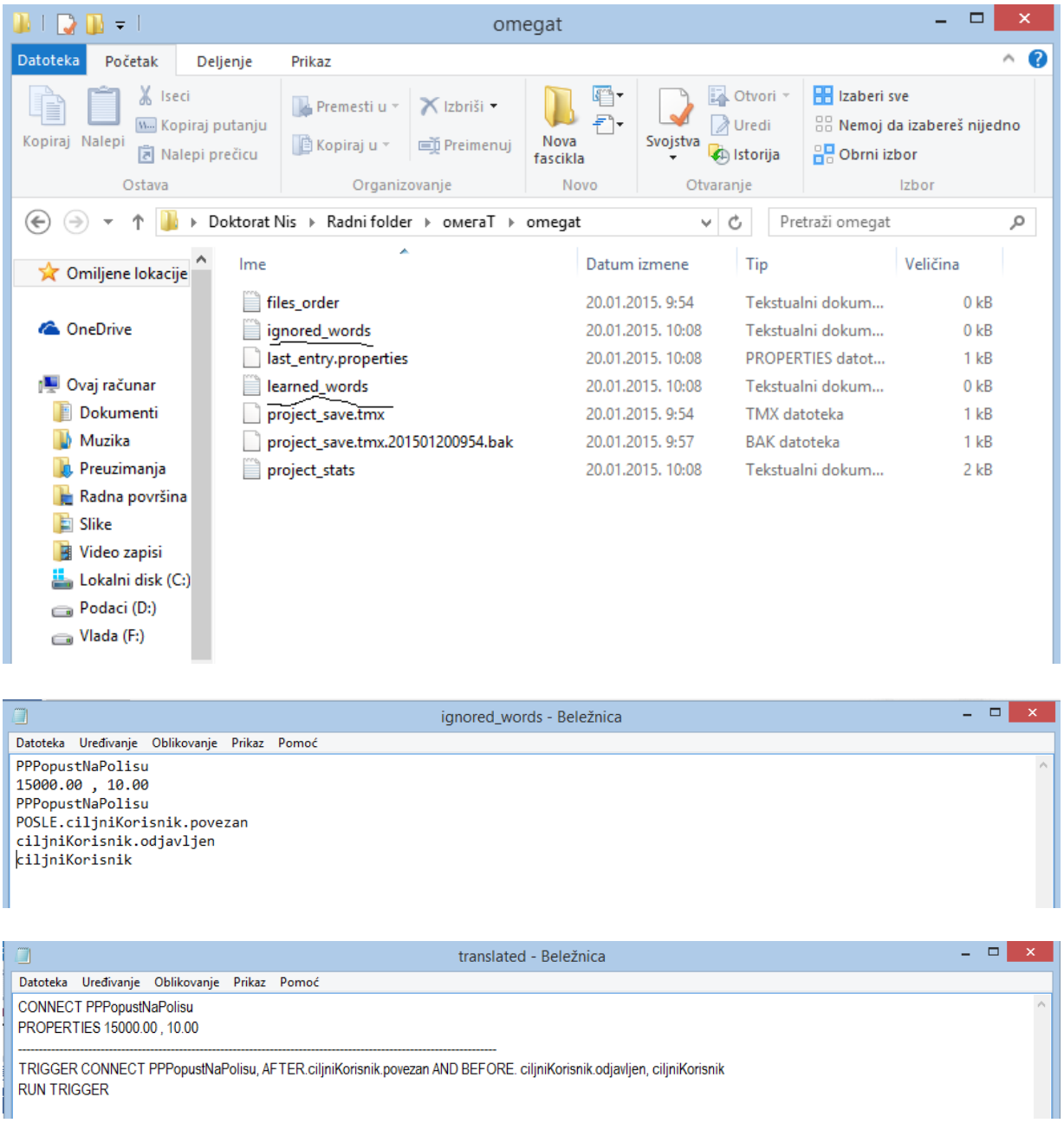
Слика 5.10. Преузимање узорка компонентом *NDL/Generator*



Слика 5.11. Издвојено знање из узорка исказано NDL језиком


OmegaT-3.1.8 :: омегаT
Project Edit Go To View Tools Options Help
Editor - прив.docx
Fuzzy Matches
POVEŽI PPPopustNaPolisu
<segment 0001>
OSOBINE 15000.00 , 10<t0/>.00
OKIDAČ POVEŽI PPPopustNaPolisu, POSLE:<t0/>ciljniKorisnik.povezan I PRE:<t1/> <t2/>ciljniKorisnik.odjavljen, ciljniKorisnik
POKRENI OKIDAČ
Glossary
Dictionary
Machine Translation
Multiple Translations
Notes
Comments
Project autosaved on 10:04 AM
0/5 (0/5, 5)
23/0
СРП
10:05
20.01.2015.

Слика 5.12. Превођење NDL инструкција софтвером *OmegaT*



Слика 5.13. Резултати превођења NDL инструкција софтвером *OmegaT*

6. Софтверске трансформације за превођење NDL језика у извршив код

Дискусија у вези са NDL језиком, и његовом применом за кодирање знања у претходном поглављу, била би само слово на папиру уколико се не обезбеде начини имплементације његових инструкција. Увођењем софтверских трансформација, које би биле задужене за пресликавање NDL кода у JAVA и AspectJ+ код, добила би се заокружена дефиниција софтверског решења *Meridian* које је предмет истраживања овог рада.

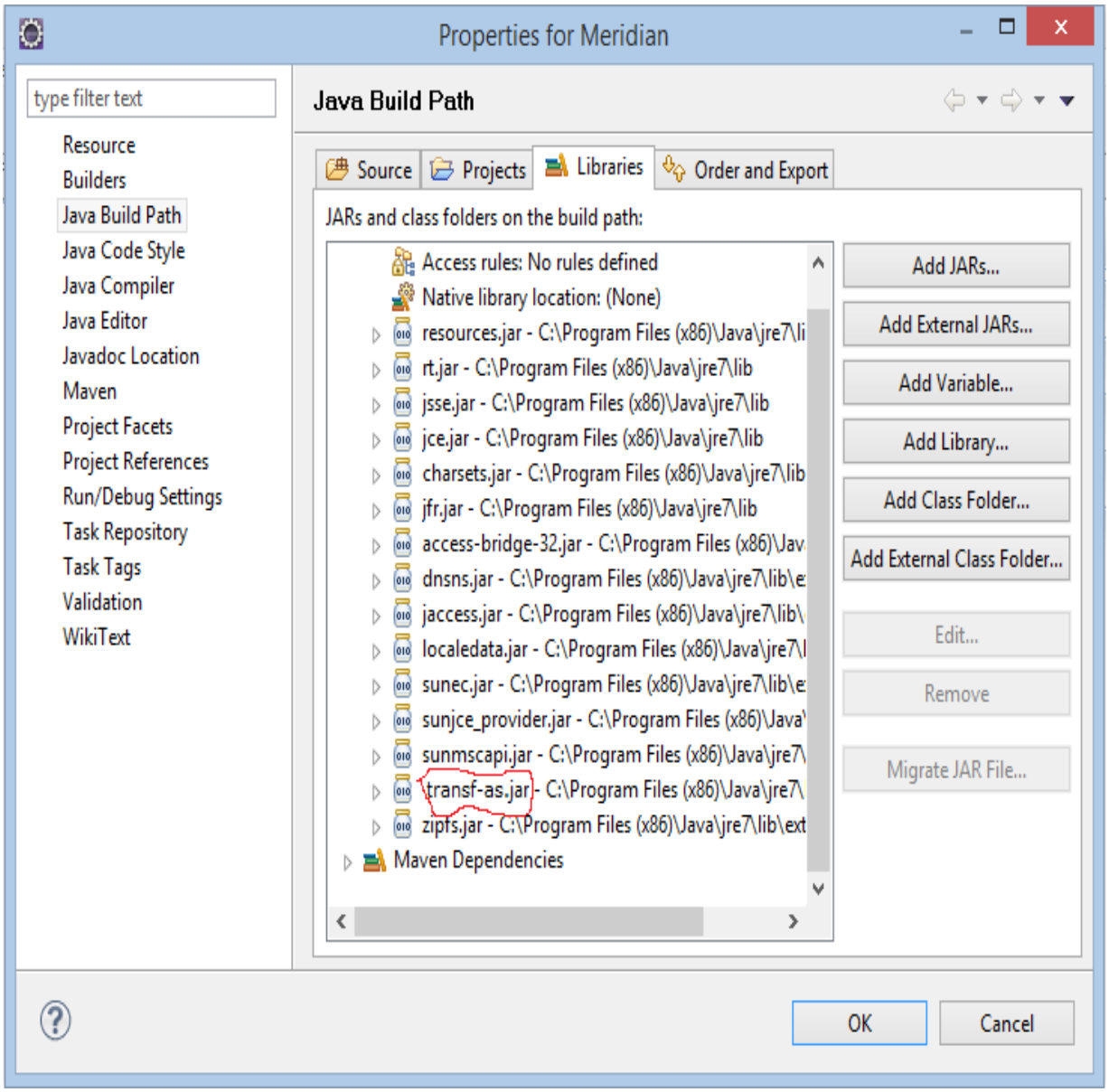
Софтверске трансформације имају веома велику примену у изградњи великог броја различитих софтверских решења. Неки од најпознатијих примера примена софтверских трансформација су:

- софтверске трансформације за унапређење процеса откривања малициозног софтвера;[88]
- софтверске трансформације за превођење модела домена у извршив код;[89] [90] [91] [92] [93]
- софтверске трансформације као интерпретација математичких трансформација;[94]
- софтверске трансформације за управљање квалитетом у софтверу[95], итд.

Од посебног значаја за овај рад јесте развој софтверске компоненте из класе транслатора која има за задатак превођење језика NDL у JAVA и AspectJ+ код. Ова компонента је добила име Translator/AspectJ+ и припада веома флексибилном нивоу класа главног дела софтверског решења. Посебно, ове класе морају да имају веома висок степен поновне употребљивости. Због тога, неопходно их је чувати у погодном репозиторијуму

[88] Christodescu M, Jha S, Kinder J, Katzenbeisser S, Veith H. 2007. *Software Transformations to Improve Malware Detection*, JComput Viral 3: 253 – 265 - Springer
[89] Cibran M. A. 2007. Connecting High Level Business Rules With Object Oriented Applications. Vrije Universiteit Brussel
[90] Lammer R. 2004. *Coupled Software Transformatins*, First International Workshop on Software Evolution Transformations
[91] Meyers B, Mannadir R, Vangheluwe H. 2009. *Evolution of Modeling Languages*, Benevol
[92] ibm.com. www.-01.ibm.com/support/docviews?uid=swg21123472 – Creating a Schema, Tables, Object Models and Data Models in Rose (приступљено 22.01.2015 у 9:00)
[93] Meng N, Kim M, McKinley K. S. 2011. *Generating Program Transformations from an Example*, The University of Texas at Austin, Papers
[94] kutasoftware.com. www.cdukutasoftware.com/worksheet/Geo/12-AllTransformations.pdf (приступљено 22.01.2015 у 9:30)
[95] Kavimandan A, Gray J. 2011. *Managing the Quality of Software Product Line Architectures through Reusable Model Transformations,* QoSA + ISARC

тако да увек буду доступне када се за то укаже потреба. Користећи развојни алат Eclipse и његов додатак Maven, трансформације за превођење NDL кода у JAVA и AspectJ+ код, похрањене су у посебну архиву којој је могуће приступити директно преко Интернета, применом наведених алата (Слика 6.1). Овако дефинисана JAVA архива примењива је на ширем спектру проблема који су у вези са превођењем једног типа програмског кода у други.



Слика 6.1. Креирана JAVA архива софтверских трансформација

Квалитет и брзина извршавања софтверских трансформација су у директној вези са бројем инструкција извршивог кода које одговарају једној инструкцији NDL кода. Одавде проистиче још једно унапређење приступа ауторке М. А. Сибран[89] у чијим радовима су приказане трансформације језика домена у JAVA и JasCo језике. Табелом 6.1. приказано је време извршавања софтверских трансформација које преводе везе пословних правила из NDL кода у JasCo и AspectJ+ респективно. Будући да је време извршавања софтверских трансформација у директној вези са квалитетом и стабилношћу истих, наведеном табелом је приказан значајан напредак квалитета класе софтверских решења транслатора.

Табела 6.1. Поређење времена извршавања софтверских трансформација

NDL везе правила	JasCo (ms)	AspectJ+ (ms)
PP_1	8,02	7,22
PP_2	7,99	7,19
PP_3	7,76	6,98
PP_4	8,12	7,31
PP_5	8,44	7,60
PP_6	8,01	7,21
PP_7	7,44	6,70
PP_8	7,99	7,19
PP_9	7,22	6,50
PP_10	8,34	7,51
PP_11	7,04	6,34
PP_12	6,99	6,29
PP_13	7,12	6,41
PP_14	7,12	6,41
PP_15	7,33	6,60
PP_16	7,20	6,48
PP_17	6,88	6,19
PP_18	7,98	7,18
PP_19	7,34	6,61
PP_20	9,01	8,11

Из табеле је могуће приметити да је за превођење NDL кода веза правила у AspectJ+ аспекте веза неопходно приближно 10% мање процесорског времена него када се ради о превођењу у JasCo аспекте веза.

У даљем излагању као циљ се поставља реализовање следећих задатака:

- креирање трансформација за превођење кода правила и одговарајућих веза у извршив програмски код;
- превођење правила и веза у извршив код;
- повезивање извршивих правила, помоћу аспеката веза, са делом софтверског решења за управљање знањем.

У оригиналном софтверском решењу *Meridian* ови задаци се одвијају аутоматски, секвенцијално и било би немогуће презентовати наведене задатке. Из наведеног разлога, у овим специфичним тачкама, а за потребе презентације у овом раду, омогућена је интервенција корисника – администратора.

6.1. Креирање трансформација за превођење NDL кода правила у извршив код

Трансформације за превођење NDL кода могуће је поделити у две категорије:

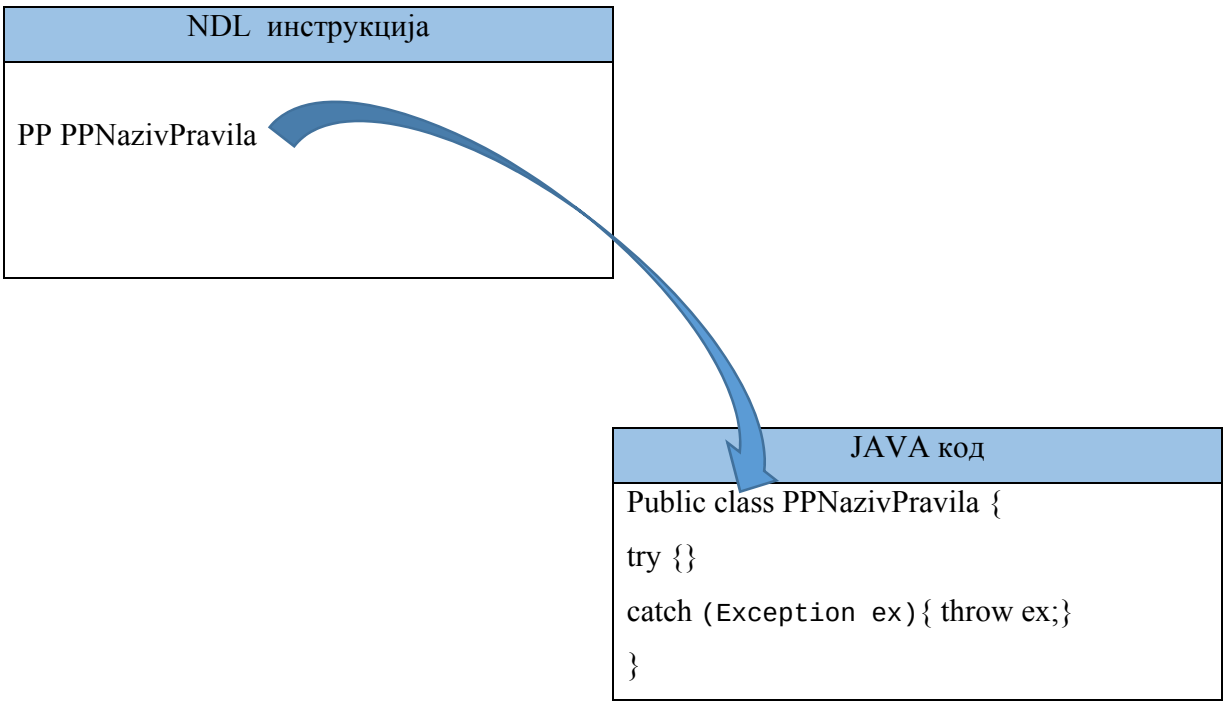
- трансформације којима се пословна правила преводе у JAVA класе објеката правила;
- трансформације којима се везе пословних правила преводе у AspectJ+ аспекте веза пословних правила.

У наставку излагања биће управо елаборирање наведене групе трансформација. Овде је потребно истаћи још једну разлику у односу на поменути приступ исказан у радовима ауторке М. А. Сибран. Логика пословних правила, исказана унутар класа правила, ослања се искључиво на концепт управљања изузецима. Инструкцијама *try – catch* омогућена је пуна контрола над извршавањем трансформација чак и у случајевима јављања изузетака у форми системских захтева или грешака. У приступу поменуте ауторке управљање изузецима препуштено је централном делу софтверског решења. То за последицу има спорије извршавање централних класа софтвера и веће меморијске захтеве због сталне провере изузетака.

6.1.1. Превођење назива пословног правила у JAVA код

Трансформацијом која преводи назив пословног правила у имплементацију креира се класа, којој се придружује назив пословног правила, из које је могуће креирати велики број објеката правила. Класа, такође, поседује два блока наредби: *try* и *catch*.

Следећом сликом је графички приказана трансформација NDL инструкције назива пословног правила у извршиву JAVA класу. Креирана JAVA класа мора бити јавна да би било могуће неограничено пута користити је из архиве репозиторијума *transf-as.jar* (Слика 6.1).



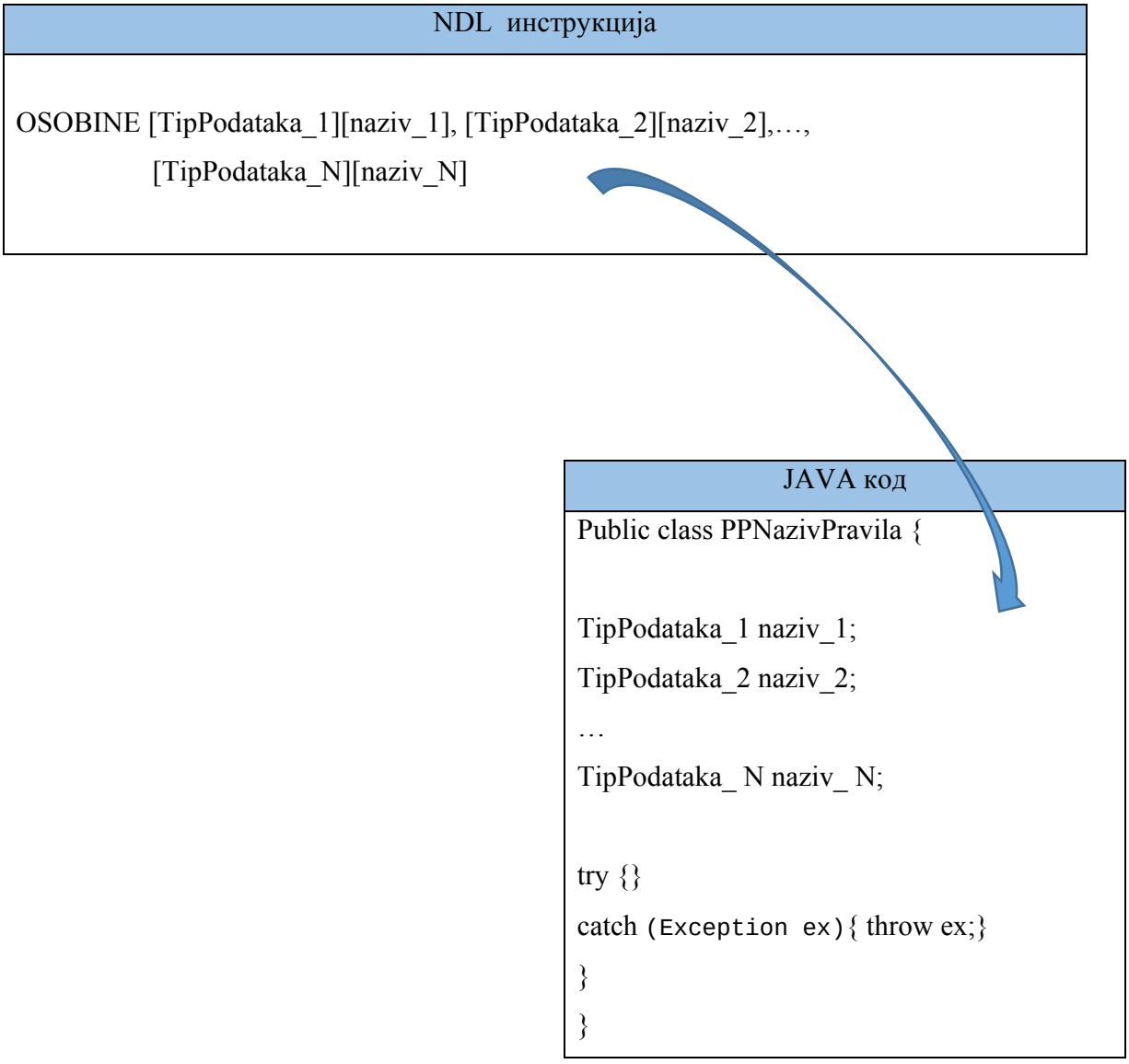
Слика 6.2. Трансформација NDL инструкције назива пословног правила у извршиву JAVA класу

6.1.2. Превођење особина пословног правила у JAVA код

Особинама правила одређени су атрибути класе објеката правила. Трансформацијама за превођење особина пословног правила, особине се ређају, по редоследу јављања, одмах

иза назива класе објеката правила. Свакој особини придружен је и одговарајући JAVA тип података.

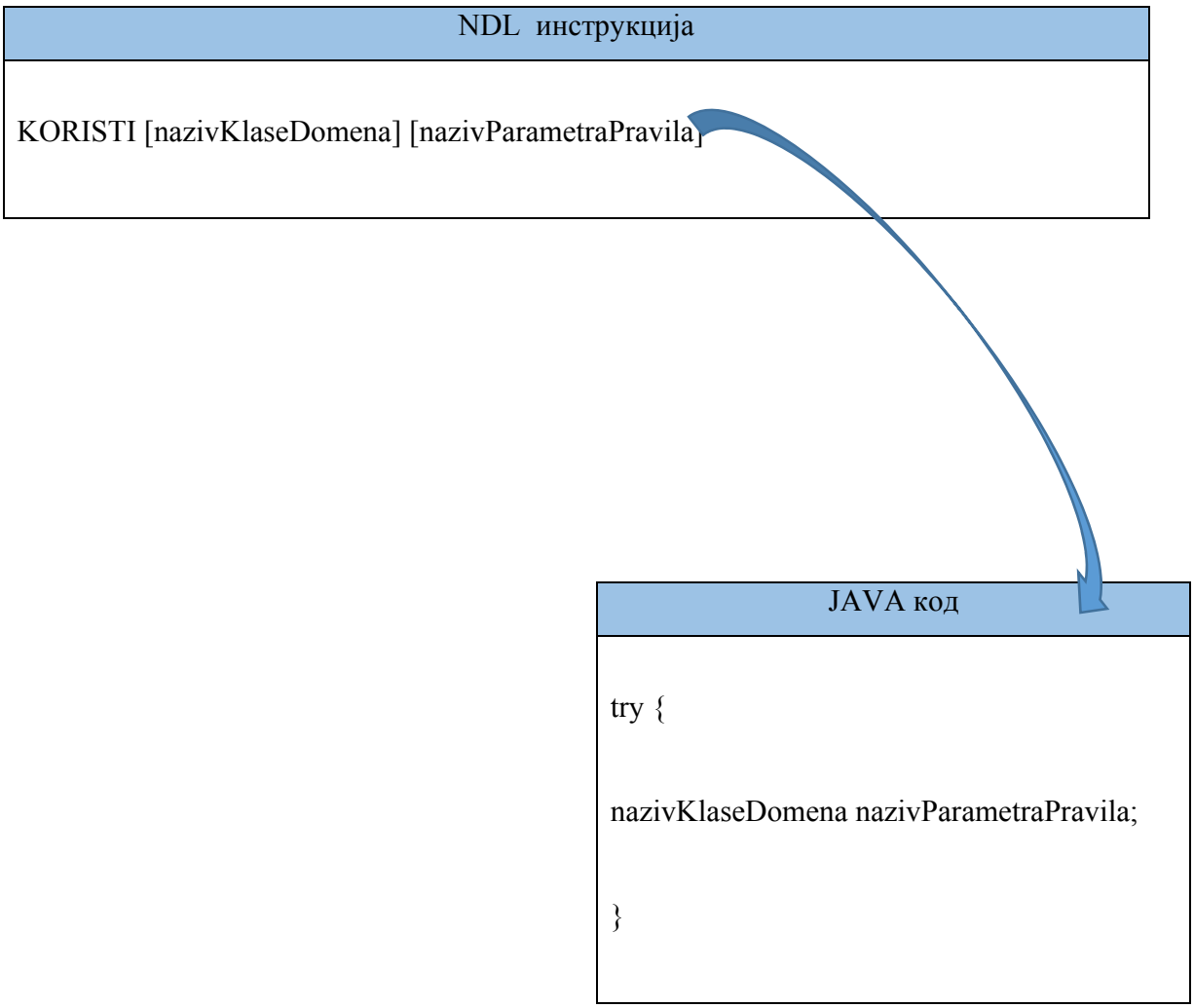
Следећом сликом је графички приказана трансформација NDL инструкције *OSOBINE* пословног правила у извршиве детаље JAVA класе објеката пословних правила.



Слика 6.3. Трансформација NDL инструкције *OSOBINE* пословног правила у детаље извршиве JAVA класе

6.1.3. Превођење инструкције *KORISTI* пословног правила у JAVA код

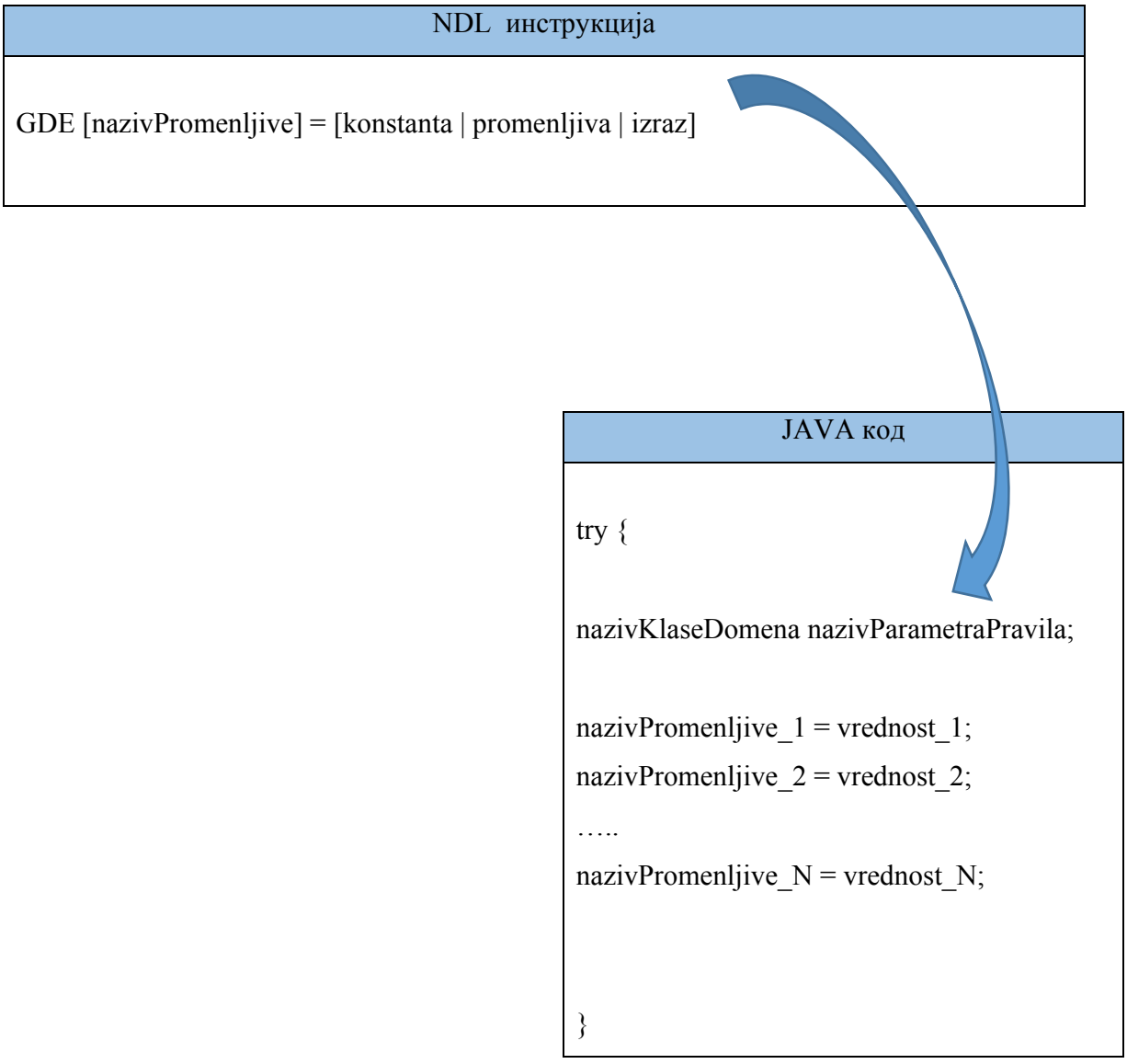
Као прва инструкција извршивог блока наредби *try*{}, класе објеката пословних правила, применом трансформација, смешта се садржај NDL клаузуле *KORISTI*, а на начин приказан следећом сликом. Управо ова инструкција представља прву JAVA инструкцију за реализовање логике примене пословног правила.



Слика 6.4. Трансформација NDL инструкције *KORISTI* пословног правила у детаље извршиве JAVA класе

6.1.4. Превођење инструкције *GDE* пословног правила у JAVA код

Софтверским трансформацијама за превођење пословних правила у класе објеката пословних правила, садржај NDL инструкције *GDE* додаје се у блок инструкција *try*{} одмах иза садржаја који одговара инструкцији *KORISTI*, а на начин приказан следећом сликом.

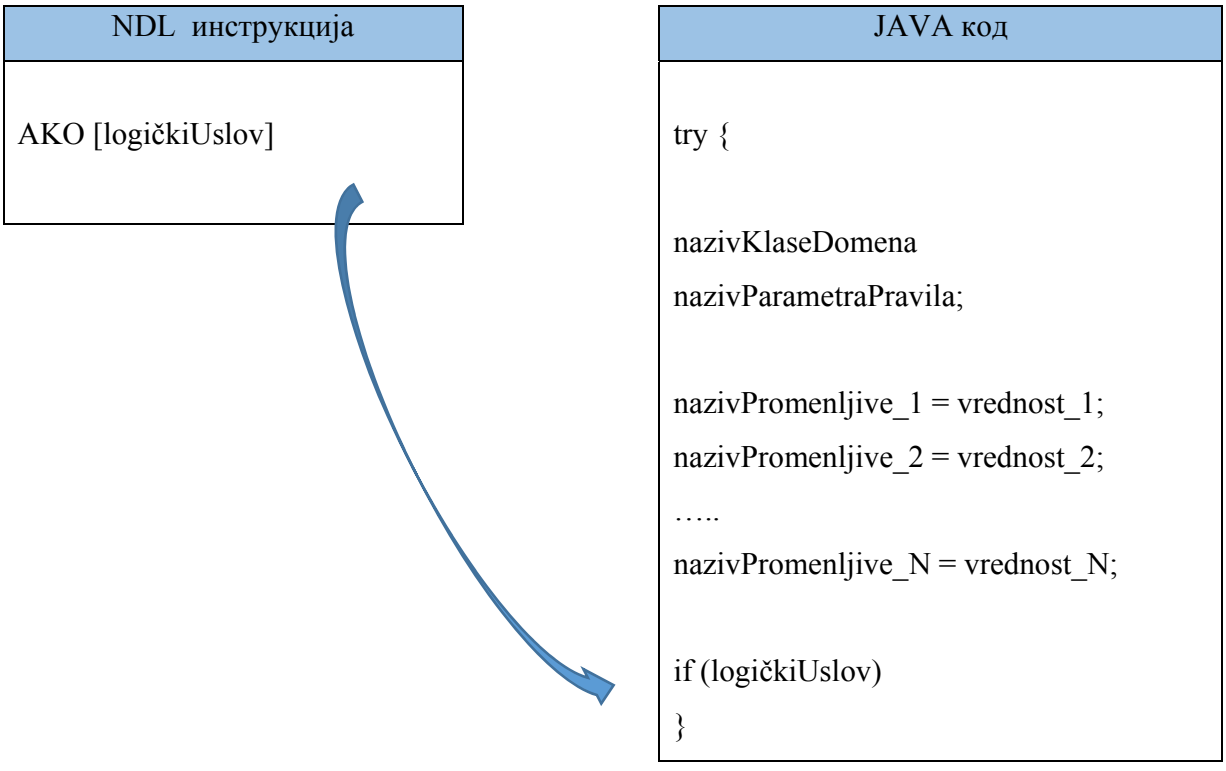


Слика 6.5. Трансформација NDL инструкције *GDE* пословног правила у детаље извршиве JAVA класе

Веома је важно напоменути да уколико назив неке промењиве инструкције *GDE* не одговара неком од атрибута класе, а јавља се први пут, испред њеног назива трансформација ће додати и одговарајући JAVA тип података.

6.1.5. Превођење инструкције *AKO* пословног правила у JAVA код

Применом софтверских трансформација за превођење NDL инструкција пословних правила у инструкције класе објеката правила, садржај клаузуле *AKO* заокружује дефиницију логике блока наредби *try*{}. Уколико су услови из овог блока задовољени, логика пословног правила биће извршена и контрола се пребацује на блок наредби *catch*.{}. Важно је напоменути да логички услов може бити прост или сложен од простих услова применом логичких оператора: *и* (&&), *или* (||) и *не* (!).

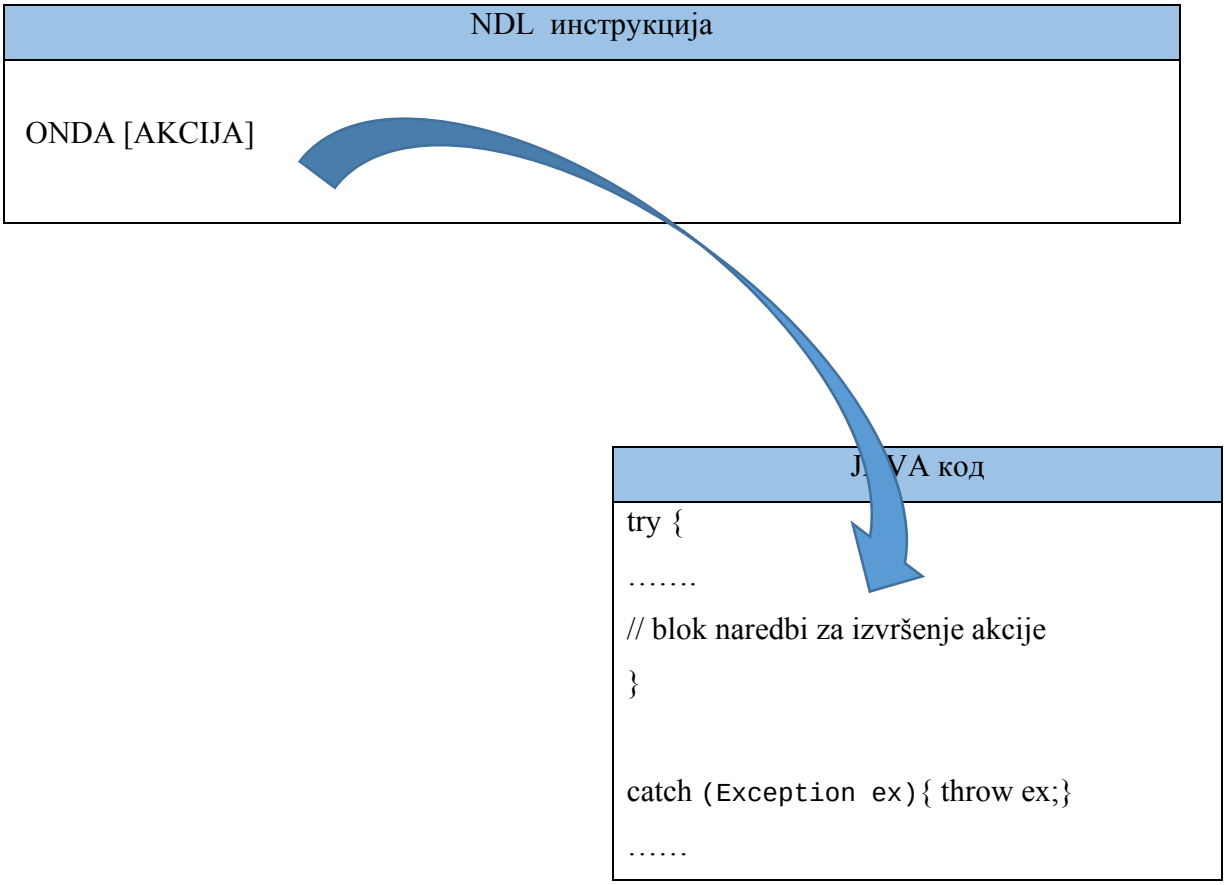


Слика 6.6. Трансформација NDL инструкције *GDE* пословног правила у детаље извршиве JAVA класе

6.1.6. Превођење инструкције *ONDA* пословног правила у JAVA код

Применом трансформација, садржај NDL инструкције *ONDA* смешта се у блок *try*{}, непосредно пре блока *catch{}* са инструкцијама за управљање изузецима. Уколико наведене инструкције нису *пресреле* ниједан системски изузетак, који би резултовао прекидом извршења логике правила, проверавају се услови из блока *try*{}, а затим се инструкцијом *return* враћа резултат примене правила.

Следећом сликом је графички приказана трансформација NDL инструкције *ONDA* пословног правила у извршиве детаље JAVA класе објеката пословних правила.



Слика 6.7. Трансформација NDL инструкције *ONDA* пословног правила у детаље извршиве JAVA класе

Сликом 6.8. приказани су резултати превођења комплетног пословног правила исказаног NDL језиком у JAVA класу објеката правила.

NDL инструкција

PP PPPopustNaPolisu
OSOBINE Real vrednostPolise, Real popust
KORISTI korisnik.brojUgovora nalog
GDE ciljniKorisnik = izbor.korisnik
AKO ciljniKorisnik.nalog.vrednostPolise >= vrednost
ONDA izbor.primeniPopust (ciljniKorisnik)

JAVA код

```
Public class PPPopustNaPolisu {
double vrednostPolise;
double popust;
try {

    korisnik.brojUgovora nalog;
    ciljniKorisnik = izbor.korisnik;
    if( ciljniKorisnik.nalog.vrednostPolise>= vrednost)
    return izbor.primeniPopust (ciljniKorisnik);
}

catch (Exception ex){ throw ex;}
}
```

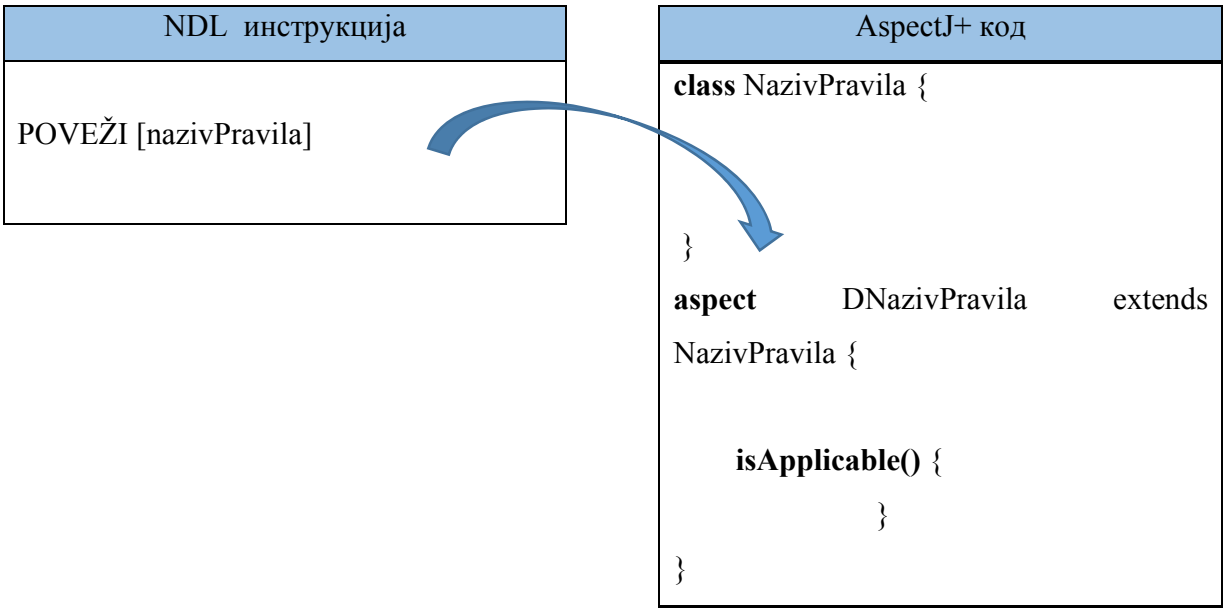
Слика 6.8. Трансформација NDL пословног правила у извршиву JAVA класу

6.2. Креирање трансформација за превођење NDL кода веза правила у извршив код

Применом трансформација на везе правила долази до креирања две специфичне класе које заједно чине аспект везе, којим је имплементиран алгоритам повезивања конкретног пословног правила са тачно одређеном класом управљачког дела софтверског решења. Прва класа крије логику повезивања пословног правила у тачки прекида, друга класа управља догађајем који представља *окидач* за покретање правила. Почетак друге класе, уместо стандардне резервисане речи *class*, истакнут је резервисаном речју *aspect*. Друга класа, директно проширује прву класу.

6.2.1. Превођење инструкције *POVEŽI* везе пословног правила у AspectJ+ код

Када се покрене трансформација за превођење NDL инструкције *POVEŽI*, софтверска компонента *Translator/AspectJ+* генерише празан и делимично дефинисан аспект конекције, а то је приказано следећом сликом.

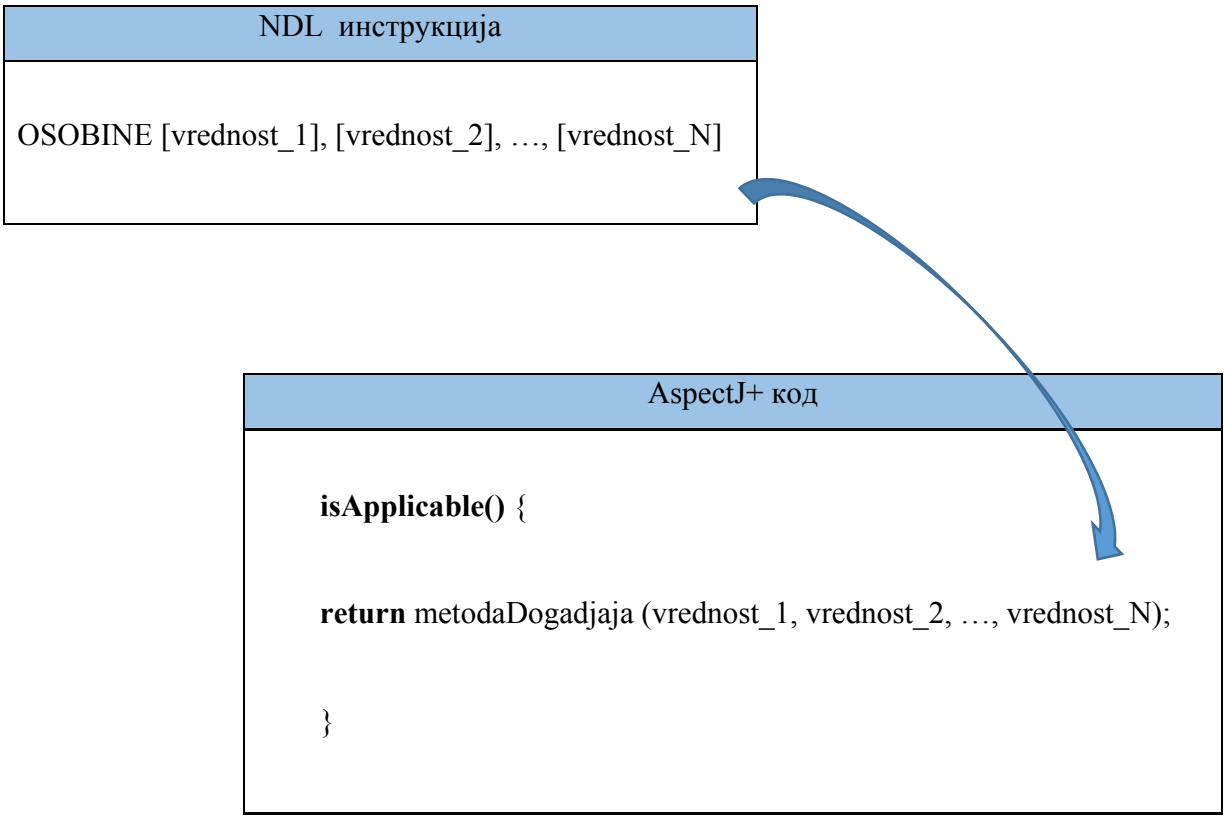


Слика 6.9. Трансформација NDL инструкције *POVEŽI* везе пословног правила у извршив *AspectJ+* аспект везе

AspectJ+ методом *isApplicable()* је омогућено реализовање догађаја који иницирају покретање пословног правила. Називом *DNazivPravila* додељено је генеричко име догађају којег преузима метода *isApplicable()*.

6.2.2. Превођење инструкције *OSOBINE* везе пословног правила

Особинама везе пословног правила, након примене трансформација, предају се неопходне информације за извршење логике пословног правила. Наведене вредности се прослеђују у методу *isApplicable()* као аргументи методе догађаја, а то је представљено следећом сликом.



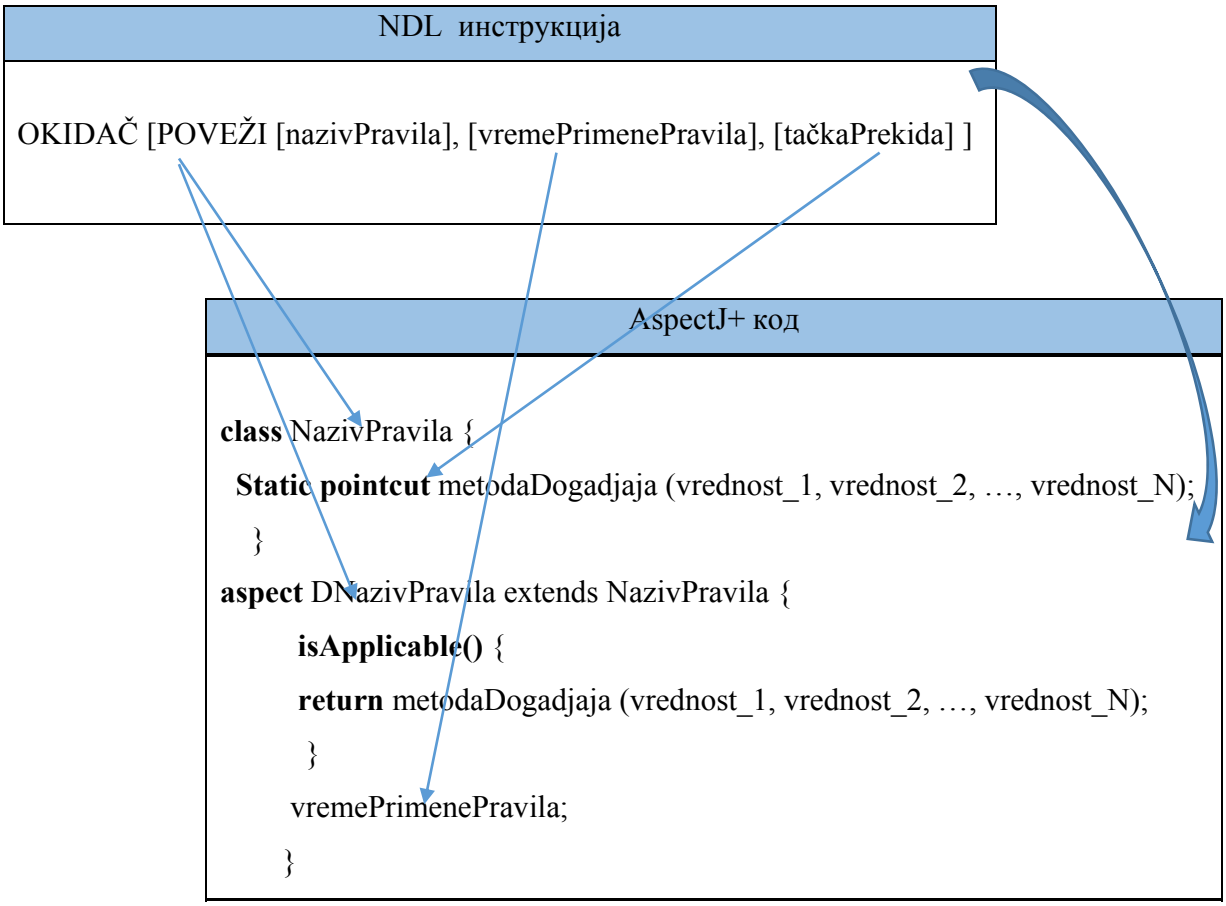
Слика 6.10. Трансформација NDL инструкције *POVEŽI* везе пословног правила у извршиве детаље *AspectJ+* аспекта везе

6.2.3. Превођење покретања догађаја

Превођење инструкције:

OKIDAČ [POVEŽI [nazivPravila], [vremePrimenePravila], [tačkaPrekida]],

представља најсложенији облик имплементације наведених софтверских трансформација. Трансформацијама са тачка прекида везује за класу управљачког дела софтверског решења у којој се правило примењује. Време примене правила наводи се одмах иза дефиниције методе *isApplicable()*, а инструкција *POVEŽI [nazivPravila]* везује догађај *окидач* за назив пословног правила.

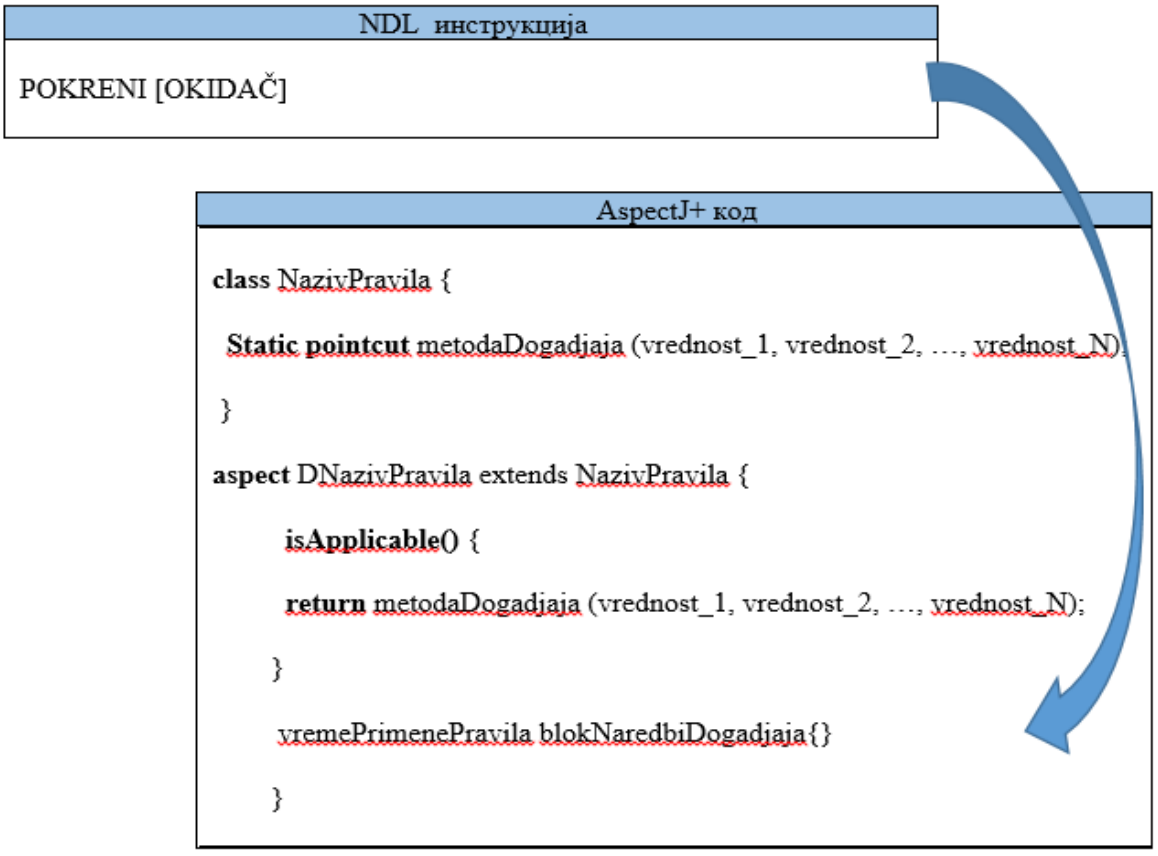


Слика 6.11. Трансформација покретања догађаја у детаље *AspectJ+* аспекта везе

Након извршавања трансформација приказаних Сликом 6.11, статичка метода pointcut() ставља показивач на блокове наредби try{} и catch{} одговарајуће класе објеката пословних правила.

6.2.4. Превођење инструкције *POKRENI [OKIDAČ]* везе пословног правила

Превођењем ове NDL инструкције заокружује се дефиниција трансформација за превођење пословних правила и њихових веза у извршиве класе објеката правила и аспекте њихових веза респективно. Превођењем инструкције *POKRENI [OKIDAČ]*, одмах иза резервисане речи, којом је одређено време примене пословног правила, додаје се логика догађаја којом се иницира покретање одговарајућег објекта пословног правила. Наведено је приказано Сликом 6.12.



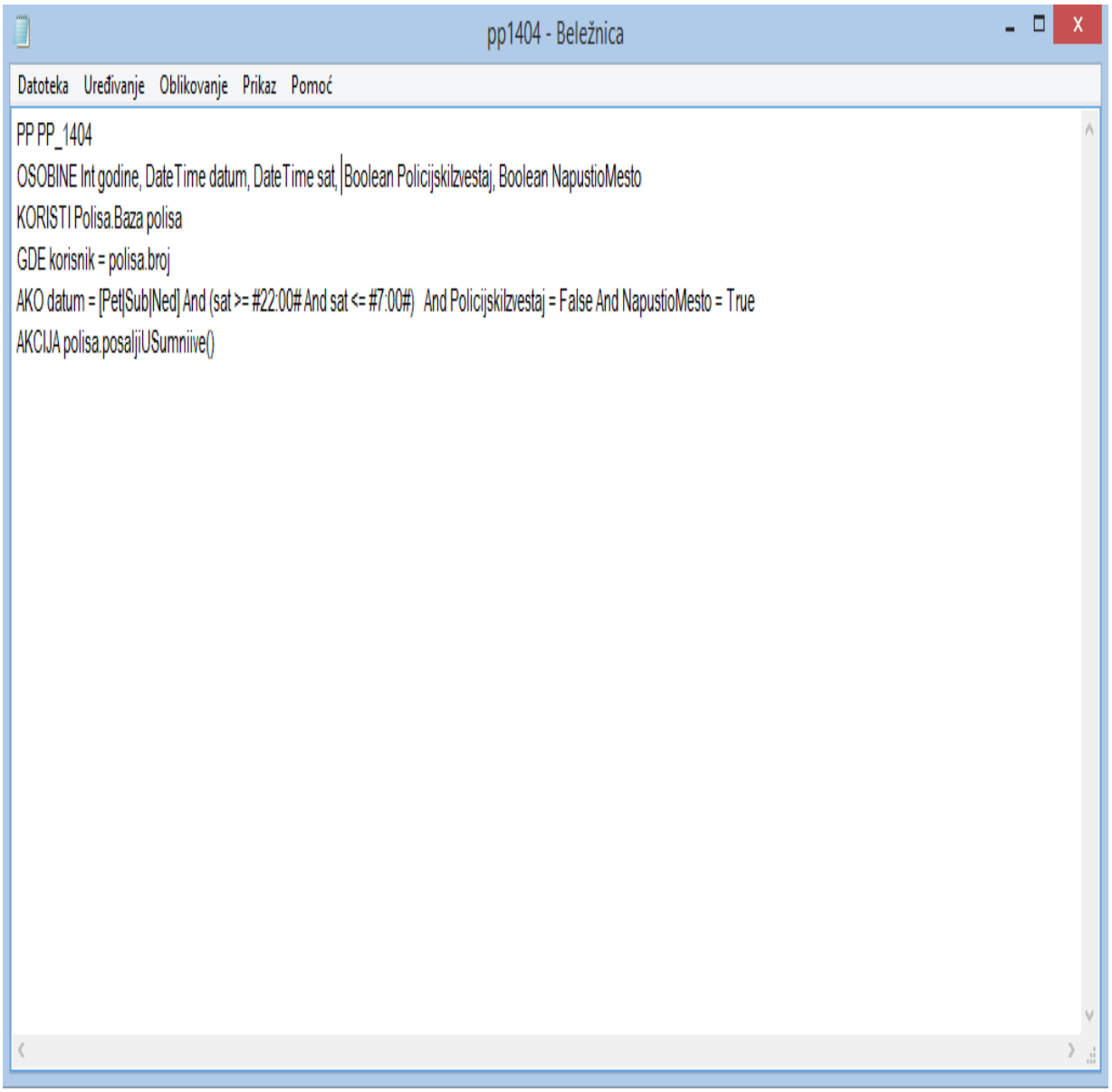
Слика 6.12. Трансформација инструкције инструкције *POKRENI [OKIDAČ]* у детаље *AspectJ+* аспекта везе

AspectJ+ кодом, са Слике 6.12. у потпуности је заокружена општа дефиниција аспеката везе пословних правила. У даљем излагању акценат ће бити на демонстрацији управљања трансформацијама софтверском компонентом *Translator/AspectJ+*.

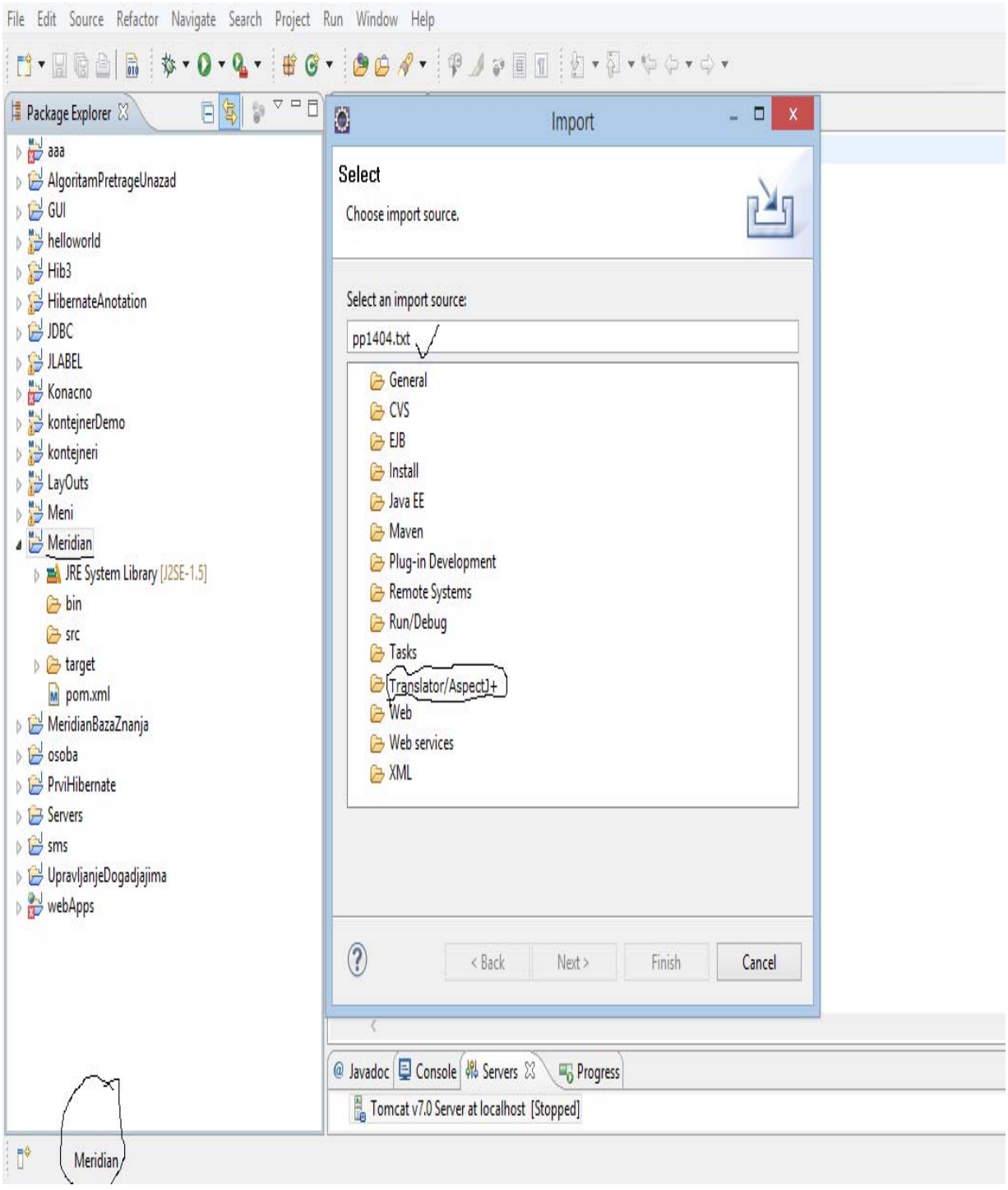
6.3. Превођење правила и веза у извршив програмски код

Оперативност софтверске компоненте Translator/AspectJ+ биће демонстрирана на примеру пословног правила изолованог софтверском компонентом NDL/Generator (Поглавље 5, део 5.1.1) датим кодом са Слике 5.11. и одговарајуће везе правила. Наведени код правила сачуван је као текстуални документ *pp1404.txt* (Слика 6.13), а затим предати на превођење (Слика 6.14). Софтверско решење ће реаговати тако што ће применом компоненте Translator/AspectJ+ прво превести правило и везу у извршив код, а затим аутоматски применити логику правила. Потребно је још додати да је дато правило генерисано на потпуно исти начин, након обраћања питањем HelpDesk систему, као што је случај описан у Поглављу 7, део 7.1.1.

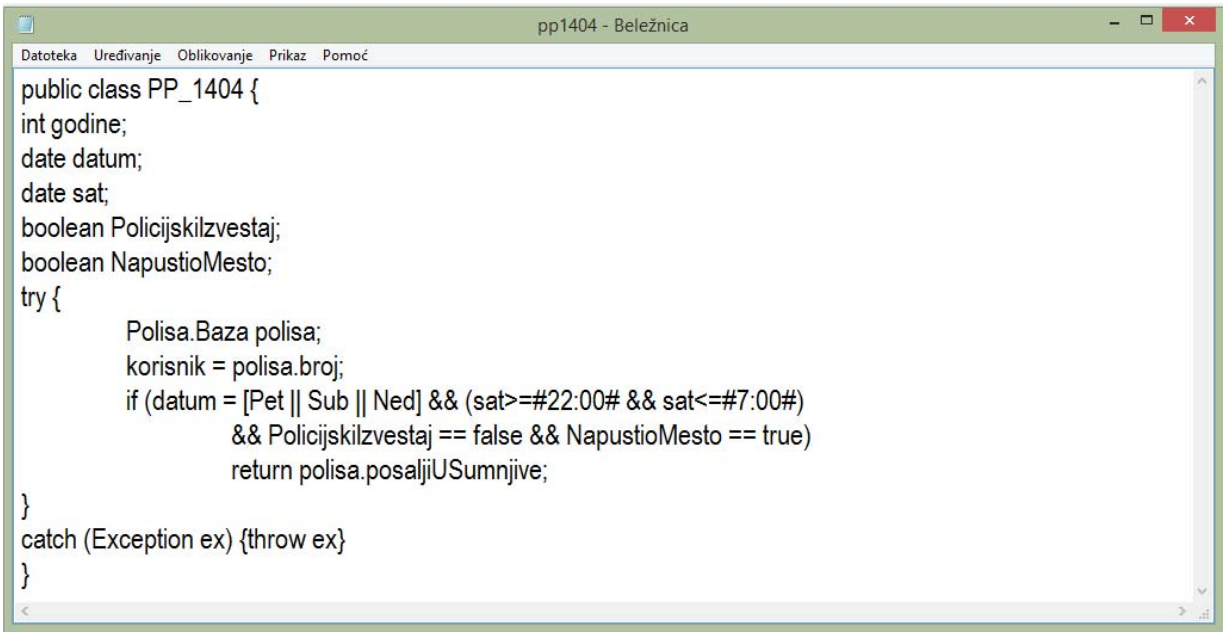
Као резултат, софтверско решење ће генерисати два текстуална документа. Први документ, означен као *pp1404.java*, садржаће резултате превођења изолованог правила у извршив код (Слика 6.15). Другим документом, *report_ pp1404.txt*, ће бити презентован резултат примене пословног правила, у комбинацији са додељеном везом, у форми извештаја (Слика 6.16).



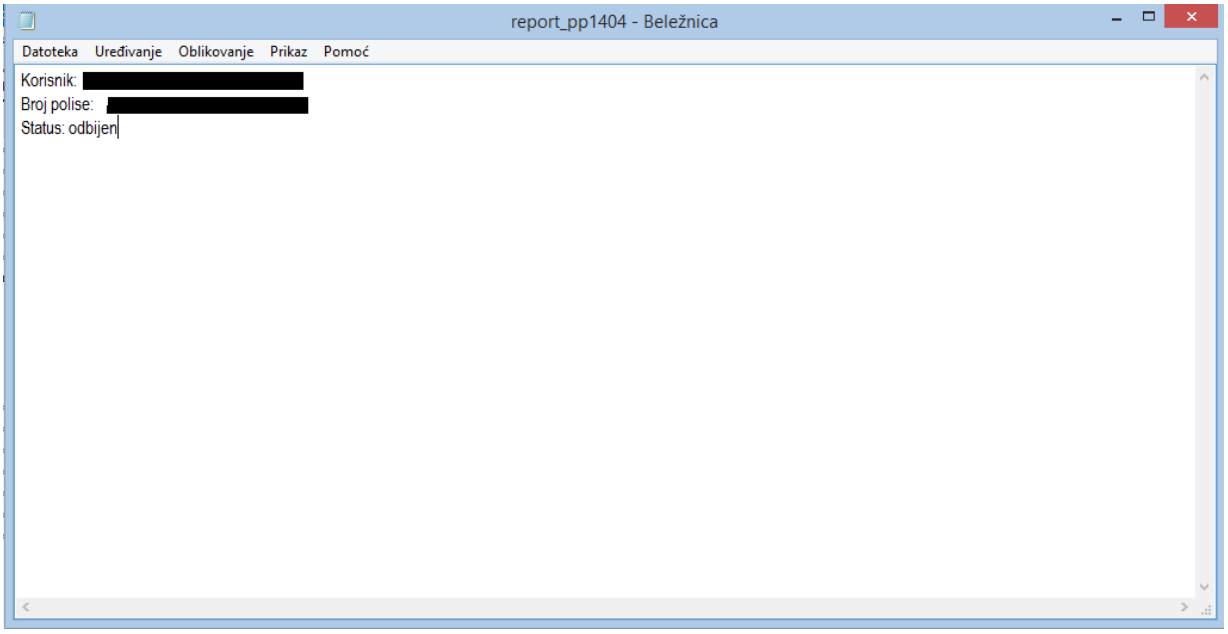
Слика 6.13. Текстуални документ са пословним правилом и његовом везом



Слика 6.14. Предавање правила и његове везе компоненти *Translator/AspectJ+*



Слика 6.15. Извршив код пословног правила након превођења



Слика 6.16. Извештај примене пословног правила

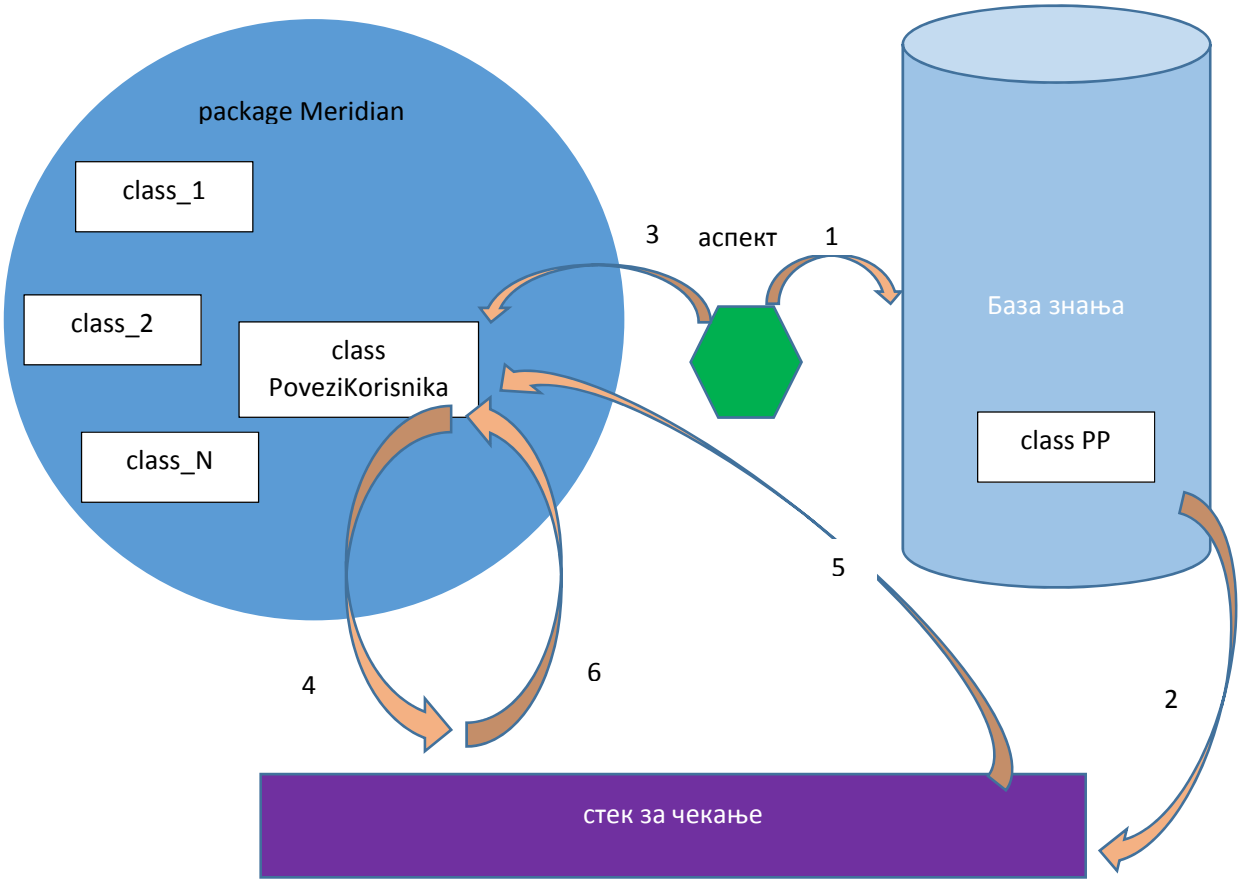
6.4. Повезивање правила са делом софтверске апликације за управљање знањем

Иако је Сликама 6.14, 6.15 и 6.16 демонстрирано повезивање правила са главним делом софтверске апликације, неопходно је детаљније објаснити саму логику повезивања. Све класе главног дела софтверског решења припадају програмском пакету са називом *Meridian*. У овом пакету се налази велики број класа којима је имплементирана логика пословног информационог система за подршку рада компаније из области осигурања. Од посебног значаја за овај део рада јесте скуп класа које припадају тзв. флексибилном слоју за управљање знањем и које имају за намену прихватање пословних правила преко одговарајућих аспеката веза. Овде посебно долази до изражаја објектно – оријентисани концепт *колаборација*[96] помоћу које класе сараднице комуницирају на веома флексибилан начин помоћу својих операција, односно дужности. Помоћу догађаја, чију логику имплементира аспект везе, посматрани објекат правила ставља се на чекање, а затим се, датим аспектом, врши обраћање класи. У зависности од три могућа сценарија: пре, после или уместо, аспект везе зауставља извршавање класе управљачког дела софтверског решења, скида објекат правила са чекања и омогућава му имплементацију. Након завршене примене логике правила, аспект везе затвара конекцију између базе знања и управљачког дела софтверског решења, а затим укида прекид са класе која је повезала објекат правила. У наставку, софтверско решење наставља нормално са својим извршавањем, а то је илустровано Сликом 6.17.

Наведено излагање има за последицу да велики број класа, било да припадају главном делу софтверског решења или бази знања, у сваком тренутку извршавања софтверског решења налази се у статусу мировања и активира се искључиво у ситуацијама када преко аспекта везе извесни догађај то иницира. Ово је од великог значаја за стабилност и утрошак меморије од стране посматраног софтверског решења из разлога што програмски језик JAVA не подржава концепт деструктора објеката. Када се креира објекат, класе правила или управљачког дела, одговарајућим конструктором он иницира извесни софтверски процес. Све док постоје инструкције које указују на дати објекат, посматрани процес је активан и користи меморију. Када се истроше поменуте инструкције, не долази до аутоматског укидања објекта већ се он шаље у тзв. *JAVA сакупљач смећа* (енг. *JAVA garbage*

[96] uml.org. www.uml.org.ck/umlapplication/pdf/crcmodeling.pdf (приступљено 22.01.2015. у 22:15)

collector).[97] То значи да иако више не постоји потреба за њим, креирани објекат ће још неко време користити меморију, све док се поменуто складиште у неком тренутку системски не испразни. Што је више активних JAVA процеса, овај проблем је израженији. Наведено тврђење може најбоље да се илуструје на примеру оперативног система *Android*.[98] На овом оперативном систему доминирају софтверска решења креирана JAVA програмским језиком. У сваком тренутку покренут је велики број апликација, који из горе наведеног разлога могу да искористе сву RAM меморију и корисници се често сусрећу за успореним радом или блокирањем мобилних уређаја. Начином на који је пројектовано и кодирано софтверско решење, које је предмет истраживања овог рада, ови недостаци су сведени на најмању могућу меру.



Слика 6.17. Процес извршавања логике пословног правила по корацима

[97] javabook.compuware.com. www.javabook.compuware.com/content/memory/how-garbage-collection-works.aspx (приступљено 22.01.2015. у 22:30)
[98] phonearena.com. http://www.phonearena.com/news/Why-Android-phones-need-3GB-of-RAM-and-iOS-gets-by-with-1GB-of-the-stuff_id62901 (приступљено 22.01.2015. у 22:40)

7. Развој интелигентног HelpDesk система за подршку компанијама из индустрије осигурања

Савремени информациони системи увелико користе интелигентне алате за откривање, препознавање и предвиђање новог знања. Област примене оваквих система је огромна и скоро да не постоји област индустрије и услуга где овакви системи нису заступљени. Управо зато, циљ развоја оваквих система јесте да се у будућности добије такав информациони систем који ће имати пуну функционалност уз могућност надоградње без интервенције софтверског развојног тима.

У овом делу рада приказано је интелигентно софтверско решење, као карика у низу ка оптималном софтверском решењу. Посебно, дата је методологија по којој је развијен прототип предложеног софтверског решења и која представља валидан основ за развој софтверских решења за различите области индустрије и услуга. Посебна пажња посвећена је интелигентним решењима која већ имају практичну примену и која могу бити надограђена и прилагођена конкретном проблему у светлу излагања из претходних поглавља. Такође, у раду су предложена побољшања конкретног интелигентног механизма претраге[99] [100] на основу недостатака уочених током вишегодишње експлоатације и тестирања. Посебно, дефинисањем језика високог нивоа као подскупа српског језика, посматрани систем може бити надограђен на основу захтева експерата, који не само да не морају да имају програмерско знање, већ не морају да се служе ни енглеским језиком. Применом, преводилачког софтвера наведена решења могу да добију додатну димензију.

У савременом пословном окружењу сваки пословни субјект нужно је усмерен ка употреби информационо комуникационих технологија. Квалитет модерних софтверских решења за подршку пословању, у директној је вези са ефикасношћу и ефективношћу обављања делатности и постизањем стратешке предности над конкуренцијом. Због повећања моћи претраге, препознавања и предвиђања стања на тржиштима производа и услуга, све чешће је у развоју информационих система акценат на напредним интелигентним техникама и алатима, попут неуронских мрежа, генетских алгоритама, фази

[99] Stankić R, Milićević V, Popović M, Savić Z. 2012. *Contribution to Intelligent System for Automatic Business Ruless Management Development*, TTEM Vol. 7/1, 2012.
[100] Милићевић В. 2012. *Допринос развоју инрелигентног рачунарског система базираног на приступу пословних правила*, ФАМС

логике и сл. Област примене оваквих система је огромна, од препознавања понашања потрошача у великим ланцима хипермаркета[100], преко примене у индустрији осигурања[101], за предвиђање кретања на берзама и тржиштима[102], до роботике[103], система паметних сензора[104], мобилних уређаја[105], напредних система рекламирања и оглашавања[106] итд.

Од посебног је значаја могућност прилагођавања софтверског решења за подршку пословања динамичким променама савременог пословног окружења. Аутори Кодингтон и Вилсон тврде да: *Информационе технологије (ИТ) и информациони системи имају централну улогу у пословима осигурања и осигуравајуће компаније препознају потребу за новим димензијама пословања, освајајући применом ИТ нова тржишта, држећи тако корак са конкуренцијом*.[107]

Такође, неопходно је напоменути и различите облике примене савремених софтверских решења у области осигурања:[108]

- Управљање односима са потрошачима;
- Управљање ефективношћу продаје;
- Обезбеђивање тренутног приступа информацијама и решењима корисницима;
- Подсећање корисника на производе и услуге електронским сервисима;
- Подршка за електронске упите и трансакције који подижу ниво задовољства корисника;
- Аутоматско извештавање и праћење перформанси и продуктивности.

---

[101] Shapiro A. F. 2007. *An Overview of Insurance Uses of Fuzzy Logic*, Springer

[102]stanford.edu. http://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Applications/stocks.html (приступљено 29. 11. 2014. у 14:40)

[103] Medina-Santiago A, Camas-Anzueto J. L, Vascuez-Feijoo J. A, Hernández-de León H. R, Mota-Grajales R. (2014). *Neural Control System in Obstacle Avoidance in Mobile Robots Using Ultrasonic Sensors*, Journal of Applied Research and Technology, vol. 12 (2014), 104-110

[104] Rivera-Mejía J, Léon-Rubio A. G, Arzabala-Contreras E. (2012). PID Based on a Single Artificial Neural Network Algorithm for Intelligent Sensors, Journal of Applied Research and Technology, vol. 10 No.2 (2012), 262-282.

[105] Nalepa, G. J., Bobek, S.: Rule-Based Solution for Context-Aware Reasoning on Mobile Devices. Computer Science and Information Systems, Vol. 11, No. 1, 171–193. (2014)

[106] Rodríguez-González, A., Torres-Niño, J., Jimenez-Domingo, E., Gomez-Berbis, J. M., Alor-Hernandez, G.: AKNOBAS: A Knowledge-based Segmentation Recommender System based on Intelligent Data Mining Techniques. Computer Science and Information Systems, Vol. 9, No. 2, 713-740. (2012)

[107] Codington, S. & Wilson, T.D. (1994). Information System Strategies in the U.K. Insurance Industry International Journal of Information Management, 14(3), 188-203

[108] safaribooksonline.com . http://my.safaribooksonline.com/book/management/9788131759844/case-study-on-reliance-life-insurance-use-of-information-system-as-a-strategic/c08-sec1-011 (приступљено 29. 11. 2014. у 14:50)

Наведене тезе требало би проширити и могућностима електронског комуницирања унутар саме компаније. Развој одговарајућег HelpDesk система, где би запослени добијали савете од експерата различитог нивоа, могао би у великој мери да унапреди пословање осигуравајуће компаније. Добијањем прецизних и правовремених информација, задовољство запослених било би на вишем нивоу, а самим тим и ефективност и ефикасност њиховог рада. Међутим, брзина одговора на постављено питање зависи од тога када ће експерт, којем је питање упућено, реаговати. Отуда, намеће се потреба за размишљањем и истраживањем на пољу развоја софтверских решења која ће експертизу, у највећој могућој мери, померити од човека ка рачунару.

7.1. Компоненте система

Софтверско решење, које је предмет истраживања, веома је комплексно и рађено је за потребе компаније *Meridian Project*, где је прототип имплементиран и тестиран. Услови у којима је ангажовано посматрано софтверско решење подразумевају:

- Централни рачунар са подигнутим Linux сервером (слика 7.1) на којем је инсталиран управљачки део софтверског решења, програмиран JAVA програмским језиком, са базама података и знања;
- База података је подигута над MySQL сервером, а објектно-релационо је управљана алатом *Hibernate ORM*;
- База знања је креирана *AspectJ*+ нотацијом и дозвољава мануелно и аутоматско ажурирање;
- Клијент софтверска JAVA решења су инсталирана на двадесет рачунара, повезаних локално и Интернетом са сервером;
- Мобилна клијент софтверска JAVA решења су инсталирана на двадесет паметних телефона, повезаних Интернетом са сервером;
- Резервне копије база података и знања редовно се чувају и ажурирају на „облак“ серверу (eng. Cloud Server).
- Унапређени алгоритам претраге знања инсталиран је као део HelpDesk модула управљачког дела софтверског решења.

Слика 7.1. Подизање Linux сервера

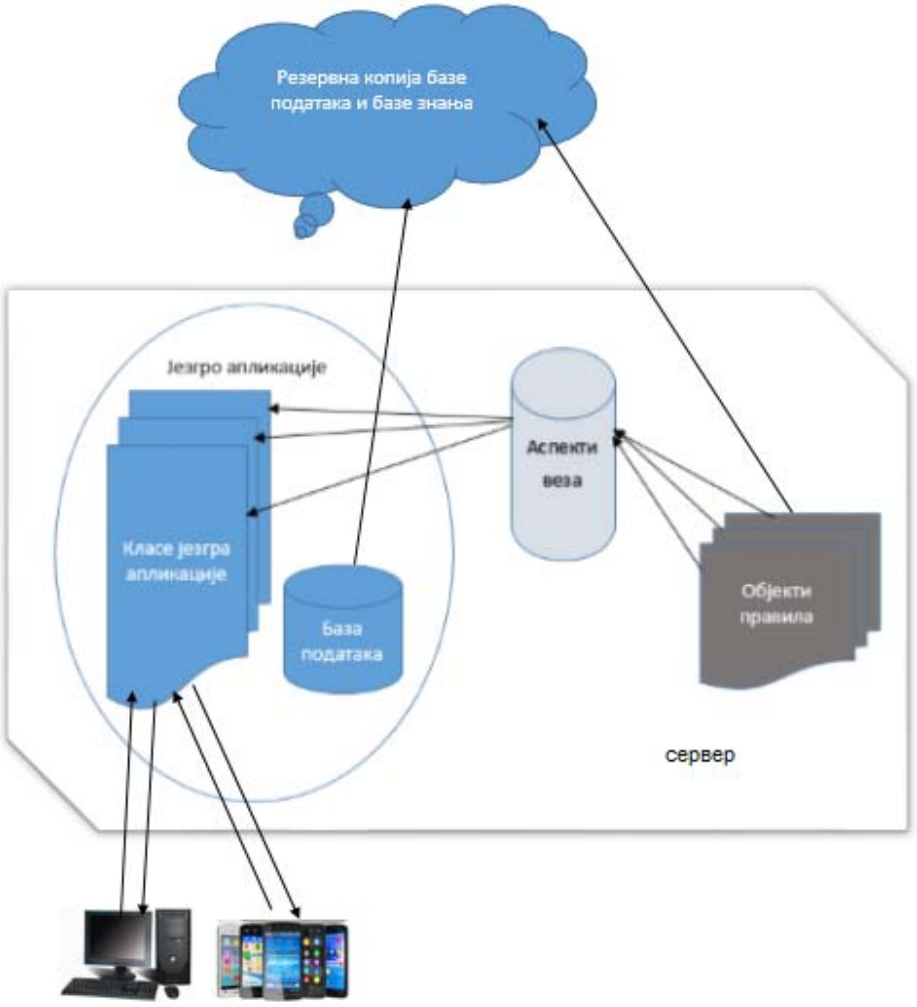
На основу предочених алата и технологија, Сликом 7.2. је могуће презентовати архитектуру посматраног софтверског решења.

7.1.1. Аспектно–оријентисана база знања

Користећи аспектно-оријентисани приступ AspectJ+ креирана је база знања у којој су објекти правила организовани по категоријама:

- Правила која су у вези са питањима везаним за производе и услуге осигуравајуће компаније;
- Правила која су у вези са питањима везаним за законску регулативу која је од значаја за индустрију осигурања;
- Правила која се односе на финансијско пословање компаније и доступна су искључиво уз посебну дозволу приступа;

- Правила којима је могуће открити лажну изјаву дату приликом подношења одштетног захтева.



Слика 7.2. Архитектура софтверског решења *Meridian*

Путем HelpDesk система, корисник производа и услуга осигуравајуће компаније, може добити одговор на питање које је у директној вези само са првом категоријом корисника. Остале категорије креирају одговоре на постављена питања службеника компаније путем HelpDesk компоненте.

Само пословно правило, заједно са одговарајућом везом, има две репрезентације:

- Као правило исказано језиком високог нивоа, препознатог механизмом претраге;
- Као правило преведено у проширени JAVA код, применом одговарајуће софтверске трансформације (претходно поглавље).

Будући да је софтверско решење, током тестирања, са успехом обавило задатке откривања лажних изјава подносилаца одштетних захтева осигуравајућој компанији, на следећим примерима биће демонстрирани управо наведени начини репрезентације правила информационим системом *Meridian*.

Пример број 1:

Путем HelpDesk система службеник је проследио питање за које база знања не поседује шаблон за идентификацију. Питање је идентично следећој изјави: "*Дана 8. 6. 2014, око 1 сат после поноћи, враћао сам се из града. На извесној раскрсници, аутомобил црвене боје прошао је кроз црвено светло и ударио моје возило у задња врата. Возач који је изазвао удес је побегао са лица места, а мени је била празна батерија мобилног телефона и нисам могао да позовем полицију. Аутомобил је био у возном стању и ја сам отишао кући. Молим Вас да ми на основу полисе број БП 0003062014 надокнадите штету. – Петар Петровић, ЈМБГ 0706993******"*.

Из наведене изјаве, експерту је јасно да је мушка особа, стара 21 годину (на основу ЈМБГ), на властити рођендан била у граду. Такође, тај дан је био викенд, а на основу искустава осигуравајуће компаније управо тада се највише конзумира алкохол. Оштећени је напустио место удеса без сачињеног полицијског записника који је највалиднији документ за одштетни захтев.

Након обраде узорка, експерт је добио изјаву, у форми новог знања, коју мора да класификује као валидну или сумњиву. Софтвер је генерисао следећи код који одговара новом правилу:

```
PP PP_1404

OSOBINE Int godine, DateTime datum, DateTime sat,

        Boolean PolicijskiIzvestaj, Boolean NapustioMesto

KORISTI Polisa.Baza polisa

GDE korisnik = polisa.broj

AKO datum = [Pet|Sub|Ned] And (sat >= #22:00# And sat <= #7:00#)
And PolicijskiIzvestaj = False And NapustioMesto = True

AKCIJA polisa.posaljiUSumniive()
```

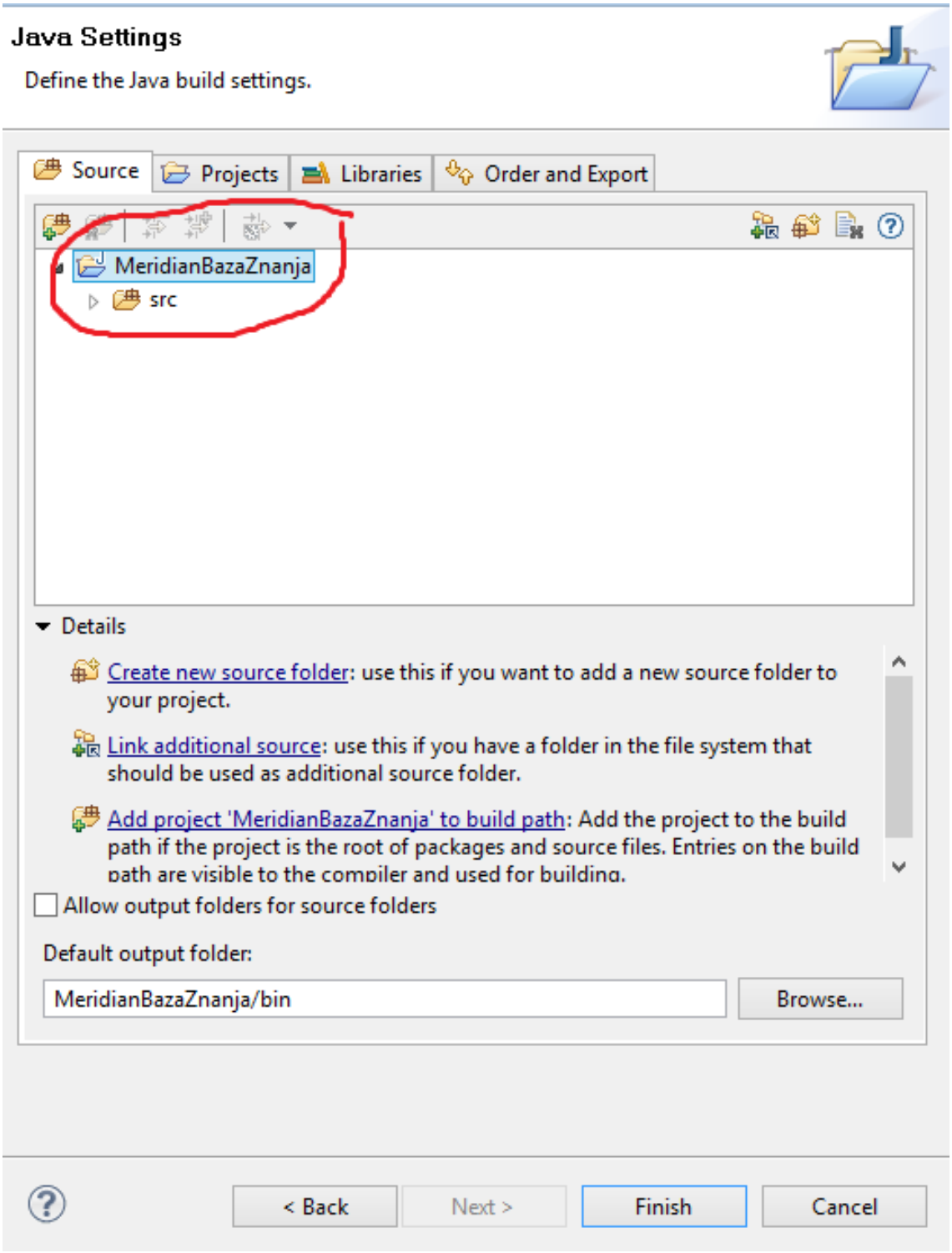
Код 7.1. Правило креирано након процесирања изјаве

Правило је аутоматски добило генеричко име по редном броју у бази знања, типови података су задржани као речи енглеског језика и идентични су JAVA типовима података, а то значајно олакшава посао трансформације у програмски код, инструкције су на српском језику и означене су као службене речи за механизам који врши њихову трансформацију у инструкције објеката правила.

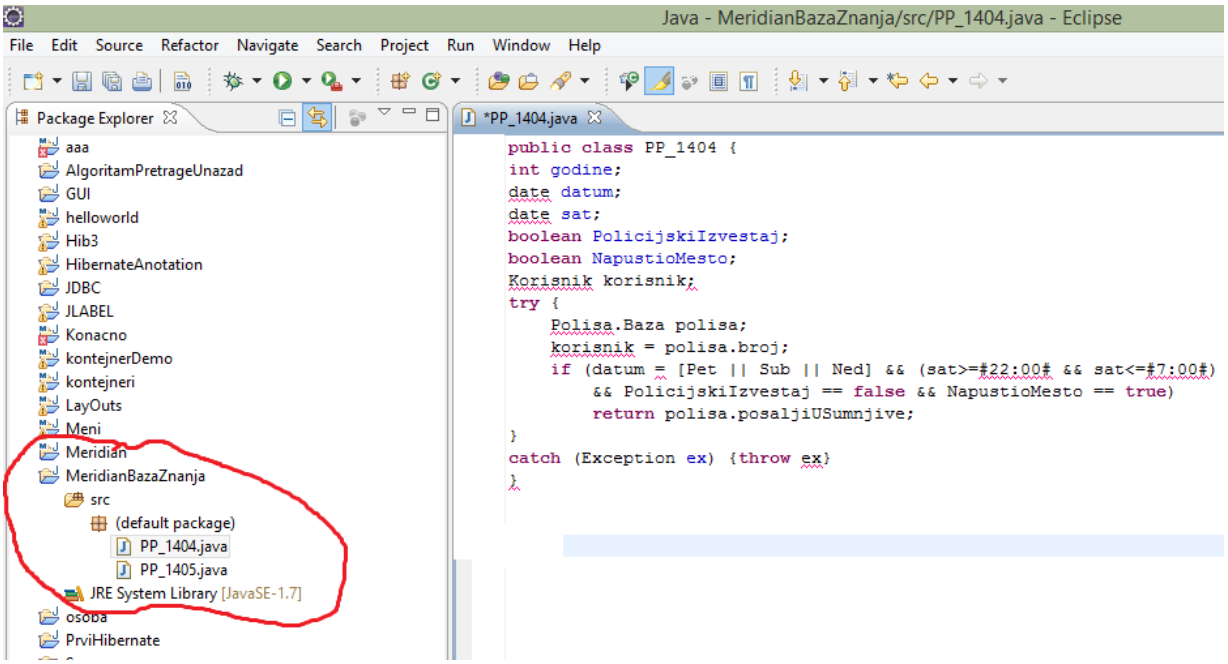
Рад система, у овој тачки може да има два сценарија и оба су успешно тестирана:

- Једним кликом експерт шаље ново правило у базу знања;
- Систем аутоматски шаље правило у базу знања.

Оба сценарија извршавају идентичну софтверску трансформацију која има за резултат похрањено правило у бази знања (слика 7.4). Да би могао да буде приказан сваки корак у креирању новог правила, за овај пример ће бити примењен први сценарио (слика 7.3).



Слика 7.3. Слање новог правила у базу знања



Слика 7.4. Имплементирано правило у бази знања

Након ангажовања овог правила службеник, који је путем HelpDesk система поставио питање, добиће системски извештај да одбије одштетни захтев.

Пример број 2:

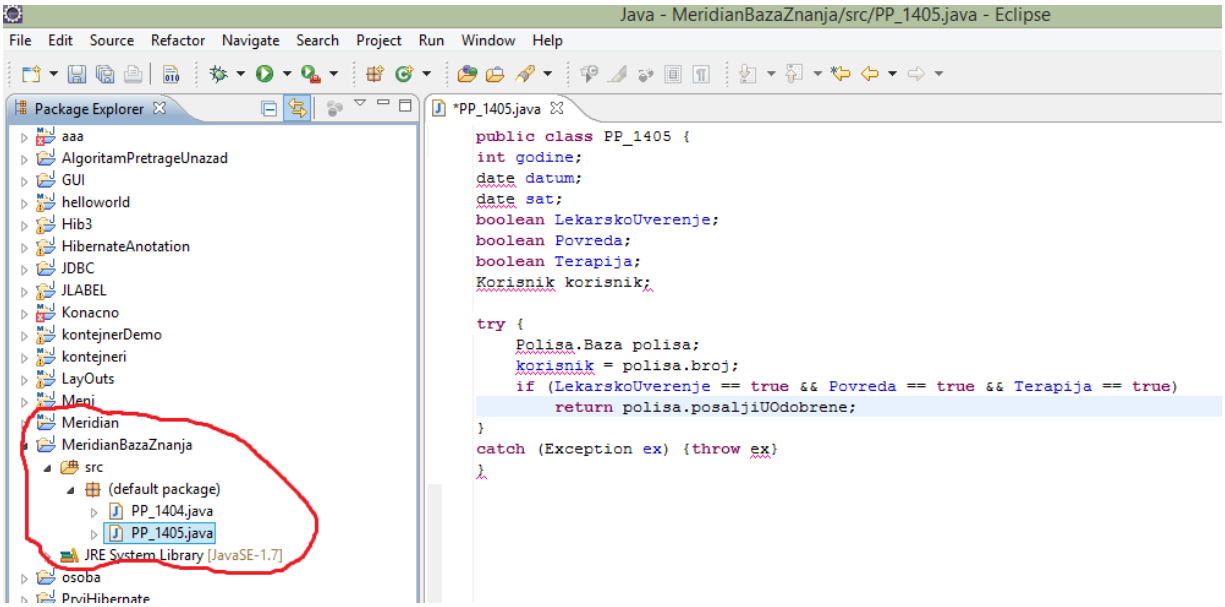
Такође, у бази знања не постоји шаблон који одговара следећој изјави: „Дана 6. 6. 2014, на радном месту, доживео сам пад низ степенице. Истог тренутка, одвезен сам код лекара где ми је констатована повреда – ишчашење зглоба. Добио сам терапију и лекарско уверење као доказ за покретање одштетног захтева и одсуствовање са посла у периоду од 20 дана. Молим Вас да ми на основу полисе број *БП* 0003062028 надокнадите штету. – Никола Николић, ЈМБГ 1106973******”.

На основу прослеђене изјаве, експерт има податке који му указују да је особа која поседује полису која покрива повреду на послу, благовремено се обратила лекару и добила све неопходне доказе за одобравање одштетног захтева. На основу претраге, софтвер је

генерисао правило (код 7.2) које је аутоматски прослеђено класама за трансформацију са циљем превођења у извршив код (слика 7.5):

```
PP PP_1405

OSOBINE Integer godine, DateTime datum, DateTime sat,
      Boolean LekarskoUverenje, Boolean Povreda,
      Boolean Bolovanje, Boolean Terapija

KORISTI Polisa.Baza polisa

GDE korisnik = polisa.broj

AKO LekarskoUverenje = True And Povreda = True And Bolovanj = True
And Terapija = True

AKCIJA polisa.posaljiUOdobrene()
```

Код 7.2. Правило креирано након процесирања изјаве



Слика 7.5. Имплементирано правило у бази знања

Након ангажовања овог правила службеник, који је путем HelpDesk система поставио питање, добиће системски извештај да прихвати одштетни захтев.

Наведеним тестовима показано је да је могуће изградити функционалан информациони систем, заснован на знању, који на веома флексибилан начин, аутоматски открива, генерише и интегрише ново знање у властиту базу знања. Наведеним тестовима доказана је Хипотеза бр. 1.

7.1.2. Интелигентни подсистеми за идентификовање новог и препознавање постојећег знања

Као валидан основ за пројектовање и изградњу интелигентног подсистема за аутоматско препознавање, кодирање и имплементацију новог знања, овај рад је имао у познатим радовима[99] [100] у којима је посебан акценат стављен на избор оптималног алгоритма претраге. Међутим, током вишегодишње експлоатације у систему електронске трговине уочени су извесни недостаци изабраног алгоритма. За претрагу је био задужен перцептрон – неуронска мрежа која је тренирана алгоритмом претраге уназад. Алгоритам је испочетка показивао одличне резултате. Са интензивним повећавањем броја итерација претраге јавила су се прва успорења софтвера да би након приближно пет хиљада итерација алгоритам *пукао* што се решавало рестартовањем сервера уз могућност губитка података који су били у оптицају у тренутку отказивања софтвера. Главни разлог оваквог понашања система представља недостатак синхронизације између суседних итерација претраге, а тај проблем није било, уопште, лако отклонити, поготово што је истовремено била укључена синхронизација са још једним интелигентним системом, генетским алгоритмом, који је мерио степен мутације у новооткривеном знању као превентиву редуданси у бази знања.

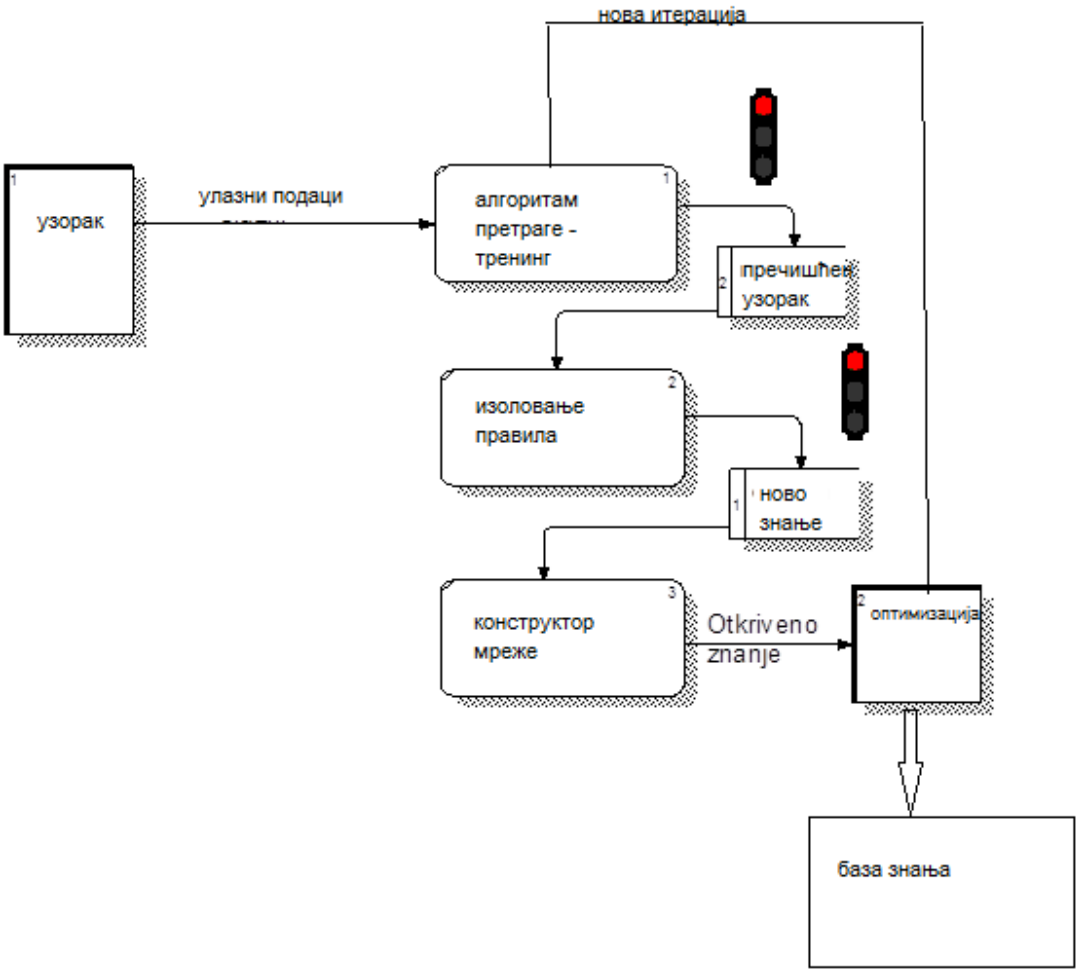
Већ тада је било јасно да алгоритам нема жељену структуру, а самим тим ни неуронска мрежа коју тај алгоритам тренира нема довољно добру архитектуру. У светлу тврђења из претходних поглавља овог рада, било је неопходно дефинисати нове софтверске трансформације за превођење идентификованог знања у објекте правила (Поглавље 6). Такође, скуп резервисаних речи језика високог нивоа добија сасвим нови облик јер је циљ био омогућавање активног учествовања експерата, различитих домена, у изградњи и одржавању софтверског решења, без нужног познавања програмирања и енглеског језика.

Из наведених разлога, познати алгоритам није преправљан већ је приступљено његовом пројектовању и програмирању од самог почетка. Уочено је да су критична места у алгоритму складишта података. Захваљујући JAVA механизму за прикупљање искоришћених процеса (скупљач смећа - енг. garbage gollector[109]), процес пражњења складишта података није пратио брзину поновног пуњења. Након великог броја итерација, долази до загушења у складиштима података и, на крају, до отказивања информационог система. Проблем је решен тако што је испред сваког складишта података постављен конкурентни софтверски концепт *семафор* чије време отварања одговара просечном времену трајања процеса (Поглавље 4). Будући да је AspectJ+ објекти *мањи* од JasCo објеката, брже се кодирају, а самим тим и процеси софтверских трансформација резултата претраге у програмски код, брже ослобађају складишта података ако се трансформисање врши у AspectJ+ објекте и аспекте веза (Поглавље 6). Стога, гледано са економског аспекта, изградња софтвера чија је интелигентна компонента изграђена AspectJ+ нотацијом, даје решења која брже и стабилније сервисирају пословне процесе, а то за последицу има уштеду у новцу, времену и ресурсима.

Избор идеалног алгоритма за тренинг неуронске мреже за претрагу знања

У светлу наведених тврдњи постављен је задатак кодирања алгоритма претраге, за тренинг неуронске мреже, који ће у основи бити алгоритам претраге уназад са имплементираним семафорима у критичним тачкама у којима, након обављене итерације претраге, алгоритам генерише нову структуру неуронске мреже. Такође, сасвим је било јасно да ће морати да се изврше, у наведеном светлу, нова тестирања ради одређивања идеалне архитектуре неуронске мреже за оптималну претрагу и рад посматраног информационог система. Архитектура неуронске мреже мора бити изграђена по следећим принципима (слика 7.6):

[109] www.oracle.com http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html (приступљено 29.11. 2014. у 18:01)

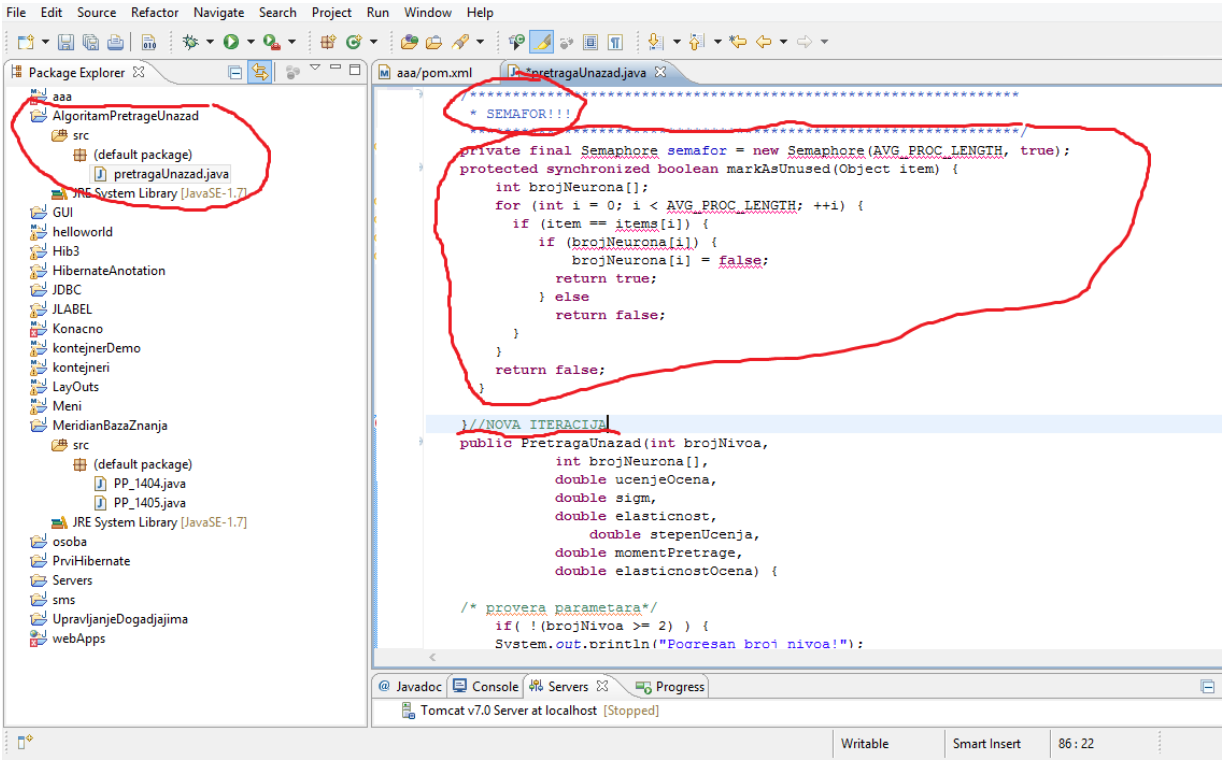


Слика 7.6. Архитектура неуронске мреже

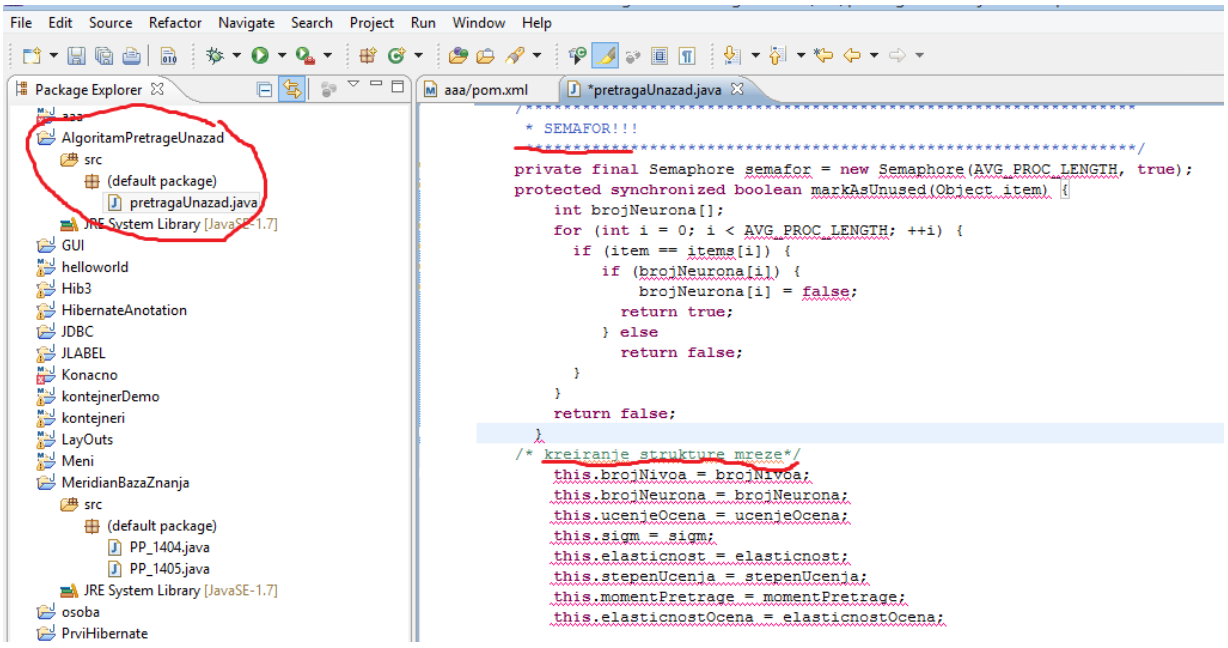
На основу дефинисане класе за имплементацију семафора, која је већ део стандардне JAVA 1.7. документације, а чија дефиниција гласи:

```
public class Semaphore extends Object implements Serializable {…},
```

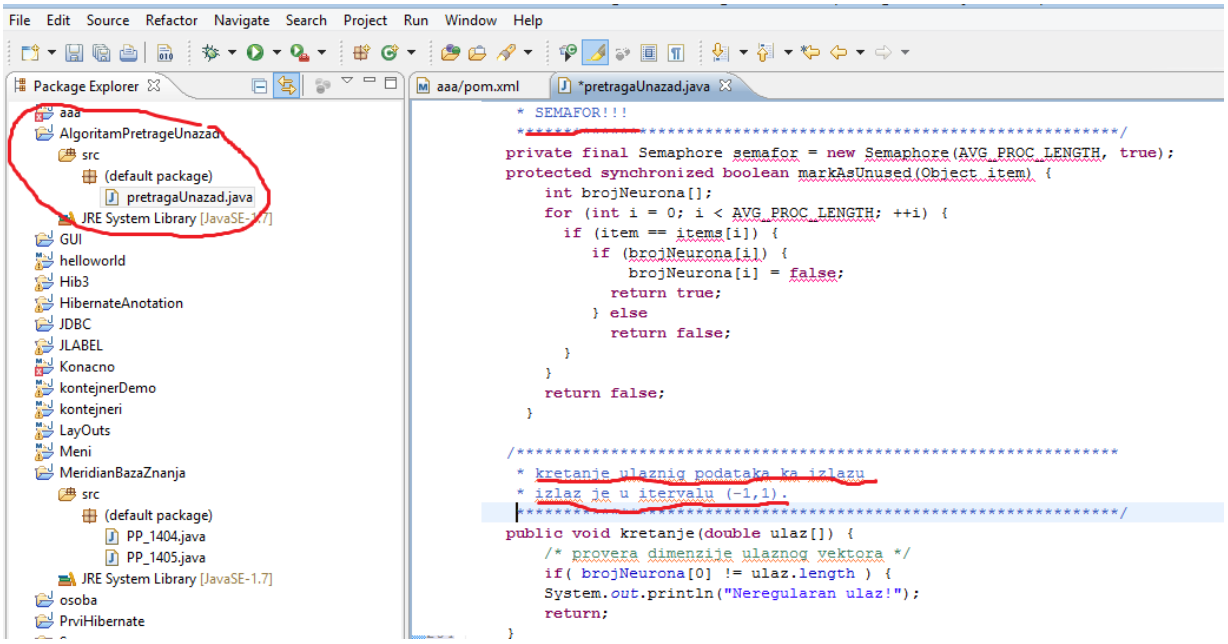
извршена је имплементација семафора у добро познати алгоритам претраге уназад. Сам код алгоритма је изузетно обиман и могуће га је наћи и преузети из већ помињаног JAVA репозиторијума. У овом делу рада биће приказане тачке у програмском коду у којима је дошло до имплементације семафора у алгоритам претраге уназад (слика 7.7, слика 7.8. слика 7.9).



Слика 7.7. Инсталирање семафора након итерације претраге



Слика 7.8. Инсталирање семафора пре генерисања нове структуре мреже

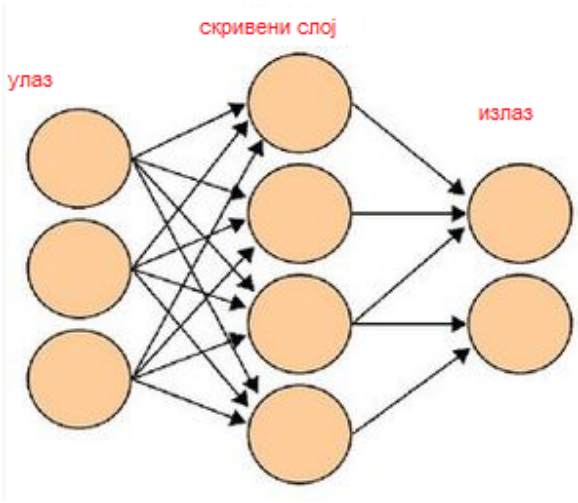


Слика 7.9. Инсталирање семафора пре уласка података у излазно складиште података

Тренирање мреже алгоритмом претраге уназад

Овако конципиран алгоритам претраге уназад отклонио је све претходно уочене недостатке и показао изузетно висок степен стабилности за много већи број итерација од оних на којима је долазило до његовог отказивања.

Остао је још један проблем, за овакав тренинг алгоритам неопходно је одредити оптималну архитектуру неуронске мреже. То значи да је, за трослојну неуронску мрежу, неопходно одредити оптималан број неурона у скривеном, унутрашњем, нивоу (слика 7.10).



Слика 7.10. Трослојна неуронска мрежа

Најефикаснији начин, који је могуће пронаћи у литератури[99] [100] [110], је тестирање перформанси неуронске мреже, трениране алгоритмом претраге уназад, за различите бројеве неурона у средњем слоју мреже, по средњој квадратној грешци. Од посебног значаја је дозволити да тест изврши довољан број итерација, а то значи неупоредиво више од броја итерација које су доводиле до отказивања претходних информационих система. Пре пуштања тест примера у рад, неопходно је било у потпуности изменити и скуп резервисаних речи на основу којих се врши претрага текстуалних узорака – базе постављених питања од стране корисника и службеника. Скуп резервисаних речи је креиран тако, а то је наглашено и у претходном излагању, да би се механизам претраге са успехом изборио са питањима постављеним на српском језику и да би експерти који не говоре енглески језик могли да активно учествују у унапређену софтверског решења. Иновиран скуп резервисаних речи дат је табелом 7.1.

[110] Скочец Ф.2006. , *Аутоматско распознавање природних језика*, Факултет електротехнике и рачунарства веучилишта у Загребу.

Табела 7.1. Скуп резервисаних речи механизма претраге ИС *Meridian*

Скуп резервисаних речи	JAVA/AspectJ+ инструкција
PP	class
AKO	try, if, catch
AKCIJA	return
I	&&
ILI	||
NE	!
OSOBINE	int, float, double, String, boolean
KORISTI	Polisa.baza polisa
GDE	korisnik = polisa.broj
POVEŽI	pointcut – isApplicable
PRE	before
POSLE	after
UMESTO	around

Претрага текстуалних узорака неуронском мрежом може се наћи у бројној литератури[99] [111] [112] [113] и данас има огромну примену на пољу друштвених мрежа[114]. Поред избора погодног алгоритма тренинга, подједнако је битно добро дефинисати скуп резервисаних речи за дати проблем и претрагу базирати на примени Зифовог закона[99] [100] [115] који додатно даје тежину скупу резервисаних речи, олакшава претрагу, кодирање и имплементацију новог знања. Због обимности документације и програмског кода, као и опште прихваћене и познате примене, у овом делу рада неће бити посебно приказана имплементација претраге текста алгоритмом претраге уназад.

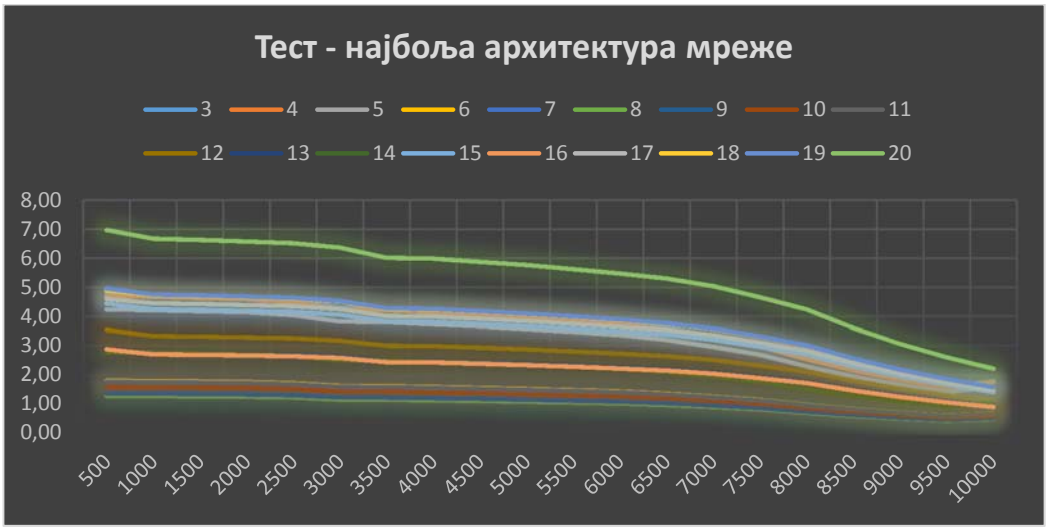
[111] nsdu.edu,http://www.ndsu.edu/pubweb/~irfan/Survey%20on%20text%20mining%20networks.pdf (приступљено 29. 11. 2014. у 20:30)
[112] Govindarajan M, Chandrasekaran R. M. 2007. *Classifier Based Text Mining for Neural Network*, EBSCO
[113] Nordbotten S. 2006. *Data Mining with Neural Networks*, Svein Nordbotten & Associates
[114] mashable.com.http://mashable.com/2012/10/09/google-artificial-intelligence/ (приступљено 29. 11. 2014. у 20:40)
[115] wolfram.com. http://mathworld.wolfram.com/ZipfsLaw.html (приступљено 29. 11. 2014. у 20:50)

Будући да су предочени сви елементи на којима се базира претрага базе питања неуронском мрежом, неопходно је извести тест којим ће софтверско решење добити интелигентну технику претраге чија је дефиниција коначна.

Тест – Избор оптималне архитектуре неуронске мреже

Тестом se испитују перформансе механизма претраге, у зависности од броја чворова у скривеном слоју, по средњој квадратној грешци. Механизам претраге је оптерећен унапред припремљеним узорком, послатим са двадесет клијент десктоп рачунара и двадесет мобилних уређаја синхронизовано. Број резервисаних речи у узорку обезбеђује преко десет хиљада итерација механизма претраге, а то је далеко већи број од броја итерација на којима је долазило до отказивања система пре модификације.

Очекивања пре теста су била да ће доћи до значајних побољшања у архитектури механизма претраге у смислу да ће због смањења броја *мртвих* итерација, а које су се јављале као последица заосталих процеса у механизму претраге, доћи и до смањења броја чворова, у средњем слоју, неопходних да се процес претраге успешно обави. Тест је управо то и потврдио – перформансе механизма претраге побољшане су за скоро 20% (Tabela 7.3). Идеална архитектура подразумева смањење броја неурона, са десет [99] [100] на осам (Табела 7.2. и Графикон 7.1).



Графикон 7.1. Графички приказ резултата теста

Табела 7.2. Резултати теста

број неурона	500	1000	1500	2000	2500	3000	3500	4000	4500	5000	5500	6000	6500	7000	7500	8000	8500	9000	9500
3	4,77	4,74	4,70	4,66	4,55	4,30	4,28	4,20	4,12	4,02	3,91	3,79	3,60	3,33	3,03	2,58	2,19	1,86	1,59
4	4,70	4,67	4,63	4,59	4,48	4,24	4,22	4,14	4,05	3,96	3,85	3,73	3,55	3,28	2,99	2,54	2,16	1,84	1,56
5	4,24	4,22	4,18	4,14	4,05	3,82	3,81	3,74	3,65	3,58	3,48	3,37	3,20	2,96	2,69	2,29	1,95	1,66	1,41
6	1,80	1,79	1,78	1,76	1,72	1,63	1,62	1,59	1,56	1,52	1,48	1,43	1,36	1,26	1,15	0,97	0,83	0,70	0,60
7	1,76	1,75	1,74	1,72	1,68	1,59	1,58	1,55	1,52	1,48	1,44	1,40	1,33	1,23	1,12	0,95	0,81	0,69	0,59
8	1,28	1,28	1,27	1,25	1,22	1,16	1,15	1,13	1,11	1,08	1,05	1,02	0,97	0,90	0,82	0,69	0,59	0,50	0,43
9	1,37	1,36	1,35	1,33	1,30	1,23	1,23	1,20	1,18	1,15	1,12	1,09	1,03	0,95	0,87	0,74	0,63	0,53	0,45
10	1,59	1,58	1,57	1,55	1,52	1,43	1,43	1,40	1,37	1,34	1,30	1,26	1,20	1,11	1,01	0,86	0,73	0,62	0,53
11	1,76	1,75	1,73	1,72	1,68	1,58	1,58	1,55	1,52	1,48	1,44	1,40	1,33	1,23	1,12	0,95	0,81	0,69	0,58
12	3,54	3,33	3,31	3,28	3,26	3,18	3,00	2,99	2,93	2,88	2,81	2,73	2,65	2,51	2,33	2,12	1,80	1,53	1,30
13	2,84	2,78	2,76	2,74	2,72	2,65	2,51	2,50	2,45	2,40	2,34	2,28	2,21	2,10	1,94	1,77	1,50	1,28	1,09
14	2,78	2,76	2,74	2,72	2,65	2,51	2,50	2,45	2,40	2,34	2,28	2,21	2,10	1,94	1,77	1,50	1,28	1,09	0,92
15	4,44	4,25	4,23	4,19	4,16	4,06	3,83	3,82	3,74	3,67	3,58	3,49	3,38	3,21	2,97	2,70	2,30	1,95	1,66
16	2,88	2,71	2,69	2,67	2,65	2,59	2,44	2,43	2,39	2,34	2,28	2,22	2,15	2,05	1,89	1,72	1,46	1,25	1,06
17	4,60	4,45	4,42	4,39	4,35	4,25	4,01	3,99	3,92	3,85	3,75	3,65	3,54	3,36	3,11	2,83	2,41	2,05	1,74
18	4,89	4,73	4,70	4,66	4,62	4,51	4,26	4,24	4,16	4,08	3,98	3,87	3,76	3,57	3,30	3,00	2,55	2,17	1,85
19	4,96	4,75	4,72	4,68	4,64	4,53	4,28	4,26	4,18	4,10	4,00	3,89	3,77	3,58	3,31	3,02	2,56	2,18	1,86
20	6,97	6,67	6,63	6,57	6,52	6,36	6,01	5,99	5,87	5,76	5,62	5,47	5,30	5,03	4,66	4,24	3,60	3,07	2,61

Тестом је показано да повећањем броја чворова у скривеном слоју до броја осам расту перформансе, након тога долази до значајног пада. Након избора оптималне архитектуре требало је практично и доказати да изабрано решење, поред тога што је поједностављено, заиста доминира по перформансама у односу на решење без семафора и са 10 неурона у скривеном слоју.

Постоје бројни, реалтивно једноставни, тестови[116] којима се мери брзина завршавања претраге неуронске мреже. Упоређивањем решења без семафора, са десет неурона, и решења са семафорима, са 8 неурона у скривеном слоју, по обављених 4012 итерација претраге, добијени су следећи резултати:

Табела 7.3. Брзина обављања претраге изабраних мрежа

	Архитектура мреже	
	10 без семафора	**8 са семафорима**
Укупно време (s)	**98**	**83**
Време по итерацији (s)	**0,024**	**0,020**

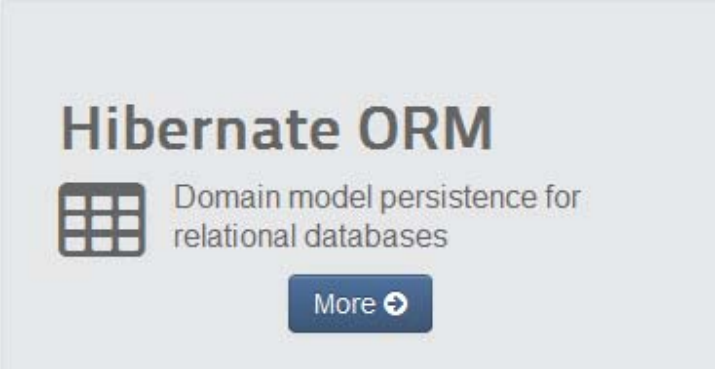
На основу добијених резултата, доказано је значајно побољшање у перформансама механизма претраге предложеног овим радом.

7.1.3. Језгро софтверског решења

Главни део софтверског решења представља класу софтверских решења познату као JAVA сервлет. У суштини то је једно савремено клијент – сервер веб базирано решење изграђено на логици отвореног кода. Изградња софтверског решења *Meridian* у потпуности је реализована у складу са следећим смерницама:

[116]stackoverflow.com. http://stackoverflow.com/questions/10308784/how-to-speed-up-neural-network-performance-in-matlab (приступљено 30. 11. 2014. у 14:38)

- Приликом пројектовања коришћен је објектно-оријентисани алат за пројектовање – UML;[117]
- Програмирање софтверског решења реализовано је програмским језиком JAVA, са предложеним аспектно-оријентисаним проширењем AspectJ+;
- Прелазак са релационог на објектно-оријентисано управљање базом подака реализовано је применом JAVA базираног алата Hibernate ORM (слика 7.11)[118] на MySQL базу података;
- Преузимање богате JAVA архиве и прослеђивање новокреираних JAVA класа назад у репозиторијум реализовано је JAVA базираним алатом Maven (слика 7.12)[119];
- Коришћене су такође интензивно и серверске технологије Apache (слика 7.13)[120] и Oracle JSP (JavaServer Pages) (слика 7.14)[121].
- Главни развојни алат који је коришћен и који је објединио горе поменуте алате и технологије био је Eclipse, тј. његова дистрибуција Indigo (слика 7.15, слика 7.16)[122].



Слика 7.11. *Hibernate ORM* (извор: hibernate.org)

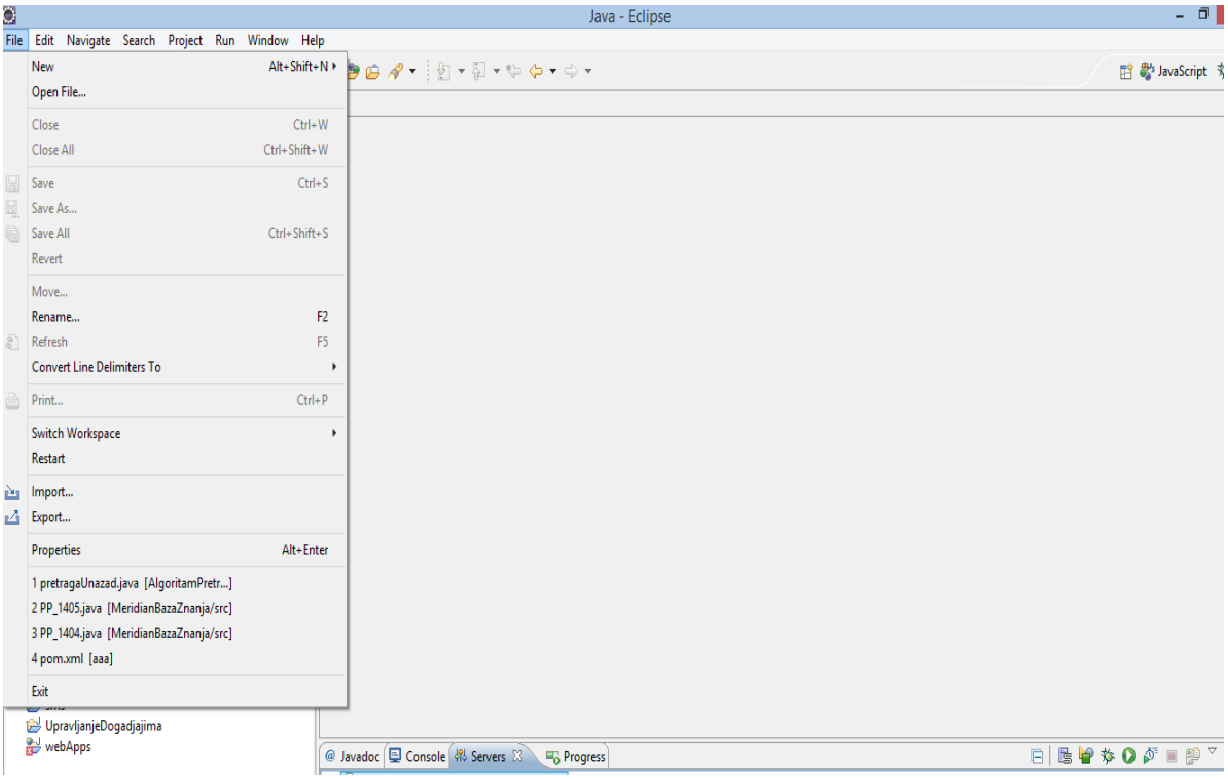
[117] http://www.uml.org/ (приступљено 30. 11. 2014. у 14:48)
[118] http://hibernate.org/ (приступљено 30. 11. 2014. у 15:19)
[119] http://maven.apache.org/ (приступљено 30. 11. 2014. у 15:25)
[120] http://httpd.apache.org/ (приступљено 30. 11. 2014. у 15:35)
[121] www.oracle.com. http://www.oracle.com/technetwork/java/javaee/jsp/index.html (приступљено 30. 11. 2014. у 15:40)
[122] Eclipse.org. https://eclipse.org/indigo/ (приступљено 30. 11. 2014. у 15:45)

Слика 7.12. *Maven* (извор: maven.apache.org)

Слика 7.13. *Apache server* (извор: apache.org)

Слика 7.14. *JSP* (извор: *oracle.com*)

Слика 7.15. *Eclipse Indigo* веб страница (извор: eclipse.org)



Слика 7.16. *Eclipse Indigo* развојно окружење

Клијент решења посматраног информационог система за подршку функционисању делатности осигуравајуће компаније реализована су применом следећих технологија:

- JAVA Swing Applet, за десктоп клијент (слика 7.17)[123];
- *Android SDK*, за мобилни клијент. (слика 7.18)[124];

Слика 7.17 JAVA Swing (извор: java2s.com)

Слика 7.18. Додавање *Android SDK Eclipse Indigo* развојном окружењу

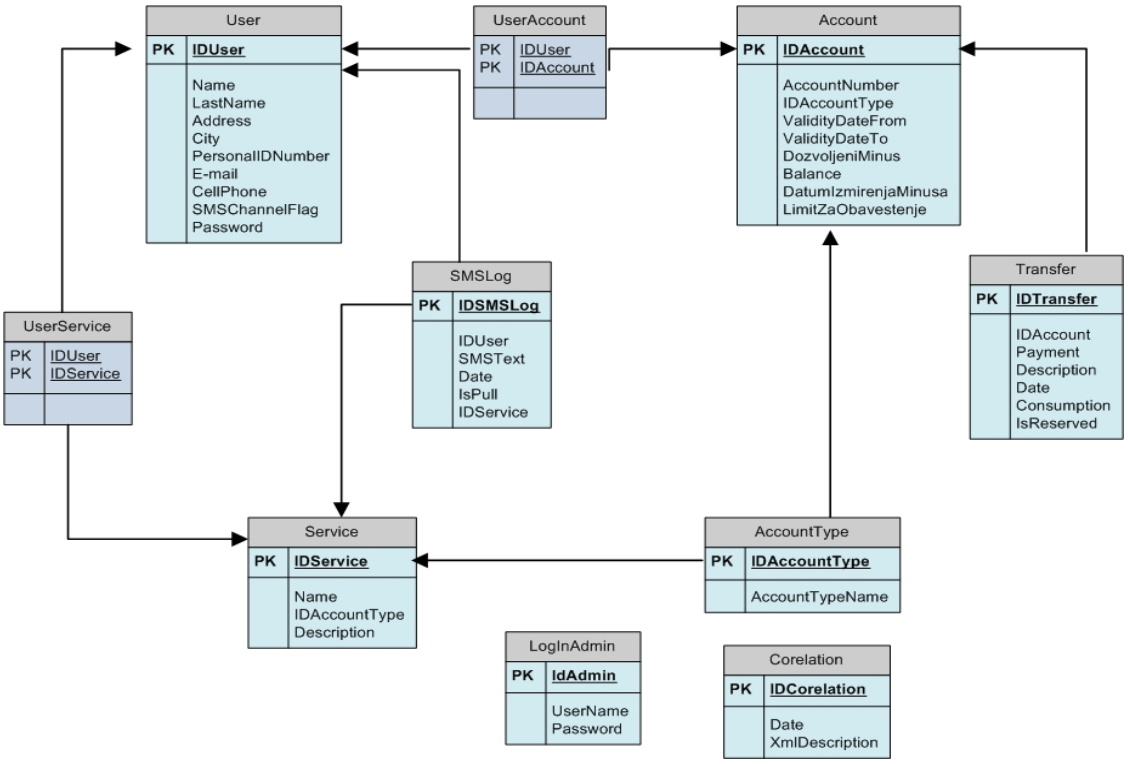
[123] www.java2s.com .http://www.java2s.com/Tutorial/Java/0120__Development/AsimpleSwingbasedapplet.htm (приступљено 30. 11. 2014. у 16:01)
[124] developer.android.com. https://developer.android.com/sdk/index.html?hl=i (приступљено 30. 11. 2014. у 16:05)

Мобилни клијент, такође, представља унапређење софтверског решења које је презентовано у ранијим радовима.[99] [100] Оптимизовано је за мобилне уређаје и, такође, користи технологију семафора због чега не оптерећује рад уређаја који функционишу на *Linux Android* платформи. Ова апликација се извршава на свим новим *Android* платформама и тренутно је предмет доградње за могућност рада на петој генерацији *Android* оперативних система[125]. Архитектура клијент решења, преузета из оригиналне пројектне документације, дата је Сликом 7.18.

Као могуће правце унапређења софтверског решења *Meridian* могуће је истаћи следеће:

- Унапређење клијент софтверског решења тако да може да се користи на мобилним уређајима које покрећу iOS[126] и Windows Mobile[127] оперативни системи;
- Применом преводилачког софтвера[128] дати нову димензију превођењу језика високог нивоа у извршив код објеката правила и аспеката њихових веза. Тренутно, софтверско решење има могућност превођења правила са језика виског нивоа који су подскупови енглеског (за потребе послодавца) и српског језика (за потребе истраживања).

[125] www.android.com. https://www.android.com/versions/lollipop-5-0/ (приступљено 30. 11. 2014. у 16:20)
[126] www.apple.com. https://www.apple.com/ios/ (приступљено 30. 11. 2014. у 16:20)
[127] http://www.windowsphone.com/en-us (приступљено 30. 11. 2014. у 16:25)
[128] http://emisia.net/ (приступљено 30. 11. 2014. у 16:35)



Слика 7.18. Мобилни клијент информационог система *Meridian*

7.2. Имплементација HelpDesk компоненте информационог система *Meridian*

Као што је приказано у делу рада означеним под 3.1, HelpDesk компонента, без предложене интелигентне подршке, у великој мери се ослања на људски фактор. То значи да брзина реаговања HelpDesk система на постављено питање, директно зависи од времена уочавања питања од стране експерта и брзине његовог решавања проблема који је у вези са питањем. На овај начин долази до гомилања задатака које службеници разних нивоа нису способни да реше без асистенције одређеног експерта. Веће паузе у послу значе мање обављеног посла за одређено време, а то за компанију представља смањење прихода и повећање незадовољства запослених. Такође, правовремена информација кориснику производа и услуга осигуравајуће компаније значи подизање нивоа задовољства и поверења у компанију.

Као што је показано, у делу рада означеним под 7.1.1, интелигентни рачунарски систем у великој мери може да преузме улогу обављања експертизе – да препознаје ново

знање (код 7.1. и код 7.2) и да га аутоматски преведе у извршиве објекте пословних правила са одговарајућим апсектима веза (слика 7.4. и слика 7.5). Поред преузимања експертизе од човека, рачунар значајно унапређује кључни концепт савременог пословања – време доношења пословних одлука. Интелигентни HelpDesk систем тренутно нуди решења за велики број постављених питања. У поређењу са примером домаћег HelpDesk система Пореске Управе Републике Србије (видети под 3.1), где су поједина решења добијена након неколико дана, брзина реаговања HelpDesk система *Meridian* је неупоредива. Такође, значајно унапређење у односу да напредне HelpDesk системе јесте могућност самосталног интегрисања новог знања од стране HelpDesk система *Meridian*.

Посебно, у овом делу рада, неопходно је приказати у којој мери, заправо, овај систем доноси предности компанији која га уводи у властито пословање. Најбољи начин за тестирање квалитета функционалности јесте поређење рада информационог система по следећим сценаријима:

- HelpDesk функционише без интелигентне подршке, све одлуке доносе експерти;
- HelpDesk функционише са предложеном интелигентном подршком, аутоматски препознаје ново знање и дозвољава експерту да га једним кликом проследи на аутоматско кодирање, софтверским трансформацијама у објекте правила са одговарајућим аспектима веза (слика 7.19).

Од посебног значаја за овај део рада јесу пословна правила која су већ откривена и приказана кодом 7.1. и кодом 7.2. Показано је да су након трансформисања, успешно сачувана у бази знања као две нове класе, са великом могућношћу поновне вишеструке употребе у форми шаблона за креирање великог броја објеката правила истог типа (слика 7.4. и слика 7.5). Други тест сценарио почиње пре препознавања два позната правила.

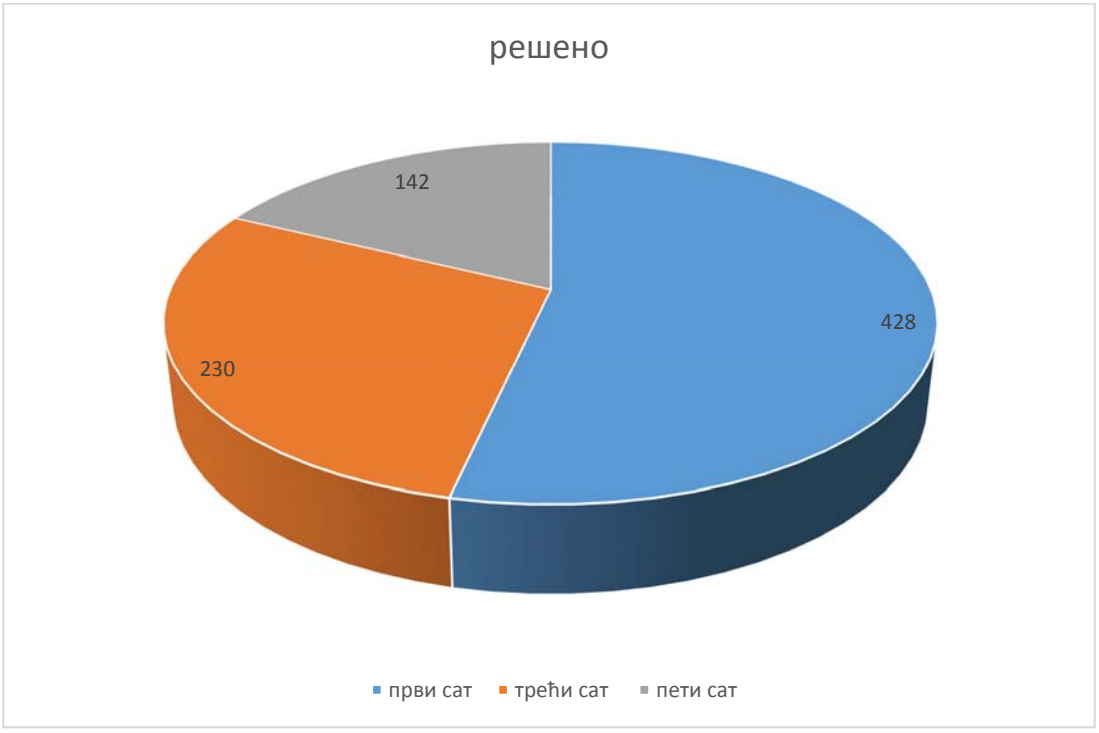
Сценарио 1 – Тестирање времена одговарања на питања путем HelpDesk система без интелигентне подршке:

Помоћу HelpDesk система, службеници задужени за тестирање софтвера су у току једног сата, путем двадесет клијент десктоп рачунара и двадесет клијент мобилних уређаја, проследили 1000 различитих питања, у вези са одштетним захтевима, експертима

осигуравајуће компаније на разматрање. Резултати су приказани Табелом 7.4. и Графиконом 7.2.

Табела 7.4. Забележено време решавања свих питања HelpDesk системом без интелигентне подршке

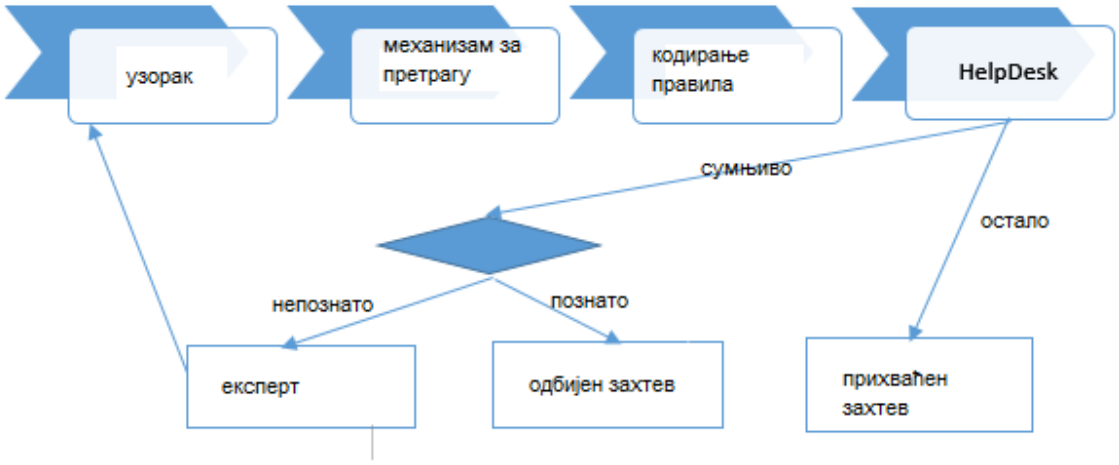
	први сат	трећи сат	пети сат	следећи дан	дужи период
решено	428	230	142	98	102
%	42,8	23	14,2	9,8	10,2



Графикон 7.2. Графички приказ забележеног времена решавања свих питања HelpDesk системом без интелигентне подршке

Тест показује да је услед великог броја постављених питања, тим експерата био *затрпан* захтевима за њихово решавање. У првом сату је решен највећи број питања, углавном тривијалног карактера, скоро 43%. У току радног дана, решено је још приближно 37% постављених питања. Најтежа питања, а чине око 20% од укупног броја, углавном у вези са законском регулативом, решена су наредних дана. Ово указује на претерану зависност HelpDesk система компаније од људског фактора и његове способности за доношење одлука.

Сценарио 2 – Тестирање времена одговарања на питања путем HelpDesk система са интелигентном подршком:



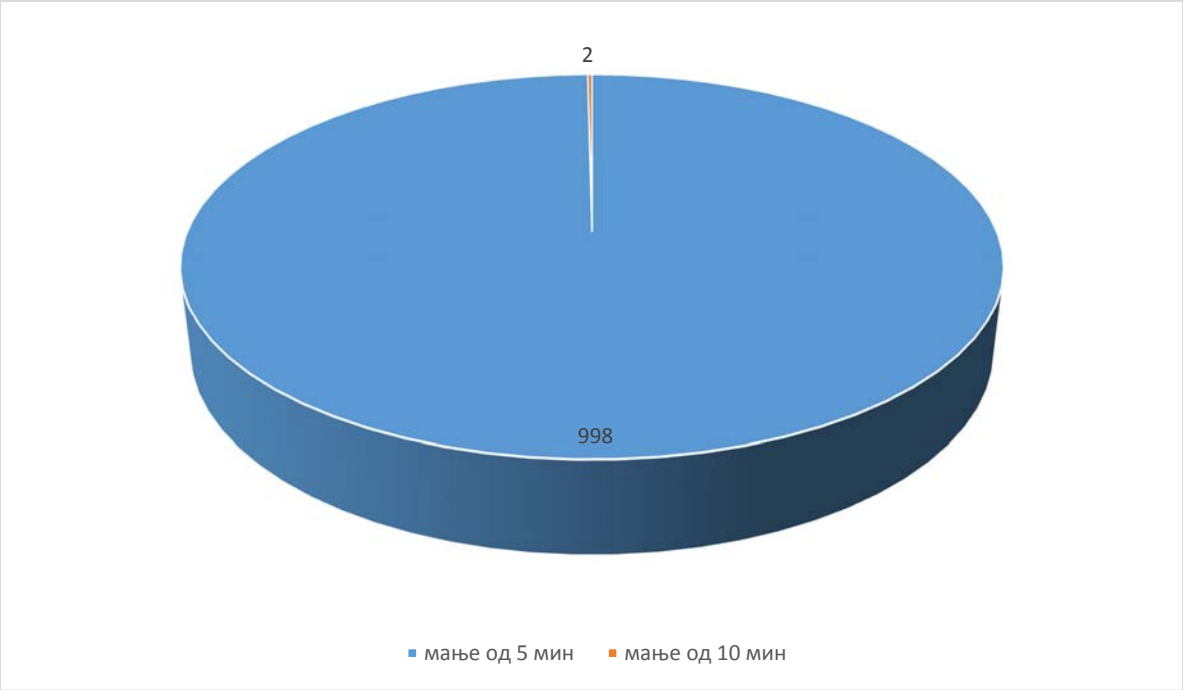
Слика 7.19. Други тест сценарио примене *Meridian* HelpDesk система

Проширеним софтверским решењем је путем HelpDesk система прослеђен идентичан узорак. Шаблони за 998 питања већ постоје у бази знања. Правила приказана кодом 7.1. и кодом 7.2 још увек нису део базе знања и њих, у овом случају, представљају преостала два питања.

Табелом 7.5. и Графиконом 7.3. приказано је време којим су обрађени прослеђени захтеви.

Табела 7.5. Забележено време решавања свих питања HelpDesk системом са интелигентном подршком

	мање од 5 мин	мање од 10 мин
решено	998	2
%	99,8	0,2



Графикон 7.3. Графички приказ забележеног времена решавања свих питања HelpDesk системом са интелигентном подршком

Као последица примене овог теста, правила приказана кодом 7.1. и кодом 7.2 су откривена. Након одобрења експерта, софтверском компонентом – транслатором, на основу дефинисаних софтверских трансформација, ова два правила се преводе у извршив програмски код (слика 7.4. и слика 7.5).

На основу приказаних тестова могуће је закључити да примена интелигентних техника развоја софтвера може резултовати великим предностима за компаније које такве софтвере користе у властитом пословању. Предности су следеће:

- Повећан ниво ефикасности и ефективности пословања;
- Нижи трошкови пословања – компанији треба мање људи да обави већу количину посла;
- Маркетиншка страна увођења оваквог система се огледа кроз побољшан имиџ компаније;
- Повећано задовољство запослених кроз правовремено добијање прецизних информација путем HelpDesk система
- Повећано задовољство корисника услуга осигуравајуће компаније због скраћеног времена реаговања на њихове одштетне захтеве итд.

Уз наведена разматрања могуће је закључити да ће компанијама из области осигурања инвестирање у информационе системе, изграђене на наведеним објектно-оријентисаним техникама са проширењима за подршку развоја интелигентних компонената софтвера, брзо бити оправдано. Као резултат тога доћи ће до повећања обима пословања и већег профита.

Индустрија осигурања је само једна област у којој интелигентни рачунарски системи доживљавају експанзију. Да би овакав систем у потпуности био независан од људске (експертске) интервенције, проћи ће још времена. Главно ограничење овог приступа огледа се у одбојности експерата и менаџера различитих нивоа да одређени део процеса доношења одлука у потпуности препусте информационо-комуникационим технологијама.

На самом крају овог излагања могуће је уочити да је реализацијом интелигентног система претраге знања, трениранога алгоритмом претраге унатраг изграђеним над скупом резервисаних речи и унапређеним конкурентним концептима – семафорима, **доказана и последња *Хипотеза број 5*.**

7.3. Безбедност веб – базираних компонената система *Meridian*

Веома битна ставка у развоју оваквог информационог система јесте заштита података од неовлашћеног и злонамерног коришћења. Будући да се ради о клијент – сервер веб апликацији, информациони систем *Meridian* мора да испуњава све савремене безбедносне захтеве, а посебно по питању заштите база података и знања од:

- SQL напада (енг. SQL injection);
- DoS и DDoS напада.

SQL напад подразумева додавање SQL кода у улазне параметре које управљачки део софтверског решења прослеђује одговарајућем SQL серверу. Приликом оваквог веб напада код се директно додаје у параметре који се повезују са SQL клаузулама које се извршавају. Такође, напад може бити реализован убацивањем злонамерног кода у податке који треба да буду похрањени у табелама базе података или скупу класа базе знања. Када ови подаци динамички буду ангажовани кроз SQL клаузуле, нападач може да коригује њихов садржај.[129] Измењене SQL клаузуле ће се извршавати са истим правима и приоритетима као и оригиналне клаузуле. Јасно је да уколико систем не буде поседовао одговарајући одбрамбени механизам, оваквим нападом може да претрпи велике штете и да директно угрози пословање компаније која га користи. Сликом 7.20. приказан је стандардни сценарио SQL напада.

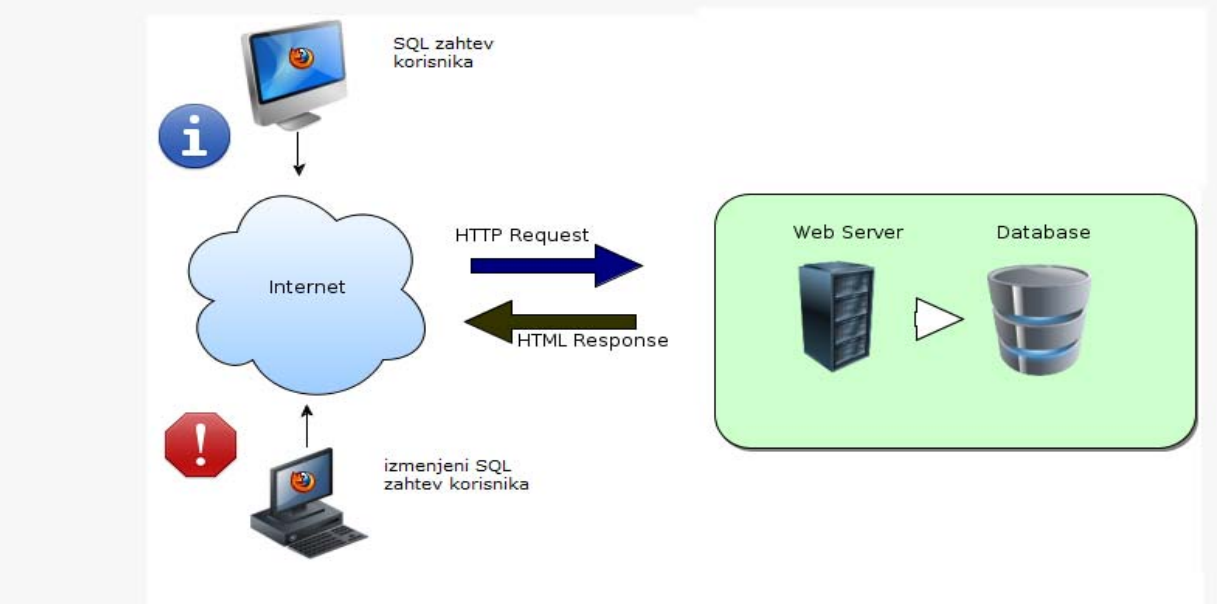
Максимални степен модуларности софтверског решења, омогућен приступом који је изложен у овом раду представља најбољи алат за обезбеђивање заштите од SQL напада. Софтверско решење *Meridian* енкапсулира SQL код унутар заштићених параметаризованих процедура којима је могуће приступити искључиво на строго контролисан начин. Управо тако, ово софтверско решење је имуно на све познате SQL нападе.

Напади који су много опасније природе и теже их је спречити су:

- ускраћивање услуге (енг. DoS – Denial of Service);
- дистрибуирано ускраћивање услуге (енг. DDoS – Distributed Denial of Service).[130]

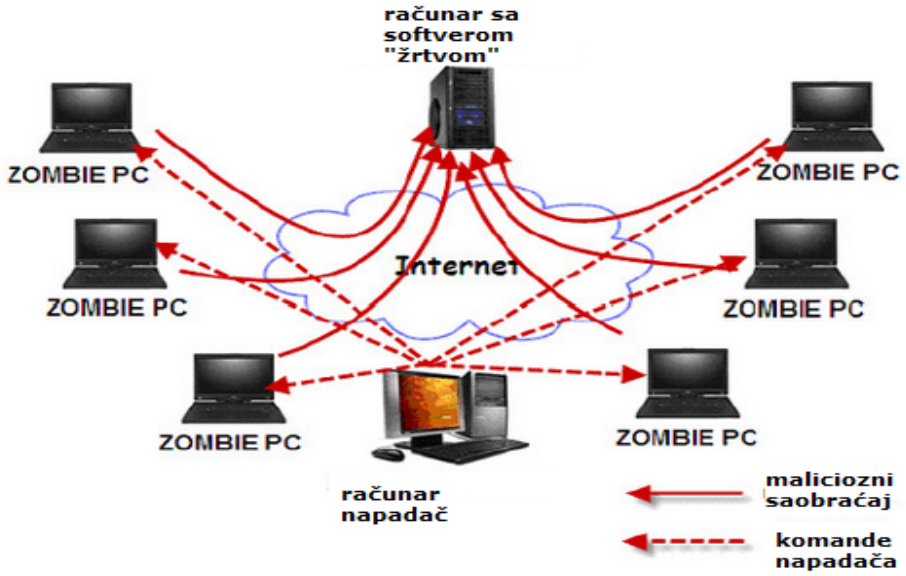
[129] Clarke J. 2012. *SQL Injection Attacks and Defense, Second Edition*, SYNGRESS
[130] Chau H. 2013. *Network Security – Defense Against DoS/DDoS Attacks*, infosecwriters.com

Наведени напади представљају преоптерећење веб софтверског решења захтевима са циљем његовог рушења (Слика 7.21). Ради се о нелегалним, али веома ефикасним, нападима на базе података и знања или било који други тип јавног сервера путем веба. Посебно, ризици од оваквих напада су велики када је дозвољено креирање динамичких упита који омогућавају претрагу по више критеријума. Ограничавањем истовремених конекција на SQL сервер и енкапсулацијом SQL кода систем *Meridian* штити своје податке од наведених напада.



Слика 7.20. SQL напад (извор: hackertarget.com [131])

[131]hackertarget.com. http://hackertarget.com/sql-injection/ (приступљено 20.02.2015. у 11:40)



Слика 7.21. DoS/DDoS напад (извор: slashgear.com [132])

[132] www.slashgear.com. http://www.slashgear.com/whats-a-ddos-attack-zombies-shopping-help-explain-it-all-11333110/ (приступљено 20.02.2015. у 12:10)

8. Закључак

Савремени концепт електронског пословања подразумева интензивну употребу најсавременијих информационо – комуникационих технологија. Као подршка функционисању наведених технолошких решења намећу се све комплекснији и захтевнији софтвери. Да би у потпуности били у стању да сервисирају пословање компанија које послују у веома динамичном пословном окружењу, софтверска решења морају да имају могућност тренутног прилагођавања променама у захтевима, у реалном времену. Из наведених разлога било је неопходно трагати за новим развојним алатима и техникама чијом применом софтверска решења добијају један нови квалитет – функционисање, одржавање и мењање у реалном времену, без потребе за накнадним интервенисањем софтверског развојног тима. Такође, омогућено је и активно учешће, у наведеним задацима, експертима који много боље познају природу пословања него пројектанти софтвера и програмери, а то за последицу има софтверска решења којима је обезбеђено ефикасније и ефективније сервисирање свих делатности компанија у којима су инсталирана.

Будући да је квалитетна информација најзначајнији ресурс за доношење пословних одлука, почетна идеја за истраживање је била могућност унапређења савремених софтверских подсистема, за информисање запослених и корисника производа и услуга, познатих као HelpDesk системи. Отуда је консултована бројна страна и домаћа, научна и стручна, електронска и писана литература из које је сагледано актуелно стање у наведеној области. Посебан акценат је стављен на изучавање развојних софтверских алата који су идентификовани као могућа помоћ у реализацији постављених идеја и циљева. Посебно, упошљавањем интелигентних рачунарских алата и техника, у претходним поглављима је приказана могућност унапређења HelpDesk система конкретним приступом.

Од посебног значаја је било истицање области примене унапређеног HelpDesk система. Компаније из области осигурања успех у свом пословању директно захваљују квалитетном протоку информација између менаџмента и службеника, са једне, и службеника и корисника са друге стране. Отуда је имплементација посматраног софтверског решења пронашла идеалну област – компанију из области осигурања. Захваљујући компанији *Meridian Project* и развојном тиму *ProSoft* омогућен је развој,

тестирање и имплементација, а самим тим и реализовање циљева дисертације приказаних у претходним поглављима. У наведеном светлу, текло је и излагање у овом раду.

Уводни део рада био је резервисан за дефинисање предмета и циљева истраживања. Од посебног значаја било је постављање проблематике HelpDesk система у оквире интелигентних информационих система са највећим могућим степеном модуларности њихових функционалних делова. Овде је јасно подвучено да традиционални *if – then – else* сценарио кодирања знања није довољан за реализовање идеја дисертације. Отуда је било неопходно посегнути за унапређењем стандардног објектно – оријентисаног приступа увођењем нових, аспектно – оријентисаних, проширења којима знање, а и одговарајуће везе са језгром софтверског решења, добија форму скупа независних објектно – оријентисаних модула. На тај начин, логика правила обухваћена је објектима правила док су везе исказане специфичним објектима – аспектима веза. Посебно, у овом делу рада, било је неопходно елаборирати постојеће аспектно – оријентисане приступе са циљем избора најквалитетнијег решења за конструисање базе знања интелигентног HelpDesk система. Овде је, такође, први пут представљен концепт конкурентног развоја софтвера *семафор* којим су отклоњени недостаци актуелних софтверских решења креираних постојећим објектно – оријентисаним алатима са аспектним проширењима и интелигентним механизмима претраге. Након јасно изложених теоријских ставова и студија, у овом делу рада су, потом, прецизно постављени предмет и циљ истраживања, а затим је постављено пет хипотеза које су доказане у поглављима која следе након уводног.

Прво наредно поглавље, бавило се применом HelpDesk система на свим нивоима електронског пословања, а са циљем оправдавања постављених циља и предмета истраживања. У овом делу рада дат је опис HelpDesk система кроз различите еволутивне периоде, од кол (енг. call) центра до аутоматизованог HelpDesk система, од канцеларијског до корпоративног вида пословања. Посебно је апострофирано да у домаћим оквирима HelpDesk обуке готово да и не постоје и локализоване су на кратке тренинге у оквиру компанија које користе овакве системе. У развијенијим економијама, HelpDesk систем је препознат као кључни фактор успешног пословања и, сходно томе, доста се улаже у квалитетне обуке HelpDesk менаџера и осталих службеника по моделу Фила Вергиса (видети поглавље број 2). Такође, овде је јасно елаборирана Интернет усмереност

савремених HelpDesk система што има за последицу раширеност њихове примене на све нивое електронског пословања. Тако је, у наставку, апострофиран значај правовремене и прецизне информације за потенцијалног купца извесног производа или услуге. За домен електронске трговине је кључан концепт лојалног потрошача. Само задовољан потрошач ће поново куповати на истом месту. Управо из наведених разлога је дошло до развоја великог броја различитих HelpDesk система за подршку електронској трговини. Посебно, у овом делу рада, истакнуто је једно од најквалитетнијих HelpDesk софтверских решења на тржишту, а за подршку системима електронске трговине, Oracle – *Siebel e-Commerce* (видети под 2.1.2). Софтверско решење је детаљно описано уз истицање бројних бенефита којих нуди крајњим корисницима. Такође, овде је уочен и један недостатак од великог значаја за развој софтверског решења по моделу представљеном у овом раду. Посматрано комерцијално софтверско решење још увек није способно да самостално учи и проширује властиту базу знања.

У наставку излагања идентификовани су и преостали сегменти електронског пословања у којим HelpDesk системи имају велику примену. Тако, електронска управа представља део електронског пословања чији сервиси директно зависе од квалитета и правовремености сервираних информација. Отуда, подизање квалитета комуникације између појединачних делова јавне управе, са циљем унапређења квалитета електронских услуга, снижавања трошкова пословања, као и обављања више посла за исто време, представља кључ за успешну савремену електронску управу. У ту сврху је детаљно елаборирано једно од најквалитетнијих софтверских решења из ове области *teleNetwork* (видети под 2.2.2).

Након наведених излагања уследио је пресек стања примене HelpDesk система у области која је од посебног значаја за овај рада, а то је индустрија осигурања (видети под 2.3). Ова област пословања, као изузетно профитабилна област производа и услуга, одавно привлачи велике произвођаче софтвера који су већ понудили властита HelpDesk решења. Тако су, у наставку излагања, приказана најквалитетнија HelpDesk софтверска решења за подршку пословању осигуравајућих компанија: *Dell HelpDesk*, *LBi HR HelpDesk* и *IBM Claims Fraud Solutions*. Последње наведено решење, а то је од посебног значаја за овај рад, на основу богате базе знања има могућност препознавања лажних изјава у одштетним

захтевима осигураника. Софтверско решење, које представља циљ овог рада, дало је унапређење наведене функционалности кроз могућност динамичког проширења базе знања правилима која се односе и на детектовање лажних изјава у захтевима за надокнаду штете.

У даљем излагању, HelpDesk системи су анализирани кроз призму развијености тржишта на којем послује компанија која користи такав систем. У домаћим оквирима, најрепрезентативније HelpDesk решење представља HelpDesk систем Пореске управе републике Србије. У овом делу рада (видети под 3.1) детаљно је описан оквир у којем је интегрисан наведени HelpDesk систем, као и његова функционалност кроз механизам *питање – одговор*. Као главни недостатак овог система наведена је његова база знања која не представља софтверску компоненту експертног система већ скуп текстуалних података са одговорима на постављена питања. Овакав систем захтева интензивно администрирање од стране човека и не представља конкурентно решење на тржишту савремених HelpDesk система. Као доказ претходном тврђењу, у наставку рада, презентовани су, и анализирани, HelpDesk системи *KronoDesk* и *SysAid* (видети под 3.2) који имају могућност аутоматизованог одговарања на постављена питања, али на основу пословних правила структурно организованих по шаблону *if – then – else*. Због овакве организације базе знања онемогућено је аутоматизовано учење посматраних HelpDesk система. Отуда, било је могуће извести следећи закључак: *на тржишту HelpDesk софтверских решења недостаје решење засновано на интелигентним техникама које самостално може да проширује експертизу.*

Управо, из наведеног разлога, у даљем излагању акценат је био на идентификовању могућих праваца унапређења савремених HelpDesk система, а у горе наведеном светлу. Као технологија за постизање циљева дисертације наметнуо се приступ развоју софтверских решења познат као *аспектно – оријентисан развој софтвера* (енг. AOSD – Aspect Oriented Softwere Development). Оправдање за увођење овог приступа пронађено је у наслеђеним недостацима, а који се односе на степен модуларности софтвера, са којима стандардни објектно – оријентисани приступ није био у стању да се у потпуности избори (видети под 3.3.1). Управо зато, у овом делу рада, показано је да искључиво софтверско решење са максималним степеном модуларности и са минималним везама између властитих модула може да оствари динамичко проширивање базе знања без поновног превођења софтверског

решења. Дакле, у раду је презентована организација базе знања у форми скупа класа (шаблона правила) од којих је могуће креирати велики број различитих објеката правила. Посебан проблем представља реализовање објеката којима се знање повезује са централним класама софтверског решења. Управо због тога је, са посебном пажњом, било неопходно изабрати најпогоднији аспектно – оријентисани приступ који би омогућио динамичко повезивање новог објекта правила са апликацијом и чување његовог шаблона у бази знања. У овом делу рада, посебно је показано да два постојећа аспектно – оријентисана приступа, AspectJ и JasCo, имају недостатке које је могуће отклонити комбиновањем најбољих индивидуалних карактеристика. На овај начин добијен је оригинални, унифицирани, приступ AspectJ+ којим је заокружена дефиниција пирамиде објектно оријентисаног дизајна. Од посебног значаја је еволуција софтверског дизајна која, ослањајући се на AspectJ+ у комбинацији са анализом модела домена, помера објектно – оријентисану парадигму развоја софтвера са MDE (енг. Model Driven Engineering) приступа (слика 8.1) ка MDSD (енг. Model Driven Software Development) приступу (слика 8.2).

Међутим, да би дефиниција унифицираног приступа била у потпуности заокружена, било је неопходно истражити релевантне предности и недостатке AspectJ и JasCo приступа, везаних за постављене циљеве дисертације. Критеријуми по којима су тестирани приступи су били:

- перформансе;
- могућност динамичког повезивања објеката правила са управљачким делом софтверског решења.

Перформансе су у директној вези са брзином извршавања и стабилношћу софтверског решења и односе се на брзину одзива неког правила из базе знања, као и на укупно трајање процеса његовог ангажовања. Уочено је одржање пропорционалности између ових величина, за два различита објекта правила, применом унифицираног AspectJ+ приступа (видети под 4.2):

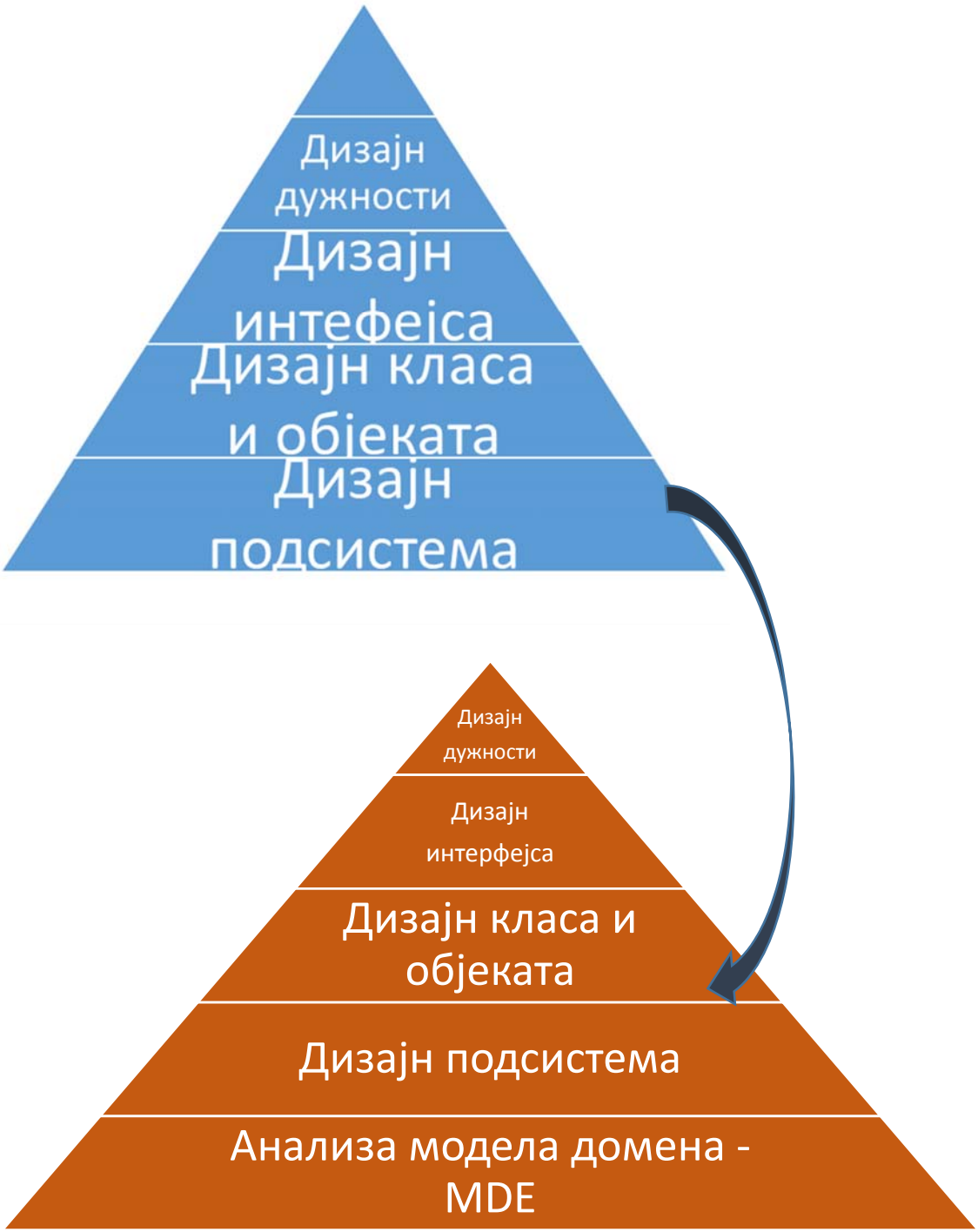
$$ping_i : ping_j = proc_i : proc_j$$

У овом делу рада је показано да AspectJ аспекти веза, у просеку, за 18% брже се активирају него JasCo аспекти, а то за последицу има 24% краће просечно време укупног трајања ангажовања објеката правила повезаних AspectJ аспектима у односу на идентичне објекте правила повезаних JasCo аспектима. Са друге стране, AspectJ аспектима је немогуће реализовати динамичко повезивање новог знања као што је случај са JasCo аспектима. Предефинисањем AspectJ аспеката на начин да им се даје могућност манипулисања шаблонима (класама објеката правила), баш као што је случај са JasCo аспектима, наведени недостаци AspectJ приступа су елиминисани. На овај начин је добијен оригинални AspectJ+ приступ који има занемарљиво лошије перформансе у односу на изворни AspectJ приступ, али са могућношћу динамичког повезивања новооткривеног знања са делом софтверског решења за управљање знањем. Класе новог приступа похрањене су у веб JAVA репозиторијум и могуће их је слободно користити приликом израде различитих софтверских решења.

У наставку, као посебан део рада јавља се анализа модела домена. Ту је, посебно, приказан развој софтверског решења као резултат примене концепта софтверског дизајна изведеног из модела који подразумева двосмеран начин развоја софтвера:

- развој кода на основу модела;
- реинжењеринг модела на основу измена у коду.

Наведено је од посебног значаја за развој квалитетних софтверских решења из разлога што је током процеса анализе домена, у развој софтверског решења, могуће активно укључити експерте који много боље разумеју пословање него што је случај са програмерима и пројектантима софтвера. Дефинисањем језика високог нивоа (језика домена), блиског говорном, а у форми програмског језика, на нивоу домена је омогућено исказивање пословних правила и њихових веза са језгром софтверског решења. Прецизна дефиниција граматике овог језика (видети под 5.1) омогућава повезивање његових инструкција са инструкцијама уграђеним у одговарајуће објекте правила, посебним JAVA класама којима се реализују софтверске трансформације превођења кода модела у извршив код програмског језика. Ово представља кључни део рада јер се у њему поставља основ по којем интелигентни алгоритам претражује текстуални узорак (скуп питања) и по шаблону (скуп класа базе знања) даје одговор на постављено питање путем HelpDesk система.



Слика 8.1. Проширена пирамида објектно – оријентисаног развоја софтвера

Слика 8.2. Еволуција проширене пирамида објектно – оријентисаног развоја софтвера

У овом делу рада, поред објеката правила и веза, уведен је и концепт догађаја – сценарија по којем се пословно правило активира и долази до примене његове логике од стране централних класа софтверског решења. За покретање правила, односно креирање објекта правила из одговарајућег шаблона, задужена је окидачка класа која манипулише следећим вредностима:

- активирање везе;
- време примене правила;
- време придруживања неопходне информације правилу.

Посебно, да би текстуални узорак (скуп питања), преко кода високог нивоа, могао да буде исказан као извршиво знање (класе и објекти правила) било је неопходно дефинисати квалитетан и комплетан скуп софтверских трансформација за превођење језика домена у извршив JAVA и AspectJ+ програмски код. Скуп класа којима је омогућено аутоматско превођење пословних правила и веза, исказаних језиком домена, у одговарајуће JAVA објекте правила и AspectJ+ аспекте веза, гради јединствену софтверску компоненту информационог система *Meridian,* под називом Translator/AspectJ+. Такође, и ове класе су доступне програмерима преко веб JAVA репозиторијума у којем су похрањене након тестирања (видети поглавље број 6). Од посебног значаја, у овом делу рада, је било тестирање перформанси транслатора са AspectJ+ функционалностима у односу на транслатор Translator/JasCo који је креиран на основу трансформација које је предложила ауторка М. А. Сибран у свом раду.[133] Уочено је, и доказано, да AspectJ+ трансформације користе, у просеку, око 10% мање процесорског времена што представља још један помак у развоју наведених класа софтверских решења. Требало би још напоменути да је, у овом делу рада, за сваку од NDL инструкција високог нивоа дефинисана конкретна трансформација при чему су оне добиле властиту објектно – оријентисану или аспектно - оријентисану репрезентацију било да се ради о инструкцијама објеката правила или аспеката веза респективно. Специјално, приликом трансформисања кода високог нивоа, инсистирано је на управљању софтверским изузецима. У дефиницији класа правила, а самим тим и објеката тих класа, логика пословних правила је реализована кроз програмске блокове *try…catch*. На

[133] Cibran M. A. 2007. Connecting High Level Business Rules With Object Oriented Applications. Vrije Universiteit Brussel

основу окидачког догађаја, логика правила се извршава уз перманентно праћење понашања система одговарајућим системским класама *ослушкивачима*. На овај начин, избегнуто је отказивање објекта правила у случају неочекиваних системских догађаја, а то за последицу има унапређење степена поузданости класа и одговарајућих објеката базе знања. У наставку излагања, након презентације трансформација којима се пословна правила преводе у објекте правила, а њихове везе у одговарајуће аспекте, демонстрирано је превођење конкретног правила софтверском компонентом Translator/AspectJ+. Правило је преузето као питање постављено путем HelpDesk система, анализирано је као текстуални узорак од којег је креирано, језиком домена, пословно правило помоћу софтверске компоненте NDL/Generator. Излаз из овог скупа класа преузео је нови скуп класа, Translator/AspectJ+, који је аутоматски креирао извршив програмски код за реализовање одговора на постављено питање.

Као посебан део рада, који претходи закључним разматрањима, истиче се демонстрирање развоја интелигентног HelpDesk система за подршку компанијама из области осигурања, а на основу теоријских разматрања, технологија и алата презентованих у претходним поглављима рада. Такође, у овом делу рада узета су разматрање постојећа позната интелигентна софтверска решења са уоченим недостацима механизма претраге. Са циљем елиминисања наведених недостатака укључени су конкурентни софтверски концепти - *семафори* (видети поглавље број 7). У овом делу рада је, такође, детаљно демонстрирана и архитектура софтверског решења:

- серверска страна са језгром апликације и базама података и знања;
- PC и мобилна клијент страна;
- резервна копија на *cloud* серверу;
- унапређени алгоритам претраге.

За сваку од наведених компонената приказан је алат и технологија којом је изграђена.

Посебан део овог поглавља представља тестирање функционалности система преузимањем конкретних правила путем HelpDesk подсистема. Кроз систем су пропуштена два питања са циљем одобравања одштетних захтева након повреда насталих на радном месту и као последица саобраћајне несреће. Систем је аутоматски генерисао два правила и,

потом, упутио одговоре на адресе са којих су питања постављена, у веома кратком временском року.

Након демонстрације функционалности аутоматизованог HelpDesk система било је неопходно описати кључну компоненту система којом је омогућено претраживање скупа питања и идентификовање постојећег и новог знања. Као идеално решење наметнуо се алгоритам претраге уназад за тренирање неуронске мреже која врши претрагу узорка. На самом почетку, истакнути су недостаци овог алгоритма који су уочени кроз вишегодишњу експлоатацију на конкретним примерима претраге. Главни недостатак овог алгоритма огледа се у високом степену отказивања, након извесног броја итерација, услед загушености софтверским процесима. Интегрисањем семафора, на кључним местима у алгоритму, дошло се до потпуне синхронизације процеса, а то за последицу има извршавање алгоритма претраге уназад, глатко и без прекида. Посебан квалитет унапређеном алгоритму претраге уназад, што је показано конкретним тест примером (видети под 7.1.2), даје чињеница да он након модификације захтева 20% мање неурона, а самим тим и софтверских инструкција, у скривеном слоју мреже за обављање истих задатака. Наведено представља значајан помак у перформансама у односу на конвенционални алгоритам претраге уназад чију је примену могуће наћи у бројној научној и стручној литератури и на разним примерима.

У наставку рада, а са циљем обезбеђивања неопходног максималног степена модуларности информационог система, елаборирано је коришћење алата за објектно – релационо мапирање којима се подаци преузети из базе података директно предају објектима језгра софтверског решења. На овај начин традиционално релационо управљање базом података замењено је презентованим алатом Hibernate ORM за постизање наведених софтверских функционалности.

У наставку, након демонстрације функционалности управљачког дела, на серверској страни информационог система, приказане су и две клијент компоненте – PC и мобилна клијент компонента са одговарајућим технологијама.

Основе преласка са прототипа у имплементирано софтверско решење дате су у наредном делу рада (видети под 7.3). Софтверско решење је тестирано по два сценарија:

- време одговора на постављена питања путем HelpDesk система без интелигентне подршке;
- време одговора на постављена питања путем HelpDesk система са интелигентном подршком.

Путем клијент софтверских компонената прослеђено је 1000 различитих питања HelpDesk компоненти информационог система. У првом случају, поједина питања, а чине око 20% од укупног броја постављених питања, нису решена до краја текућег радног дана и дуже. Систем са интелигентном подршком је решио сва питања процесирањем које је трајало краће од 10 минута. Очигледно да је софтверско решење са интелигентним системом претраге показало доминантне карактеристике које воде ка:

- већем степену ефикасности и ефективности пословања;
- већем степену задовољства корисника система на основу веће брзине добијања информација;
- нижим трошковима пословања;
- унапређењу имица компаније;
- већем степену задовољства корисника производа и услуга услед краћег времена решавања њихових захтева.

Као незаобилазан део проблематике развоја информационих система јавља се дискусија о безбедносним аспектима, како трансакција, тако и софтверских процеса. Посебно, акценат је стављен на познате технике масовног напада на веб системе. Будући да се ради о веб базираном клијент – сервер софтверском решењу, информациони систем *Meridian* морао је да укључи све познате технике одбране од веб напада, а то је приказано у делу рада обележеним са 7.4.

На самом крају излагања могуће је истаћи да је у раду приказана методологија развоја пословних софтверских решења, напредним објектно – оријентисаним алатима са аспектним проширењима и са интелигентном HelpDesk подршком, која је ставила посебан акценат на следећу проблематику:

- **Развој унифицираног приступа за дефинисање објеката правила кроз комбиновање доминантних карактеристика постојећих приступа;**
- **Дефинисање иновативног приступа за развој широког спектра експертних система;**
- **Могућност проширења и унапређења постојећих експертних система предложеним алатима и техникама.**

9. Литература

1. Ballou M. C. 2008. *Improving Software Quality to Drive Business Agility*, IDC – White Paper
2. Curry S, Kirda E, Schwartz E, Stuart W, Yoran A. 2013. *Big Data Fuels Intelligence-Driven Security*, RSA Security Brief, January 2013
3. dobney.com. http://www.dobney.com/market_intelligence.htm (приступљено 12.02. 2015. у 10:20)
4. Sousa K, Effy O. 2014. *Management Information Systems*, Cengage Learning
5. Simur J, Schulte W. R, Hill B. J, Jones T. 2012. *Magic Quadrant for Intelligent Business Process Management Suites*, Gartner 2012.
6. utwente.nl. www.home.math.utwente.nl/~omnerenjcw/153084/sheetsApp.pdf (приступљено 12.02. 2015. у 10:44)
7. comarch.com. www.comarch.com/files_eu/file_1585/ComarchAltum.pdf (приступљено 12.02. 2015. у 10:48)
8. Kiczales G, Lamping J, Mendhekar A, Maeda A, Videira Lopes C, Loingtier J. M, Irwin J. 2007. *Aspect-oriented Programming*, Xerox Palo Alto Research Center - 2007.
9. Hilsdale E, Hugunin J. 2004. *Advice Weaving in AspectJ*, 3rd International Conference on Aspect-Oriented Software Development (AOSD). April 2004.
10. Kiczales G, Hilsdale E, Hugunin J, Mik Kersten, Palm J, Griswold V. B. 2001 . An Overview of AspectJ. In *Proceedings of the 15th European Conference on Object-Oriented Programming (ECOOP)*. Springer, 2001.
11. Hunt J. 2006. Aspect oriented programming with Java, The Register.
12. Kiselev I. 2008.Aspect-oriented programming with AspectJ ,Sams.
13. Suvee D, Jonckers V, Vanderperren W. 2004. Supporting JAsCo AOP, ENTCS vol. 107.
14. Suvee D, Vanderperen W, Jonckers V. 2003. JAsCo: An Aspect-Oriented Aproach forComponent Based Software, AOSD Boston.
15. Cibrán M. A, Suveé D, D'Hondt M, Vanderperren W, Jonckers V. 2004. Integrating Rules with Object-Oriented Software Applications using Aspect-Oriented Programming, Proceedings of ASSE'04, Argentine Conference on Computer Science and Operational Research, Córdoba, Argentina, 2004.

16. Cibrán M. A, D'Hondt M, Vanderperren W. 2003. Aspect-Oriented Programming for Connecting Business Rules. Proceedings of BIS International Conference, Colorado Springs, USA, June 2003.

17. Cibran M. A. 2007. Connecting High Level Business Rules With Object Oriented Applications. Vrije Universiteit Brussel

18. Rojas R. 1996. *The BackPrpagation Algorithm*, Neural Networks, Springer Verlag Berlin - 1996

19. Cilimkovic M. 2008. *Neural Networks and Back Propagation Algorithm*, Institute of Technology Blanchardstown

20. Милићевић В. 2012. *Допринос развоју интелигентног рачунарског система базираног на приступу пословних правила*, ФАМС

21. Zelenski J. 2008. *Thread and Semaphore Examples*, Stanford, CS107

22. ufsc.br. http://www.inf.ufsc.br/~bosco/ensino/ine5645/Semaphore-Monitor.pdf (приступљено 19.02.2015. у 8:30)

23. techtarget.com. http://searchcrm.techtarget.com/definition/help-desk (приступљено 06.10. 2014. у 22:50)

24. http://www.verghisgroup.com/ (приступљено 06.10. 2014. у 23:50)

25. Verghis P. 2013. *The Ultimate Customer Support Executive*, Silicon Press, ISBN 092930634

26. asponline.com. http://www.asponline.com/essentials (приступљено 07.10. 2014. у 00:14)

27. philverghis.com. http://www.philverghis.com/helpdesk.html (приступљено 07.10. 2014. у 00:20)

28. freshdesk.com. http://www.freshdesk.com/industries/help-desk-software-retail-ecommerce (приступљено 07.10. 2014. у 01:10)

29. oracle.com. http://www.oracle.com/us/products/applications/siebel/051165.pdf. (приступљено 21.10. 2014. у 17:10)

30. www.siebel.com. https://s3.amazonaws.com/rmc_docs/siebel_business_analytics_brochure.pdf. (приступљено 21.10. 2014. у 19:30)

31. Agraval A, Mittal M, Rastogi L. 2002. *Enabling e-Governance Integrated Citizen relationship Management Framework – the Indian Perspective*, Deloitte & Touche (2002) Budget 2002 Report, India.

32. www.telenetwork.com.https://www.telenetwork.com/insights/articles/case_studies/Internet_Help_Desk_Performance_Improvement.pdf (приступљено 29.10. 2014. у 11:30)
33. www.telenetwork.com (приступљено 02.11. 2014. у 11:30)
34. www.bnamericas.com.www.bnamericas.com/news/telecommunications/Atlantic_Tele-Network_Q3_net_income_up_18* (приступљено 02.11. 2014. у 12:01)
35. www.pubblicaamministrazione.net.
36. www.pubblicaamministrazione.net/e-government/news/929/p1/telenetwork-il-progetto-per-il-telelavoro.html (приступљено 02.11. 2014. у 11:55)
37. www.rankingthebrands.com.
38. www.rankingthebrands.com/The-Brand-Rankings.aspx?rankingID=65&year=76 (приступљено 02.11. 2014. у 11:15)
39. www.my.safaribooksonline.com

    www.my.safaribooksonline.com/book/management/9788131759844/case-study-on-reliance-life-insurance-use-of-information-system-as-a-strategic/c08-sec1-011 (приступљено 02.11. 2014. у 15:10)
40. www.quest.com. www.quest.com/documents/insurance-company-saves-75-percent-of-a-service-desk-fte-within-nine-months-of-launch-casestudy-26716.pdf (приступљено 02.11. 2014. у 15:50)
41. www.lbisoftware.com.
    www.lbisoftware.com/casestudies/LBi-HR%20HelpDesk-Case%20Study.pdf
    (приступљено 02.11. 2014. у 16:10)

42. www.ibm.com.
    www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=SP&infotype=PM&appname=SWGE_IM_IM_USEN&htmlfid=IMS14396USEN&attachment=IMS14396USEN.PDF (приступљено 02.11. 2014. у 16:40)

43. Torpey D. (2011). Fraud in insurance on rise – survey 2010-2011, Ernst & Young 2011

44. www.lbihrhelpdesk.com.www.lbihrhelpdesk.com/HRHelpDesk/login.do (приступљено 03.11. 2014. у 10:10)

45. www.kineticdata.com.http://www.kineticdata.com/news/media-coverage/5-steps-to-a-better-service-desk.html (приступљено 25.11.2014. у 16:30)

46. https://www.forrester.com (приступљено 25.11.2014. у 16:45)

47. Беговић Б, Атанасијевић С, Дулић С, Милићевић В. *Примена HELPDESK-а у функцији повећања ефикасности пословања у државној управи*, (Копаоник 2014, YU INFO 2014 – зборник радова, 23-28)

48. www.inflectra.com. https://www.inflectra.com/Company/In-The-News.aspx (приступљено 26. 11. 2014. у 9:20)

49. www.capterra.com. http://www.capterra.com/help-desk-software/ (приступљено 26.11. 2014. у 9:55)

50. www.inflectra.com. http://www.inflectra.com/KronoDesk/Documentation.aspx (приступљено 26.11.2014. у 10:20)

51. www.sysaid.com. https://www.sysaid.com/features (приступљено 26.11.2014. у 10:55)

52. Highly T. J, Lack M, Meyers P. 2013. *A Critical Analysis Of A New Programming Paradigm* , Programming Languages – CS655 Semester Project

53. Cibran M. A. 2007. *Connecting High Level Business Rules With Object Oriented Applications.* Vrije Universiteit Brussel

54. Kiczales G., Lamping J., Mendhekar A., et al. 1999. *Aspect-Oriented Programming*, Xerox PARC, Palo Alto, CA. 1997.

55. JasCo Community. http://ssel.vub.ac.be/jasco/community.html (приступљено 26.11.2014. у 12:55)

56. https://eclipse.org/ *(приступљено 27.11.2014. у 12:26)*

57. hibernate.org. http://hibernate.org/orm/ (приступљено 27.11. 2014. у 11:08)

58. Schwanninger C, Wuchner E, Kircher M. 2012.Encapsulating Crosscutting Concerns in System Software, Siemens AG

59. Stankić R, Milićević V, Popović M, Savić Z. 2012. Contribution to Intelligent System for Automatic Business Ruless Management Development, TTEM Vol. 7/1, 2012

60. D'Hondt M, D'Hondt T. 2002. The Tirany of Dominant Model Decomposition, Vrije Univesitat Brussel

61. Kisalesz G, Sung J. 2002. AspectJ , XEROX PARC

62. Kiczales G., Lamping J., Mendhekar A., et al. 1999. *Aspect-Oriented Programming*, Xerox PARC, Palo Alto, CA. 1997.

63. www.cambridgecollegesummerschool.co.uk. http://www.cambridgecollegesummerschool.co.uk/spring-school (приступљено 28.11.2014. у 13:04)

64. www.ics.uci.edu. http://www.ics.uci.edu/~lopes/(приступљено 28.11.2014. у 13:15)

65. www.cs.uu.nl. http://www.cs.uu.nl/docs/vakken/swe/13-swe-aspectj.pdf (приступљено 28.11.2014. у 13:10)

66. Stahl T, Völter M, Bettin J, Haase A, Helsen C. 2006. *Model-Driven Software Development: Technology, Engineering, Management*, Wiley

67. Schmidt D. 2006. Model Driven Engineering, IEEE Computer Society 0018-9162/06

68. Kent K. 2001. Model Driven Engineering, IT Topics - 2001

69. Милићевић В. 2009. Објектно-оријентисано пројектовање пословних информационих система, Економски факултет у Београду.

70. www.trados.com (приступљено 20.01.2015. у 9:30)

71. www.translationzone.com (приступљено 20.01.2015. у 9:35)

72. www.omegat.org. http://www.omegat.org/en/dl_overview.php (приступљено 20.01.2015. у 9:40)

73. www.smallbiztrends.com.http://smallbiztrends.com/2014/08/phrase-exact-keyword-match-adwords.html (приступљено 20.01.2015. у 9:44)

74. www.wordstream.com (приступљено 20.01.2015. у 9:50)

75. google.com. https://support.google.com/adwords/answer/6324?hl=en (приступљено 20.01.2015. у 9:55)

76. Christodescu M, Jha S, Kinder J, Katzenbeisser S, Veith H. 2007. *Software Transformations to Improve Malware Detection*, JComput Viral 3: 253 – 265 – Springer

77. Lammer R. 2004. *Coupled Software Transformatins*, First International Workshop on Software Evolution Transformations

78. Meyers B, Mannadir R, Vangheluwe H. 2009. *Evolution of Modeling Languages*, Benevol

79. ibm.com. www.-01.ibm.com/support/docviews?uid=swg21123472 – Creating a Schema, Tables, Object Models and Data Models in Rose (приступљено 22.01.2015 у 9:00)

80. Meng N, Kim M, McKinley K. S. 2011. *Generating Program Transformations from an Example*, The University of Texas at Austin, Papers

81. kutasoftware.com. www.cdukutasoftware.com/worksheet/Geo/12-AllTransformations.pdf (приступљено 22.01.2015 у 9:30)

82. Kavimandan A, Gray J. 2011. *Managing the Quality of Software Product Line Architectures through Reusable Model Transformations,* QoSA + ISARC

83. uml.org. www.uml.org.ck/umlapplication/pdf/crcmodeling.pdf (приступљено 22.01.2015. у 22:15)

84. javabook.compuware.com.www.javabook.compuware.com/content/memory/how-garbage-collection-works.aspx (приступљено 22.01.2015. у 22:30)

85. phonearena.com.http://www.phonearena.com/news/Why-Android-phones-need-3GB-of-RAM-and-iOS-gets-by-with-1GB-of-the-stuff_id62901 (приступљено 22.01.2015. у 22:40)

86. Shapiro A. F. 2007. *An Overview of Insurance Uses of Fuzzy Logic*, Springer

87. stanford.edu.http://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Applications/stocks.html (приступљено 29. 11. 2014. у 14:40)

88. Medina-Santiago A, Camas-Anzueto J. L, Vascuez-Feijoo J. A, Hernández-de León H. R, Mota-Grajales R. (2014). *Neural Control System in Obstacle Avoidance in Mobile Robots Using Ultrasonic Sensors*, Journal of Applied Research and Technology, vol. 12 (2014), 104-110

89. Rivera-Mejía J, Léon-Rubio A. G, Arzabala-Contreras E. (2012). PID Based on a Single Artificial Neural Network Algorithm for Intelligent Sensors, Journal of Applied Research and Technology, vol. 10 No.2 (2012), 262-282.

90. Nalepa, G. J., Bobek, S.: Rule-Based Solution for Context-Aware Reasoning on Mobile Devices. Computer Science and Information Systems, Vol. 11, No. 1, 171–193. (2014)

91. Rodríguez-González, A., Torres-Niño, J., Jimenez-Domingo, E., Gomez-Berbis, J. M., Alor-Hernandez, G.: AKNOBAS: A Knowledge-based Segmentation Recommender System based on Intelligent Data Mining Techniques. Computer Science and Information Systems, Vol. 9, No. 2, 713-740. (2012)

92. Codington, S. & Wilson, T.D. (1994). Information System Strategies in the U.K. Insurance Industry International Journal of Information Management, 14(3), 188-203

93. safaribooksonline.com. http://my.safaribooksonline.com/book/management/9788131759844/case-study-on-reliance-life-insurance-use-of-information-system-as-a-strategic/c08-sec1-011 (приступљено 29. 11. 2014. у 14:50)

94. www.oracle.com http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html (приступљено 29.11. 2014. у 18:01)

95. Скочец Ф.2006. , *Аутоматско распознавање природних језика*, Факултет електротехнике и рачунарства веучилишта у Загребу.

96. nsdu.edu,http://www.ndsu.edu/pubweb/~irfan/Survey%20on%20text%20mining%20networks.pdf (приступљено 29. 11. 2014. у 20:30)

97. Govindarajan M, Chandrasekaran R. M. 2007. *Classifier Based Text Mining for Neural Network*, EBSCO

98. Nordbotten S. 2006. *Data Mining with Neural Networks*, Svein Nordbotten & Associates

99. mashable.com.http://mashable.com/2012/10/09/google-artificial-intelligence/ (приступљено 29. 11. 2014. у 20:40)

100. wolfram.com. http://mathworld.wolfram.com/ZipfsLaw.html (приступљено 29. 11. 2014. у 20:50)

101. stackoverflow.com.

http://stackoverflow.com/questions/10308784/how-to-speed-up-neural-network-performance-in-matlab (приступљено 30. 11. 2014. у 14:38)

102. http://www.uml.org/ (приступљено 30. 11. 2014. у 14:48)

103. http://hibernate.org/ (приступљено 30. 11. 2014. у 15:19)

104. http://maven.apache.org/ (приступљено 30. 11. 2014. у 15:25)

105. http://httpd.apache.org/ (приступљено 30. 11. 2014. у 15:35)

106. www.oracle.com. http://www.oracle.com/technetwork/java/javaee/jsp/index.html (приступљено 30. 11. 2014. у 15:40)

107. Eclipse.org. https://eclipse.org/indigo/ (приступљено 30. 11. 2014. у 15:45)

108. www.java2s.com .http://www.java2s.com/Tutorial/Java/0120__Development/AsimpleSwingbasedapplet.htm (приступљено 30. 11. 2014. у 16:01)

109. developer.android.com. https://developer.android.com/sdk/index.html?hl=i (приступљено 30. 11. 2014. у 16:05)

110. www.android.com. https://www.android.com/versions/lollipop-5-0/ (приступљено 30. 11. 2014. у 16:20)

111. www.apple.com. https://www.apple.com/ios/ (приступљено 30. 11. 2014. у 16:20)

112. http://www.windowsphone.com/en-us (приступљено 30. 11. 2014. у 16:25)

113. http://emisia.net/ (приступљено 30. 11. 2014. у 16:35)

114. Clarke J. 2012. *SQL Injection Attacks and Defense, Second Edition*, SYNGRESS

115. Chau H. 2013. *Network Security – Defense Against DoS/DDoS Attacks*, infosecwriters.com

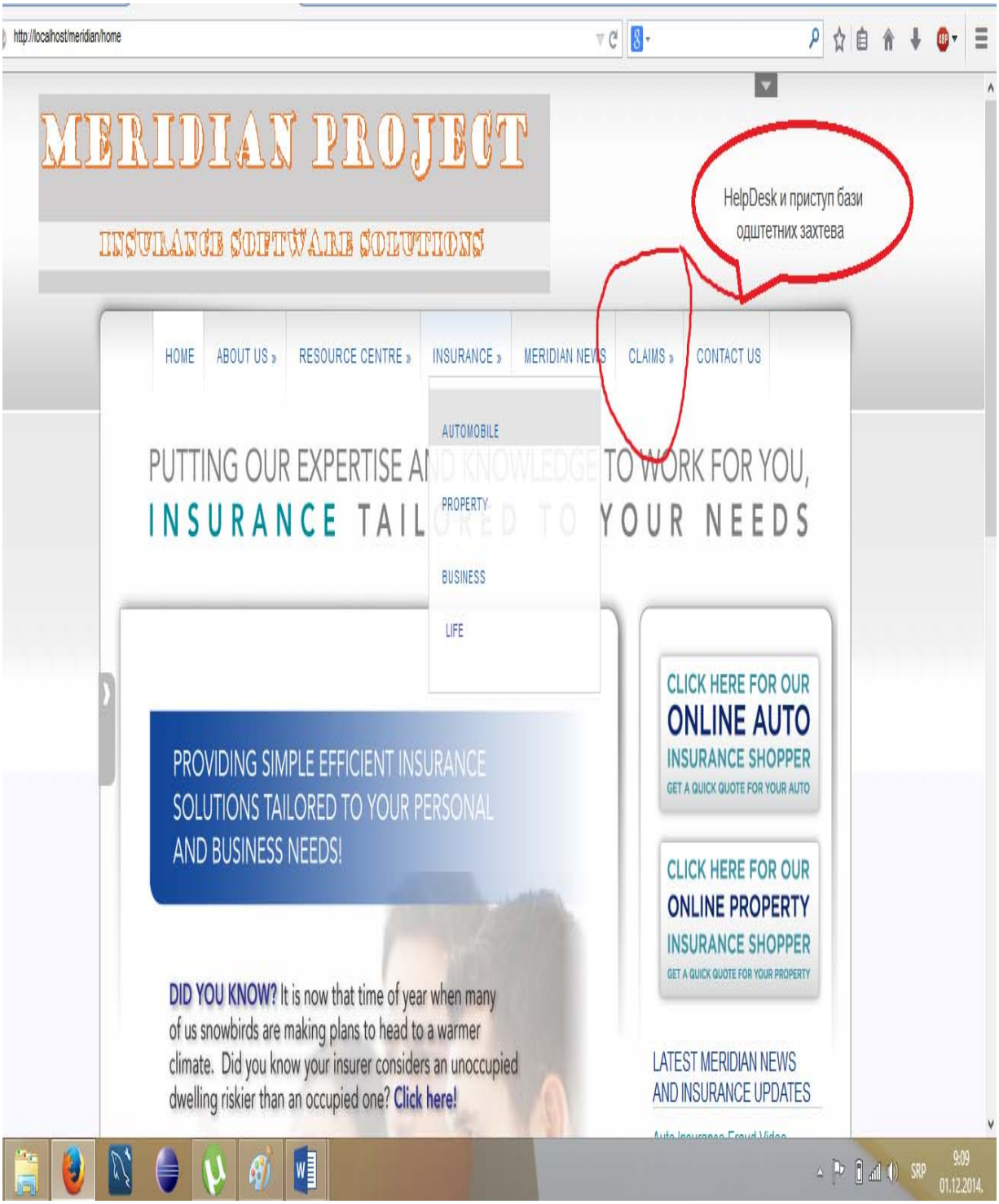
116. hackertarget.com. http://hackertarget.com/sql-injection/ (приступљено 20.02.2015. у 11:40)

117. www.slashgear.com. http://www.slashgear.com/whats-a-ddos-attack-zombies-shopping-help-explain-it-all-11333110/ (приступљено 20.02.2015. у 12:10)

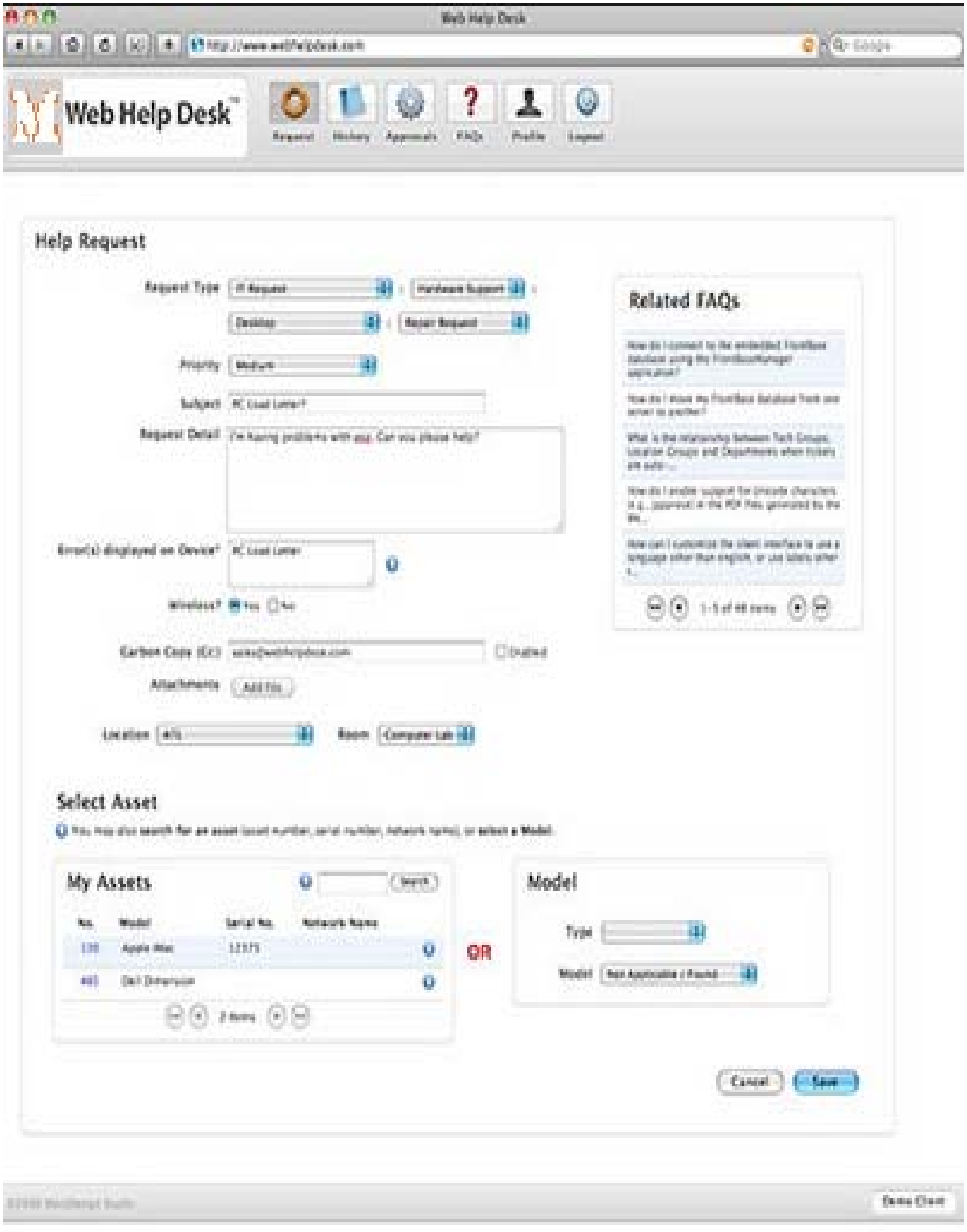
Прилог А

Информациони систем са интелигентном HelpDesk подршком *Meridian*

Сервер веб апликација



Десктоп клијент

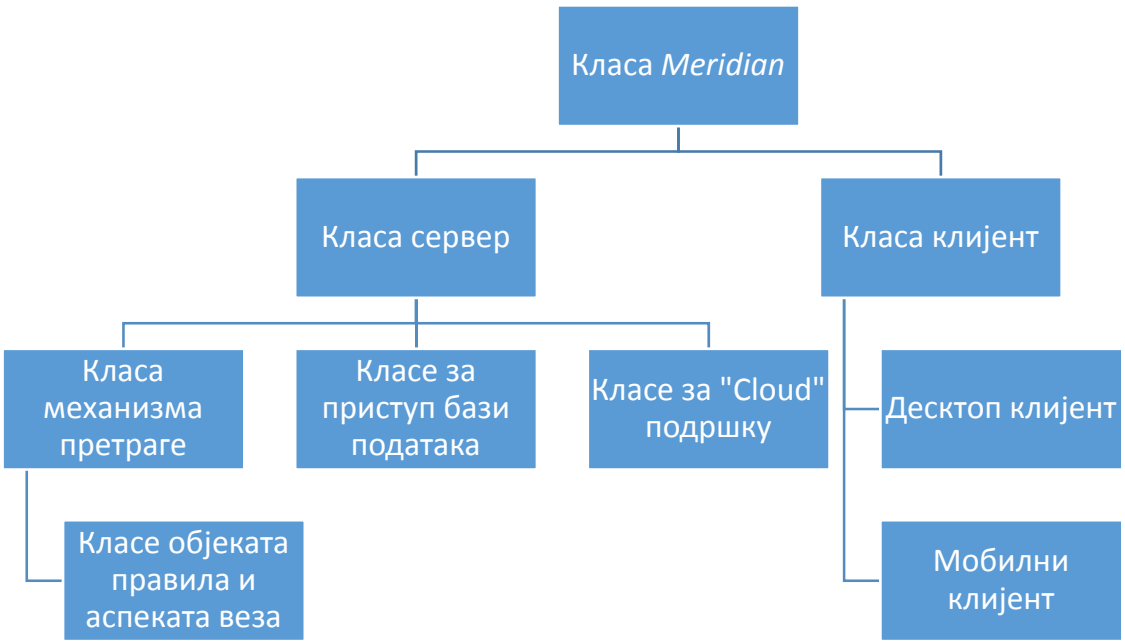


Мобилни клијент

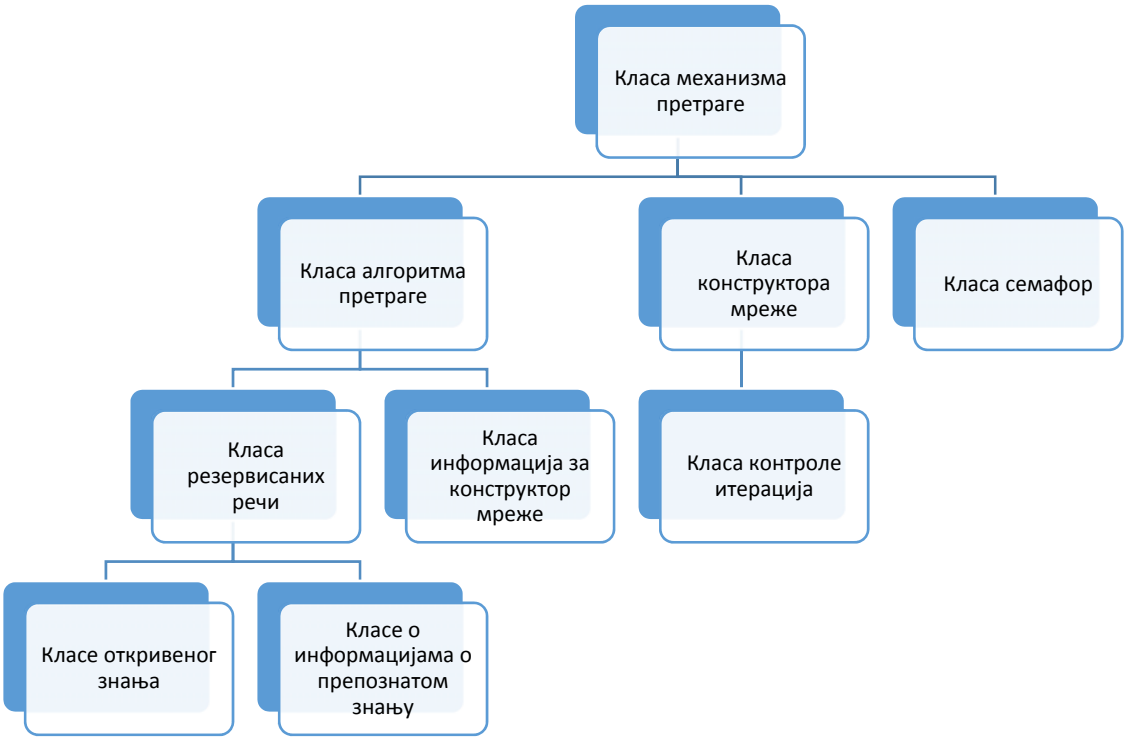
Прилог Б

Избор из пројектне документације

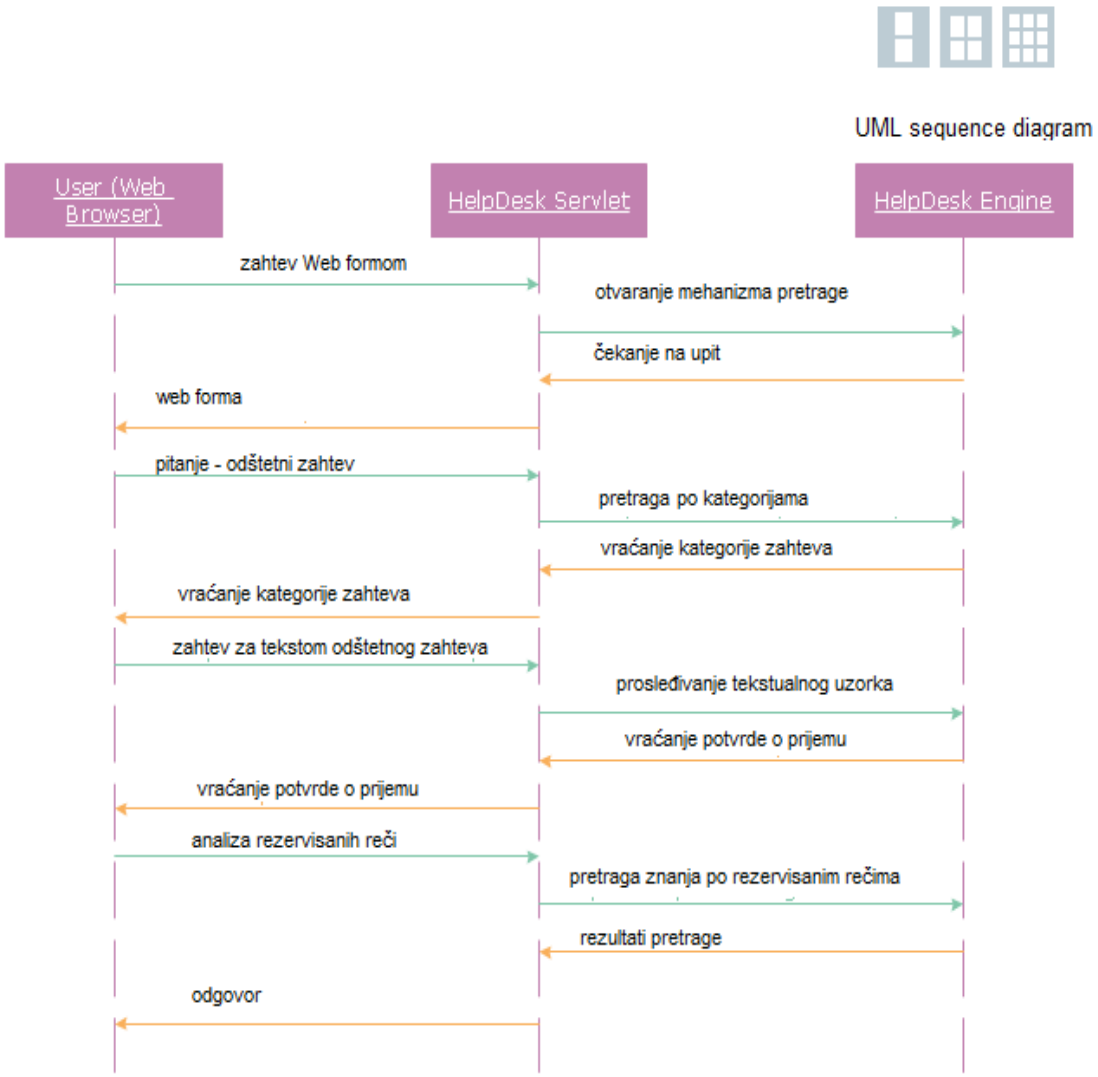
1. Хијерархијски дијаграм класа информационог система *Meridian*



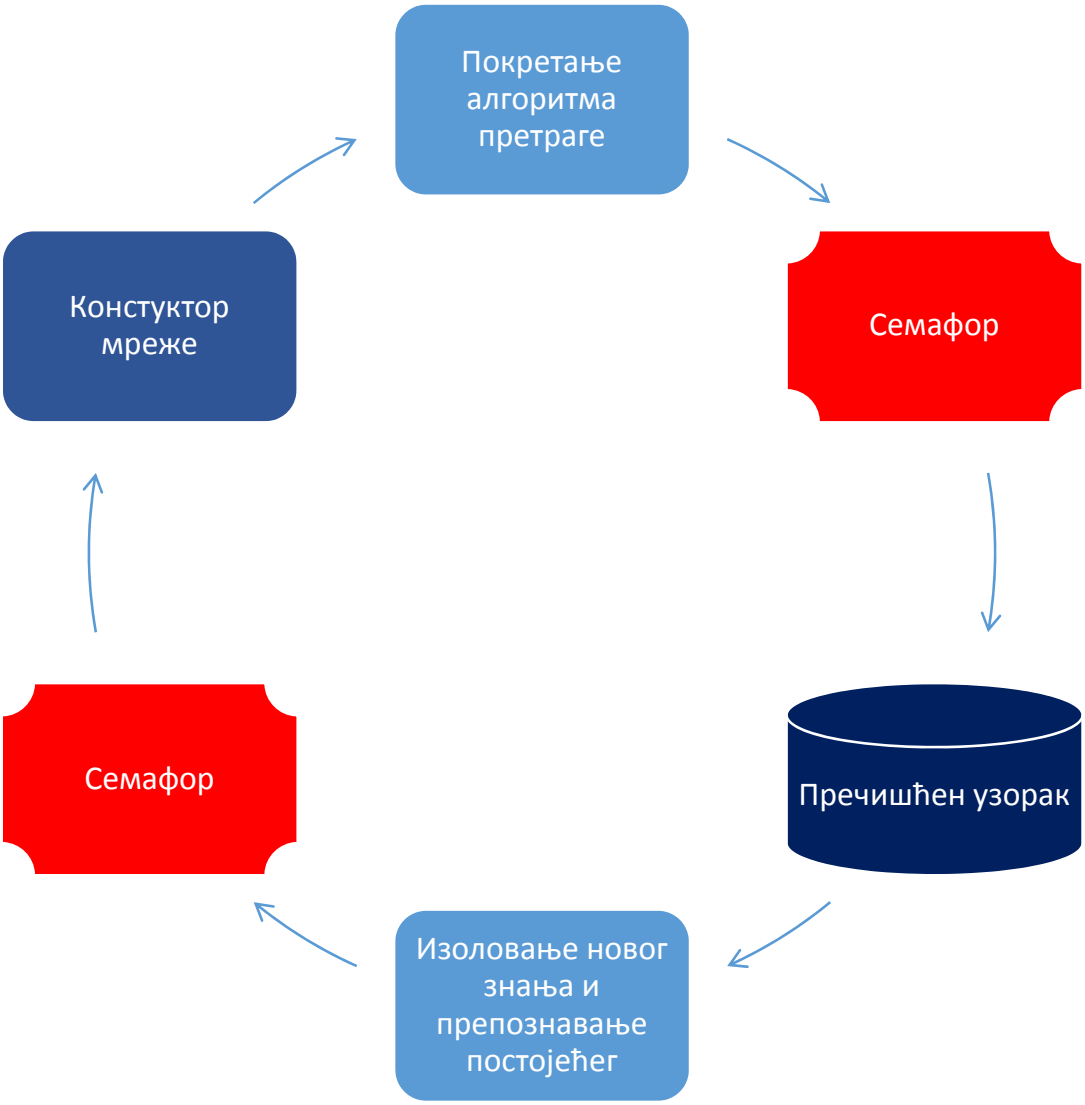
2. Хијерархијски дијаграм класа система за претрагу информационог система *Meridian*



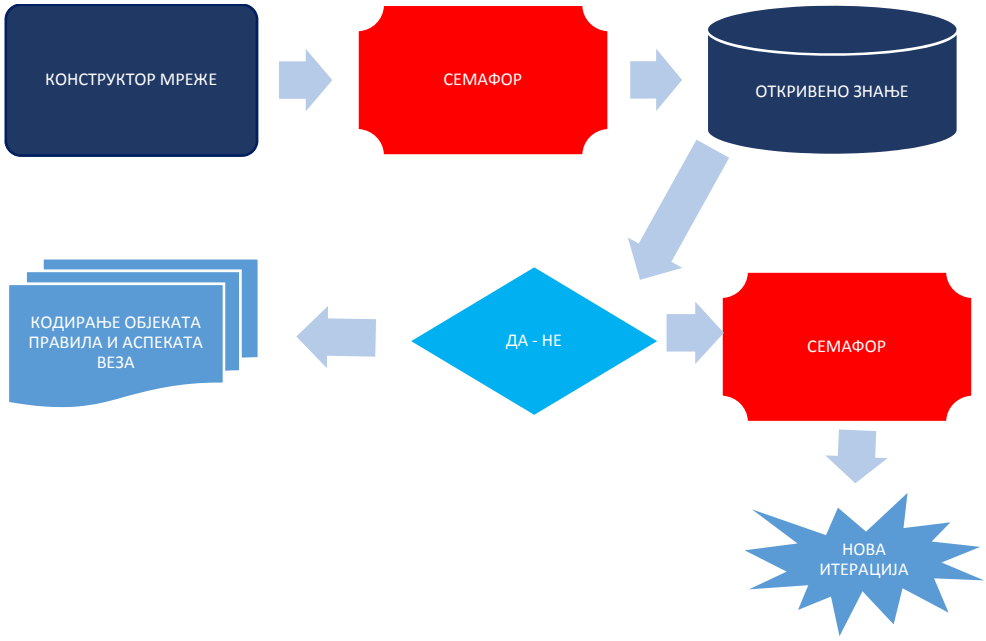
3. Секвентни дијаграм система за претрагу информационог система *Meridian*



4. Токови процесирања система за претрагу информационог система *Meridian*



5. Процесирање конструктора мреже



6. Програмски код алгоритма претраге информационог система *Meridian* са имплементираним семафорима

```
package AlgotitamPretrageUnazad.defaultPackage;

import java.util.Slucajno;
import java.io.Datoteka;
import java.io.SlucajnoPristup;
import java.io.IOException;

public class PretragaUnazad {

   //sprečavanje uporednog procesiranja
   protected double prilagodljivost;
   // indeks 1 – učenje, indeks 0 – nema promena
   protected double korak;
   protected double trenutno;
    // prilagodljivost (0, 1)


   protected double prilagodljivostOcena;

   // default broj neurona je 8 bez ulazbog i izlaznog sloja
   protected int brojNeurona = 8;

   protected Neuron iteracije[][];

   // Prisustvo greske

   public double apGR;

   // Početna iteracija pretrage

   public PretragaUnazad(int brojIteracija,
                         int brojNeurona[],
                         double ucenjeStatus,
                         double sigmoidnaoid,
                         double prilagodljivost,
                    double korak,
                         double trenutno,
                         double prilagodljivostOcena) {

         //korektnost parametara
     if( !(brojIteracija >= 2) ) {
           System.out.println("Greška u iteraciji!");
```

```
            return;
}

if( brojNeurona.length != brojIteracija ) {
            System.out.println("Greška u iteraciji!");
            return;
        }

boolean greska = false;
        for( int i = 0; i < brojNeurona.length && !greska; ++i ) {
            if( !(brojNeurona[i] > 0) ) {
                greska = true;
            }
        }
        if( greska ) {
            System.out.println("Greška u iteraciji!!!");
            return;
        }

        if( !(ucenjeStatus > 0.0 && ucenjeStatus <= 1.0) ) {
            System.out.println("Greška u učenju!!!");
            return;
}

        if( !(korak > 0.0 && korak <= 1.0) ) {
            System.out.println("Greška u iteraciji!!!");
            return;
}

        if( !(trenutno >= 0.0 && trenutno <= 1.0) ) {
            System.out.println("Greška u iteraciji!!!");
            return;
}

        if( !(prilagodljivostOcena >= 0.0 && prilagodljivostOcena <= 1.0) ) {
            System.out.println("Greška u učenju!!!");
            return;
}

        //korekcija strukture – prvi semafor

//SEMAFOR
    private final Semaphore available = new Semaphore(AVG_PROC_LENGTH, true);
    protected synchronized boolean markAsUnused(Object item) {
        for (int i = 0; i < AVG_PROC_LENGTH; ++i) {
          if (item == brojneurona) {
            if (brojneurona) {
```

```
                brojneurona = false;
                return true;
              } else
                return false;
            }
          }
          return false;
        }

      }


      this.brojIteracija = brojIteracija;
      this.brojNeurona = brojNeurona;
      this.ucenjeStatus = ucenjeStatus;
      this.sigmoidna = sigmoidna;
      this.prilagodljivost = prilagodljivost;
      this.korak = korak;
      this.trenutno = trenutno;
      this.prilagodljivostOcena = prilagodljivostOcena;

      nivoi = new Neuron[brojIteracija][];

      // kreiranje ulaza

      for( int i = 1; i < brojIteracija; ++i ) {
         nivoi[i] = new Neuron[brojNeurona[i]];
         for( int j = 0; j < brojNeurona[i]; ++j) {
            nivoi[i][j] = new Neuron(this,brojNeurona[i - 1]);
         }
      }

         this.init();
   }

// pomeranje ulaznih podataka ka izlazu – novi semafor

         //SEMAFOR
      private final Semaphore available = new Semaphore(AVG_PROC_LENGTH, true);
      protected synchronized boolean markAsUnused(Object item) {
          for (int i = 0; i < AVG_PROC_LENGTH; ++i) {
            if (item == brojneurona) {
              if (brojneurona) {
                brojneurona = false;
                return true;
              } else
                return false;
```

```
            }
          }
          return false;
        }

      }

   public void kretanje(double ulaz[]) {
// kontrola toka
      if( brojNeurona[0] != ulaz.length ) {
            System.out.println("Neregularan ulaz!");
            return;
          }
      for( int i = 1; i < brojIteracija; ++i ) {
            double izlaz[] = new double[brojNeurona[i]];
        for( int j = 0; j < brojNeurona[i]; ++j ) {
          izlaz[j] = nivoi[i][j].kretanje(ulaz);
        }
        ulaz = izlaz;
      }
   }

// čuvanje međurezultata

          //SEMAFOR
        private final Semaphore available = new Semaphore(AVG_PROC_LENGTH, true);
        protected synchronized boolean markAsUnused(Object item) {
            for (int i = 0; i < AVG_PROC_LENGTH; ++i) {
              if (item == brojneurona) {
                if (brojneurona) {
                  brojneurona = false;
                  return true;
                } else
                  return false;
              }
            }
            return false;
          }

        }
   public void ucenje(double ulaz[],double izlaz[]) {

     // izlaz provera

      if( brojNeurona[brojIteracija - 1] != izlaz.length ) {
            System.out.println("Nekorektan izlaz!");
            return;
```

```
    }

kretanje(ulaz);

// ključni deo pretrage, pomeranje unazad
double greska[] = new double[brojNeurona[brojIteracija - 1]];
double prethodnaGreska[];
apGR = 0.0;
for( int j = 0; j < brojNeurona[brojIteracija - 1]; ++j ) {

        //određivanje tekuđe greške

   greska[j] = (izlaz[j] - nivoi[brojIteracija - 1][j].izlaz) *
           trenutno(nivoi[brojIteracija - 1][j].state);

        // učenje po vrednosti greške

   nivoi[brojIteracija - 1][j].ucenje(greska[j]);

        // ukupna greška mreže

   apGR += Math.abs(greska[j]);
}

// procesiranje u skrivenom sloju mreže

for( int i = brojIteracija - 2; i > 0 ; --i ) {
   prethodnaGreska = greska;
   greska = new double[brojNeurona[i]];
   for( int j = 0; j < brojNeurona[i]; ++j ) {

                //određivanje tekuđe greške

                for( int k = 0; k < brojNeurona[i + 1]; ++k ) {
      greska[j] += prethodnaGreska[k] *
                    nivoi[i+1][k].tezine[j];
      }
      greska[j] *= trenutno(nivoi[i][j].state);

                  // učenje po vrednosti greške

      nivoi[i][j].ucenje(greska[j]);

   }
}

                ucenjeStatus *= korak;
```

```
    prilagodljivost += (1.0 - prilagodljivost) * prilagodljivostOcena;
  }


// određivanje tež. faktora u čvorovima

  public void noveTezine(){
        Slucajno slucajno = new Slucajno();

    for( int i = 1; i < brojIteracija; ++i ) {
      for( int j = 0; j < brojNeurona[i]; ++j ) {
        nivoi[i][j].nove(slucajno);
      }
    }
  }


  // skladiste podataka koeficijenti – sledeći semafor

      //SEMAFOR
      private final Semaphore available = new Semaphore(AVG_PROC_LENGTH, true);
      protected synchronized boolean markAsUnused(Object item) {
          for (int i = 0; i < AVG_PROC_LENGTH; ++i) {
            if (item == brojneurona) {
              if (brojneurona) {
                brojneurona = false;
                return true;
              } else
                return false;
            }
          }
          return false;
        }

       }

  public void snimiNeuron(String putanja, String naziv){
        try{
          Datoteka datoteka = new Datoteka(putanja, naziv);
          try{
            SlucajnoPristup datoteka = new
            SlucajnoPristup(datoteka,"sirov");
                // čuvanje tipa String
                datoteka.writeUTF("datoteka_NN");
                // čuvanje trenutnih parametara structure mreže
```

```
                datoteka.writeInt(brojIteracija);
                for( int i = 0; i < brojIteracija; ++i ) {
                   datoteka.writeInt(brojNeurona[i]);
                }
                datoteka.writeDouble(ucenjeStatus);
                datoteka.writeDouble(sigmoidna);
                datoteka.writeDouble(prilagodljivost);
                datoteka.writeDouble(korak);
                datoteka.writeDouble(trenutno);
                datoteka.writeDouble(prilagodljivostOcena);

                //

                for( int i = 1; i < brojIteracija; ++i ) {
                   for( int j = 0; j < brojNeurona[i]; ++j ) {
                           nivoi[i][j].writeUTF(datoteka);
                   }
                }
                datoteka.close();
           } catch(IllegalArgumentException iae){
                System.out.println("Greška ulaza:" +  iae);
           }
        } catch(IOException ioe){
           System.out.println("Greška datoteke:" +  ioe);
        } catch(SecurityException si) {
           System.out.println("Sigurnosni izuzetak:" +  si);
        }
}
```

```
//SEMAFOR
private final Semaphore available = new Semaphore(AVG_PROC_LENGTH, true);
protected synchronized boolean markAsUnused(Object item) {
    for (int i = 0; i < AVG_PROC_LENGTH; ++i) {
      if (item == brojneurona) {
        if (brojneurona) {
          brojneurona = false;
          return true;
        } else
          return false;
      }
    }
    return false;
  }

 }
```

```
// Učitavanje prethodne iteracije

 public static PretragaUnazad ucitajMrezu(String putanja, String naziv){

        if(naziv == null || putanja == null) return null;

        try{
           Datoteka datoteka = new Datoteka(putanja, naziv);
           try{
                      SlucajnoPristup datoteka = new
                      SlucajnoPristup(datoteka,"r");

                   //Provera tipa String

                   if(datoteka.readUTF().compareTo("datoteka_NN") != 0){
                      datoteka.close();
                      System.out.println("Nekorektan podatak!!!");
                      return null;
                   }

                  // nova struktura mreže
                  PretragaUnazad bpn = new PretragaUnazad();
                  bpn.brojIteracija = datoteka.readInt();
                  bpn.brojNeurona = new int[bpn.brojIteracija];
                  for( int i = 0; i < bpn.brojIteracija; ++i ) {
                     bpn.brojNeurona[i] = datoteka.readInt();
                  }

                  bpn.nivoi = new Neuron[bpn.brojIteracija][];

                  //ulaz

                  for( int i = 1; i < bpn.brojIteracija; ++i ) {
                     bpn.nivoi[i] = new Neuron[bpn.brojNeurona[i]];
                     for( int j = 0; j < bpn.brojNeurona[i]; ++j) {
                              bpn.nivoi[i][j] = new Neuron(bpn,
                                               bpn.brojNeurona[i - 1]);
                     }
                  }

                  // novi parametri

                  bpn.ucenjeStatus = datoteka.readDouble();
                  bpn.sigmoidna = datoteka.readDouble();
                  bpn.prilagodljivost = datoteka.readDouble();
                  bpn.korak = datoteka.readDouble();
```

```
                bpn.trenutno = datoteka.readDouble();
                bpn.prilagodljivostOcena = datoteka.readDouble();

                // nove težine

                for( int i = 1; i < bpn.brojIteracija; ++i ) {
                   for( int j = 0; j < bpn.brojNeurona[i]; ++j ) {
                           bpn.nivoi[i][j].readUTF(datoteka);
                   }
                }

                datoteka.close();

                return bpn;

            } catch(IllegalArgumentException iae){
                System.out.println("Greška ulaza:" +  iae);
            }
        } catch(IOException ioe){
            System.out.println("Greška datoteke:" +  ioe);
        } catch(SecurityException si) {
            System.out.println("Sigurnosni izuzetak:" +  si);
        }

        return null;
}



// vraćanje izlaza

public double izlaz(int i){
   return nivoi[brojIteracija - 1][i].izlaz;
}
   public double hidden(int i){
   return nivoi[brojIteracija - 2][i].izlaz;
}

//vraćanje vrednosti parametara

public double getUcenjeStatus(){
   return ucenjeStatus;
}


public void setUcenjeStatus(double ucenjeStatus){
   this.ucenjeStatus = ucenjeStatus;
```

```
    }
    //konstruktor klase
    protected PretragaUnazad() {
    }

    protected double trenutnoid( double x){
       return 2.0 / (1.0 + Math.exp(-1.0 * prilagodljivost * x + sigmoidna))
                                          - 1;
    }
    // rezultat primene funkcije trenutno.
    protected double trenutno(double x){
            double r = Math.exp(-1.0 * prilagodljivost * x + sigmoidna);
            return (2.0 * prilagodljivost * r) /((r + 1) * (r + 1));
    }
    /*************************************************************
     *
     * inicijalizacija mreze sa razlicitim tezinama
     *
     *************************************************************/
    protected void init(){
            Slucajno slucajno = new Slucajno();

       for( int i = 1; i < brojIteracija; ++i ) {
          for( int j = 0; j < brojNeurona[i]; ++j ) {
             nivoi[i][j].init(slucajno);
          }
       }
    }
}
```

Биографија

Владимир Л. Милићевић рођен је 20.09.1974. у Брчком, Реп. Босна и Херцеговина, где је одличним успехом завршио основну школу. Средњу стручну спрему стекао је у Првој крагујевачкој гимназији након чега уписује Природно математички факултет у Крагујевцу, смер Рачунарство и информатика. Након успешно завршеног Факултета стиче назив дипломирани математичар-информатичар.

Од фебруара 2005. до септембра 2007. Ангажован је по уговору на Економском факултету у Брчком, Универзитета у Источном Сарајеву, у звању асистента, а на следећим пословима:

- Извођење вежби из предмета: *Базе података, Електронско пословање, Пројектовање информационих система;*

- Администрирање и одржавање информатичког центра и рачунарске мреже.

Од октобра 2005. по уговору је ангажован на Високој техничкој школи у Крагујевцу на радном месту сарадника у настави. Послови на којима је био ангажован обухватају:

- Извођење вежби на предметима: *Програмирање на рачунару, Математика 1, Математика 2, Познавање производних процеса.*

У 2009. успешно је завршио последипломске студије на смеру *Управљање и информациони системи* и одбранио је магистарски рад под називом *Објектно-оријентисано пројектовање пословних информационих система* и стекао звање магистра економских наука, ужа област пословна информатика.

Исте године прелази у стални радни однос на ВТШСС у Крагујевцу у звању сарадника у настави где је 2011. изабран за предметног наставника за области *Математика и програмирање* и *Рачунари и пројектовање*, а на предметима *Увод у програмирање, Објектно-оријентисано програмирање, Примена WEB система, Интелигентни системи у саобраћају* и *Методе одлучивања.*

Аутор је већег броја апликативних софтверских решења из области објектно-оријентисаног програмирања, као и коаутор збирки задатака из програмирања који се користе у настави на Економском факултету у Брчком и ВТШ струковних студија у Крагујевцу. Активно користи следеће програмске језике и развојне алате:

- JAVA, AspectJ, JasCo, JBoss, Eclipse, Maven, Hibernate;
- VB, VB Net, C# (.net);
- MySQL, SQL server;
- Php, html;
- ERWin, BPWin, UML.

У периодима 2007-2012. и 2012 – 2015. био је ангажован на пословима сарадника за развој софтвера од компаније *Meridian Project*. Од 2015. године са компанијом *Limessoft* учествује на *Fractals* пројекту BeeWeb који се финансира из ЕУ фондова.

Овлашћени је ECDL испитивач. Поседује активно знање енглескоj језика.

Живи и ради у Крагујевцу. Ожењен је и отац троје деце.

Изјаве аутора

Прилог 1.

ИЗЈАВА О АУТОРСТВУ

Изјављујем да је докторска дисертација, под насловом

Унапређење савремених HelpDesk пословних система применом напредних интелигентних софтверских алата

- резултат сопственог истраживачког рада,
- да предложена дисертација, ни у целини, ни у деловима, није била предложена за добијање било које дипломе, према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио ауторска права, нити злоупотребио/ла интелектуалну својину других лица.

У Нишу, 02.07.2015.

Аутор дисертације: Владимир Милићевић

Потпис докторанда:

Прилог 2.

ИЗЈАВА О ИСТОВЕТНОСТИ ШТАМПАНЕ И ЕЛЕКТРОНСКЕ ВЕРЗИЈЕ ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Име и презиме аутора: Владимир Милићевић

Наслов рада: *Унапређење савремених HelpDesk пословних система применом напредних интелигентних софтверских алата*
Ментор: Проф. др Славољуб Миловановић

Изјављујем да је штампана верзија моје докторске дисертације истоветна електронској верзији, коју сам предао за уношење у **Дигитални репозиторијум Универзитета у Нишу.**

Дозвољавам да се објаве моји лични подаци, који су у вези са добијањем академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада, и то у каталогу Библиотеке, Дигиталном репозиторијуму Универзитета у Нишу, као и у публикацијама Универзитета у Нишу.

У Нишу, 02.07.2015.

Аутор дисертације: Владимир Милићевић

Потпис докторанда:

Прилог 3.

ИЗЈАВА О КОРИШЋЕЊУ

Овлашћујем Универзитетску библиотеку „Никола Тесла“ да, у Дигитални репозиторијум Универзитета у Нишу, унесе моју докторску дисертацију, под насловом:

Унапређење савремених HelpDesk пословних система применом напредних интелигентних софтверских алата

која је моје ауторско дело.

Дисертацију са свим прилозима предао сам у електронском формату, погодном за трајно архивирање.

Моју докторску дисертацију, унету у Дигитални репозиторијум Универзитета у Нишу, могу користити сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons), за коју сам се одлучио.

1. Ауторство
2. Ауторство – некомерцијално
3. Ауторство – некомерцијално – без прераде
4. Ауторство – некомерцијално – делити под истим условима
5. Ауторство – без прераде
6. Ауторство – делити под истим условима

У Нишу, 02.07.2015.

Аутор дисертације: Владимир Милићевић

Потпис докторанда: