УНИВЕРЗИТЕТ У БЕОГРАДУ

ФАКУЛТЕТ ОРГАНИЗАЦИОНИХ НАУКА

Ивана В. Огњановић

СЕМАНТИЧКЕ ТЕХНОЛОГИЈЕ ЗА КОНФИГУРИСАЊЕ СЕРВИСНО-ОРИЈЕНТИСАНИХ АРХИТЕКТУРА НА ОСНОВУ НЕФУНКЦИОНАЛНИХ ЗАХТЕВА

докторска дисертација

Београд, 2012.

UNIVERSITY OF BELGRADE

FACULTY OF ORGANIZATIONAL SCIENCES

Ivana V. Ognjanović

CONFIGURATION OF SERVICE ORIENTED ARCHITECTURES WITH SEMANTIC TECHNOLOGIES BASED ON NON-FUNCTIONAL REQUIREMENTS

Doctoral Dissertation

Belgrade, 2012

Ментор:

др Јелена Јовановић, доцент

Факултет организационих наука, Универзитет у Београду

Чланови комисије:

др Владан Девеџић,редовни професор

Факултет организационих наука, Универзитет у Београду

др Драган Гашевић, ванредни професор

Атабаска Универзитет, Канада

Датум одбране: _____

Семантичке технологије за конфигурисање сервисно-оријентисаних архитектура на основу нефункционалних захтева

Резиме:

Ова дисертација је фокусирана на примену семантичких технологија за решавање проблема оптималне конфигурације сервисно-оријентисаних архитектура (енгл. *Service Oriented Architecture – SOA*) на основу нефункционалних захтева корисника. Решење је базирано на проширењу AHP алгоритма за рад са различитим врстама захтева и развоју хеуристичког приступа заснованог на генетичким алгоритмима за решавање проблема оптималне конфигурације. Постојећа решења у овој области су показала изузетно мали ниво персонализације, тј корисницима није дозвољено дефинисање разних софистицираних врста захтева који осликавају њихове жеље, очекивања и строге захтеве за које захтевају потпуно испуњење. Такође, постојећа решења су била перманентно фокусирана на испуњење захтева функционалности, након чега се врши одабир конфигурације сходно захтевима о смањењу вредности карактеристика које имају тенденцију раста (нпр., цена и време извршавања), односно повећању вредности карактеристика које имају тенденцију опадања (нпр., поузданост и доступност). Међутим, када се посматрају целе фамилије SOA, од посебног значаја постаје проблем конструкције конфигурације при истовременом задовољењу функционалних и нефункционалних захтева.

Предложено интегрално решење под називом *OptConfSOAF*обезбеђује представљање различитих врста захтева (безусловни, условни, захтеви о лексикографском поретку) о нефункционалним карактеристикама и оптималну конфигурацију фамилија SOA на основу дефинисаних захтева. Приступ који се предлаже обезбеђује истовремено задовољење захтева који се тичу функционалности система као и нефункционалних захтева који могу бити различитог нивоа приоритета, односити се на поједине делове или сервисно-оријентисану архитектуру у целости.

Предложено решење је опште и није ограничено само на веб сервисе, иако се појам семантичких технологија обично везује за дати домен примене. Решење

се може применити у било ком домену у којем се SOA парадигма може применити посматрањем сервиса као било које компоненте (необавезно софтверске) дате функционалности.

У дисертацији је такође развијено софтверско решење базирано на Eclipse платформи које омогућује представљање различитих врста захтева и примену развијеног CS-AHP алгоритма за њихову анализу и рангирање расположивих одлика у моделу одлика (енгл. *feature model*).

Кључне речи: сервисно-оријентисане архитектуре, семантичке технологије, нефункционални захтеви, онтологије, правила, генетички алгоритми, AHP метод
Научна област:рачунарство
Ужа научна област:софтверско инжењерство
УДК број:004.41

Configuration of Service Oriented Architectures with Semantic Technologies based on Non-Functional Requirements

Abstract:

This dissertation is focused on the application of semantic technologies for solving the problem of optimal configuration of service-oriented architectures (SOA) based on stakeholders' non-functional requirements. The proposed solution is developed as an extension of the AHP algorithm to allow for processing of different kinds of requirements. To address the problem of optimal configuration of SOA, a heuristic approach based on genetic algorithms has also been proposed and validated.

Existing approaches in this field have shown low level of personalization, i.e. stakeholders are neither enabled to define sophisticated requirements that reflect their own expectations and attitudes, nor they are able to indicate hard requirements that have to be fully satisfied. Furthermore, existing approaches were primarily addressing the problem of fulfilling functional requirements, while the selection of an appropriate configuration is driven by the goal of decreasing the values of monotonically decreasing features (e.g., price and execution time) and simultaneous increasing the values of monotonically increasing features (e.g., availability and reliability). By considering the whole SOA families, the problem of configuration based on both functional and non-functional requirements gets special importance for research and further applications.

The proposed solution, titled OptConfSOAF provides a framework for specification and processing of different kinds of requirements (unconditional, conditional, and requirements about lexicographical order) over non-functional features, and further optimal configuration of SOA families. The proposed approach provides simultaneous fulfillment of functional requirements (i.e., requirements related to the system's functionalities) and non-functional requirements, where the latter could be defined with different level of importance, for specific parts of a SOA-based system or the system in its entirety.

The proposed solution is general and is not bound to web services, even though semantic technologies are often associated with that domain. Since the solution considers a service as a component (no mandatory to be software component) with the

specified functionality, it is applicable and easily adaptable to any specific application domain where SOA paradigm may be applied.

This dissertation also includes a software solution, developed as an extension of the Eclipse platform, that provides a user interface for specification of different kinds of requirements, and makes use of the developed CS-AHP algorithm for the analyses of the specified requirements and ranking of available features in the featuremodel.

Key words: servis-oriented architectures, semantic technologies, non-functional requirements, ontologies, rules, genetic algorithms, AHP method
Scientific field: computing
Major scientific field: software engineering
UDK number:004.41

Садржај

1. Увод

Савремени трендови у развоју Семантичког Веба и дистрибуираних система су довели до велике промене у самом приступу развоју софтвера: од самосталног и изолованог начина развоја софтвера до развоја интеграцијом и спајањем одговарајућих сервиса и компоненти. Сервисно оријентисана архитектура (*Service Oriented Architecture – SOA*) се појављује као парадигма која је изузетно погодна за дистрибуирано пројектовање рачунарских апликација, њихов развој и примену. Ефикасан и флексибилан развој софтвера заснован на SOA парадигми, врши се првенствено на основу дефинисаних функционалних захтева за сваку од компоненти као и за систем у целини. Осим испуњења функционалних захтева често се намеће и додатни критеријум: селекција сервиса на основу њихових нефункционалних обележја и захтева дефинисаних над њима. Међутим, постојеће технологије не омогућавају аутоматизацију развоја SOA-заснованих софтверских решења на основу дефинисаних додатних нефункционалних захтева.

Постојећа решења у овој области су показала изузетно мали ниво персонализације, тј корисницима није дозвољено дефинисање разних софистицираних врста захтева који осликавају њихове жеље, очекивања и строге захтеве за које захтевају потпуно испуњење. Такође, постојећа решења су била перманентно фокусирана на испуњење захтева функционалности, након чега се врши одабир конфигурације сходно захтевима за смањење вредности карактеристика које имају тенденцију раста (нпр. цена и време извршавања),и повећање вредности карактеристика које имају тенденцију опадања (нпр. поузданост и доступност). Међутим, када се посматрају целе фамилије SOA, од посебног значаја постаје проблем конструкције конфигурације при истовременом задовољењу функционалних и нефункционалних захтева.

Захтеви над нефункционалним обележјима могу бити дефинисани квалитативно или квантитативно, а могу осликавати и међусобну зависност тј. условљеност међу тим обележјима. Семантичке технологије садрже велики потенцијал за реализацију овако дефинисаних условних захтева. Ако се нефункционалним обележјима додели експлицитно, онтолошки-

заснованозначење и одговарајуће инстанце (као могуће вредности), за представљање условних захтева се намећу *if-then*правила која су добро успостављена у интелигентним системима. Осим тога, у оквиру SOA-заснованих софтверских решења може доћи до проблема интеграције домена па се као решење намеће употреба онтологија које омогућавају формално и прецизно дефинисање домена и размену знања. Такође, потребно је имати у виду да у процесу дефинисања захтева обично учествује више корисника од којих свако посебно дефинише своје захтеве и преференце над нефункционалним обележјима, па се природно намеће и питање могућности интеграције и испуњења ових захтева, како појединачно, тако и у свеукупности. И за решавање овог проблема се уочава могућност примене онтолошки заснованог знања и резоновања над њима.

Досадашња истраживања у области представљања и анализе корисничких захтева су показала да је већина метода развијена за анализу безусловно дефинисаних захтева који у највећем броју случајева не одговарају реалности. Проблем анализе условно дефинисаних захтева је развијан у многим областима примене, као што су: операциона истраживања, базе података, вештачка интелигенција, али развијени методи имају проблем постојања цикличности међу условима у разним захтевима, који директно нарушавају комплетност датих метода. Такође, додатни проблем је проблем временске сложености за решавање упита о поређењу две опције.

Истраживачка заједница је углавном одвојено изучавала проблеме моделовања и анализе захтева са једне стране, и проблеме конфигурације SOA на основу функционалних и нефункционалних захтева, са друге; резултати су показали немогућност директне примене постојећих техника/технологија за дефинисани проблем оптималне конфигурације SOA фамилија. Међутим, савремени трендови у развоју софтвера су додатно наметнули захтеве за обезбеђењем варијабилности и поновне употребљивости делова система, чиме се додатно поспешује потреба за обезбеђењем аутоматизације конструкције SOA фамилија.

Стога, технике и технологије за развој SOA треба да омогуће нов начин за аутоматски одабир сервиса на основу ширег и већег скупа захтева који ће осим функционалности обухватати и захтеве дефинисане над нефункционалним

обележјима препознатим за дати систем. На тај начин може да се обезбеди и развој целе фамилије SOA које имају заједнички скуп функционалности, али у различитом степену испуњавају нефункционалне захтеве на основу чега су и рангиране. Овакав приступ у развоју и конфигурацији SOA-е има потенцијално велику примену у различитим областима као што су системи за учење, композиција софтверских система на бази пословних процеса, системи за одлучивање који могу укључивати и историјске податке и искуства, итд.

Из свега наведеног се може закључити да се теоријско одређење предмета истраживања може сажети тако да се предмет истраживања дефинише као: ***коришћење семантичких техника и технологија у сврху обезбеђења развоја сервисно-оријентисаних архитектура на основу условно и безусловно дефинисаних захтева над нефункционалним обележјимакомпоненти и система у целини.***

1.1. Циљеви истраживања

Семантичке технике и технологије су препознате као погодне за примену у развоју SOA-заснованих софтверских решења, на основу захтева дефинисаних над нефункционалним обележјима. Као циљ истраживања се наметнуло испитивање и уочавање предности и недостатака њихове примене у циљу:

- Дефинисања нефункционалних обележја са одређеним онтолошким значењем и додељеним могућим вредностима у форми одговарајућих онтолошких инстанци. Сваки сервис је на одговарајући начин окарактерисан (означен) у односу на дефинисани скуп нефункционалних обележја и инстанци.
- Представљања разних врста захтева над нефункционалним карактеристикама који могу бити квалитативни или квантитативни, а у оба случаја условни или безусловни. Они могу да осликавају односе и преференце над самим обележјима као и над њиховим инстанцама. Такође, за одређени скуп обележја преференце могу бити дефинисане

квалитативно, а за преостали квантитативно, па је омогућена и њихову интеграцију као и методе за трансформацију.

- Развоја метода и техника за аутоматизацију процеса конфигурације фамилија SOA на основу захтева корисника и скупа расположивих сервиса.
- Развоја метода и техника за аутоматску детекцију и отклањање могућих неконзистенција. Неконзистенције могу настати у разним фазама процеса спецификације, рангирања и/или конфигурације, па је обезбеђен процес њихове детекције и отклањања.
- Анализе могућности примене описаног поступка у разним областима које карактеришу различити услови окружења и карактеристике процеса спецификације и конфигурације.

Развијени методи и технике су интегрисане у потпуно ***ново интегрално решење*** које омогућује потпуну аутоматизацију циелог процеса, од саме спецификације и дефинисања захтева до крајњег развоја и конфигурације SOA-е која у највећој мери одговара дефинисаним захтевима. За сада није познат ни један приступ са сличним или истим циљевима, стога, ниво научног достигнућа који јепостигнут је ***научно откриће***.

Зареализацију истраживања било је неопходно сагледати искуства и резултате из сродних области, као што су вештачка интелигенција, операциона истраживања, базе података итд. У овим областима су идентификовани следећи елемети који су од значаја:

- употреба семантичких техника и технологија за развој SOA-заснованих софтверских система на основу разних врста захтева над нефункционалним обележјима
  - употреба онтолошки заснованих начина за предстаљање нефункционалних обележја;
  - употреба система заснованих на правилима за представљање разних врста захева над онтолошком структуром нефункционалних захтева;

- употреба ефективних и ефикасних техника за резоновање над онтолошком структуром као и система заснованих на правилима у комбинацији са техникама за квантитативно рангирање;

- употреба техника вештачке интелигенције за решавање оптималних задатака
  - интеграција техника интелигентног резоновања и квантитативног рангирања са техникама вештачке интелигенције за решавање оптималних задатака;
  - анализа карактеристика интегрисаних метода у зависности од различитих услова окружења у областима примене;
- употреба метода интелигентног резоновања за развој техника за детекцију и разрешавање неконзистентности у појединим фазама целокупног процеса од спецификације захтева до оптималне конфигурације.

1.2. Садржај рада по поглављима

Ова дисертација се састоји од седам поглавља и поглавља са списком литературе која је коришћена при изради дисертације. Након поглавља Увод, следи поглавље у коме је дат преглед три релевантне области: сервисно-оријентисане архитектуре, софтверски захтеви, и семантичке технологија и техника вештачке интелигенције за решавање оптималних захтева. У овом поглављу се, након увођења дефиниције SOA и навођења основних принципа SOA парадигме, проблем композиције сервиса наводи као централни принцип при контрукцији SOA архитектура и уводи појам сервисно-оријентисаног дијаграма тока. Као језик за моделовање се уводи BPMN нотација која се у дисертацији користи за представљање основних образаца композиције сервиса. Како за сваку од функционалности у SOA на располагању може бити већи број сервиса, нефункционалне карактеристике се уводе као мере квалитета расположивих сервиса и последично целе фамилије. Такође, нефункционалне карактеристике се могу карактерисати сходно доменима примене, па се наводи и неколико примера у којима су конструисани одговарајући модели за представљање

нефункционалних карактеристика. На крају, у складу са савременим захтевима варијабилности и поновне употребљивости појединих делова система, уводи се појам фамилија SOA, начини за моделовање, фазе у животном циклусу и начини за генерисање вредности нефункционалних карактеристика целе фамилије на основу вредности карактеристика расположивих сервиса. Следећи део овог поглавља се односи на проблем инжењеринга захтева корисника и фаза у животном циклусу. Посебан акценат је постављен на постојећим алгоритмима из разних области и домена примене који се односе на представљање и анализу разних врста захтева. Такође, представљени су постојећих приступи за моделовање разних врста захтева над SOA архитектурама и фамилијама. На крају, дат је преглед техника за семантичко моделовање знања (онтологије и правила) које се користе у дисертацији као и метода вештачке интелигенције за решавање оптималних захтева. Последњи део овог поглавља се односи на поређење постојећих приступа и идентификацију њихови ограничења и каракетристика примене.

Треће поглавље дефинише проблем истраживања и идентификује могућности примене семантичких технологија, технологија вештачке интелигенције и метода за анализу захтева корисника у домену дефиниције захтева о нефункционалним карактеристикама и проблем оптималне конфигурације SOA фамилија.

Четврто поглавље даје комплетно решење дефинисаног проблема које се предлаже дисертацијом. Решење се може поделити у четири јасне целине: 1) креирање двослојне структуре за представљање нефункционалних карактеристика и захтева над њима; 2) рангирање разних врста захтева и теоретска анализа коректности и комплетности CS-AHP алгоритма који се предлаже; 3) интегрално решење *ConfSOA*засновано на хеуристичком приступу за решавање проблема оптималне конфигурације SOA фамилије; 4) *OptConfSOA*решењедобијено оптимизацијом, а на основу претходне теоретске анализе ефикасности предложеног приступа.

Пето поглавље представља детаље имплементације базиране на Eclipse платформи. Имплементација укључује представљање разних врста захтева, и

развијеног CS-AHP алгоритма за њихову анализу и рангирање расположивих одлика у моделу одлика (енгл. *feature model*).

Шесто поглавље садржи симулационе анализе развијених метода/приступа. Прво се наводе резултати анализа развијеног CS-AHP алгоритма са становишта неконзистентности које могу бити идентификоване при насумичном креирању двослојне структуре и захтева над њима, на основу чега се дају препоруке за пожељну структуру модела која значајно смањује неконзистентности. Други део симулација се односи на карактеристике креираних интегралних приступа (оптимизованог и неопимизованог) у области фамилије пословних процеса. Како се хеуристичким приступима добија само приближно решење, симулације су извршене с циљем одређивања оптималности предложених решења, њиховог поређења и утврђивања да ли су извршене оптимизације утицале на смањење оптималности решења које се предлаже и, на крају, утврђивање зависности решења од карактеристика модела.

Последње поглавље даје критички осврт на допринос остварен предложеним решењима, идентификује могућности примене у разним доменима и могуће правце за даље истраживање и надградњу предложених решења.

2. Преглед досадашњих резултата истраживања

Ово поглавље даје преглед литературе у одговарајућим областима и уводи основне концепте који су неопходни за разумевање појмова и концепата који се уводе и користе у наставку дисертације. Стога, у овом поглављу су описане основе сервисно-оријентисаних архитектура и њихових фамилија, нефункционалних захтева и метода за њихову анализу, као и семантичких технологија за представање знања и метода вештачке интелигенције за решавање оптималних задатака.

2.1.Сервисно-оријентисана архитектура и фамилије

Основни концепти сервисно-оријентисане архитектуре (енгл. *Service Oriented Architecture - SOA*) датирају још из 1970.године[1]. SOA побољшава и проширује флексибилност ранијих интеграционих метода (енгл. *Enterpise Application Intergation* – EAI) и дистрибуираних архитектура, и фокусира се на поновно коришћење постојећих апликација и система, ефикасну интероперабилност и интеграцију апликација, као и композицију пословних процеса преко сервиса одређених функционалности.

Модерни развој софтверских решења иде корак даље и намеће захтеве за подршком истоветности и варијабилности софтверских компоненти, чиме се омогућава вишеструка употребљивост доменски специфичних производа. Омогућавањем варијабилности појединим елементима и идентификацијом истоветности међу SOA архитектурама, дефинишу се целе SOAфамилије.

У наставку овог поглавља се дају основни концептиSOA и SOA фамилија, начини за њихово моделовање, као и квалитети сервиса чије карактеристике директно утичу на карактеристике појединачних инстанци SOAархитектура и целе фамилије.

2.1.1. SOA дефиниција

SOA је појам који најчешће означава инфраструктурни модел система чија је функционалност заснована на интероперабилним „услугама“, тј функционалним елементима који интерагују кроз заједнички дефинисане норме. Егзактна дефиниција не постоји, Tsai [2]наводи да SOA представља само нову парадигму; Erl[3]да је реч о моделу са карактеристикама и смерницама које треба поштовати у процесу имплементације; док премаDuane et al.[4],SOA представља стриктно дефинисани скуп технологија које је неопходно посматрати заједно и само као такве представљају SOA.

Thomas Erl [5] у својој књизи о SOA-и дефинише да „[SOA] успоставља модел архитектуре који има за циљ да побољша ефикасност, агилност и продуктивност предузећа од позиционирања сервиса као основних чинилаца преко којих је представљена логика решења у циљу реализације стратешких циљева повезаних са сервисно-оријентисаним рачунарством“.

С друге стране, Raghu R.Kodali[6], Oracle програмер који је уједно и аутор многих чланака о SOA-и, дефинише да „SOA представља еволуцију дистрибуираног рачунарства заснованог на захтев/одговор парадигми дизајна за синхроне и асинхроне апликације. Пословна логика апликације или појединачних делова је моделована и представљена као сервиси у потрошач/клијент апликацијама. Кључ система је својство слабе повезаности сервиса, тј. независност интерфејса сервисаодњихове имплементације. Апликације се могу израђивати повезивањем једног или више сервиса без знања о њиховој конкретној имплементацији“. Ова дефиниција наглашава неколико кључних карактеристика SOA које се најчешће наводе у свим дефиницијама: модуларност, дистрибуираност и слаба повезаност.

Логика SOA организације осликава парадигму која се заснива на теорији софтверског инжењерства познатој под називом *раздвајање интереса (*енгл. *separation of concerns)*. Укратко, ова теорија заступа сепарацију већег проблема на мање проблеме, тзв интереси (енгл. *concerns*), које је једноставније решити. Тиме се решење логички дели на мања решења, тј. мање целине које су дизајниране тако да свака појединачно реши по један од појединачних проблема. Мања

решења која су међусобно повезана се могу груписати у веће целине које представљају логичке делове целокупног решења.

Ова теорија није нова [7];развијана је и имплементирана на различитим платформама, као нпр, објектно-оријентисано програмирање [8]и програмирање засновано на компонентама [9], где се подразумева декомпозиција проблема кроз употребу објеката, класа и компоненти. За разлику од овог традиционалног објектно-оријентисаног приступа, сервисна-оријентација се сматра другачијим приступом за раздвајање интереса [3] која се огледа у раздвајању интереса на начин који одговара појединим јединицама логике решења [10]. SOA стога тежи да обједини претходно развијене приступе као што су модуларно програмирање, поновна употреба кода и објектно-оријентисане технике развоја софтвера [11].

Примена сервисно-оријентисане логике води ка дефиницији појма сервиса као јединице функционалности. Сервиси се сматрају „скупом аутономних јединица независних од платформе које могу бити описане, објављене, откривене, динамички компоноване и програмиране употребом стандардних технологија за развој колаборативних апликација“ [5]. Надоградњом речју „архитектура“, значење овог појма добија технички смисао, тј даје се на знање да се ради о моделу који је развијен на претходно наведеном принципу. Логичке јединице, тј. сервиси, су у својој функционалности аутономне али нису међусобно изоловане; гледано у целини оне чине један организовани дистибуирани систем широко дефинисане функционалности, тј својеврсну архитектуру [5]. Јендом речју, SOA парадигма омогућава реализацију софтвера као сервиса (енгл. *Software-as-a-Service (SaaS)*) [12].

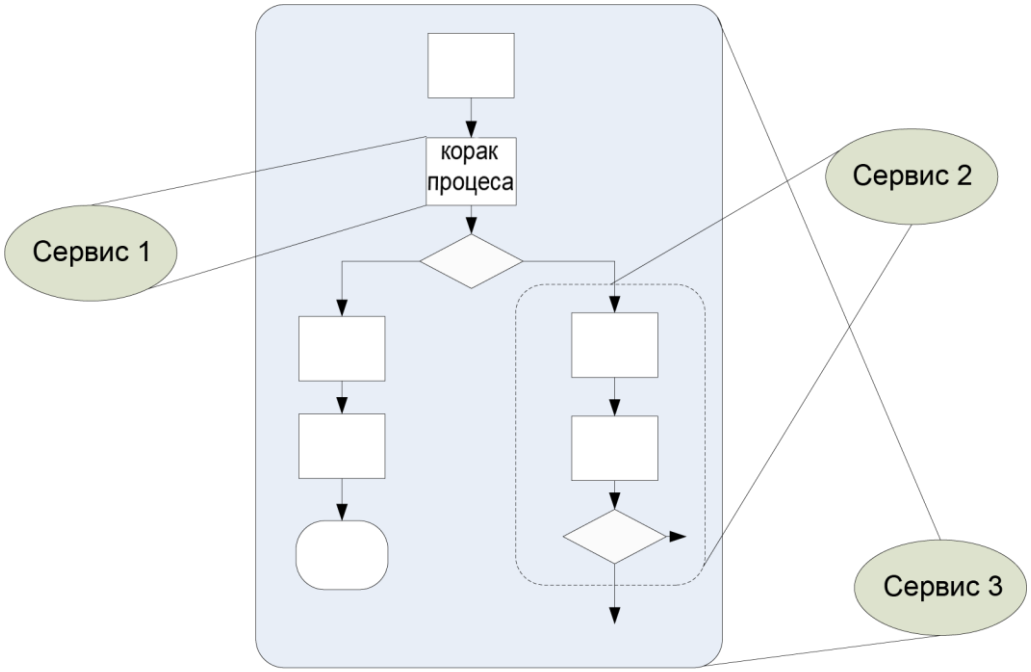
SOA сервиси се логички представљају као пар сервисни интерфејс и имплементација сервиса. Сервисни интерфејс дефинише идентитет сервиса и логистику позивања; док имплементација сервиса даје саму имплементацију функционалности за коју је дефинисано да ће је сервис пружити. Како су интерфејси независни од платформе, са било којег комуникационог уређаја уз употребу било које рачунарске платформе, оперативног система и програмског језика, се може користити сервис [11][13].

2.1.2. SOA принципи

Опсег услуге сервиса[14],тј ширина његове функционалности може бити разноврсна; сервиси могу обављати један мали, строго дефинисани део пословног процеса али може и обављати широко дефинисане пословне задатке по потреби уз помоћ других, уже дефинисаних сервиса што за крајњег корисника може бити потпуно транспарентно.

Пример процеса заснованог на сервисима различито дефинисаног опсега услуга је представљен на Слици 1. Опсег појединe услуге дефинисан је у зависности од контекста и улоге тако дефинисане услуге унутар пословног процеса. Нпр 'Сервис 1' обавља један уско дефинисани сегмент пословног процеса на најнижем нивоу декомпозиције. 'Сервис 2' такође обавља одређену функцију, али део процеса који она обавља је заправо потпроцес који се састоји од више сегмената. На крају, 'Сервис 3' је широко дефинисана услуга која енкапсулира цели процес и која заправо апстрахује постојање сервиса 'Сервис 1' и 'Сервис 2', па их за корисника сервис 'Сервис 3' може учинити скривеним. Корисник који посматра процес с највишег нивоа не мора бити свестан сервиса „нижег слоја“ с обзиром да њега занима само пословни модел сервиса (тј опсег услуга) које користи, тј резултат обраде информација које он пружа.

Стога, сервиси се морају налазити у посебно дефинисаним односима па често морају бити свесни постојања других сервиса. Корисници сервиса такође морају имати на располагању механизам помоћу кога могу добити детаље о одређеној услузи којој желе приступити и чија функционалност им је потребна. Сазнање о другим сервисима се врши кроз тзв опис сервиса (енгл. *service descriptions*) који садржи информације о називу сервиса, његовој локацији као и о неопходним условима за успостављем комуникације.



Слика 1. Услуге различито дефинисаног опсега с гледишта пословног процеса

Принципи дизајна сервиса и порука међу њима представљају јако важан сегмент SOA концепта. Неки од тих принципа су:

- ***Слаба повезаност***(енгл. *Service Loose Coupling*)- као што је већ наведено, реч је о успостављању таквог типа односа између самих сервиса, као и сервиса и корисника, који минимизује међузависност и нужно захтева само свесност о међусобном постојању.
- ***Уговор о комуникацији***(енгл. *Standardized Service Contract*)-сваки сервис мора се прилагодити дефинисаним нормама тј. уговору о комуникацији чији се детаљи могу описати на одређени начин у опису услуга.
- ***Аутономија***(енгл. *Service Autonomy*)- сервиси имају апсолутну контролу над пословном логиком коју садрже.
- ***Апстракција***(енгл. *Service Abstraction*)- све информације које сервис пружа о себи налазе се у опису услуга; сама имплементације је сакривена од спољашњег света.
- ***Поновна искористивост***(енгл. *Service Reusability*)- сервис може бити део већег бројапословних процеса, а његова функционалност не мора бити

прилагођена само посебно одабраној прилици, већ се као универзална, према потреби може поново користити.

- ***Композиција***(енгл. *Service Composability*)– скупови сервиса се могу декомпоновати у пословне процесе и/или шире дефинисане сервисе.
- ***Непостојање стања***(енгл. *Service Statelesness*)– сервис мора минимизовати информације које су специфичне за поједине случајеве употребе.
- ***Могућност проналажења сервиса***(енгл. *Service Discoverability*)– сервиси се креирају тако да се могу описати преко дефинисаних механизама како би се информације о њима могле на адекватан начин пронаћи,а сервисима на основу њих и приступити.

Може се уочити да наведени принципи нису међусобно независни и да постоји висока корелација међу њима. Принцип ***Композиције сервиса*** се може сматрати централним принципом, с обзиром да имплицира остале, што је представљено на Слици 2, а везе импликације 1-7 се објашњавају не следећи начин:

(1) разматрања која произилазе из декомпозиције сервиса захтевају већи ниво дефиниције правила и норми дизајна како би били испуњени захтеви декомпозиције;

(2) у циљу максимизације потенцијала декомпозиције сервиса, неопходно је да њихова имплементација буде флексибилна и самостална што је више могуће;

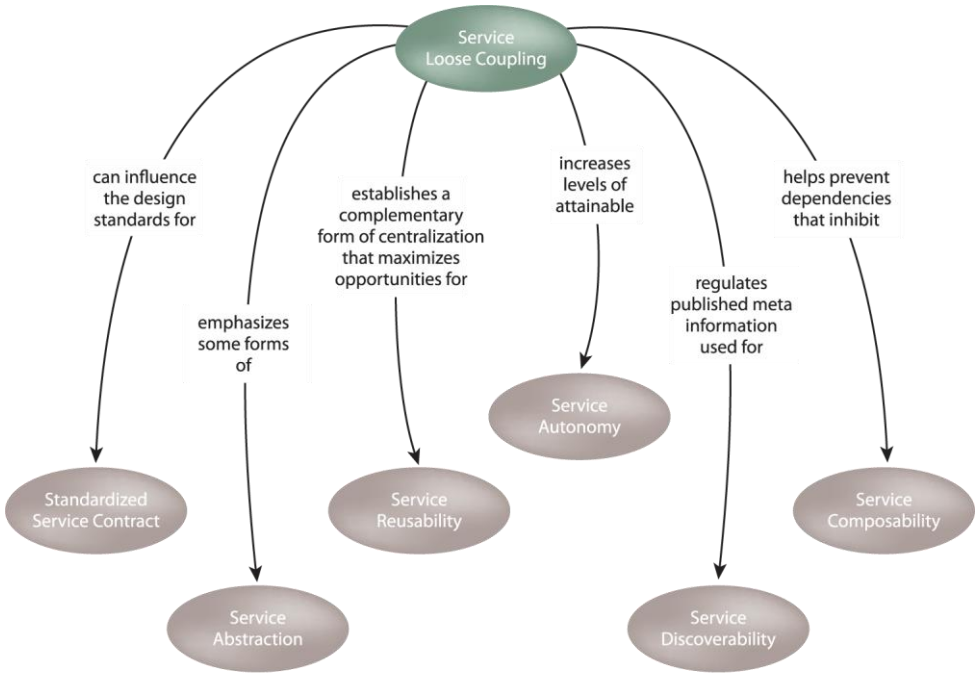
(3) при дизајнирању комплексних композиција сервиса, природно је очекивати да се објави комплетна перспектива на све сервисе који су укључени у композицији. Ипак, могуће је да се појави потреба/жеља да се информације о неким од сервиса желе сакрити, што води ка концепту *сакривених елемената композиције;*

(4) теоријски, принципи композиције сервиса и поновне искористивости могу постојати независно један од другог. Очекује се да су композитни сервиси, иако не постоје у самом тренутку имплементације, имају карактеристике као и композициони сервиси, чиме се даје проширено значење појму поновне искористивости;

(5) неопходно је жртвовати аутономију сервиса када они учествују у композицији. Стварна вредност аутономије композитног сервиса је еквивалентна комбинованој мери аутономија свих сервиса који су укључени у композицију;

(6) у циљу обезбеђења ефективне композиције сервиса, неопходно је да информације о стањима сервиса буду ефективне и конзистентне (тј. скоро независне од појединих случајева употребе)

(7) неопходно је идентификовати могућности композиције у облику објављивања енкапсулиране композиционе логике.



Слика 2. Принцип Композиције сервиса као централни принцип SOA[5]

Аутоматизацији реализације принципа композиције сервиса (као централног принципа SOA архитектуре) је у литератури дата изузетно велика пажња, где се са различитих аспеката[3] [1] [15], у области веб сервиса, анализира могућност конструкције функционалности сервиса и одабира одговарајуће имплементације. Међутим, у овој дисертацији се дати приступ ослобађа ограничења разматрања само веб сервиса, па се под сервисом сматра било која компонента (која у општем случају не мора бити софтверска) која обезбеђује тражену функционалност, а

окарактерисана је у складу са претходно наведеним принципима.Као потврда оправданости оваквог приступа, наводи се технолошканезависност SOA-е:SOA принципи су генерички и за одређену функционалност је потребно одабрати конкретну платформу за имплементацију, а технологија веб сервиса представља једну од најраширенијих и најчешће примењивих платформи.

2.1.3. Сервисно-оријентисани дијаграм тока

Композиција сервиса подразумева идентификацију хијерархије сервиса, а један од начина за њено представљање је у облику оркестрације (енгл. *orchestration*). Идеја оркестрације подразумева да се одговорност око координисања извршења композитног сервиса додели јединственом ентитету (оркестратору). Оркестратор (енгл. *orchestrator*) је задужен за прихватање улазних захтева и интеракцију са сервисима који чине композицију (тзв. компонентни сервиси) како би се добијени захтев испунио. Интеракција између оркестратора и компонентних сервиса се представља у облику орекстрационог модела који обично има форму пословног процеса у којем сваки задатак представља или неку интерну активност (нпр. трансформација података) или интеракцију са компонентним сервисом.

У пракси, модели процеса се представљају помоћу специјализованих језика као што су Business Process Execution Language (WS-BREL) и Business Process Modeling Notation (BPMN). Помоћу ових језика се модел процеса представља у облику радног тока (енгл. *workflow*) [15]који описује активности пословног процеса на концептуалном нивоу као и дијаграме тока и информација међу активностима. Он описује аутоматизацију пословног процеса у целини или деловима помоћу елемената који представљају логичке кораке, познате као задаци (енгл. *task*) или активности (енгл. *activity*), зависности међу задацима и активностима, правилима рутирања и учесницима. У радном току, задатак може представљати елементарну активност или неки софтверски систем.

Наведено разматрање је истраживачима већ наметнуло употребу модела процеса за подршку и менаџмент веб сервиса[16] [17] [18].По аналогији, један

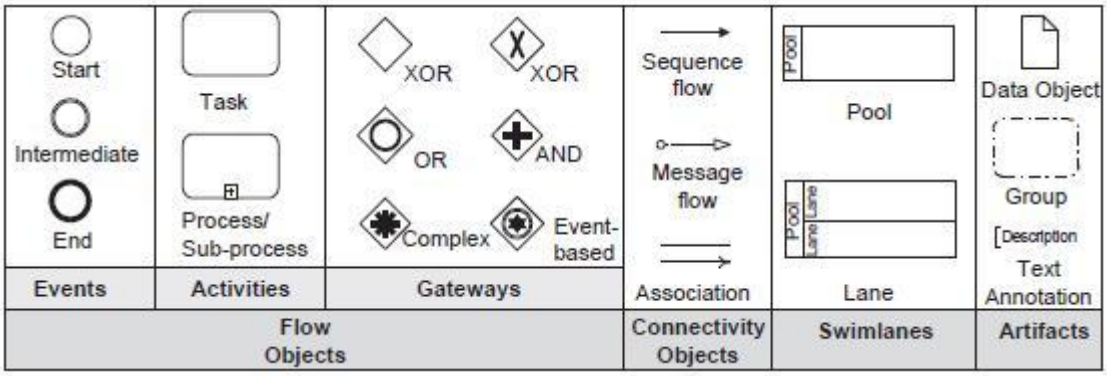
задатак модела процеса се представља као атомични сервис SOA архитектуре, односно активност модела процеса као композитни сервис чији се начин декомпозиције управо представља током који одговара датој активности. Истоветан приступ се користи у овој дисертацији за наведени шири контекст употребе појма сервиса и SOA архитектуре. На тај начин се, једним задатком у пословном току сматра било који систем (који ради општости не мора бити софтверски) дате функционалности, а активности се идентификују у складу са објашњењем опсега услуга сервиса (претходно илустрованим наСлици 1).

Сходно наведеној јакој повезаности међу управљањем пословним током и композицијом сервиса [19], постојећи језици за ток композиције сервиса базирани су на аналогним језицима за моделовање пословних токова. За овакав начин моделовања се као предност истиче могућност директног поређења композиција сервиса у односу на обезбеђену функционалност, пре него на нефункционалне карактеристике, које ће бити анализиране у поглављу 2.1.4. У овој дисертацији се за представљање користити BPMN нотација, па су у наставку представљени њени основни елементи, основни обрасци пословног тока као и кореспондентни обрасцикомпозиције.

2.1.3.1.BPMN нотација

УBPMN нотацији, пословни ток је представљен у облику дијаграмакоји је састављен од скупа графичких симбола и елемената. Они омогућавају лак развој једноставних дијаграма, и имају интуитивно јасну интерпретацију. Основни симболи BPMN нотације су сумарно представљени наСлици 3.

За представање понашања и управљање пословним током користе се објекти тока и објекти веза, помоћу којих се и конструишу одговарајући обрасци (енгл. *patterns*) у пословном току, на основу којих ће се касније креирати и одговарајући обрасци (тј правила) декомпозиције сервиса. У наставку се наводе кратки описи објеката тока и објеката веза, као и основни обрасци тока.



Слика 3. BPMN нотација, основни симболи [20]

2.1.3.1.1. Објекти тока

Објекти тока представљају основне графичке елементе за дефинисање понашања пословних процеса. Објекти тока су *Event*, *Activity*и*Gateway*елементи. *Event* елемент представља нешто што се дешава током извршавања пословног процеса, утиче на извршавање тока процеса и обично има узрок (окидач) и последицу (резултат). *Event*елементи су категорисани по фази у којој се дешавају у процесу (*Start*, *Intermediate*и*End*) и по типу (*Basic*, *Message*,*Timer*, *Rule*, *Exception*, *Cancellation*, *Compensation*, *Link*, *Multiple*, *Termination*).

*Activity*елемент представља општи појам за посао који се обавља, а може бити прост или сложен. Сложена активност, која се назива и процес, такође може имати свој скуп простих или сложених активности, као и друге BPMN елементе.

*Gateway*елемет представља главни начин законтролу токова у оквиру BPMN дијаграма.*Gateway Exclusive OR* елемент, који се базира на подацима, користи контролне структуре *if-then-else* и *switch* са међусобно искључивим случајевима; у ситуацији када се врши гранање у оквиру процеса, тачно један услов мора бити испуњен. Други тип *Gateway Exclusive OR* елемента, који је базиран на догађају, користи *pick* контролну структуру. У ситуацији када се врши гранање у оквиру процеса, свака грана води до једног чвора са догађајем. Контрола пролази кроз грану где се окида први догађај, а све остале гране се игноришу. *Gateway Inclusive OR* елемент је сличан *Gateway Exclusive OR* елементу, који је базиран на

подацима, изузев што дозвољава пролазак кроз сваку грану *switch* контролне структуре, чији услов је испуњен. *Gateway Complex* елемент се углавном користи код спајања путева у оквиру процеса, када се проверава дефинисани излаз да би се одредила путања којом ће се наставити извршавање процеса. *Gateway Parallel* елемент користи *flow* контролну структуру, и код гранања у оквиру процеса, дозвољава пролазак кроз сваку од дефинисаних путања. Код спајања, блокира наставак извршавања процеса, све док се не изврше све дефинисане гране.

2.1.3.1.2. Објекти веза

Објекти тока се међусобно повезују коришћењем објеката веза. Постоје три основна типа елемената који обезбеђују ову функцију:

- *Ток секвенце* (енгл. *sequence flow*) која се користи за дефинисање редоследа извршавања активности у оквиру процеса.
- *Ток поруке* (енгл. *message flow*) која се користи за представљање тока порука између два одвојена учесника у процесу који примају и шаљу поруке.
- *Ток асоцијације* (енгл. *association*) којим се повезују подаци, текст и други артифакти са објектима тока; овим елементима се представљају улазни и излани подаци из активности.

2.1.3.1.3. Обрасци пословног тока

Употребом дефинисаних објеката тока и веза, добијају се основни обрасципословног тока. У складу са препорукама датим у [21] [20], за сваки образац се даје дефиниција, опис, најчешће употребљавани синоними и графички симбол(и).

Образац 1: Секвенца (енгл. ***Sequence***)

Образац секвенце се користи за представање узастопних корака у пословном току. За сваку активност (атомичну или сложену) у секвенци важи да је омогућена за извршавање након завршетка извршавања активности која јој директно претходи.

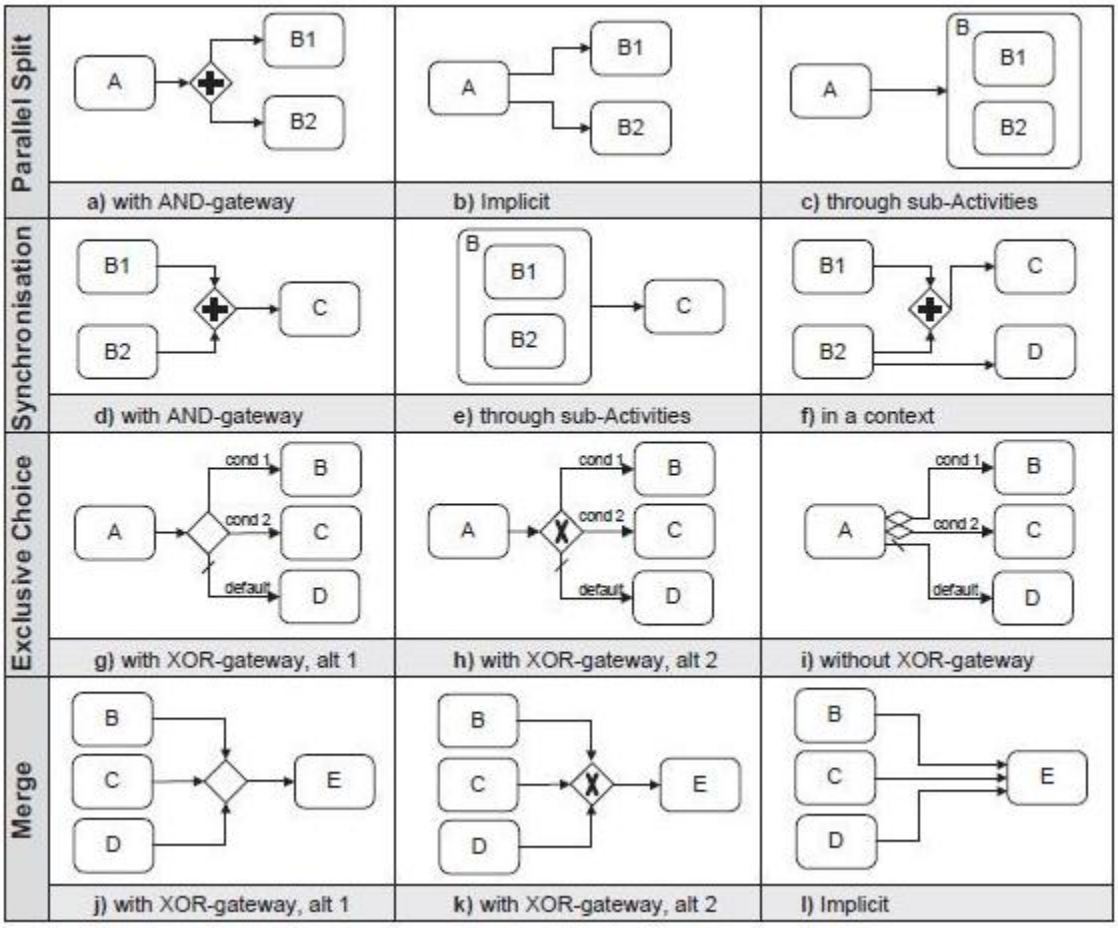
Синоними: Секвенцијално рутирање, серијско рутирање
Графичка презентација: две активности повезане секвенцијалном везом тока

Следећа два обрасца се користе за представљање паралелног рутирања што значи да се одговарајуће контроле тока извршавају у паралели. Према томе, логичка веза која се користи за њихово објашњење и представљање је логичко AND.

Образац 2: Паралелно раздвајање (енгл. ***Parallel split***)
Овај образац се користи за представљање места у току пословног процеса где се јединствена нит тока раздваја на више нити котроле које се могу извршавати у паралели, чиме се дозвољава истовремено извршавање активности.
Синоними: AND-раздвајање, паралелно ритурање
Графичка презентација: за презентацију се користи *AND split Gateway* (Слика 4a). Такође, паралелно раздвајање може бити представљено имплицитно, помоћу веза тока директно међу активностима, чиме се избегава навођење *AND split Gateway*-а (Слика 4b). Трећи начин за представљање подразумева представљање паралелизма у извршавању активности у облику под-активности у датом процесу/потпроцесу (Слика 4c).

Образац 3: Синхронизација (енгл. ***Synchronization***)
Супротно од претходног обрасца који врши раздвајање јединствене нити тока, овај образац врши спајање више нити у једну. Стога, овај образац се користи за представљање места у току пословног процеса где више паралелних потпроцеса/активности конвергирају ка јединственој нити тока, чиме се врши њихова синхронизација. Претпоставка овог обрасца је да се свака улазна грана (тј нит) извршава тачно једном приликом уласка у синхронизатор (тј *AND join Gateway*).
Синоними: AND спајање, рандеву образац, синхронизатор
Графичка презентација: за презентацију се користи *AND join Gateway* (Слика 4d). И у овом случају, за представљање се може користити приступ заснован на под-активностима (Слика 4e). Међутим, овакав приступ се не може користити у

сложеним ситуацијама, нпр Слика 4f где само једна од наредних активности захтева синхронизацију.



Слика 4. Основни обрасци пословног тока [20]

Наредна два обрасца се користе за представљање условних рутирања (тј гранања). За разлику од паралелних рутирања, само једна од нити гранања може бити одабрана и активирана. Стога, логичка веза која се користи за објашњавање и представљање је логичко XOR.

Образац 4: Екслузивни одабир (енгл. ***Exclusive choice***)

Овај образац се користи за представљање места у току пословног процеса где се на основу контролних података доноси одлука о одабиру једног тока од понуђених.

Синоними: XOR-раздвајање, условно рутирање, switch

Графичка презентација: за презентацију се користи *XORsplit Gateway* (Слика 4g и Слика 4h). Могуће је један од токова прогласити подразумеваним и у том случају је загарантовано да, без обзира на дефинисани захтев гранања тачно један излазни ток ће бити одабран. Понашање екслузивног одабира може бити представљено и помоћу више веза тока за које се дефинишу вредности атрибута *ConditionType и ConditionExpression* (Слика 4i), али је на дизајнеру да обезбеди еквиваленцију са логичкимXOR оператором.

Образац 5: Једноставно спајање (енгл. ***Simple merge***)

Овај образац се користи за спајање нити под претпоставком непостојања паралелног извршавања алтернативних нити. Стога, користи се у местима у току пословног процеса где се две или више алтернативних нити спајају без претходне синхронизације.

Синоними: XOR-спајање, асинхронизационо спајање

Графичка презентација: за презентацију се користи *XOR join Gateway* (Слика 4j и Слика 4k) или пак, по аналогији са примерима претходних обрасца, без употребе Gateway-а, као што је приказано на Слици 4l.

Следећи обрасци су чести при имплементацији реалних сценарија и фокусирани су на напредније гранања и синхронизације.

Образац 6: Вишеструки избор (енгл**. *Multi-choice*)**

Насупрот патерну експлузивног избора, који подразумева да се врши одабир тачно једне алтернативе од наведених, овај образац омогућава да се одаберу више алтернатива које ће бити извршене у паралели. Стога, логичка веза која се користи за објашњавање и представљање је логичко OR.

Синоними: условно рутирање, селекција, OR-гранање

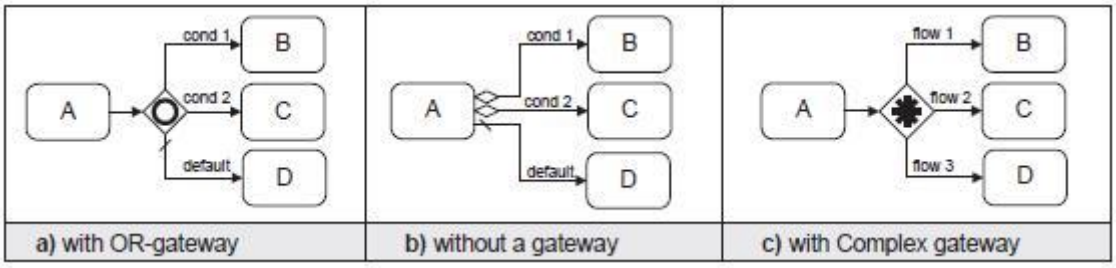
Графичка презентација: Постоје три начина за графичко представљање овог обрасца:

а) помоћу *ORsplit Gateway*-а (Слика 5a). На дизајнеру је да обезбеди да ће у сваком тренутку најмање један од дефинисаних услова бити испуњен;

алтернатива је да се дефинише подразумевани ток који ће бити извршен у случају да ни један од услова није испуњен;

b) без употребе *Gateway* елемента (Слика 5b). За разлику од имплементације представњене на Слици 4и, условни изрази за разне токове не морају бити међусобно искључиви;

c) помоћу *Complex Gateway*-а (Слика 5c) где се логичким изразима дефинишу правила за одабир сваког од токова у различитим могућим ситуацијама.



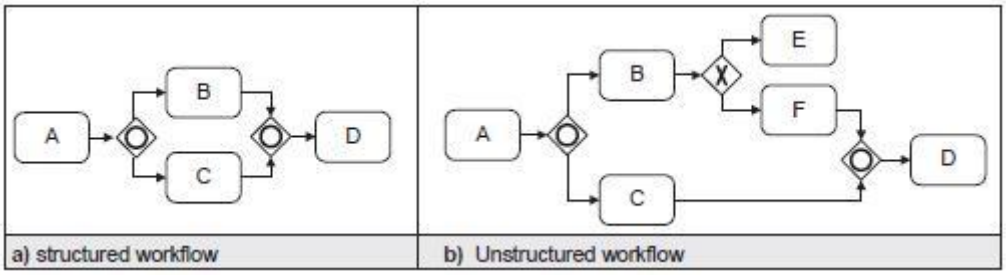
Слика 5. Обрасци вишеструког избора [20]

Образац 7: Синхронизационо спајање (енгл. ***Synchronizing merge***)

Овај образац се користи на местима у току пословног процеса где две или више нити конвергирају ка једној нити. Уколико се две или више нити извршавају, неопходно је прво урадити синхронизацију. Међутим, овај образац подразумева ограничење да, уколико је једна нит активирана, не може бити поново активирана док не сачека остале нити који су активне да заврше са извршавањем.

Синоними. Синхронизациони приступ

Графичка презентација. За презентацију се користи *OR join Gateway*. Решење представљено на Слици 6а подразумева структурирани пословни ток. Међутим, уколико се овај образац појављује у неструктурираном току, као што је представљено у примеру датом на Слици 6b, ток може доћи у ћорсокак јер произведена активност E неће бити поново спојена у *OR join Gateway*, као што се очекује.



Слика 6. Обрасци синхронизационог спајања [20]

Образац 8: Вишеструко спајање (енгл.***Multi-merge***)

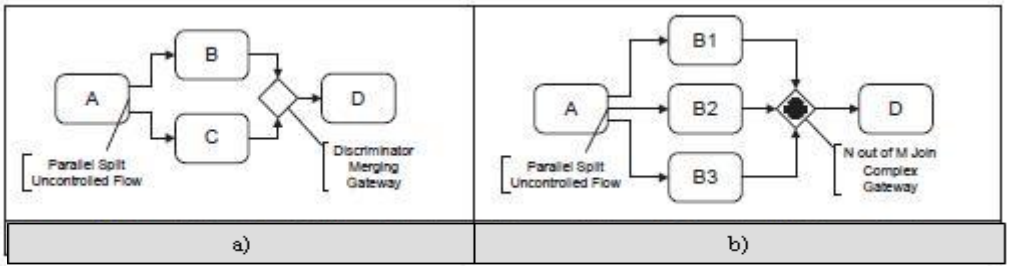
Овај образац описује специфичан оператор спајања паралелних нити на месту, у току пословног процеса, где стижу до заједничке јединстевене нити. Уколико је активно више од једне нити, могуће и конкурентних, активност која следи након спајања се активира посебно за сваку долазећу нит. Стога, овај образац се имплементира задржавањем различитих извршавања у паралели без синхронизације, па се може имплементирати помоћу *AND split*и *AND join* оперција за агрегацију.

Графичка интепретација. На исти начин као иобразац једноставног спајања представљен на Слици 4j, k и l.

Образац 9. Образац дискриминатор (енгл. ***Discriminator***)

Дискриминтор је месту у току пословног процеса на којем се чека да се једна од улазних грана (нити) изврши пре активирања наредних активности. Након што све улазне гране буду извршене, образац врши ресетовање самог себе, тако да може бити активиран поново (што је јако важно јер у супротном не би могао бити коришћен у својству петљи).

Графичка презентација. Дискриминатор је специјална врста *N-out-of-M* Joinобрасца, где је *N=1*. *N-out-of-M* Joinобразацсе заснива на могућности синхронизације тока након извршења *N*паралелних нити од укупно *M*могућих. Начини за представљање обрасца доскриминатора и *N-out-of-M* Joinобрасца су дати наСлици 7.



Слика 7. Образац дискриминатор [20]

Образац 10. Произвољни циклуси (енгл. ***Arbitrary cycles***)
Овај образац омогућава вишеструко извршавање дате активности.
Графичка презентација: За представљање се користи циклични представљена веза секвенце са могућношћу назнаке броја понављања.

Наведени обрасци су само примери за које су конструисани еквиваленти који се користе за моделовање декомпозиције сервиса као централног дела разматрања SOA архитектура у овој дисретацији. Проширена евалуација контрола тока у BPMN нотацији је дата у [20].

2.1.3.2.Обрасци композиција сервиса

Као што је предложено у раду [22], обрасци тока се могу користити као мотивација за увођење *апстрактних композиционих обрасца* (енгл. *abstract composition patterns*) који представљају основни структурни елемент композиције. Композициони обрасци се деле у две основне групе: 1) *секвенцијалне патерне* и 2) *паралелне и условне патерне*, који одређују како се ток процеса одвија у секвенци, како се секвенца дели у гране и како се оне касније спајају.

При дефиницији паралелних и условних обрасца, није примењен једноставан приступ аналогије са релевантним обрасцима тока. Истраживања проблема одређивања нефункционалних карактеристика композиционих сервиса (која ће бити анализирана у секцији 2.1.4.) су наметнула представљање композитних сервиса помоћу добро-стуктурираног оркестрационог модела, тј. модела који се може представити у облику графа. Такви модели, осим чворова који представљају једничне компонентне сервисе, садрже и чворове спајања и гранања, тако да за

сваки чвор гранања постоји одговарајући чвор спајања помоћу којег дио графа између гранања и спајања представља подграф са јединственим улазом и јединственим излазом (енгл. *single-entry-single-exit (SESE)*).

Оба поменута језика за моделовање, WS-BREL и BPMN, дозвољавају конструкцију неструктурираног модела оркестрације.Стога, дефинишу се следећа *правила добре структурираности графа*[23]:

i. Модел оркестрација има један почетни чвор (тј. чвор без улазних грана) и један завршни чвор (тј. чвор без излазних грана) и сваки чвор се може обићи на путу од почетног до крајњег чвора.
ii. Сваки атомични чвор има само једну улазну и једну излазну грану, а сваки gateway-елемент је или гранање или спајање. Ако овај услов није задовољен, низом једноставних трансформација, модел оркестрације се трансформише у модел који задовољава дефинисане услове.
iii. Сума вероватноћа додељених излазним гранама из XOR-гранања је 1;
iv. Свака грана чији изворни чвор није XOR-гранање има вероватноћу 1, што значи да се њоме увек врши обилазак уколико је досегнут почетни чвор из којег грана излази.

Уколико се пак, по аналогији са обрасцима пословног тока користе неки од алтернативних начина за представљање (нпр без употребе *Gateway* елемената), правило (ii) дефинише неопходност трансформације тако дефинисаног тока.

Примери композиционих образацакоји су релевантни за проблем композиције сервиса у фази моделовања SOA архитектуресу представљени на Слици 8, а анализа њихове релевантности за проблем композиције сервиса је дата у [22]:

CP_1- секвенца

CP_2- петља са датим бројем понављања

CP_3- XOR гранање праћено са XOR спајањем

CP_4- AND гранање праћено са AND спајањем

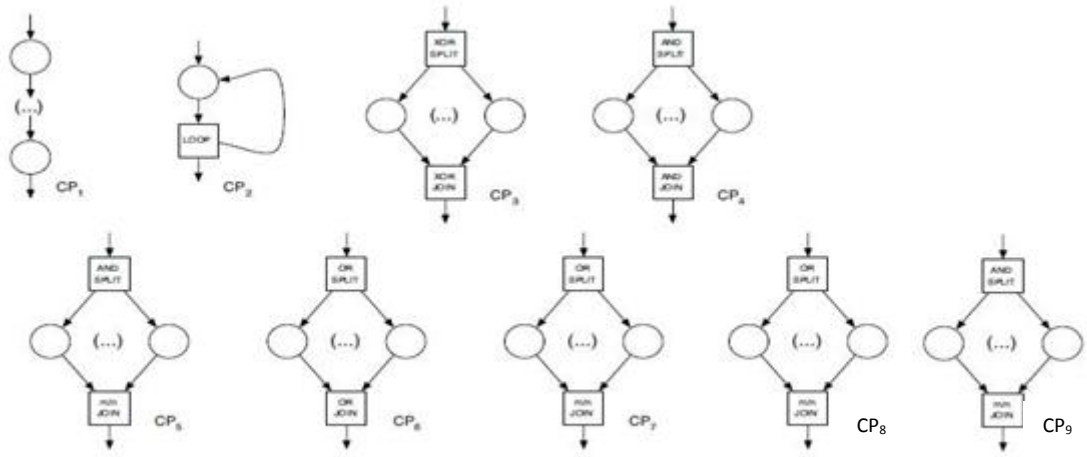
CP_5- AND гранање праћено са *m-out-of-n*спајањем

CP_6- OR гранање праћено са OR спајањем

CP_7- OR гранање праћено са *m-out-of-n*спајањем

CP_8- OR гранање праћено са XOR спајањем

CP_9- AND гранање праћено са XOR спајањем



Слика 8. Композициони обрасци[22]

На крају, SOAдијаграм тока, сагледана као композитни сервис се може формално дефинисати на следећи начин, при чему се додатно намеће испуњење услова *(i)-(iv)*:

Дефиниција 1 (Сервисно оријентисани дијаграм тока). ***Сервисно-оријентисани дијаграм тока*** је граф $G_{PM} = (\nu, \varepsilon)$, где $\nu = \{n_\varepsilon, \nu_\sigma, \nu_A, \nu_G\}$ представља скуп међусобно дисјунткних чворова: n_ε - јединствени почетни чвор, ν_σ - јединствени завршни чвор, ν_A -скуп активности и ν_G - скуп gateway-елемента. Скуп $\varepsilon \subseteq \nu \times \nu$ представља скуп грана међу чворовима.

Током процеса декомпозиције, одређена функционалност која треба да буде задовољена се дефинише као *апстрактнисервис* (или *интерфејс*). На овај начин, под моделовањемсервисно-оријентисаног дијаграма тока се подразумева моделовање декомпозиције апстрактног сервиса који представља функционалност целе архитектуре, до нивоа елементарних сервиса.

2.1.5. Нефункционалне карактеристике сервиса

Током процеса декомпозиције сервиса није неопходно да једном елементарном интерфејсу одговара тачно један сервис који га имплементира, шта више, јако честа је ситуација да постоји читав скуп сервиса дате функционалности, нпр сервиси за резервацију хотелског смештаја или сервиси за проверу дневне температуре. У већини случајева, мерљива разлика међу датим сервисима се директно односи на њихове нефункционалне карактеристике, познате под називом атрибути квалитета сервиса (енгл. *Quality of Service (QoS) attributes*). Међународни стандард квалитета ISO 8402 (који је дио ISO 9000 стандарда [24]) QoS атрибуте дефинише као „*укупност карактеристика производа или сервиса који се односе на његову способност да задовољи наведене или имплициране потребе*“.

Идентификација скупа нефункционалних карактеристика представља предмет истраживања у многим доменима, с обзиром да сервисе и пословне токове карактерише јако широка и добра применљивост. Овом проблему се приступа са различитих становишта, а најчешћа су: операциони менаџмент организације и квалитет сервиса за потребе софтверских система[16].

С организационе стране, Stalk и Hout [25], и Rommel [26]су анализирали карактеристике које утичу и одређују успех компанија на светском тржишту. Резултати анализе показују да су то: време, цена и квалитет, што одговара реалним захтевима организација и чињеници да се највише пажње поклања управо њима. Међутим, Garvin[27]придружује осам додатних димензија квалитета, укључујући перформансе и поузданост. С друге стране, квалитети сервиса са становишта софтверских система су анализирани и проучавани у различитим доменима примене. Највећи допринос је дат у области мрежа [28] [29], апликацијама реалног времена [30], дистрибуираним системима[31] [32]. На пример, за дистрибуиране посредничке системе, Frolund и Koistinen[33]су представили практичне димензије за поузданост система и перформансе дистрибуираних објеката, које укључују време потребно за опоравак (енгл. *time to repair (TTR)*), време до отказа (енгл. *time to failure (TTF)*), доступност (енгл. *availability*), маскирање нерегуларног рада (енгл. *failure masking*) и отказ сервера

(енгл. *server failure*). С друге стране, у случају мрежа података[34], QoS карактеристике се генерално односе на доменски специфичне димензије, као што су пропусни опсег (енгл. *bandwidth*,), латентност (енгл.*latency*,), губици (енгл. *loss*), итд.

Постоји неколико модела карактеристика квалитета за софтверске системе. Један од најчешће применљивих, објављен стандрадом ISO9126[35], класификује карактеристике (и под-карактеристике) квалитета софтвера на следећи начин:

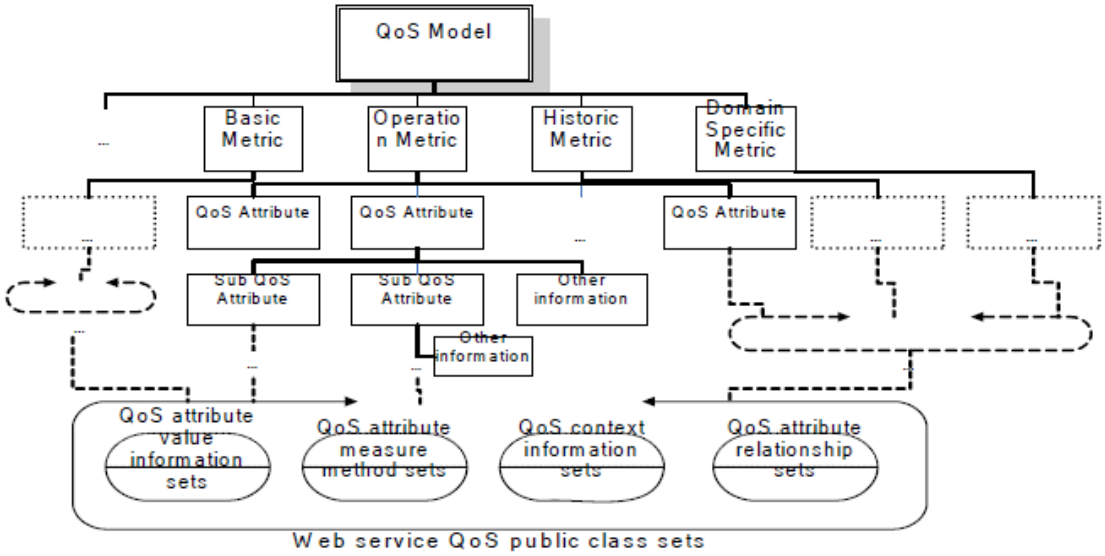
- **Функционалност** (енгл. *Functionality*) се дефинише као намена производа или сервиса представљена као скуп суштинских функција које дати софтверски производ пружа. Она се декомпонује на следеће под-карактеристике: погодност (енгл. *Suitability*), тачност (енгл. *Accuracy*), интероперабилност (енгл. *Interoperability*), усаглашеност (енгл. *Compliance*) и сигурност (енгл. *Security*).
- Када је задовољена функционалност софтверског система, карактеристика **поузданости** (енгл. *Reliability*) дефинише могућност система да одржи услуге које пружа под одређеним условима за дефинисани временски период. Ова карактеристика се декомпонује на: постојаност (енгл. *Maturity*), могућност опоравка (енгл. *Recoverability*) и толеранцију грешке (енгл. *Fault Tolerance*)
- **Употребљивост** (енгл. *Usability*) се односи на лакоћу употребе за дату функцију. Она се декомпонује на следеће под-карактеристике: лакоћа учења (енгл. *Learnability*), разумљивост (енгл. *Understandability*) и оперативност (енгл. *Operability*).
- **Ефикасност** (енгл. *Efficiency*)се односи на ресурсе система који се користе у циљу обезбеђења дате функционалности.Она се декомпонује на следеће под-карактеристике: утрошња времена (енгл. *TimeBehaviour*) и утрошња ресурса (енгл. *Resource Behaviour*).
- **Одрживост** (енгл. *Maintainability*)се односи на способност идентификације и отклањања грешке у оквиру саме софверске компоненте.Она се декомпонује на следеће под-карактеристике: стабилност (енгл. *Stability*), могућност анализе (енгл. *Analyzability*), променљивост (енгл.*Changeability*) и могућност тестирања (енгл. *Testability*).

- **Преносивост** (енгл. *Portability*) се односи на могућност софтвера да се прилагоди променама у окружењу или у самим захтевима. Ова карактеристика се декомпонује на: могућност инсталирања (енгл. *Installability*), замењивост (енгл.*Replaceability*), адаптивност (енгл.*Adaptability*) и потврду (енгл. *Conformance*).

Наведене области примене јасно доводе до закључка, да се у сваком домену(области) примене, скуп QoS атрибута посебно дефинише. Међутим, осим саме дефиниције, неопходно је дефинисати: метрике за сваки од дефинисаних атрибута као иначине мерења одговарајућих вредности. За неке нефункционалне карактеристике мере нису квантитативне (нпр. висок ниво сигурности), док је за неке могуће дефинисати више различитих мера.Тако на пример, за QoS атрибут доступност, могу се дефинисати следеће метрике: 1) просечно време међу периодима нерегуларног рада (недоступности); 2) просечна доступност, тј просечно време регуларног рада сервиса када је он доступан; 3) укупно време недоступности (тј нерегуларног рада) у току године.

Осим различитих метрика, квантитативних (целобројне, реалне вредности) и квалитативних, QoS атрибуте карактеришу и различите тенденције монотоности. Тако нпр, цена и време одзива су карактеристике које имају опадајућу тенденцију, тј.минимизација наведених QoS атрибута је од интереса свим учесницима у процесу развоја апликација и софтверских система. С друге стране, просечна доступност и поузданост су карактеристике које је потребно максимизовати, па се сматрају монотоно растућим карактеристикама.

На основу наведених карактеристика QoS атрибута, креиран је модел (представљен на Слици 9) који представља широк опсег скалабилних типова, који се могу односити и на различите области практичне примене [36]. Такође, на основу датих карактеристика, QoS атрибути веб сервиса обично се деле у четири групе истоветних карактеристика и тенденција, чији су представници *cost, response time, reliability*, и*availability,* па се сва разматрања односе на дате представнике. Истраживања су показала да се аналогни приступи и модели користе и при ослобађању ограничења посматрања веб сервиса, што одговара приступу који се користи у овој дисертацији.



Слика 9. QoS модел [36]

Осим модела и начина представљања атрибута квалитета сервиса, као додатни задатак се намеће процена вредности самих атрибута. Многа истраживања су фокусирана управо на решење овог задатка[16] [37].Најшире прихваћени приступ је комбинација процена добијених од стране самих дизајнера са вредностима које се добијају на основу мониторинга извршавања и анализом стохастичких података. На истоветан начин се врше процене и осталих елемената који карактеришу модел композиције сервиса:*(i)*вероватноће гранања на основу услова гранања; *(ii)*корелације (тј. зависности) међу индивидуалним сервисима, и *(iii)* број понављања (тј. итерација) петље.

Карактеристике квалитета сваког од атомичних сервиса омогућавају карактеризацију сервисног интерфејса који се њима имплементира, формално дефинисаним на следећи начин:

Дефиниција 2.*Класа сервиса* за задатак $a \in A$ (у ознациS_a)представља скуп сервиса дате функционалности који могу бити окарактерисани међусобно различитим вредностима нефункционалних карактеристика. ***Сервисни интерфејс*** обухвата класу сервиса дате функционалности окарактерисане параметром класе $\left(Q_a^{LB}, Q_a^{UB}\right) = \left(\left\langle Q_1^{LB},..,Q_k^{LB}\right\rangle, \left\langle Q_1^{UB},..,Q_k^{UB}\right\rangle\right) \in R^k \times R^k$ који представља доњу и горњу границу вредности сваке од нефункционалних карактеристика. Стога, сваки

атомични сервис ***s*** који припада датој класи сервиса реализује интерфејс, а вредности његових нефункционалних карактеристика задовољавају неједнакости:

$$Q_i^{LB} \le q_i(s) \le Q_i^{UB}, 1 \le i \le k.$$

2.1.6. Агрегација вредности нефункционалних карактеристика сервиса

На основу вредности атрибута квалитета атомичних сервиса, могуће је одредити дате вредности и у случају композитног сервиса, као агрегатне вредности сервиса који учествују у композицији [16].Агрегација вредности се врши једноставним рекурзивним обиласком структуре графа (која представља сервисно-оријентиснаи дијаграм тока) и применом правила агрегације за сваки од чворова. Претходно наведена правила добре структурираности омогућавају трансформацију структуре графа (која представља пословни ток) у структуру стабла.

У Табели 1 су представљена правила агрегације за основне патерне композиције и одговарајуће QoS атрибуте. У складу са наведеним груписањем атрибута истоветних карактеристика, наводе се само атрибути *cost, response time, reliability,* и*availability* као представници одговарајућих класа атрибута, при чему се не губи на општости и генералности приступа. Такође, како је корисник у могућности да дефинише неке од доменских карактеристика, тј. карактеристика које се директно односе на област примене и проблем који се решава, у последњем реду Табеле 1 се наводи општа функција *f*која одређује правила агрегације за такве класе атрибута. Тиме се обухвата и случај генералног приступа ове дисертације, када се уместо веб сервиса посматрају сервиси у општем смислу.

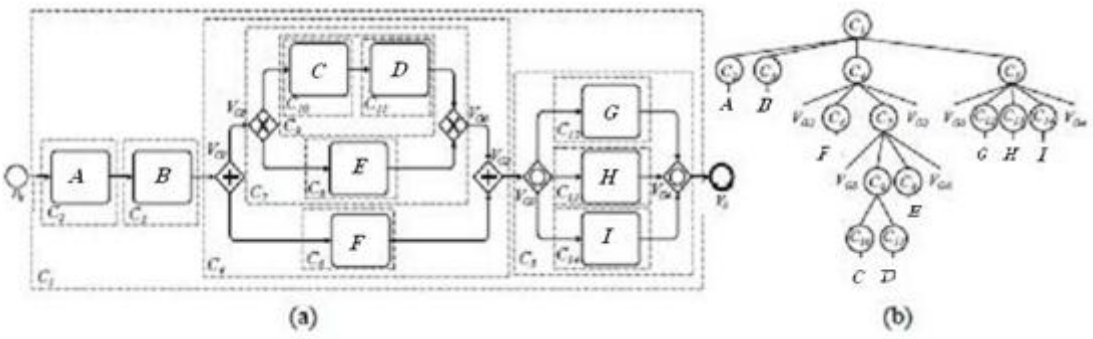
У многим анализама се за процену узимају једнаке вероватноће изршавања свих грана код условних гранања (нпр. XOR/OR-гранања), међутим у циљу прецизнијих процена, потребно је одредити вероватноће за сваку од грана гранања. Стога, са $p_1, p_2, \ldots, p_n$, где је $\sum_{i=1}^{n} p_i = 1$ се означавају вероватноће сваке од грана у паралелном гранању дефинисане као вероватноће да ће дата грана бити

употребљена приликом обиласка за време извршавања процеса; а са *c* број понављања (тј итерација) петље. Наведене вредности су процењене у складу са поступцима наведеним у Секцији 2.1.4, а могу бити ажуриране и у току самог извршавања.

Табела 1.QoS атрибути и правила агрегације у односу на патерне композиције

QoSProperties	***Sequential patterns***		***Parallel-Conditional patterns***	
	Sequence	**Arbitrary Cycle (Loop)**	**Flow** AND-AND	**Switch** XOR/OR
Cost (c)	$\sum_{i=1}^{n} c_i$	$c \cdot c_i$	$\sum_{i=1}^{n} c_i$	$\sum_{i=1}^{n} p_i \cdot c_i$
Response time (t)	$\sum_{i=1}^{n} t_i$	$c \cdot t_i$	$\max(t_i, i \in \{1,..., n\})$	$\sum_{i=1}^{n} p_i \cdot t_i$
Reliability (r)	$\min(r_i, i \in \{1,..., n\})$	r_i	$\min(r_i, i \in \{1,..., n\})$	$\sum_{i=1}^{n} p_i \cdot r_i$
Availability (a)	$\prod_{i=1}^{n} a_i$	a_i^c	$\prod_{i=1}^{n} a_i$	$\sum_{i=1}^{n} p_i \cdot a_i$
Custom property (q)	$f_S(q_i, i \in \{1,.., n\})$	$f_L(c, q_i, i \in \{1,.., n\})$	$f_F(q_i, i \in \{1,.., n\})$	$f_B(p_i, q_i, i \in \{1,.., n\})$

За илустрацију се наводи пример представљен на Слици 10; у левом делу слике је представљен дијаграм тока који се састоји од девет активности (означених саA, B, C, D, E, F, G, H, I), док је одговарајућа структура стабла дата у десном делу слике. За представљање секвенцијалних и паралелних образаца користе се помоћни чворови c_i *(i=1,…14)*.



Слика 10. а) Модел тока представљен у BPMN спецификацији; б) Одговарајућа структура стабла

На основу правила агрегације датих у Табели 1, укупно време одзива T и цена C су одређени са:

$$T = t_A + t_B + \max\{t_F, p_1(t_C + t_D) + p_2 t_E\} + p_G t_G + p_H t_H + p_I t_I$$
$$C = c_A + c_B + c_F + p_1(c_C + c_D) + p_2 c_E + p_G c_G + p_H c_H + p_I c_I$$

где t_i и c_i представљају редом процењене вредности времена одзива и цена за *i*-ти сервис, $i \in \{A,B,C,D,E,F,G,H,I\}$.

С друге стране, уколико нису задовољена правила добре структурираности, *Stochastic Workflow Reduction* (SWR) алгоритам [38] се може користити за рачунање агрегираних вредности QoS карактеристика за дати дијаграм тока. SWR алгоритам се заснива на итеративној примени правила редукције којима се елементи из дијаграма тока елиминишу све док не остане једна атомична активност. Тиме се, након сваке итерације, врши измена саме структуре дијаграма тока, при чему ће атомична активност која на крају остане садржати свеукупне вредности атрибута квалитета, сходно правилима агрегације која се примењују у свакој итерацији. И овај алгоритам има ограничења која се односе на непостојање невалидних елемената у дијаграму тока, тј. грешака у дизајну, као што су незавршене гране и ћорсокаци [39].

Иако се може сматрати да су дефинисана правила добре структурираности ограничавајућа, показало се да добра структура стабла садржи низ пожељних карактеристика које на крају резултирају мање софистицираним механизмом верификације [40]. Читав низ истраживања је спроведен у циљу анализе начина за трансформацију неструктурираних и насумице креираних дијаграма тока у

стуктурирани [41] [42] [40]. На основу датих трансформација, могуће је све анализе вршити искључиво на структурираним дијаграмима тока чиме је обезбеђено да ће се оне односити и на неструктуриране (који се применом наведених механизама трансформишу у структуриране).

На крају, како је у овој дисертацији посебан акценат на композицији апстрактног сервиса SOA архитектуре и карактеристикама квалитета, у наставку се под SOA архитектуром подразумева сервисно оријентисани дијаграм тока $G_{PM} = (v, \varepsilon)$ као композитни модел и скуп расположивих сервиса (класа сервиса) S_a за сваку одатомичнихактивности a; у ознаци $SOA = \left(G_{PM}, \{S_a\}_{a \in A}\right)$, где је А скуп свих атомичних активности у моделу G_{PM} .

2.1.7. SOA фамилије

Поновна употреба (енгл. *reusability*) се сматра једним од неопходних захтева при развоју софтвера. Такође, све већи број софтверских система са истоветним функционалностима је водио променио д развоја појединачних SOA архитектура ка развоју целих фамилија. Под фамилијом се сматра скуп архитектура које деле већину својих функционалности[43]. Овакав приступ, с обзиром на дељеним истоветностима, поспешује поновност употребе и самим тим представља знатно ефикаснији приступ по питању свеукупне цене развоја[44].

Управо су Gomma et al. [45] [46]дали анализу разних добрих примера познатих у теорији развоја софтверских архитектура који су указали на потенцијалне начине заразвојсервиса који подржавају концепт поновности и даљу декомпозицију система датим сервисима. Закључци њихове анализе јасно указују на неопходност адресирања концепта поновности пре саме конструкције SOAрешења[46], с обзиром да SOA-у карактерише недостатак подршке варијабилности на архитектураном нивоу [47] [48].

Линије софтверских производа (енгл. *software product line (SPL)*)су такође развијане за подршку поновности употребе, али садругачијом филозофијом развоја у основи. У случају SPL,софтверски елементи са поновном употребом су

развијани интерно, након изузетно систематичног процеса планирања и анализе. С друге стране, код SOA, поновност употребе типично се остварује кроз одабир сервиса који су обично развијени од стране неке треће стране у целом процесу. Стога, као решење које се наметнуло је управо интеграција ова два приступа, тј коришћење начина за представљање варијабилности и поновљивости из SPL у SOA архитектурама [49] [50] [51].

У складу са тим, развој SOAфамилије се односи на развој три модела: *модел простора проблема* (енгл. *problem space models*), *модел простора решења* (енгл. *solution space models*) и *мапирање међу моделима* (енгл. *mapping models*) [52]. *Модел простора проблема* дефинише одлике које карактеришу дату фамилију као и зависности међу њима. Одлике се обично користе како би корисници одабрали жељене одлике крајњег производа. Скуп одлика се назива конфигурацијом одлика[1]. *Модел простора решења* представља модел који дефинише саму реализацију конкретне фамилије. Како је претходно, у поглављу 2.1.3.,SOA представљена у облику сервисно-оријентисаног модела процеса, SOA фамилија ће бити представљена у облику шаблона (енгл. *template*) модела који ће такође бити представљени језиком за моделовање пословних процеса (у овој дисретацији BPMN нотација). На крају, *модел мапирања* дефинише релације мапирања између наведена два модела. Оно дефинише који делови из шаблона модела могу бити уклоњени на основу одабраних одлика у моделу простора решења.

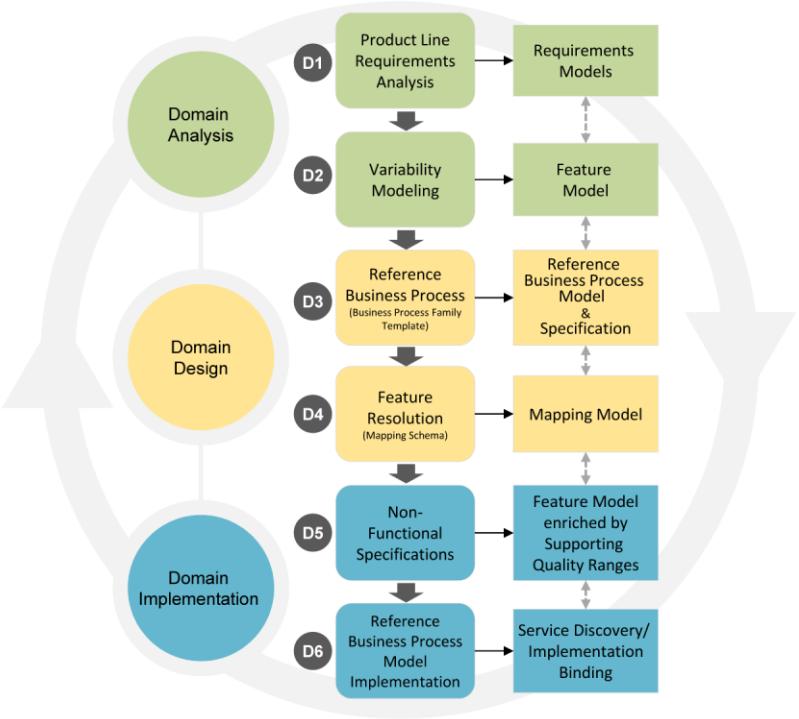
2.1.7.1.Моделовање SOA фамилије

У овом поглављу се описују детаљи начина представања SOA фамилије као и животни циклус доменске анализе и дизајна који као крајњи резултат дају управо дизајн SOA фамилије. Ове активности, означене са D1-D4 представљене на Слици 11 се извршавају итеративно у току доменске анализе (о чему ће речи бити више у секцији 2.2.). Такође, уводе се и неки формални концепти који ће се користити у наредном делу дисертације.

[1] У [36] се често среће назив конфигурација, међутим у овој дисертацији ће термин конфигурација бити касније дефинисан и представљаће конкретну имплементацију конфигурације одлика помоћу расположивих сервиса

2.1.7.1.1. Анализа захтева о линијама производа (D1)

Аналогно традиционалним техникама инжењерства захтева, анализа доменских захтева треда да обухвата следеће активности [46]: (1) *прикупљање захтева*, у којој се идентификују пословни циљеви и захтеви корисника; (2) *спецификација*, у којој се захтеви анализирају и (формално) спецификују; (3) *валидација*, у којој се врши анализа валидности захтева и провера конзистенције, и (4) *управљање захтевима*, у којој се врше евентуалне измене у захтевима или представљање појединих захтева у облику више једноставнијих (мање сложених) захтева. Додатно наведеним активностима, анализа доменских захтева обухвата и идентификацију варијабилности и истоветности међу захтевима дефинисаним од стране више корисника. На крају, резултат ове анализе је модел захтева који може бити представљен у облику модела циљева (енгл. *goalmodel*), случајева коришћења (енгл. *use-cases*), документација и детаља који се користе за даљу анализу варијабилности линија производа.



Слика 11. Активности у анализи и дизајну SOA фамилија [51]

2.1.7.1.2. Моделовање варијабилности (D2)

Моделовање одлика (енгл. *feature modeling (FM)*) је техника која омогућава моделовање атрибута целе фамилије система [53] [54], а у контексту *SPL*је језик који обезбеђује представљање варијабилности [55].

Одлике представљају важне препознатљиве аспекте, квалитете или карактеристике фамилије система [53] [56]. Оне се користе за представљање дељених структура и понашања скупа сличних система. У циљу креирања фамилије производа, све разноврсне одлике скупа сличних или сродних система су компоноване у облику модела одлика [57]. Стога, FM представља начин (средство) за представљање могућег простора свих производа фамилије у смислу њихових одлика. Из овог разлога, неопходно је да FM представља истовремено варијабилност и истоветност међу скуповима одлика разних апликација доступних у датом домену.

Као што ће у наставку бити представљено, FMјетехника погодна за моделовање истоветности, с обзиром да при доменском моделовању омогућава развојистоветних модела одлика који истовремено представљају више апликација, док се варијабилности представљају помоћу различитих компетитивних одлика уједињених заједничким моделом. Пример представљања истоветности је када је једна одлика, која постоји у више апликација представљена јединственом одликом у свеукупном моделу; пример варијабилности је када један појам који се среће у различитим апликацијама се представља различитим одликама које су међусобно компетитивне [57].

FM се описује како формалном семантиком тако и јасном графичком интерпретацијом (нпр.FMдијаграм). Он представља хијерархијску декомпозицију одлика у својству родитељ-дете релације на различитим нивоима апстракције. Како се при конструкцији овог модела полази од чињенице да сви његови елементи не морају бити присутни у крајњим инстанцама фамилије, разлике међу одликама су дефинисане у својствима типова одлика и релација међу њима. Обавезна одлика (енгл. *mandatory feature*) је увек селектована уколико је њена родитељска одлика селектована, док опциона одлика (енгл. *optional feature*) може, а не мора бити селектована. Нпр, на Слици 12 сви производи морају

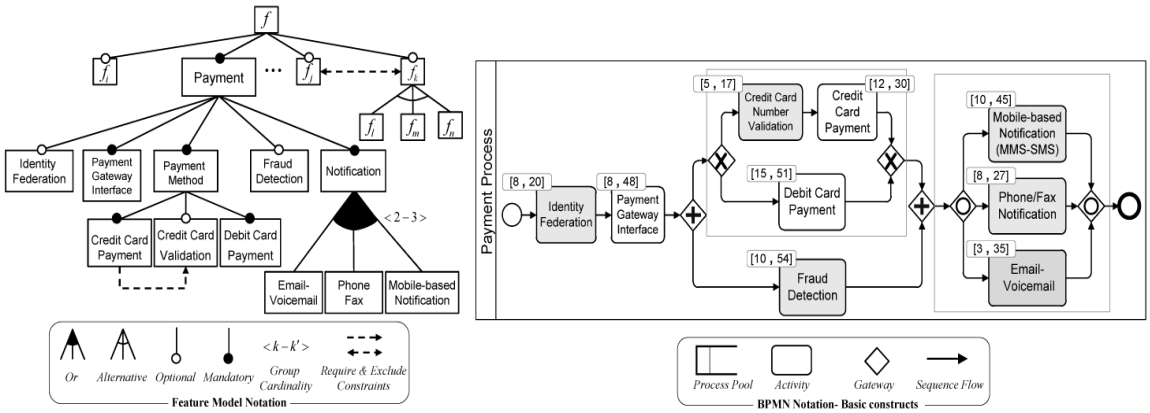
укључитинпр*CreditCardPayment* одлику с обзиром да је обавезна, док су све остале опционе, па не морају бити укључене.

Графички, модел одлика се представља у облику стабла при чему корен стабла представља доменски концепт, нпр домен апликације, а остали чворови и листови представљају одлике. У овом контексту, одлика је концепт или својство које се односи на функционални или нефункционални захтев дефинисан од стране корисниика. У моделу одлика, осим наведених обавезних и опционих одлика, хијерархијска организација се представља помоћу:

- Алтернативне групе одлика(енгл.*Alternative Feature Group*), једна и само једна одлика из дате групе може бити укључена у опис родитељске одлике.
- ИЛИ-групе одлика(енгл.*OR Feature Group*), једна или више одлика из дате групе може бити укључена у опис родитељске одлике. Могући број одлика који може бити укучен се може експлицитно дефинисати навођењем кардиналности опсега, у облику $\langle k - k' \rangle$.

У оквиру легенде симбола на Слици 12 дато је графичко представљање релација међу одликама. Како би се омогућило и представљање релација међусобне зависности међу одликама, уводе се додатна ограничења као ограничења интенгритета (енгл.*intengrity constraints*). Најчешће коришћена ограничења интенгритета су:

- Ограничење укључења(енгл.*include*), које дефинише да присуство дате одлике (или скупа одлика) захтева постојање друге одлике (или скупа одлика).
- Ограничење искључења(енгл.*exclude*), које дефинише да присуство дате одлике (или скупа одлика) елиминише постојање друге одлике (или скупа одлика).



(a) (b)

Слика 12. FM модел за е-процес плаћања

На крају, FMмодел се формално може дефинисати на следећи начин [58]:

Дефиниција 3.(FMмодел)*Модел одлика* $FM = G_{FM}(V, E)$ је усмерени ациклични граф који се састоји од скупа чворова V који представљају одлике и грана $E \subseteq V \times V$ које означаавју релације родитељ-дете међу одликама, $E = \left\{ \overset{\bullet}{f}, \overset{\circ}{f}, f_{or}^{\,k-k'}, f_{xor} \right\}$, где $\overset{\bullet}{f}$ и $\overset{\circ}{f}$ представљају обавезне и опционе родитељ-дете релације, а $f_{or}^{\,k-k'}$ и f_{xor} представљајуИЛИ (са кардиналношћу $k-k'$) и Алтернативне групе одлика са заједничким родитељем.

Значи, након детаљне анализе захтева над целом фамилијом (Слика 11-D1), све варијације у фамилији су представљене помоћу FM модела (Слика 11- D2), чиме су променљиви и истоветни елементи фамилије представљени, редом, као опционе и обавезне одлике у FM моделу.

2.1.7.1.3. Шаблон SOA фамилије (D3)

Дизајн свих елемената фамилије као резултат даје шаблон SOA модела, који је део простора решења. Шаблон SOA фамилије $\mathrm{M}^{T}(SOA)$ садржи унију свих валидних SOA архитектура које могу припадати датој фамилији.

2.1.7.1.4. Модел мапирања (D4)

Да би се повезала дефинисана два модела, модел одлика и шаблон SOA модела, дефинише се мапирање међу њиховим елементима, тј одликама и активностима. Приступом аналогним приступу представљеном у [59], за сваку одлику f_i дефинишу се логичке променљиве Ψ_i; док се за сваку активност у шаблону $\mathrm{M}^T(SOA)$ дефинише атрибут,у литератури познат под називом услов присутности (енгл. *presence conditions (PC)*) [51]. *PC*неке активности се дефинише као логички израз од променљиве Ψ_i којом је представљена одговарајућа одлика у моделу FM. Дефинисане одлике и активности у шаблону $\mathrm{M}^T(SOA)$ користе референтне елементе који се односе на бијективно кореспондентне елементе, редом, у шаблону модела и моделу одлика.

Значи, при самом моделовању, врши се и мапирање одлика у активности, чиме се дефинише *PC* сваке од активности. Касније, уколико се одлика f_i не укључи у инстанцу модела, вредност одговарајуће варијабле Ψ_i се поставља на *false*. Након доделе вредности свим променљивима Ψ_i, врши се евалуација вредности *PC*свих активности и оне активности чија је вредност додељеног *PC* једнака *false,* уклањају се из шаблона модела, као и све остале активности чије је присуство било условљено присуством дате активности. На тај начин се добија једна инстанца целе фамилије. Верификација целог приступа је дата у [60], о чему ће више речи бити у поглављу 2.2.

Тако, у примеру представљеном на Слици 12, претпоставимо да одлика *CreditCardValidation*није селектована, тј да је вредност променљиве Ψ_i постављена на *false.* За одговарајућу активност *PC*је дефинисан као логички израз $\psi_1 \wedge \psi_4 \wedge \varphi_5$ који након евалуације има вредност *false,* па је дата активност уклоњена из модела.

На крају, формална дефиниција фамилије SOA архитектура је дата са:

Дефиниција 4 (Модел SOA фамилије).*Фамилија SOAархитектура* одређена моделом одлика $FM = G_{FM}(V,E)$ представља граф $G_{PM}^{FM} = (\nu,\varepsilon)$,где је ν скуп чворова представљен скупом активности $A = \dot{A} \cup \mathring{A}$ и gateway елемената G процес модела. Скуп gateway елемената је типа $T(G) \in \{xor, or, and, disc\}$. $\varepsilon \subseteq \nu \times \nu$ представља скуп грана међу чворовима. Постоји мапирање $f : (V,E) \to (\nu,\varepsilon)$ које задовољава услове $f(\dot{A}) = E_{|\dot{f}}$ и $f(\mathring{A}) = E_{|\mathring{f}}$, тј скупактивности $\dot{A}$, скуп активности $\mathring{A}$ и скупови gateway елемената су дефинисани бијективним мапирањем, редом, са обавезним одликама, опционим одликама, и релацијама родите-дете у моделу одлика.

Као што је наведено у поглављу 2.1.5. за разматрања у овој дисертацији су од посебног значаја вредности нефункционалних карактеристика, у наставку се посматра пар $SOA = (G_{PM}, \{S_a\}_{a \in A})$, где је A скуп свих атомичних активности у моделу G_{PM}. Аналогно, након разматрања представљених у наредној секцији о мерењима вредноси нефункционалних карактерисика,SOA фамилије и нефункционалне каракетристике се представљају као уређени пар $SOA^{FM} = (G_{PM}^{FM}, \{S_a\}_{a \in A})$, тј осим модела SOA фамилија, посмтра се и скуп свих расположивих сервиса са одговарајућим нефункционалним карактеристикама.

2.1.7.2.Нефункционалне карактеристике SOA фамилије

Као што је представљено у секцији 2.1.4., нефункционалне карактеристике се дефинишу на нивоу сервиса и омогућавају дефинисање мерљиве разлике међу сервисима исте функционалности. Досадашња истраживања у области SOA фамилија су претежно била фокусирана на проблеме моделовања и интергације модела одлика (*Feature Model* – FM)[51]. Међутим, јасна је неопходност посматрања нефункционалних карактеристика у циљу развоја крајње инстанце целе фамилије коју, осим задовољења тражене функционалности карактеришу и

жељене нефункционалне карактеристике као мере квалитета. Стога, у фази D5 (Слика 11) свакој од одлика на нивоу листа у FMмоделу се додјељује параметар класе атомичног интерфејса (из процес шаблона) (алгоритамски кораци су представљени наСлици 13), па се промењивој Ψ_i додјељује још једна карактеризација у виду $k \times k$-димензионог интервала $\left(Q_a^{LB}, Q_a^{UB}\right) = \left(\left\langle Q_1^{LB},..,Q_k^{LB}\right\rangle, \left\langle Q_1^{UB},..,Q_k^{UB}\right\rangle\right) \in R^k \times R^k$, где је *κ*-укупан број нефункционалних карактеристика који се посматра у фамилији, а Q_j^{LB} и Q_j^{UB} су редом минимална и максимална вредност за *j*-ту нефункционалну карактеристику.

За све одлике које нису на нивоу листа, потребно је извршити одређена рачунања, која се дефинишу у зависности од структуре и ограничења у оба модела:FMмоделу и шаблону SOA фамилије. Правила агрегације, предложена у [12]су наведена за класе нефункционалних карактеристика које су претходно разматране у секцији 2.1.4, док се генерални приступ дефинише по аналогији, помоћу функције агрегације *f*која се одређује при идентификацији одговарајућих атрибута квалитета у сваком од домена примене.

Algorithm 1: Aggregate QoS range for feature: AggregateQoSRange(f)

Input: f: given feature of feature model FM
Output: QoS reange- $q^{\mathrm{R}}offeaturef$

```
1  begin
       // All direct child features of f;
2      S_f[] ⟵ ∀ChildFeatureOf(f) ;
3      if S_f = ∅ then return q^R(f);
4      else
5          for i = 1 to |S_f| do
6              AggregateQoSRange(S_f[i]);
7              PST ⟵ ProcessStructure(S_f[i]);
8              if PST = ∅ then return q^R(f) else q^R(f)= ComputeQoSR(PST)
9
10 end
```

Слика 13. Алгоритам *AggregateoSRange(f)*[12]

Конкретно, процена интервала могућих вредности атрибута квалитета се добија на основу хијерархијске агрегације у складу са обрасцима варијабилности

у моделу одлика FM и кореспондентних композиционих обрасца (у моделу сервисно-оријентисаног дијаграма тока).

Табела 2. Правила агрегације дефинисана за обавезне-опционе (енгл. *Mandatory-Optional*) патерне варијабилности[12]

	QoS Properties			Cost (q_c)	Response Time (q_{rt})
Or$^{(*)}$ / *Alternative*$^{(**)}$ *Variability Patterns*	Seq. Pattern	1	Sequence	$\left[\min\left(\sum_{f_i \in F_{Stub}} q_{pr}^{LB}(f_i) : \forall F_{Stub} \in F_{C_k^n}\right), \max\left(\sum_{f_i \in F_{Stub}} q_{pr}^{UB}(f_i) : \forall F_{Stub} \in F_{C_{k'}^n}^{(*)} \mid F_{C_k^n}^{(**)}\right)\right]$	$\left[\min\left(\sum_{f_i \in F_{Stub}} q_{rt}^{LB}(f_i) : \forall F_{Stub} \in F_{C_k^n}\right), \max\left(\sum_{f_i \in F_{Stub}} q_{rt}^{UB}(f_i) : \forall F_{Stub} \in F_{C_{k'}^n}^{(*)} \mid F_{C_k^n}^{(**)}\right)\right]$
	Parallel Patterns	3	AND-AND		$\left[\min\left(\max\left(q_{rt}^{LB}(f_i) : \forall F_{Stub} \in F_{C_k^n}\right)\right), \max\left(\max\left(q_{rt}^{UB}(f_i) : \forall F_{Stub} \in F_{C_{k'}^n}^{(*)} \mid F_{C_k^n}^{(**)}\right)\right)\right]$
		4	AND-DISC		$\left[\min\left(\min\left(q_{rt}^{LB}(f_i) : \forall F_{Stub} \in F_{C_k^n}\right)\right), \max\left(\max\left(q_{rt}^{UB}(f_i) : \forall F_{Stub} \in F_{C_{k'}^n}^{(*)} \mid F_{C_k^n}^{(**)}\right)\right)\right]$
		5	AND-XOR		
		6	OR-XOR		
		7	OR-OR		$\left[\min\left(\max\left(q_{rt}^{LB}(f_i) : \forall F_{Stub} \in F_{C_k^n}\right)\right), \max\left(\max\left(q_{rt}^{UB}(f_i) : \forall F_{Stub} \in F_{C_{k'}^n}^{(*)} \mid F_{C_k^n}^{(**)}\right)\right)\right]$
		8	OR-DISC		
		9	XOR-XOR		

Због бијективне кореспонденције међу моделима FM и шаблона SOA фамилије, свакој одлици се може придружити подграф који одговара декомпозицији дате функционалности. Због дефинисаног SESE ограничења, сваки од наведених подграфова има структуру стабла, тако да се може применити приступ заснован на *Refined Process Structure Tree* (RPST) парсирању. Тиме се агрегација врши *post-order depth-first*обиласком FM модела, тј рачунање граница интервала могућих вредности агрегираних атрибута квалитета се врши супституцијом почев од нивоа листова редом до кореног елемента, са приоритетом чворова који су на десној страни.Алгоритамски кораци су представљени на Слици 13.

Табела 3. Правила агрегације дефинисана за *Or u Alternative* патерне варијабилности[12]

	QoS Properties			Cost (q_c)	Response Time (q_{rt})
Mandatory-Optional Variability Pattern	Seq. Patterns	1	Sequence	$\left[\sum_{i=1}^{n} q_{pr}^{LB}(f_i) : \forall f_i \in \dot{f}, \sum_{i=1}^{n} q_{pr}^{UB}(f_i) : \forall f_i \in \dot{f} \vee \mathring{f}\right]$	$\left[\sum_{i=1}^{n} q_{pr}^{LB}(f_i) : \forall f_i \in \dot{f}, \sum_{i=1}^{n} q_{pr}^{UB}(f_i) : \forall f_i \in \dot{f} \vee \mathring{f}\right]$
		2	Arbitrary Cycle	$\left[cq_{pr}^{LB}(f_i) : f_i \in \dot{f} \vee \mathring{f}, cq_{pr}^{UB}(f_i) : f_i \in \dot{f} \vee \mathring{f}\right]$	$\left[cq_{rt}^{LB}(f_i) : f_i \in \dot{f} \vee \mathring{f}, cq_{rt}^{UB}(f_i) : f_i \in \dot{f} \vee \mathring{f}\right]$
	Parallel Patterns	3	AND-AND	$\left[\sum_{i=1}^{n} q_{pr}^{LB}(f_i) : \forall f_i \in \dot{f}, \sum_{i=1}^{n} q_{pr}^{UB}(f_i) : \forall f_i \in \dot{f} \vee \mathring{f}\right]$	$\left[\max\left(q_{rt}^{LB}(f_i) : \forall f_i \in \dot{f}\right), \max\left(q_{rt}^{UB}(f_i) : \forall f_i \in \dot{f} \vee \mathring{f}\right)\right]$
		4	AND-DISC		$\left[\min\left(q_{rt}^{LB}(f_i) : \forall f_i \in \dot{f}\right), \max\left(q_{rt}^{UB}(f_i) : \forall f_i \in \dot{f} \vee \mathring{f}\right)\right]$
		5	AND-XOR		
		6	XOR-XOR	$\left[\min\left(q_{pr}^{LB}(f_i) : f_i \in \dot{f} \vee \mathring{f}\right), \max\left(q_{pr}^{UB}(f_i) : f_i \in \dot{f} \vee \mathring{f}\right)\right]$	$\left[\min\left(q_{rt}^{LB}(f_i) : \forall f_i \in \dot{f} \vee \mathring{f}\right), \max\left(q_{rt}^{UB}(f_i) : \forall f_i \in \dot{f} \vee \mathring{f}\right)\right]$
		7	OR-XOR	$\left[\min\left(\sum_{f_i \in F_{Stub}} q_{pr}^{LB}(f_i) : \forall F_{Stub} \in F_{C_m^n}\right), \max\left(\sum_{f_i \in F_{Stub}} q_{pr}^{UB}(f_i) : \forall F_{Stub} \in F_{C_m^n}\right)\right]$	
		8	OR-OR		$\left[\min\left(\max\left(q_{pr}^{LB}(f_i) : f_i \in F_{Stub}, \forall F_{Stub} \in F_{C_m^n}\right)\right)^1, \max\left(q_{rt}^{UB}(f_i) : \forall f_i \in \dot{f} \vee \mathring{f}\right)\right]$
		9	OR-DISC		

Субституција се врши применом правила агрегације представљених у Табели 2 и Табели 3, у зависности од патерна варијабилности у моделу одлика и патерна композиције у дијаграму тока, сходно корацима представљеним на Слици 14.

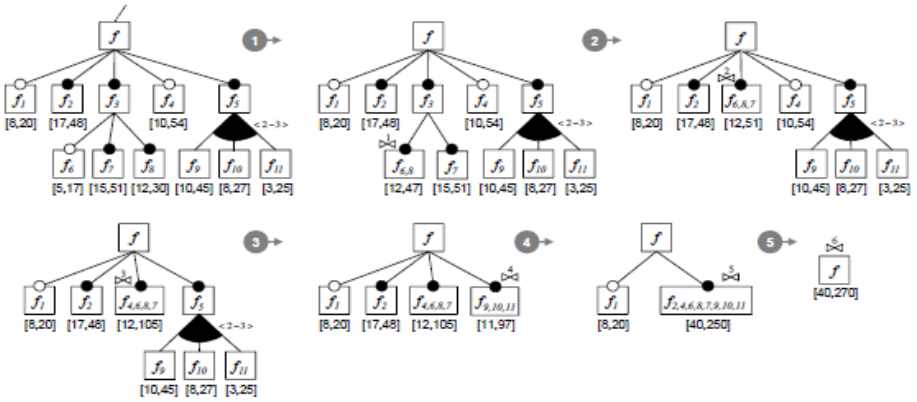
Algorithm 2: Compute QoS range values of business process associated to feature f: ComputeQoSR(C)

Input: C : Process component- node of process structure tree PST
Output: q^{R}: aggregated QoS range of process component

1 **begin**
2 **foreach** $C_i \in ChildOf(C)$ **do** ComputeQoSR(C_i)
3 **forall** $f_k \in C_i$ **do**
   // [⋈]Group features w.r.t. variability patterns and step-wise collapsing features by means of virtual features;
4 **switch** *feature* f_k.*Type* **do**
5 **case** $f_k \in \overset{\circ}{f} \vee \overset{\bullet}{f}$
6 $f_{V_{mo}}[\,] \longleftarrow f_k$;
7 $q^{\mathrm{R}}(f_{V_{mo}}) = \mathrm{AggQoS}(f_{V_{mo}})$ w.r.t. Formulas in Table 1;
8 **if** $\forall f_k \in f_{V_{mo}} : f_k \in \overset{\circ}{f}$ **then** $f_{V_{mo}}.Type = \overset{\circ}{f}$ **else** $f_{V_{mo}}.Type = \overset{\bullet}{f}$;
9 **case** $f_k \in$ *an Or-group*
10 $\overset{\bullet}{f}_{V_{or}}[\,] \longleftarrow f_k$;
11 $q^{\mathrm{R}}(\overset{\bullet}{f}_{V_{or}}) = \mathrm{AggQoS}(\overset{\bullet}{f}_{V_{or}})$ w.r.t. Formulas in Table 2;
12 **case** $f_k \in$ *an Alternative-group*
13 $\overset{\bullet}{f}_{V_{xor}}[\,] \longleftarrow f_k$;
14 $q^{\mathrm{R}}(\overset{\bullet}{f}_{V_{xor}}) = \mathrm{AggQoS}(\overset{\bullet}{f}_{V_{xor}})$ w.r.t. Formulas in Table 2;
15 **end**
16 $f_V[\,] \longleftarrow f_{V_{mo}}, \overset{\bullet}{f}_{V_{or}}, \overset{\bullet}{f}_{V_{xor}}$;
17 $q^{\mathrm{R}}(f_V) = \mathrm{AggQoS}(f_V)$ w.r.t. Formulas given in Table 1,2;
18 **if** $\exists f_k \in f_V : f_i \in \overset{\bullet}{f}$ **then** $f_V.Type = \overset{\bullet}{f}$ **else** $f_V.Type = \overset{\circ}{f}$;
19 **end**
20 **return** $q^{\mathrm{R}}(f_V)$
21 **end**

Слика 14. Алгоритам *ComputeQoSR(f)*[12]

Илустративни пример са наводима алгоритамских корака је дат на Слици 15.



Слика 15. Пример агрегације интервала вредности нефункционалних карактеристика

Наведеним алгоритмом агрегације омогућено је да се на основу скупа расположивих сервиса одреди процена могућих вредности атрибута квалитета целе фамилије. Кораци алгоритма гарантују да ће агрегиране вредности било које конкретне комбинације сервиса (добијене помоћу алгоритма из секције 2.1.5.) припадати добијеним интервалима, с тим да не постоји импликација у супротном смеру; тј за сваку од вредности из процењеног интервала не мора постојати комбинација сервиса која је реализује.

Стога, један од предмета истраживања који ће бити анализиран у овој дисертацији се односи на одређивање вредности атрибута квалитета за коју се жели одредити одговарајућа комбинација сервиса, па се уводе следећи термини:

(i) *Конфигурација одлика* - скуп одлика из FM модела, који у коресподенцији са шаблоном SOA фамилије дефинише једну инстанцу фамилије;

(ii) *Конфигурација SOA фамилије* – скуп сервиса који имплементирају конфигурацију одлика дате фамилије.

Јасно је да проблем одабира конфигурације SOA фамилије директно зависи од одабира конфигурације одлика као и жељених вредности атрибута квалитета целе фамилије. У том циљу, од посебног значаја су разне врсте захтева које могу бити дефинисане од стране корисника, а које су представљене и анализиране у наредном поглављу.

2.2.Анализа корисничких захтева

Анализа корисничких захтева је опште прихваћена као једна од фаза у дизајну и развоју софтвера која је од круцијалног значаја за проблем развоја софтвера који ће одговарати кориснику и његовим захтевима. Емпиријска истраживања су показала да су грешке у анализи захтева санајвећом учестаношћу у целом животном циклусу развоја софтвера, а такође захтевају највише времена за отклањање што проузрокује и додатно највећу цену [61].

Уопштено, под захтевом се сматра опис онога шта се очекује да један софтверски производ извршава. Обично, захтевом се наводе само неки аспекти новог или побољшаног производа или неког сервиса. Најчешће цитирани IEEE 610.12-1990 стандард [62] дефинише захтев као документовану форму:

(1) Услова или способности дефинисаних од стране корисника који су му неопходни за постизање циља или решење проблема.

(2) Услова или способности који морају бити обезбеђени од стране система у целини или његових компоненти како би се задовољио уговор, стандард, или нека друга врста формалног обавезујућег документа.

Стога, под захтевом се не подразумева само захтев дефинисан од стране једног корисника већ може бити дефинисан од стране неке организације, може представљати индустријских стандард или бити дефинисан од стране неких виших државних органа [61]. У сваком случају, захтев представља колекцију потреба дефинисаних од стране једног или више корисника или других наручилаца (организације, заједнице, државних органа или индустријски стандарди) који морају бити задовољени у целости. У идеалном случају, захтеви су независни од дизајна и представљају шта се од система очекује да треба да ради, а не како да ради. Ипак, ово није увек могуће у пракси, посебно ако се узме у обзир да значење речи шта и како може да зависи и од особе која интерпретира[63]. Ово управо води ка робусној подели и дефиницији корисничких и техничких захтева. Кориснички захтеви су написани са стране корисника и његовог погледа на систем(производ) који треба развити. Они дефинишу неку функционалност, ограничење или пак својство које мора бити задовољено како би се развијени систем(производ) могао сматрати одговарајућим.

С друге стране, технички захтеви описују како систем(производ) треба да буде имплементиран у циљу испуњења дефинисаних корисничких захтева.

У овој дисетрацији је фокус само на корисничким захтевима, па ће у наставку бити описане врсте корисничких захтева као и основне фазе у животном циклусу процеса њихове обраде при развоју софтвера.

2.2.1. Врсте корисничких захтева

Постоје разни начини класификације захтева, као што је приказано у Табели 4. Најчешће, захтеви се деле на функционалне и нефункционалне. Међутим, не постоји стриктна дефиниција појма функционалности, већ постоји широк консензус дефиниција, које ипак прате две опште прихваћене нити.

Табела 4. Врсте захтева [61]

Requirements Classification
Functional requirements — what the system will do *Non-functional requirements* — constraints on the types of solutions that will meet the functional requirements e.g. accuracy, performance, security and modifiability
Goal level requirements — related to business goals *Domain level requirements* — related to problem area *Product level requirements* — related to the product *Design level requirements* — what to build
Primary requirements — elicited from stakeholders *Derived requirements* — derived from primary requirements
Others classifications, e.g. *Business requirements* versus *technical requirements* *Product requirements* versus *process requirements* - i.e. business needs versus how people will interact with the system *Role based requirements*, e.g. customer requirements, user requirements, IT requirements, system requirements, and security requirements

Прва нит [64] се фокусира на функцију: функционални захтев дефинише функцију коју систем мора бити у могућности да обезбеди [62], шта производ мора да ради [65] и шта би систем требао да ради [66]. Друга нит се фокусира на понашање: функционални захтеви описују аспекте понашања система [67], акције које систем мора бити у могућности да изврши без обзира на физичке захтеве и

условљености, као и захтев који дефинише улазно/излазно понашање система[63][68].

Табела 5. Дефиниције појма 'нефункционални захтев' [64]

Source	***Definition***
[67]	Describe the non behavioral aspects of a system, capturing the properties and constraints under which a system must operate.
[70]	The required overall attributes of the system, including portability, reliability, efficiency, human engineering, testability, understandability, and modifiability.
[62]	Term is not defined. The standard distinguishes design requirements, implementation requirements, interface requirements, performance requirements, and physical requirements.
[71]	Term is not defined. The standard defines the categories functionality, external interfaces, performance, attributes (portability, security, etc.), and design constraints. Project requirements (such as schedule, cost, or development requirements) are explicitly excluded.
[68]	A requirement that specifies system properties, such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensibility, and reliability. A requirement that specifies physical constraints on a functional requirement.
[69]	Requirements which are not specifically concerned with the functionality of a system. They place restrictions on the product being developed and the development process, and they specify external constraints that the product must meet.
[72]	"... global requirements on its development or operational cost, performance, reliability, maintainability, portability, robustness, and the like. (...) There is not a formal definition or a complete list of nonfunctional requirements."
[73]	The behavioral properties that the specified functions must have, such as performance, usability.
[65]	A property, or quality, that the product must have, such as an appearance, or a speed or accuracy property.
SCREEN (1999)[2]	A requirement on a service that does not have a bearing on its functionality, but describes attributes, constraints, performance considerations, design, quality of service, environmental considerations, failure and recovery.
[74]	A description of a property or characteristic that a software system must exhibit or a constraint that it must respect, other than an observable system behavior
Wikipedia: *Non-Functional Requirements*[3]	Requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviors.
Wikipedia: *Requirements Analysis* [4]	Requirements which impose constraints on the design or implementation (such as performance requirements, quality standards, or design constraints).

С друге стране, нефункционални захтеви или захтеви квалитета су захтеви који нису посебно захтевани функционалностима система [69]. Показало се да је у општем случају теже задовољити дефинисане нефункционалне захтеве, зато што

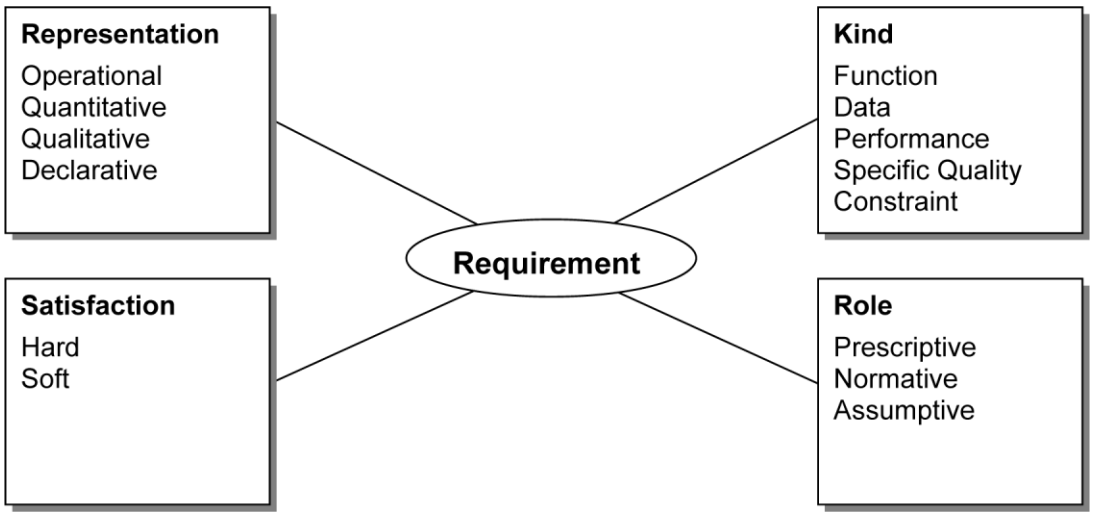
[2] *Glossary of EU SCREENProject*.http://cordis.europa.eu/infowin/acts/rus/projects/screen/glossary/glossary.htm

[3] http://en.wikipedia.org/wiki/Non-functional_requirements

[4] http://en.wikipedia.org/wiki/Requirements_analysis

се обично не могу локализовати на само једну компоненту система, већ се обично односе на систем у целини. Постоји читав спектар различитих дефиниција које наводе како терминолошки, тако и концептуално различита питања која се односе на нефункционалне захтеве. Неке од њих су предствљене у Табели 5.

Из табеле се јасно види да свака наведена дефиниција садржи појмове као што су својство, карактеристика, атрибут, без њиховог претходног дефинисања. Сваки захтев (укључујући и функционалне) се може сматрати одређеним квалитетом, јер сходно стандарду ISO 9000:2000[24] под квалитетом се сматра степен у којем карактеристике система (производа) задовољавају дефинисане захтеве. Аналогно, сваки захтев истовремено може бити посматран као ограничење, с обзиром да одређује простор потенцијалних решења који испуњава дефинисане захтеве [64].



Слика 16. Класификација затева[75]

Осим наведеног проблема дефиниције функционалних и нефункционалних захтева и њихове јасне сепарације, евидентан је и даљи проблем класификације захтева. У [75] се предлаже начин класификације захтева у којем више не постоје појмови функционалних и нефункционалних захтева. Приступ се занива на идентификацији следећих концепата: врсте, начина презентације, нивоа задовољења и улоге, као што је представљено на Слици 16.

За концепт *Врста* су идентификоване следећих пет могућих вредности:

- *Функција*–захтев који се примарно односи на очекивано понашање система у целини или неке његове компоненте.
- *Податак* – захтев који се односи на улазне податке чијом се анализом и обрадом добијају тражени излазни подаци.
- *Перформансе* – захтев који се највећим делом односи на време извршавања, брзину, обим и пропусност система.
- *Специфичан квалитет* – захтев који може обухватати посебне врсте квалитета, сходно [35].
- *Ограничење* – захтев који смањује (ограничава) простор могућих решења сходно траженим функционалностима, перформансама и специфичним карактеристикама квалитета.

Концепт*Степен задовољења* дефинише да ли је испуњење дефинисаног захтева обавезујуће, тј да ли његово неиспуњење директно води ка незадовољавајућем производу. У том случају, реч је о строгим захтевима, чије нарушавање посматрани систем (производ) одмах елиминише из даљег разматрања као потенцијалног решења. Насупрот строгим захтевима, постоје и захтеви који дозвољавају да се мери степен њиховог испуњења с обзиром да у целости не морају бити испуњени. За такве захтеве се каже да је реч о нестрогим захтевима (енгл. *soft requirements*).Постоји цели скуп разних метода за анализу дефинисаних захтева [76] [77], као што ће бити наведено у поглављу 2.2.2.3. С становишта концепта *Степен задовољења*, разне технике за анализу захтева могу бити интерпретиране као технике које обезбеђују максимизацију испуњења комбинације већине нестрогих захтева уз обавезно испуњење дефинисаних строгих захтева.

Концепт *Начин предствљања* се односи на начин на који систем може верификовати испуњење самог захтева. Могуће вредности су:

- *Операционо* – тестови и контроле обезбеђују процес верификације.
- *Квантитативно* – дефинише се нађин мерења степена испуњења дефинисаних захтева. Неопходно је да дефинисане мере буду најмање ординалне.
- *Квалитативно* – у принципу, нема директне верификације. У неким случајевима могуће је дате вредности трансформисати у квантитативне,

или верификација може бити спроведена на основу прототипа или идиректно раздвајањем циљева.

- *Декларативна* – контрола је једина врста верификације.

Додатно наведеним концептима, у литератури се наводе врсте различитих скала за представљање улазних и излазних података као и њихове семантике. Сходно [78], улазни и излазни подаци могу бити редом: квантитативни, квалитативни и/или вербални. Додатно, разне врсте скала података могу бити коришћене за престављање захтева, као што су: ординална скала (енгл. *ordinal scales*), која дефинише редослед међу опцијама; интервална скала (енгл. *interval scales*), код које су нумеричке вредности дефинисане до реда афиних трансформација; непрекидна скала (енгл. *ratio scales*), код које су вредности дефинисане до нивоа мултипликативног фактора [79].

Наведени спектар дефиниција и подела намеће да се проблем дефиниције и поделе захтева директно односи на домен примене, и да сходно томе, различити аутори дефинишу и различите принципе. Ипак, општа и најчешће прихваћена подела обухвата класификацију на функционалне и нефункционалне захтеве. У суштини, систем (производ) је одређен његовим функционалностима и његовимнефункционалним карактеристикама, као што су употребљивости, флексибилност, перформансе, интероперабилности и сигурност [80]. Ипак, јасан је нагласак на траженим функционалностима система, иако систем није употребљив или није довољно добар да би био употребљив, уколико не задовољава и дефинисане нефункционалне захтеве. Иако је у литератури увек јасна граница и разлика између ове две врсте захтева, у пракси их није увек могуће јасно разликовати [81]. Тако нпр, захтев корисника о сигурности система може бити разматран као захтев за обезбеђењем функције безбедности, док с друге стране, сам ниво безбедности може представљати и нефункционални захев.

2.2.2. Менаџмент захтева при конструкцији софтвера

Као што је дефинисано у [82], основни проблем у области Анализе корисничких захтева је како идентификовати шта корисници заиста желе и како њихове захтеве испунити. У питању је сложен процес који се итеративно одвија током целог софтверског пројекта, а због саме сложености целог процеса нема ни консензуса око активности у животном циклусу. Најчешће се наводе следеће активности:

- *Прикупљање, спецификација и моделовање захтева*: ове активности обухватају идентфикацију захтева корисника, њихово прикупљање и формално моделовање. Управо је ово једна од критичних фаза у развоју софтвера с обзиром да су захтеви корисника обично јако тешки за разумевање, обимни и неконзистентни.
- *Рангирање*: неопходно је одредити приоритет дефинисаних захтева за корисника, тј који од захтева су неопходни за испуњење, а за које међу њима је могуће правити и компромисе уколико се у потпуности не могу испунити. У том циљу је развијен читав скуп метода и техника, при чему одабир одговарајуће директно зависи од карактеристика система и врсте захтева које дефинише корисник.
- *Анализа зависности међу захтевима и анализа утицаја*: У многим примерима, од изузетног значаја је сагледавање промена које могу настати у самом софтверу као последица промене захтева и њиховог директног утицаја [83].
- *Балансирање захтева различитих интересних група*: Различите групе корисника, дизајнера, пројектних менаџера и каснијих одржаваоца система, могу учествовати у процесу дефинисања захтева. Сваки члан групе има различите интересе при дефинисању захтева, који стога могу бити у конфликту. Зато је неопходно извршити балансирање међу њима, и сходно улогама сваке од интересних група, наћи начине за превазилажење конфликта и њихово разрешавање. Ово је такође једна од круцијалних фаза у анализи захтева јер може имати директан утицај на крајњи производ.

- *Обезбеђење квалитета*: у овој фази је неопходно обезбедити високе захтеве квалитета и благовремено их документовати. Од изузетног значаја је да обе стране, корисници и развојни тим, учествују у спецификацији ових захтева јер је то од директног утицаја на успех целог пројекта. Што ранија идентификација грешака смањује цену каснијих корекција и исправки.

2.2.2.1.Прикупљање корисничких захтева

Процес прикупљања захтева се обично сматра почетном фазом, а обухвата идентификацију захтева путем комуникације и интервјуа са корисницима, наручиоцима и осталим учесницима у процесу развоја система [84]. Процес прикупљања захтева је у јакој спрези са осталим фазама у целом процесу јер је неопходно извршити анализу комплетности, како би се даљи развој заснивао на испуњењу дефинисаних захтева.

Најважнији задаци фазе прикупљања захтева су: идентификација класа корисника и учесника (наручиоци система, развојни тим који дизајнира и развија комплетан систем, крајњи корисници који интерагују са системом и користе га), њихових циљева (које систем мора да задовољи) и задатака над системом (тј који корисници шта треба да раде са системом), као и сценарија употребе (у циљу бољег схватања аспеката употребе система). У литератури постоје разне класе техника прикупљања захтева, које се међусобно обично разликују по времену и ресурсима потребном за њихову примену, као и врсти захтева који се прикупљају. Неке од тих категорија су[84]: традиционалне технике (нпр. организациони дијаграми исл.), технике групног прикупљања захтева[82] [85] [86], прототипови[87] [88] [89], технике вођене моделом [90][80][91], и когнитивне технике [92].

Одабир одговарајуће технике/техника за прикупљање захтева зависи искључиво од контекста примене и доменског знања корисника и осталих учесника у процесу инжењеринга захтева. У овој дисертацији се подразумева даје дефинисан скуп захтева, па је фокус истраживања на осталим фазама у животном циклусу.

2.2.2.2.Моделовање корисничких захтева

У овој фази је потребно обезбедити начин за формалну спецификацију корисничких захтева као и дати процену сложености репрезентације и самог развоја. Спецификација система зависи од дела система који се разматра, као и самог погледа (перспективе) на систем који се жели представити. Све већа сложеност система који се моделују намећу употребу разних врста мета-модела којима се представљају разне карактеристике система. Тиме се систем моделује помоћу различитих модела, сваки од њих одговара различитим погледима на систем, посвећен само одређеном скупу карактеристика система. Најчешће примењиване врсте мета-модела су дијаграми стања (енгл. State Charts), Пертријеве мреже (енгл.*Petri nets*), аутомати коначних стања (енгл. *finite state machines (FSMs)*), дијаграми повезаних компоненти (енгл. *component-connectivitydiagrams (CCDs)*), итд.

Након конструкције одговарајућег мета-модела, неопходно је развити и ***формални језик за његово представљање***. Формална анализа верификације је неопходна како би се испитала повезаност спецификације, за шта је неопходан математички формално заснован језик за представљање.

Следећа фаза обухвата ***процену сложености*** која се односи на две различите димензије: сложеност репрезентације и сложеност развоја. Сложеност репрезентације зависи од језика који се користи за спецификацију, која, уколико се правилно спроводи, омогућава прегледност и разумљивост спецификације. Сложеност развоја се односи на контролу евалуације спецификације система од иницијалног концептуалног представљања захтева. Овде се неки детаљи система могу оставити за касније фазе у развоју, или се применити неки од следећих приступа у процесу спецификације: од врха ка дну (енгл. *top-down*), од дна ка врху (енгл. *bottom-up)* или од средине ка крајевима (енгл. *middle-out*).

Захтеве корисници обично дефинишу природним језиком и употребом неформалних дијаграма, са релативно ниским нивоом детаља, фокусирани на сами домен проблема. Случајеви коришћења су једна од најпогоднијих техника, с обзиром да су једноставни и лако разумљиви. Захтеви корисника могу бити

различите врсте (као што је раније представљено у поглављу 2.2.1.) и за њихову анализу и процесирање су развијене разне методе и технике (као што ће бити приказано у поглављу 2.2.2.3), али свака од техника захтева захтеве представљене у одговарајућој форми. Стога, процес спецификације и моделовања захтева корисника обухвата следећа два корака: *(i)*идентификацију захтева корисника; и*(ii)* њихову трансформацију у одговарајућу форму која је директно одређена методом који ће се користити за њихову анализу. Идентификација захтева корисника је од посебног значаја за цели процес инжењеринга захтева, која није једноставна, већ суочена са многим проблемима и неконзистенцијама [77].

Многи истраживачи су се бавили проблемом развоја техника и метода за прикупљање, анализу и формализацију захтева корисника над расположивим опцијама, као што су метода поређења парова (енгл. *pair wise comparison method*), приоритетне групе (енгл. *priority groups*), мреже за одлучивање (енгл. *desicion networks*) и кумулативно рангирање (енгл. *cumulative ratings*)[93] [94]. Представљање захтева и њихово процесирање се перманентно развило у областима као што су економија, пројектни менаџмент, управљање ризицима, теорија одлучивања, социјална теорија избора, са даљим развојем и адаптацијом у областима операционих истраживања, база података, анализа безбедности, као ивештачке интелигенције. Моделовање корисничких захтева је велики изазов, а обзиром да је јако тешко представити људско мишљење и разум на начинкоји је погодан за процесирање од стране рачунара [95].

2.2.2.3.Рангирање корисничких захтева

Као што ће бити наведено у наредним поглављима, постоји мноштво формализама и метода који су развијени за различите структуре захтева, са различитим врстама улазних и излазних података, и различитим семантикама. Већина метода је развијена за анализунезависно дефинисаних захтева сходно основној хипотези о узајамно независним захтевима (енгл. *mutual preference independence – MPI*)[96]која подразумева да се кориснички захтеви над једном опцијом увек дефинишунезависно у односу на остале [95]. Ипак, Sterling at al. су показали да ова хипотеза не одговара увек ситуацијама у пракси и реалном

животу, већ да захтеви имају знатно сложенију структуру[97]. Човечијем уму је природније да формулише условне захтеве који представљају стање једне опције у случају када је стање друге тачно одређено и има одговарајућу вредност [95], стога се они и користе за представљање људског размишљања [95]. Такође, јако често се у пракси и свакодневном животу среће групно (тј. колективно) одлучивање, где различити учесници у процесу одлучивања са различитим интересима и приоритетима дефинишу своје захтеве, које је свеукупно потребно узети у разматрање и укључити у процес развоја крајњег производа.

Иако семетоде за резоновање развијају у разним областима са циљем решавања различитих проблема (из разних домена примене), једна од кључних карактеристика за њихово оцењивање је процена сложености. Овај критеријум се у области анализе захтева анализира са становишта триврсте упита [98]:*(i)**упит доминације*, који се односи на поређење две опције, *(ii)**упит сортирања*, којим се дати коначан скуп опција уређује у растући/опадајући поредак сходно приоритетима генерисаним на основу дефинисаних захтева, и *(iii)**упит оптимизације*, који одређује теоријски најбољу опцију [95]. У области анализе захтева, показано је да су сва три упита у општем случају NP-сложена[99], чиме се јасно намеће потреба за развојем ефикасније технике за оптимизацију упита *(i)* - *(iii)* у одговарајућој области примене.

У наставку се даје преглед и анализа постојећих метода за анализу захтева као и њихова поредбена анализа. Остатак секције је организован на следећи начин: секција 2.2.2.3.1. садржи преглед квантитативних захтева и анализе њихове могућности примене за анализу условних захтева, секција 2.2.2.3.2. даје преглед техника за анализу условних захтева и анализу могућности њихове примене у анализи квантитативних података. На крају, секција 2.2.2.3.3.даје сумарне резултате анализе.

2.2.2.3.1. Квантитативно рангирање и условно дефинисани захтеви

У овој секцији се наводе методе и формализми из различитих области примене и истраживања који се могу применити за квантитативно рангирање и условно дефинисане захтеве.

Традиционалне методе за анализу захтева обухватају анализу нефункционалних захтева и засноване су на поређењима парова чиме се као резултат добија квантитативна мера за поређење расположивих опција. AHP метод (енгл. *Analytical Hierarchical Process*) који је развио Saaty [100]представља опште прихваћени више-критеријумски метод за одлучивање. AHPметод омогућава одлучивање над материјалним и нематеријалним опцијама при чему се прати степен конзистентности у захтевима [101]. Још једна изузетно важна карактеристика AHPметода је могућност подршке групном одлучивању, и у том циљу постоји неколико различитих приступа са становишта вредновања појединаца из групе у одлучивању [91][92][93]. Неки од њих су: *(i)* концензус међу учесницима, *(ii)* компромис или гласање када концензус не може бити постигнут, *(iii)* агрегација индивидуалних процена *(iv)* агрегација индивидуалних приоритета; *(v)* агрегација индивидуалних захтева, и *(vi)* разматрање интервалних пресуда. Као опште прихваћени метод, AHPсе користи у многим важним областима одлучивања као што су предвиђање, менаџмент квалитета, ре-инжењерство пословних процеса[102] [103][104][101]. Иако је AHP метод једноставан за употребу, уочено је неколико недостатака, као што су квадратни број поређења, и немогућност поређења концептуално различитих опција. У циљу савладавања уочених недостатака, развијен је S-AHP метод (енгл. *Startified AHP*)[105]. S-AHP метод смањује број потребних поређења међу опцијама креирањем слојевите двослојне структуре.

Још један представник фамилије метода заснованих на квантитативним мерама (на основу категоризацијеLarichev-a[78]) је TOPSIS метод (енгл. *Technique for Order Preference by Similarity to an Ideal Solution)*. Основа приступа TOPSIS метода [100] се огледа у одабиру алтернатива на основу најкраћег растојања у односу на позитивно иделани случај и истовремено највећег растојања у односу нанегативно идеални случај. Позитивно и негативно идеални случајеви се одређују на основу статичких (временски непроменљивих) и безусловно дефинисаних захтева. Нпр, најбоља цена је она која је најмања, док је најбољи квалитет представљен највећом вредношћу итд. Овакав начин дефинисања захтева (статички и безусловни) уједно представља и ограничење примене метода за представљање и анализу различитих врста захтева. Додатно, овој групи

припадају и још многе друге методе, као што су SAW метод (енгл. *Simple Additive Weighting*)[106][107][108], LINMAP техника(енгл. *Linear Programming Techniques for Multidimensional Analysis of Preference*) [109], CORPAS метода (енгл. *Complex Proportional Assessment*) [110, 106], али се њихове карактеристике међусобно не разликују сходно основнимпоредбеним критеријумима наведеним у Табели 4.

Пример метода за компаративно поређењекоје је засновано на поређењу алтернатива у паровима је ELECTRE фамилија метода, развијенасредином шестдесетих година прошлог века. Ова фамилија метода је имала изузенто јак утицај на развој области операционих истраживања[111]. ELECTRE фамилија методаомогућава рангирањеи квантитативних и квалитативних захтева. У поређењу са традиционалним методама, ELECTRE уводи концепт прага индиферентности (енгл. *indifference threshold*), тј преференце се дефинишу у смислу: жељених, индиферентних и немогућих за поређење. Цела фамилија метода је развијена у циљу подршке хетерогених скала, интерпретације рангирања на основу *fuzzy*-дефинисаних захтева, као и у случају непотпуне дефиниције коефицијената релативног односа. Методи из ове фамилије су добро познати и применљиви у области оптимизације одлучивања под условима извесности. Слично као и код TOPSIS метода, захтеви су дефинисани статички, што уједно представља главни разлог немогућности примене ELECTRE фамилије мотода за условно дефинисане захтеве (које карактерише неизвесност).

Још једна техника која се среће у литератури, *Bubble sort* техника, је јако слична AHP методу с једином разликом да се поређења врше с циљем одређивања опције већег приоритета, али не и мере приоритетности. Стога је јасно да *Bubble sort*техника има исте недостатке као и AHP метод (нпр. велики број поређења). У литератури се могу наћи и предлози за редукцију броја потребних поређења који се обично заснивају на некомплетним поређењима парова[94]. Ове технике се заснивају на локалним и/или глобалним критеријумима у случајевима када даља поређења нису потребна јер не доносе нову информацију,која већ није садржана у претходно дефинисаним. Дате технике би биле од изузетне користи уколико би биле примењене у техникама као што су AHP и S-AHP[112].

Табела 6. Поредбена анализа метода/техника за рангирање корисничких захтева[112]

Метод/ техника	*Улазни подаци/ Излазни подаци*	*Условни захтеви*	*Специјалне врсте захтева*	*Тестови доминације/ Упит о редоследу*	*Упит опимизације*
[113]	ординална скала / ординална скала	Адресирани	Лексикографско рангирање	Непотпуно адресирани	Линеарна сложеност у односу на величину мреже под условом слабије ацикличности
[114]	ординална скала/ ординална скала	Непотпуно адресирани	-	NP- сложен, оптимизоавн у случајевима одређених топологија	Линеарна сложеност у односу на величину мреже под условом ацикличности
[115]	ординална скала/ нумеричке вредности	Непотпуно адресирани	Непрекидни атрибути, вишеструки атрибути у захтевима	Непотпуно адресирани	Непотпуно адресирани
[93]	ординална скала/ ординална скала	Адресирани	-	NP-сложен	Линеарна сложеност у односу на величину мреже под условом ацикличности
[116]	Иницијално квалитативне, трансформисане у нумеричке вредности/ нумеричке вредности	Непотпуно адресирани	-	Линеарна сложеност у доносу на величину мреже	Линеарна сложеност у доносу на величину мреже
[58]	Иницијално квалитативне, трансформисане у нумеричке вредности/ нумеричке вредности	Непотпуно адресирани	-	Полиномијал на сложеност	NP- сложен
[117]	ординална скалаи вредности корисности / нумеричке вредности	Адресирани	Вишеструки атрибути у захтевима	NP- сложен у најгорем случају	NP- сложен
[100, 111, 118]	Иницијално квалитативне, трансформисане у нумеричке вредности/ нумеричке вредности	-	-	Полиномијал на сложеност	NP- сложен

HCV метод (engl. *Hierarchical Cumulative Voting*) је техника која се заснива на одабиру више рангираног захтева на основу поена додељених од стране корисника [118]. Недостатак овог метода се огледа у проблему који се намеће кориснику који додељује поене у ситуацији када бројрасположивих опција постане јако велики. У том случају, за корисника се проблем огледа у одређивању најбоље технике гласања, тј начина одабира опције која заслужује највише поена. Додатно, HCVподразумева да је предмете од интереса могуће поделити у различите нивое, али не садржи механизам на основу којег је подела могућа [119]. Као решењекоје се намеће у циљу проширења HCVметода за анализу условних и безусловних захтева, је њихова подела на две различите групе. Тиме би било могуће да се добије само једна (или чак ниједна) група безусловних захтева, и велики број условних, чиме је добијен нови проблем поделе условних захтева на одговарајуће групе. С друге стране, уколико су све опције у истом блоку, настаје проблем компензације који је јасно идентификован за овај метод.

2.2.2.3.2. Условни захтеви

Бројне студије [116][114][93]су изведене у циљу анализе условних захтева и разних врста захтева који укључују условљеност. Обично се за представљање условних захтева употребљавају структуре података као што су мреже или графови, а најпознатијетакве технике су CP-nets и TCP-nets.

Мрежа CP-net (енгл. *Conditional Preference Network*) [114]је формализам за предствљање условних захтева у мултиваријантним проблемима. То је квалитативни начин за представљање захтева који осликава условне зависности и независности у случају интерпретације *'ceteris paribus'*(тј. под истим околностима). Изузетноважно својство CP-net мрежа је линеарна комплексност алгоритма за одређивање оптималног решења који се заснива на тополошком сортирању мреже[120]. С друге стране, проблем поређења дате две опције, сстановишта комплекности алгоритма над датом структуром није тако једноставан [79]. Стога су новија истраживања управо усредсређена на идентификацију појединих топологија CP-net мрежа које могу оптимизовати наведене упите, али у општем случају, реч је о NP-сложеном проблему [99].

TCP-nets (енгл. *Tradeoffs-enhanced CP-nets*)[93]подржавају информације о условним независностима као и о условним релативним приоритетима. Стога, реч је о моделу за представљање богатијег скупа корисничких захтева, мада је и даље реч само о квалитативно дефинисаним захтевима. У овим мрежама се препознаје посебна класа условно ацикличних TCP-nets мрежа, које задржавају линеарну комплексност за одређивање оптималне опције. Слично као код CP-nets мрежа, проблем прецизног одређивања сложености алгоритма за поређења расположивих опција је отворено истраживачко питање, сходно досадашњим сазнањима.

COP-network мрежа (енгл. *Conditional Outcome Prefrence Network*) [117]је још један пример начина представљања условних захтева помоћу графа. Овај формализам се заснива на директној презентацији захтева дефинисаних од стране корисника и каснијем проширењу мреже захтевима који семогу извести као закључци тј логичке последице (исходи) постојећих захтева. У поређењу са претходно наведним методама, COP-network мрежа користи функцију корисности за предвиђање вредности које се могу добити на основу дефинисаних захтева; на тај начин могуће јеједноставноодредитида ли једна опција има већу преферабилност у односу на другу. COP-network мрежа се креира придруживањем по једног чворасвакој могућој опцији, чиме време потребно за креирање и обилазак такве мреже постаје изузетно велико у најгорем случају. С друге стране, за разлику од осталих техника, које служе за представљање захтева који се односе на одговарајући атрибут, COP-network мреже пружају могућност представљања захтева који садрже односе међу атрибутима.

И поред изложених недостатака, CP-nets/TCP-nets мреже представљау опште прихваћени модел за представљање и анализу условних захтева, али се ипак наглашавају лоше перформансе у случају упитанамењених поређењу расположивих опција, као и ограничење цикличности у мрежама, што директно умањује њихову практичну употребљивост[113]. У том циљу је и предложено неколико надоградњи које су развијане у различитим правцима због комплексности проблема, али још увек не нуде опште решење.

Проширење у знатно општију cp-теорију (енгл. *cp-theory*) [113]уводи нови формализам, који се може посматрати као једноставна логика условних захтева. Показано је да се датом семантикоммогу представити знатно сложенији искази о

захтевима, који се даље могукористити за, нпр, конструкцију лексикографског поретка међу опцијама, што није било могуће представити помоћу CP-nets/TCP-nets. У cp-теорији се разматра слабија форма ацикличности, што представља довољан услов да cp-теорија буде конзистетнта. С друге стране, радо cp-теорији[113]не анализира проблем поређења опција, за који је претходно показан да је у CP-nets/TCP-netsјако комплексан [99][114] [93]и предствља јако важан проблем у области анализе затхева.

Док су CP-nets/TCP-netsмреже развијене за квалитативну анализу захтева,UCP-мреже[116]анализирају и квантитативне захтеве и податке о релативним приоритетима употребом функција корисности. Ове мреже комбинују теорије CP-netsi GAI-netмрежа (енгл. *Generalized Additive Decomposable Utility Fuctions*)[121].ПроширењемCP-netsмрежа квантитативним подацима корисности, постигнута је већа експресивност и упити о поређењима међу опцијама су постали ефикаснији, тј смањена им је сложеност. С друге стране, UCP-мреже садрже ограничење с обзиром да модел не садржи никакве претпоставке о врстама интеракције међу атрибутима који се пореде.

У[58], TCP-netsмреже су анализиране са становишта примене за представљање квантитативних података. Захтеви и ограничења су представљени квалитативно након чега су мапирани у квантитативни модел. Анализа захтева се врши на основу расположивих опција креирањем стабла које садржи карактеристике опција и вредности на основу дефинисаних корисничких захтева. На тај начин, упит о поређењу опција има полиномијалну сложеност, док упит о одабиру оптималне опције подразумева креирање стабла којесадржи све теоријски могуће опције у моделу и њихово поређење. Овим моделом нису обухваћене квантификације свих врста захтева који се квалитативно могу представити у TCP-netsмрежама.

CP-netsмреже су у[115]проширене помоћу концепата из елементарне геометрије, употребом адитивне линеарне функције чије вредности одговарају релацијама међу захтевима корисника и дефинисањем ограничења над парицијалним изводима тако дефинисане функције. Показано је да за сваку ацикличну CP-netsмрежу адитивна функција може представити све форме захтева[122]. Ипак, предложена надоградња има ограничења због цикличности

које се у графу појављују дефинисањем условљености међу захтевима. С друге стране, моделом нису представљене све врсте релација могућих у TCP-nets мрежама, мада су уведени нови начини за представљање релација помоћу линеарних неједначина исистема. Међутим, такав систем у општем случају не мора имати решење, чиме се добија нова сложеност у моделу.

2.2.2.3.3. Закључак анализе метода за рангирање захтева

Карактеристике наведених метода су сумарно представљене у Табели 6. За сваку од њих су приказани подаци о врстама улазних и излазних података, да ли подржавају условљености у захтевима или било која специфична врста захтева, као и временске сложености за наведене упите резоновања. Јасно је да не постоји свеобухватно најефикаснији метод/техника за анализу упита резоновања над дефинисаним захтевима, па се одабир одговарајуће технике врши сагледавањем карактеристика специфичног проблема (тј области примене) који се решава.

Осим метода за анализу корисничких захтева, у литератури постоје анализе начина за само представљање захтева, с посебним акцентом на условно дефинисане захтеве. Како условно дефинисана реченица у природном језику може имати више различитих интерпретација, условно дефинисани захтев такође може бити различитоинтерпретиран од стране различитих корисника; о семантици захтева ће бити више речи у поглављу 2.3.

2.2.2.4.Зависности међу захтевима и условљености

Како у теорији, тако и у пракси је показано да су захтеви међусобно повезани и да такви односи међу њима директно утичу на поједине фазе развоја софтвера. То даље значи да чак и они најиндивидуалнији захтеви, дефинисани у ранијим фазама не могу бити изоловани и засебно анализирани [123]. Уместо тога, они су у међусобној вези и међусобном утицају на знатно комплекснији и суптилнији начин [93] [124]. Анализе су показале да само пети део свих дефинисаних захтева је заиста сингуларан, тј независтан и некорелисан са осталима[93]. Зависности међу захтевима могу бити описане као посебан сегмент следљивости захтева[125].

У литератури постоји мношто дефиниција појма зависности међу захтевима, а једна од њих је: „способност описа и праћења животног циклуса захтева у оба смера (напред и назад), у идеалном случају током целог животног циклуса система“ [125].

Постоје разне врсте зависности међу захтевима [126]:

- Структурне зависности које подразумевају организацију скупа захтева у облику структуре у оквиру које су зависности (односи) хијерархијске или су пристуне међу самим структурама. Пословни захтеви високог нивоа се постепено декомпонују на захтеве за развој софтвера који садрже више детаља, чиме се добија хијерархија. Такође, могу постојати структурне релације међу захтевима у оквиру различитих делова и нивоа хијерархије.
- Зависности ограничења које осликавају ситуације када неки захтев може да дефинише ограничење над другим. Постоје две најважније врсте оваквих зависности: *1)*захтевање (енгл. *requires)* које дефинише зависност испуњења једног захтева у односу на испуњење другог и, *2)* конфликт (енгл. *conflicts with)* који дефинише да два захтева не могу истовремено постојати, или пак да повећање задовољења једнога директно смањује задовољење другог захтева.
- Зависности цена/вредност које се односе на цену имплементације датог захтева у односу на вредностодређеног степена испуњења датог захтева.

Јасно је да се дефинисане врсте зависности међу захтевима никада не појављују појединачно, тј пуно чешће је више њих у комбинацији, па су тиме знатно теже за идентификацију и уочавање. Додатно, идентификација условљености и повезаности међу захтевима од директног је утицаја на све активности анализе захтева, почев од саме спецификације и декомпозиције захтева, преко дизајна и имплементације, која је посебно осетљива на конфликте у захтевима и неконзистенције у њиховим међусобним односима. У литератури такође постоји читав скуп метода развијених за анализу, уочавање и след зависности међу захтевима [123] [127] [128].

2.2.2.5.Балансирање захтева различитих интересних група

Балансирање захтева се у многим дисциплинама сматра од пресудног значаја, па су многе технике и методе развијене у том циљу. Примарни циљ процеса балансирања међу захтевима се односи на идентификацију и решавање конфликата међу захтевима разних (група) корисника. То је у потпуној сагласности са циљем дефинисања изводљивих и међусобно усклађених захтева који представљају све жеље и захтеве корисника над датим системом.

Примарни проблеми процеса балансирања захтева се односе на следећа питања [129]: Како конфликти могу бити идентификовани? Како идентификовани конфликти могу бити разрешени? Како се могу одредити прихватљиве алтернативе? Ко је задужен за идентификацију и вршење процеса балансирања, сами актери или моделатори? Које технике могу да подрже процес балансирања захтева и како их употребити?

Одговори на дефинисана питања се налазе у употреби метода и техника из разних области, дисциплина и домена.Овде се додатно намеће формална верификација промене захтева код корисника/наручиоца који са различитим циљевима, одговорностима и погледима на дати систем дефинишу захтеве [130]. Постоји читав скуп различитих приступа, почев од крајње формалног који аутоматизованим процесом детектује конфликте (тј неконзистенције) у захтевима дефинисаним од стране разних корисника и учесника, преко приступа једноставног усклађивања знања из перспективе корисника са доменским система, до образаца конфликата којимасе обезбеђује систематичан приступ њиховом разрешавању. С друге стране, крајње формално представљање захтева изузетно много напора да се представе сва знања система, обезбеди валидација и одговарајуће методе резоновања.

Насупрот овом крајње формалном приступу, неформалан (или полу-аутомстки) приступ познат под занивом *Win-Win*[131][132]представља генерални приступ балансирању захтева који не захтева претходно моделовање знања, резоновање над њим и валидацију. Међутим, овај приступ не омогућује систематски приступ разрешавању конфликата, већ када се конфликт идентификује, настаје ad-hoc процес одабира алтернативе за његово разрешење.

Модел *WinWin*користи теорију*Theory W*[133] познату под називом „направи свакога победником“која кориснику прво омогући да сам идентификује победничке услове, па се затимбалансирају потенцијални конфликти међу дефинисаним условима. Употребом ове теорије у *WinWin*моделу, корисници су укључени у процес идентификације и реализације начина балансирања захтева, тј.разрешавање конфликата међу дефинисаним победничким условима.

Процес идентификације и разрешавања конфликата може бити извршен током самог процеса рангирања захтева. У том случају, методе које се примењују за рангирање и анализу захтева треба да садрже као своју интегралну компоненту начине за идентификацију и разрешавање конфликата. Тако нпр. AHP метод [77]дефинише два различита приступа анализи захтева добијених од стране разних корисника из различитих интересних група: *(1)*појединачно рангирање захтева добијених од стране сваког од корисника, након чега се врши спајање добијених резултата;*(2)*прво се врши спајање свих захтева, након чега се врши рангирање интегралног скупа захтева свих корисника укључених у процес развојасистема. Оба приступа карактерише комбиновање захтева свих корисника са одговарајућим фактором који дефинише значај корисника (групе) у процесу одлучивања, без обзира на потенцијалне конфликте који могу настати у њиховим захтевима.

Ни једна од метода представљених у поглављу 2.2.2.3. које се заснивају на структури графа, не може користити принцип решавања конфиката сличан наведеном. Разлог је у чињеници да је за презентацију већине врста конфликата управо граф најпогоднија структура података којом се такође индукују циклуси који директно нарушавају процес рангирања.

2.2.2.6.Обезбеђење квалитета анализе захтева

Традиционално, активности осигурања квалитета (енгл. *Quality Assurance* (QA))суфокусиране на касније фазе развоја, као што су имплементација и тестирање. Међутим, одређене активности је потребно почети знатно раније, чак у самој фази анализе захтева.

Дефиниција самог квалитета над захтевима постаје јако сложена ако се има у виду да сам појам квалитета директно зависи од мишљења и ставова корисника. Нпр, уколико захтеви корисника нису схваћени на прави начин, може се развити систем који треба сматрати системом лошег квалитета с обзиром да не одговара захтевима и ставовима корисника [134].

Табела 7. Атрибути квалитета захтева [134]

Quality Attribute	Definition
Correctness	The implemented requirements have to reflect the behavior expected (intended) by the users and customers. That is, everything stated as a requirement is something that shall be met by the final system to fulfill a certain purpose (suitability).
Unambiguity	Requirements should have only one possible interpretation. Note that one requirement might be unambiguous to a certain group of stakeholder but has a different meaning in another. It is important to involve all stakeholders in the requirements engineering process to gain a common understanding.
Completeness	All elements that are relevant for fulfilling the different user's tasks should be considered. This includes relevant functional and non-functional requirements and interfaces to other systems, the definition of responses to all potential inputs to the system, all references to figures and tables in the specification, and a definition of all relevant terms and measures.
Consistency	The stated requirements should be consistent with all other requirements, and other important constraints such as hardware restrictions, budget restrictions, etc.
Ranked for Importance / Stability	Each requirement specifies its importance and/or its stability. Stability expresses the likelihood that the requirement changes, while importance specifies how essential the requirement is for the success of the project (from a value based and user point of view).
Verifiability	All requirements should be verifiable. That is, there should be a process to be used by a machine or a human to check (in a cost effective way) whether a requirement is fulfilled or not.
Modifiable	All requirements should be modifiable, that is the structure of the requirements and the requirements specification should allow for the integration of changes in an easy, consistent and complete way.
Traceable	All requirements should be traceable, that is, it should be possible to reference the requirement in an easy way. Moreover, it should be possible to identify the origin of a requirement
Comprehensibility	The requirements are specified and phrased in a way that is understood by all involved stakeholders.
Feasibility	All requirements can be implemented with the available technology, human resources and budget. Moreover, all requirements contribute to the monetary success of the system, that is, they are worth to be included in the system.
Right Level of Detail	The information given in the requirements is suitable to gain the right understanding of the system and to start the implementation. There are no unnecessary implementation or design details specified in the requirements.

IEEE Стандард спецификације захтева[71] дефинише аспекте квалитета захтева који су наведени у првој колони Табеле 7, а на основу [134] су у другој

колони наведене дефиниције атрибута квалитета прилагођене контексту захтева корисника.

Приступи за обезбеђење квалитета треба да провере да ли захтеви задовољавају дефинисане карактеристике квалитета. Највећи проблем се огледа у чињеници да не постоје правила на који начин је то могуће урадити. Нпр, за један атрибут квалитета може постојати више различитих алата.

У овој дисертацији ће се користити формално развијени концепти засновани на елементима Дескриптивне Логике (енгл. *Descriptive Logic*), где се полази од чињенице да је корисник у могућности да своје захтеве формално дефинише, или пак, да постоји начин за недвосмислену формалну презентацију захтева. На тај начин се највећи акценат поставља на сам процес анализе формално дефинисаних захтева корисника, конфликата и условљености међу захтевима, као и максимизацију њиховог испуњења.

2.2.3. SOA фамилије и захтеви корисника

У претходном делу овог поглавља наведене су, у теорији познате врсте корисничких захтева које је потребно идентификовати у контексту SOA фамилија и представљеног начина за њихово моделовање. Стога, остатак ове секције је организован на следећи начин: у 2.2.3.1. дат је преглед могућих врста нефункционалних и функционалних захтева у контексту SOA фамилија, на основу чега је у 2.2.3.2. формално дефинисан проблем конфигурације SOA фамилије и дат преглед литературе са досадашњим резултатима релевантним за дефинисани проблем.

2.2.3.1.Врсте захтева над SOA фамилијом

Као што је представљено у поглављу 2.1.6.1. модел одлика (FM)обезбеђује спецификацију тражених функционалности система. На основу захтева чије је испуњење неопходно обезбедити (тзв. строги захтеви), идентификују се обавезне одлике (које морају бити укључене у сваку инстанцу фамилије), док су остале

одлике опционе и оне могу, а не морају бити укључене у инстанце фамилије. Такође, FM обезбеђује задовољење основних врста условљености (*include/exclude*) међу функционалностима (фаза у анализи корисничких захтева описана у поглављу 2.2.2.4).

Како је наведено у поглављу 2.1.6., процес одређивања конфигурације одлика подразумева одабир скупа одлика који задовољава ограничења дефинисана FM моделом. Имајући у виду величине савремених фамилија софтверских система (које могу имати реда величине 10^3 одлика [57]), провера сваке инстанце представља скуп процес који захтева велики утрошак времена, уз велику вероватноћу генерисања грешке при мануелној валидацији. Из тог разлога је развијен аутоматски начин верификације [60] који се заснива на елементима Дескриптивне Логикекоја је омогућила формализацију ограничења у моделима и валидацију процеса селекције сервиса. На тај начин је доказано да, свака конфигурација сервиса која садржи по један атомичан сервис за сваку од одлика (у конфигурацији одлика, тј. инстанци фамилије) која је валидна у односу на ограничења FM модела, представља валидну конфигурацију сервиса дате SOA фамилије. С друге стране, сваки од атомичних сервиса је окарактерисан атрибутима квалитета, за које корисник може дефинисати разне врсте захтева. Дати захтеви се могу односити на целу SOA фамилију или на поједине њене делове. Као што је претходно наведено, захтеви могу бити дефинисани са разним приоритетима, од захтева којима корисник дефинише само своја идеална очекивања и жеље, до захтева којима жели строго ограничити вредности појединих атрибута квалитета. У том случају, реч је о строгим захтевима, који се у литератури [135]представљају са: $cl_i(e_1,...,e_{|A|}) \leq u_i$, а у случају да је дефинисано ограничење најмање вредности (тзв ограничење с „доње стране“), оно се трансформише (множењем с негативним множиоцем) и такође представља у наведеном облику.

2.2.3.2.Проблем конфигурације SOA фамилије

На основу наведених функционалних и нефункционалних захтева у домену SOA фамилија, проблем конфигурације се дефинише као:

Дефиниција 5 (Конфигурација SOA фамилије). ***Задатак конфигурације SOA фамилије*** представља задатак конструкције комбинације сервиса који имплементирају инстанцу SOA фамилије валидне у односу на ограничења FM модела и дефинисане строге захтеве о нефункционалним карактеристикама уз истовремену максимизацију испуњења дефинисаних захтева о жељеним вредностима нефункционалних карактеристика.

У складу са датом дефиницијом, намећу се два приступа за решавање: један који тежи ка истовременом испуњењу дефинисаних функционалних и нефункционалних захтева, и други који прво реализује захтеване функционалне карактеристике и за добијену инстанцу фамилије врши одабир сервиса жељеног квалитета [136].

Значи, проблем конфигурације SOA фамилије се може решити итеративно: прво одабиром конфигурације одлика и касније њеном конфигурацијом одабиром сервиса који је имплементирају. Други начин за решавање је паралелни тј истовремени одабир конфигурације одлика и њене даље конфигурације на основу скупа расположивих сервиса. Како се моделом одлика може дефинисати велики број опционих одлика (тј варијабилних карактеристика система), од посебног значаја се сматра обезбеђење другог наведеног приступа у решавању дефинисанох проблема.

По досадашњим сазнањима, проблем конфигурације тј одабира сервиса је анализиран само појединачно у областима пословних процеса [137]и модела одлика[135], као и у области ланаца процеса заснованих на догађајима (енгл. *event-driven process chains*) [138]. Приступ представљен у [138] даје само делимично решење. Фокусирано је само на моделовању варијабилности, засновано на конкретном језику, без обезбеђења механизма за аутоматско или

полу-аутоматско конфигурисање и одабир сервиса у зависности од њихових QoS вредности.

Већина анализа које се односе на проблем селекције сервиса на основу нефункционалних карактеристика разматра једноставну тежинску шему (енгл. *Simple Additive Weighting* (SAW)) која осликава најједноставнији облик преференци корисника. Оне одговарају претходно наведеним растућим/опадајућим тенденцијама група нефункционалних карактеристика, па се тежински фактори односе на дефинисање приоритета сваке од група (и/или карактеристика унутар групе) у процесу одабира највише преферираних врста нефункционалних карактеристика. С друге стране, јасна је неопходност интеграције сложених и софистицираних врста захтева, што, на основу досадашњих сазнања, није адресирано ни у једној од поменутих области примене. Следећа истраживања се такође могу посматрати од интереса за проблем нефункционалних карактеристика и анализе захтева над њима:

Godse et al. [139]су представили технику за груписање QoS атрибута у кластере на основу сличности њихових типова, и тако добијени кластери представљају чворове на нивоу листова у хијерархији. Употребом AHP метода, ранга вредности за сваки од QoS атрибута и података о заступљености датих атрибута у карактеризацији сваког од расположивих сервиса, врши се рангирање сервиса и крајњи одабир сервиса који у већој мери задовољава дефинисане захтеве.

С друге стране, Srivastava et al. [140] представљају технику за поређење сервиса исте функционалности на основу колективне вредности QoS атрибута. Овом техником се врши поређење сервиса на основу преференци дефинисаних од стране корисника у односу на вредности сваког од QoS атрибута. У овом раду се такође разматра чињеница да повећање вредности неког од атрибута нема линеаран утицај на повећање/смањење задовољења преференци корисника, што у односу на претходне анализе и разматрања,у знатно већој мери одговара реалним ситуацијама.

2.3.Представљање знања и методе за резоновање

Семантичке технологије сеобично дефинишу као технологије које спецификују значење одвојено од самог податка и независно од саме имплементације [141].Ове технологије обезбеђују апстрактни слој којимсе успостављају семантичке релације међу разним подацима, садржајима и процесима. Стога, оне могу да обезбеде знатно интелигентнију, моћнију и релевантнију интеракцију у односу на стандрадне технологије.

Семантичке технологије су погодне за употребу у било ком сценарију који захтева разумевање контекста и самог значења података, као и међусобних релација, где постоји потреба и за представљањем могућих динамичких променљивости. Показано је да се применом конвенционалних релационих модела не могу обезбедити флексибилни динамички приступи за карактеризацију података [142].

2.3.1. Представљање знања и методе за резоновање

Представљање знања (енгл. *Knowledge Representation)* се сматра једним од принципијелних елемената области вештачке интелигенције и круцијалним елементом решавања свих врста задатака [143].

Представљање знања из датог домена се састоји од дефинисања:*i)базе знања* која садржи симболе концептуалног модела представљене у форми исказа о посматраном домену, и *ii)резоновања* којим се дефинише начин за манипулисањем датим симболима. Технике аутоматског резоновања омогућавају рачунарским системима извођење закључака из представљеног знања у форми разумљивој машинама [144].

Уколико се анализирају тренутне семантичке технологије, представљање знања се појављује у различитим формама, а најрелевантније међу њима су засноване на примени *семантичких мрежа, правила, онтологија* и *логике*. Логика се користи за формалну спецификацију семантичке интерпретације свих осталих наведених форми. Успостављањем формалне спецификације језика за

представљање знања, формализми засновани на логици постају основа за аутоматизацију процеса дедукције (тј. процеса доношења закључака на основу претпоставки и правила извођења)[144]. У овој дисертацији се користе онтолошке структуре, правила за представљање захтева и елементи логике за њихову формализацију, тако да се у наставку наводе њихови кратки описи.

2.3.1.1.Онтологије

Термин онтологија (енгл. *ontology*) је први пут уведен у филозофији у деветнаестом веку до стране немачког филозога Rudolfa Gockela, у његовом раду Lexicon Philosphicum, да би се означила разлика између проучавања бића односно постојања као таквог у односу на различите врсте бића у природним наукама [145].

У рачунарским наукама онтологије су усвојене да омогуће размену и поновно коришћење знања. У овој области су онтологије постале важан појам у деведесетим годинама двадесетог века и нашле су примену углавном у домену аквизиције знања. У овом контексту, Gruber је 1993.год дао основну дефиницију онтологија на следећи начин: „Онтологија је експлицитна спецификација концептуализације“ [146]. Borst (1997) је дефинисао онтологију као формалну спецификацију дељене концептуализације, да би се у [147] спојила у сада доминантној дефиницији онтологија [148]: „Онтологија представља формалну експлицитну спецификацију дељене концептуализације посматраног домена“. Ова дефиниција указује нанеколико важних карактеристика онтологија као спецификација доменског знања:

- *Формалност*: онтологија омогућује представљање знања које има формалну семантику;
- *Експлицитност*: онтологијомје знање екплицитно представљено у циљу могућности разумевања од стране машина;
- *Дељеност*: онтологија осликава договор о доменској концептуализацији међу учесницима којима је датидомен од интереса;
- *Концептуалност*: онтологија спецификује знање на концептуалном нивоу у облику симбола који предсатвљају концепте и релације међу њима [144].

Уместо дефинисања исказа о специфичним ситуацијама које се односе па појединачне индивидуе, онтологијом се обухватају све ситуације које се потенцијално могу идентификовати[149]

- *Доменска специфичност*: спецификација онтологије је ограничена знањем о појединим домену који се анализира. Експлицитна спецификација доменског знања може бити модуларно представљена са више различитих онтологија са различитим под-доменима од интереса.

Данас су онтологије широко прихваћене и представљају својеврстан стандард за представљање знања у системима за одлучивање. Технички, основни елементи онтологије су: концепти (енгл. *concepts),* релације (енгл. *relations*) и инстанце (енгл. *instances*). Концепти представљају онтолошке категорије релевантне за посматрани домен. Аналогије концептима се срећу и у осталим моделима за представљање знања, нпр чворовима у семантичким мрежама (енгл. *semantic networks*), унарним предикатима у логици (енгл. *logic*) или концептима у дескриптивној логици (енгл. *description logics*). Релације семантички повезују концепте као и инстанце, спецификујући њихове релације. Релацијама одговарају гране у семантичким мрежама, бинарни предикати у логици или улоге у дескриптивној логици. Инстанце представљају именоване и идентификоване конкретне објекте у посматраном домену, тј посебне индивидуе које су класификоване помоћу концепата. Аналогија инстанцама су индивидуални чворови у семантичким мрежама и константе у логици. Наведени елементи чине онтолошки вокабулар за посматрани домен. Стога, онтологија може бити посматрана као скуп исказа, представљених помоћу појмова из вокабулара, који се називају аксиомима [144]. Иако наведени концепти имају аналогију у појмовима из ранијих модела, главна предност и разлог велике применљивости онтологија се огледа у могућностима спецификације знања специфичног домена [150]. Концепти и својства могу имати хијерархијску структуру. Концепт може бити подконцепт неког другог концепта, а својство може бити подсвојство неког другог својства.

Дескриптивна логика(енгл. *description logic*) само је један од формализама који има могућност обликовања онтолошке структуре. Основна му је карактеристика

да има семантику засновану на логици што омогућава опис појмова логичким изразима. При томе долази до изражаја могућност извођења новог (имплицитно присутног) значења из постојећег (експрилицитно наведеног) значења коришћењем алата за аутоматско резоновање. Данас је дескриптивна логика поново у фокусу истраживања управо због појаве нових језика и алата за обликовање онтологија у дескриптивној логици, првенствено у домену технологија семантичког веба. Као последица тога, све више се у осталим сродним областима истраживања користе језици и алати засновани управо на дескриптивној логици. Нпр, уобичајено је да се при изградњи базе знања у систему за одлучивање користи OWL формализам (енгл. *Web Ontology Language*) и Protégé-OWL алат за креирање онтологија, иако је примарна намена ових алата и језика номинално у неком другом домену [151].

2.3.1.2.Резоновање засновано на правилима

Још један од природних начина запредстављање знања у посматраном домену је заснован на правилима која осликавају појам последица. Правила се представљају у форми ако-онда (енгл. *IF-THEN*) конструкција које обезбеђују представљање исказа произвољне сложености. Правила се срећу у системима за логичко програмирање (енгл. *logic programming systems*) [152], дедуктивним базама података (енгл. *deductive databases*) [153] и многим другим.
Општи облик запредстављање правила је:

IF логички_израз THEN акција1/закључак1
ELSE акција2/закључак2

Обично, системи који користе правила за представљање знања полазе од скупа чињеница (које се сматрају безусловним правилима) и применом дефинисаног скупа правила изводе се нове чињенице тј закључци. Главним задатком таквог система сматра се одређивање подскупа скупа свих правила чији

су услови задовољени на основу датих чињеница и полазних информација (тзв. задовољени подскуп правила). Након идентификације таквог подскупа правила, врши се одабир тачно једног правила које се извршава. Јако су честе ситуације да правило које је кандидат за извршавање није јединствено одређено, чиме се појављује конфликт у извршавању правила; а подскуп таквих правила се назива конфликтим скупом (енгл. *conflict set*). Одабир правила из конфликтног скупа се врши на основу стратегије за разрешавање конфликата. Познате су разне стратегије, а најчешће применљиве су следеће [154]:

Стратегија првог правила: Уколико су правила сортирана по неком специфичном редоследу, примена прво рангираног правила је стратегија која увек јединствено одређује правило које ће бити примењено. Реч је о најједноставнијој стратегији која може узроковати проблем бесконачне примене истог правила. За разрешавање датог проблема, обично се користи додатна стратегија елиминације сваког правила након прве примене.

Стратегија насумице одабраног правила: Заразлику од претходне стратегије, ова стратегија нема карактеристику предиктивности. У неким случајевима примена насумично одабраног правила за извршење може имати предности, као нпр. у системима заснованим на теорији игара исл. Сваком правилу се може доделити и одређена вероватноћа извршења, чиме се дефинише фази систем правила (енгл. *fuzzy rule-based system*) [155]

Стратегија одабира нејспецифичнијег правила: Ова стратегија је заснована на карактеризацији правила на основу сложености услова; тј. врши се одабир правила са највише услова у IF делу правила. Овакав приступ је оправдан претпоставком да, уколико се у услову који је задовољен, појављује највише различитих чињеница, значи да је реч о правилу који је најрелевантније за дати скуп чињеница.

Стратегија одабира најмање коришћеног правила: Овом стратегијом сваком правилу се додељује временски идентификатор који представља временску одредницу његове последње употребе. На овај начин се максимизује број индивидуалних правила која су извршена најмање једном. Ова стратегија је најпогоднија у системима у којима је неопходно да сва правила буду извршена.

Стратегија „најрелевантнијег“ правила: Овом стратегијом сваком правилу се додељују тежинске вредности које представљају релевантност датог правила у односу на алтернативна. Правило са највећим тежинским фактором се бира за извршавање, а у случају постојања више правила са истоветним тежинским вредностима, примењује се нека од претходно наведених стратегија.

Након одабира правила које се извршава, скуп чињеница се проширује новим чињеницама која су добијенекао последица примене датог правила. Поступак се наставља све док скуп чињеница не постане непроменљив или не постојини једно правило које би се могло применити.

Из наведеног је очигледно да одабир стратегије за разрешавање конфликата директно зависи од домена примене. На пример, посматрајмо правила која дефинишу приоритет међу опцијама које се пореде. У том случају, интерпретација ситуације када је више од једног правила задовољено представља поређење две опције на два различита начина, тј. са две различите вредности, што се сматра неконзистенцијом у захтевима. С друге стране, ситуација када ни једно правило није задовољено се може сматрати као индиферентност о односима међу датим опцијама, што води ка имплицитној дефиницији једнакости приоритета посматраних опција [156].

Наведене ситуације могу имати потпуно другачије значење уколико се правилима дефинишу, нпр препоруке у одређеној области/домену. Тада, две различите препоруке (дефинисане различитим правилима) не представљају неконзистенцију, већ обе могу бити сматране адекватним. Недостатак правила којим се дефинише препорука, се сматра неконзистенцијом у систему препорука или пак индиферентношћу и препоруком било које од опција из целог датог система.

Осим представљеног проблема разрешавања конфликата, у теорији је познат и проблем генерисања задовољеног подскупа правила. Најједноставнији начин, а истовремено начин са највећом сложеношћу извршавања је *brute-force method* који подразумева проверу редом свих правила. Сложеност овог метода је реда $(R \cdot A)^C$, где јеRброј правила, Cје просечан број услова у сваком правилу,

a*A*просечан број нових чињеница по правилу[5]. Дата експоненцијална сложеност чини да системи засновани на правилима имају лоше перформансе извршавања [154].

У теорији је познато неколико начина за оптимизацију који чине системе засноване на правилима применљивима у реалним системима. Најефикасније решење је предложено са *Rete* алгоритмом (1982)[157]. Овим алгоритмом се врши смањење броја поређења услова у захтевима и чињеница које се налазе у радној меморији. Алгоритам креира листу услова који су потпуно или делимично задовољени тренутно расположивим чињеницама, па се провера нових услова заснива на провери претходно делимично задовољених (само за нове чињенице додате у меморију) или провером услова за нова правила (која нису претходно проверена). Сложеност овог метода је линеарна *O(RAC)*, што представља знатно унапређење у односу на претходно наведену експоненцијалну сложеност.

Бенефити примене IF-THEN правила се огледјау у њиховој модуларности, тј правила су обично мала и одговарају једном малом делу знања о целом систему [158]. Овакви системи су изузетно погодни када се резоновање врши на конкретним инстанцама. Међутим, додатни проблеми се огледају у сложенијим и генералним исказима који могу бити изведени у датом домену, а истовремено не одговарају ни једном дефинисаном правилу.

2.3.2. Методе за решавање оптималних задатака

У теорији, под задатком оптимизације се подразумева комбинаторни задатак дефинисан на следећи начин:

$$\begin{aligned} &Q(x) = Q(x_1, x_2, \ldots, x_n) \to \min \\ &g_i(x) \geq 0, i = \overline{1, m} \\ &h_j(x) = 0, \; j = \overline{1, p} \end{aligned} \qquad \textbf{(1)}$$

где је *Q(x)* функција циља, а $L = \left\{ \{g_i\}_{i=\overline{1,m}}, \{h_j\}_{j=\overline{1,p}} \right\}$ скуп ограничења. Уколико је

[5]http://ai-depot.com/Tutorial/RuleBased-Methods.html

функција циља линеарна као и функције садржане у ограничењима, задатак **(1)** је задатак линеарног програмирања.

Ако је бар једна од функција нелинеарна, тада се добијени проблем назива задатком нелинернаог програмирања. Ако задатак нема услова, реч је о безусловном оптималном задатку , а у супротном се решава проблем условне оптималности.

2.3.2.1.Целобројно линеарно програмирање

Линеарно програмирање (енгл. *linear programming (LP)*)представља често примењивани начин за моделовање задатака оптимизације. На овај начин, проблем се своди на налажење минимума или максимума линеарне функције *Q(x)* при одређеним линераним ограничењима (представљеним у облику линеарних једначина и/или неједначина). Посебна класа ових задатака је у случају да су вредности непознатих $x_1, x_2, ..., x_n$ целобројне, познате под називом целобројно линеарно програмирање (енгл. *integer linear programming (ILP)*). Такође, у случају да вредности непознатих могу бити само из скупа {0, 1} реч је о класи задатака познатих под називом 0-1 ILP.

Постоје разне методе за решавање ILPзадатака. Једна од најпознатијих је Dantzig-ова метода, позната под називом симплекс метода (енгл. *simplex method*), као и многе њене модификације.

Генерално, методи који су развијени на базиILPметода омогућавају налажење само локалних екстремума, а у пракси постоји само неколико типичних задатака који се успешно решавају поменутим алгоритмима, као што су: задаци расподеле ограничених ресурса, неки транспортни задаци, задаци везани за управљање залихама, идр.

На крају, за метод ILP је показано да има добре перформансе само за решавање мањих комбинаторних проблема[136], што додатно представља проблем применљивости у општем случају.

Знатно реалнија ситуација је када су функција циља *Q(x)* и/или функције из скупа ограничења L дефинисане нелинеарно, чиме је представљен задатак нелинеарног програмирања. За разлику од задатака линеарног програмирања, ови

задаци се не могу решавати применом неког универзалног метода, већ се за сваки конкретан случај, у зависности од конструисаног математичког модела, димензија и каракетристика нелинеарности, развија нови модел или се прилагођава неки од постојећих метода. У великом броју случајева, чак и не постоји прикладан метод на основу којег се може наћи оптимално решење формализованог задатка, па се решење налази у неким од метода за приближно решавање (које ће бити представљене у наставку).

Ипак, за неке специфичне случајеве, као нпр за линеарна ограничења и нелинеарну функцију циља, за функције циља задате квадратном формом или за целобројне промењљиве уз ограничења конвексности (конкавности) функције циља и конвексности (конкавности) области која је одређена скупом ограничења [159], конструисани су конкретни алгортими за решавање.

2.3.2.2.MMKP проблем

MMKP (*Multi-choice Multi-dimension 0-1 Knapsack*) проблем [160] је комбинаторни проблем, дефинисан на следећи начин:

Нека је на располагању K група, свака група има $l_i (1 \le i \le K)$ елемената, при чему сваки елемент даје профит p_{ij} а захтева $r_{ij} = (r_{ij_1},..., r_{ij_m})$ ресурса. Укупна количина распложивих ресурса је $R = (R_1,..., R_m)$. Задатак MMKP се дефинише као одабир тачно једног елемента из сваке групе, уз поштовање расположивих ресурса и крајњи циљ максимизације профита.

Проблем MMKP је такође NP сложен [160]. У литератури је познато неколико различитих алгоритама који се користе за решавање овог задатка. Најчешће се употребљава *Branch and bound algorithm* алгоритам [161]који се заснива на креирању стабла претраге у циљу генерисања решења. Чвор у таквом стаблу претраге представља потенцијално решење, а нови кандидати се добијају изменама вредности појединих елемената у посматраној комбинацији при фиксним вредностима осталих. Овај алгоритам има велику сложеност, па није

погодан за решавање задатака велике величине, тј време извршавања расте експоненцијално у односу на број непознатих у моделу.

С друге стране, хеуристички алгоритам (енгл. *heuristic algorithm (HEU)*) [162] се заснива на одабиру једног изводљивог решења које се, затим, итеративно побољшава заменом појединачних елемената у циљу повећања вредности функције корисности (енгл. *utilityfunction*). Ако не постоји такав елемент, покушава се са заменом целе групе елемената, итд. Проблем примене овог алгоритма је често генерисање некоректних решења, па је развијен приступ који користи интергацију са одговарајућим граф моделом, тзв *WE-HEU*[163]. На тај начин, проблем селеције је трансформисан у проблем налажења путање у усмереном ациклином графу који даје највећу вредност функције корисности. Симулационе анализе су показале да *WE-HEU* има за 50% боље перформансе од *HEU*[164].

Слично као и код метода за анализу захтева корисника, алгоритми за ММКР који се заснивају на структури графа и алгоритмима за њихов обилазак, имају недостатке који се огледају у лошим перформансама у случајевима велике мреже (тј. великог броја чворова у мрежи) и могућим циклусима који директно утичу на знатно смањење перформанси.

2.3.2.3.Генетички алгоритми

Генетички алгоритми (енгл. *Genetic Algorithms* (GA)) су група метода за решавање комбинаторних проблема употребом алгоритама који симулирају нео-Дарвинову теорију еволуције. Реч је о итеративној процедури над популацијом константне величине. Свака индивидуа (тј елемент популације) који се назива хромозом (енгл. *chromosome*) представља потенцијално решење. Нови елементи популације се добијају применом оператора рекомбинације (енгл. *crossover*) и мутације (енгл. *mutation*) над елементима старије генерације, на тај начин што се од два родитељска хромозома комбинацијом њиховим битова (тј генома) добија једна или више нових индивидуа, тј. њихових потомака (енгл. *offspring*). Над њима се примењује оператор мутације како би се избегла конвергенција у локални оптимум, на тај начин што се на случајан начин врше модификације појединих

генома потомака. Тако добијени скуп потомака се евалуира помоћу функције циља (енгл. *fitness function*) која осликава преференце и сам задатак оптимизације који се решава. За индивидуе које имају веће вредности функције циља додјељују се веће вредности вероватноћа за учешће у даљим репродукцијама.

Algorithm 1. ***Genetic algorithm***

Input: Feature model FM, BPM family, set of available services S_{a_k}

Output. optimal configuration

Begin

```
initialize population;
  evaluate population;
while TerminationCriteriaNotSatisfied
  do
    select parents for reproduction;
    perform crossover and mutation;
    evaluate population;
  end while;
End.
```

Слика 17. Генетички алгоритам

Овај циклус се понавља све док се не генерише тражено решење, или се задовољи неки од критеријума заустављања (обично максимални дозвољени број итерација), а основни кораци аглоритма су представљени на Слици 17.Један од најзначајнијих елемената адаптације GA у конкретном домену примене је дефиниција функције циља [165] која осликава перформансе сваке појединачне индивидуе [166]. У литератури је познато више истраживања о начинима за генерисање фунцкије циља [167] [168] и опште применљива препорука се односи на дефинисање казнених пенала за све индивидуе које не задовољавају ограничења дефинисана у задатку [166]. Тиме се постиже изузетна моћ функције циља да обухвати разне врсте ограничења које могу понаособ дефинисати разне врсте казнених пенала. У таквим случајевима се препоручује употреба релативних вредности које одговарају интуитивним пресудама о релативном значају

задовољења одговарајућих захтева [165]. Такође, многа истраживања се односе управо на начине за дефинисање казнених пенала [169], при чему већина обухвата динамичко (у односу на број итерација) генерисање пенала на основу принципа заснованог на мерењу растојања.

Показало се да GA алгоритми имају велике могућности за решавање већих комбинаторних проблема[137], док су симулационе анализе показале да у случајевима мањих комбинтаорних задатака ILP имају боље перформансе у односу на GA [137].

2.3.2.4.Закључак анализе метода за решавање задатака оптимизације

У претходним секцијама је наведено неколико различитих модела и алгоритама за представљање и решавање разних врста задатака оптимизације. За сваки од приступа су у литератури познати проблеми који су његовом применом ефикасно решени, међутим не постоји опште применљиви модел нити метод за решавање овог задатка у различитим доменим примене.

Табела 8. Компаративна анализа модела/метода/техника за решавање задатака оптимизације

Метод/ техника	***Захтеви корисника***	***Функција корисности***	***Строги захтеви***	***Временска сложеност***	***Динамичке промене***
Целобројно линеарно програмирање	Статички	Линеарна	Линеарни	Погодно за решавање проблема мале комбинаторне сложености	-
Генетики алгоритми	Квантитативна мера може бити садржана у функцији корисности	Нема ограничења линеарности	Садржано у функцији корисноти у облику казнених пенала	Погодно за решавање проблема изузетно велике комбинаторне сложености	Непотпуно адресирани
ММК-Р	Статички	Линеарна	Линеарни	Погодно за решавање проблема мале комбинаторне сложености	-

Анализе представљених метода/техника су сумиране у Табели 6, а анализа је извршена на основу: *1)* различитих врста захтева који могу бити интегрисани у функцију циља, *2)* присутности ограничења над функцијом циља, *3)* начина за представљање строгих захтева, *4)* сложености метода и *5)* могућности за динамичке промене у задатку који се решава.

Као општи закључак дате анализе се може навести да:

(1) Задатак оптимизације датог модела (из посматраног домена примене) треба трансформисати у облик сличан неком од наведених, у теорији познатих задатака;

(2) Приступ за решавање задатка треба добити адаптацијом неког од познатих приступа у складу са доменом примене.

2.3.3. SOA и проблеми резоновања и оптималне претраге

У овом поглављу се наводи преглед досадашњих резултата из области примене семантичких технологија и метода за решавање задатака оптимизације у области конфигурације SOA фамилија и других сродних области.

2.3.3.1.Семантичко моделовање карактеристика квалитета SOA фамилије

Као што је наведено у поглављу 2.2.3, досадашња истраживања су била фокусирана појединачно на модел одлика FM и шаблон SOA фамилија.

Многи истраживачи су предложили управо QoS атрибуте за унапређење процеса селекције веб сервиса, а њихово моделовањеи спецификације су разматране из различитих аспеката. Неки од њих су засновани на проширењу постојећих стандарда WSDL[6] и UDDI[7] како би се представиле карактеристике веб сервиса, нпр[170][171][172] [173]. Такође, у[174]је разматрано доменски независно

[6]E. Christensen et al. Web Services Description Language(WSDL) version 1.1. http://www.w3.org/TR/wsdl, 2001.

[7]Latest UDDI Version (3.0.2), UDDI SpecTechnical Committee Draft, Dated 20041019.http://uddi.org/pubs/uddi-v3.0.2-20041019.htm,2004.

представљање нефункционалних каракетристика (као што су цена, доступност, сигурност итд).

Међу истаживачима у области веб сервиса, познат је OWL-S[8]оквир (енгл. *OWL-Sframework*) који обезбеђује ограничен приступ за опис QoS вредности у облику парова (атрибут, вредност), па је, стога велика пажња поклоњена семантичком моделовању QoS карактеристика. Приступи као што су SWSO[9]и WSDL-S [175]су претежно фокусирани на функционалне аспекте и још увек не омогућавају експресиван начин описа сервиса.

У [176] је креирана OWL-QoS онтологија која је заснована на OWL-S оквиру са флексибилнијом спецификацијом QoS параметара веб сервиса. С друге стране, [177] представља онтолошки засноване трансформације различитих појмова *WS-Agreement*стандарда. Приступ који је делом заснован на истоветном семантичком моделовању QoS карактеристика и захтева над расположивим сервисима је дат у [178]. У овом приступу је креирана онтологија за представљање QoS вредности (представљена на Слици 18) која је погодна за каснији одабир и селекцију сервиса заснованих на корисничким захтевима и преференцама. Преференце корисника се дефинишу у облику њихових жељених вредности, а компонента за одабир сервиса је моделована у облику система за анализу упита чији су основни делови засновани на системима за филтерисање и процену QoS вредности, као и алгоритмима за рангирање сервиса. Рангирање сервиса је квалитативно, чиме дати приступ има недостатак у могућности примене за одабир комбинација сервиса који у највећој мери посматрани свеукупно задовољавају дефинисане захтеве.

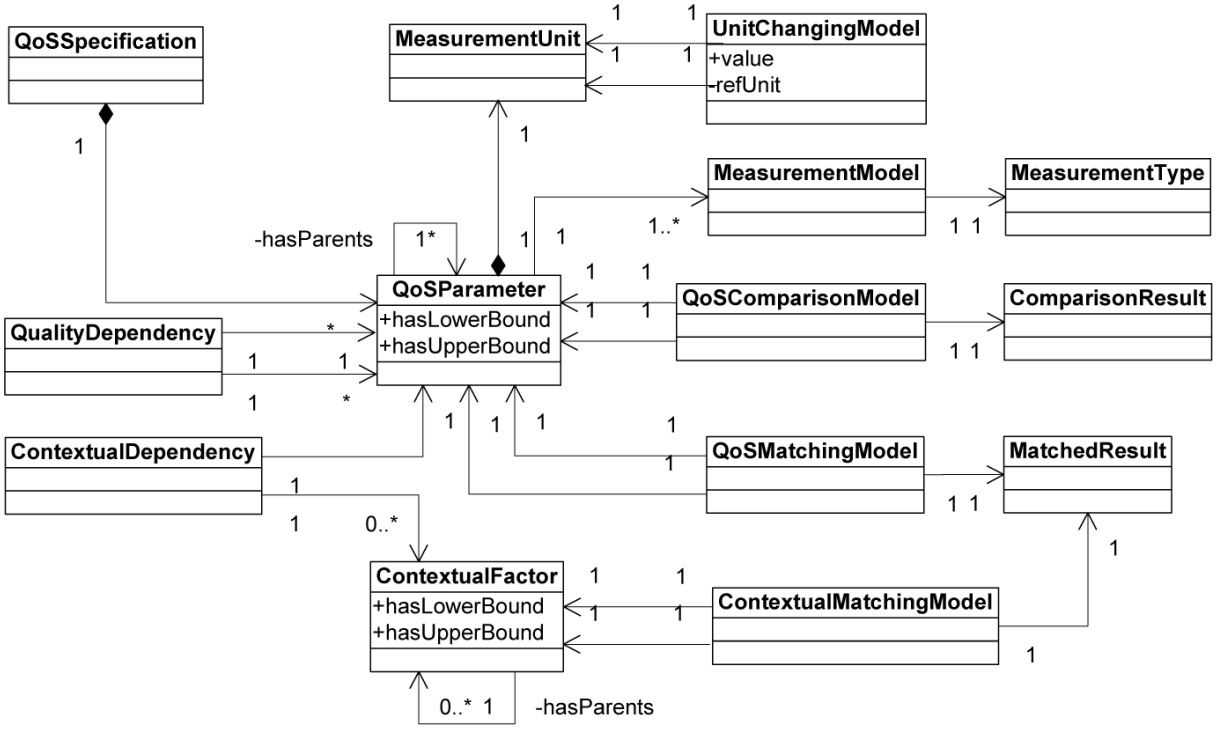
Квантификована приширења стандардног QoS модела су такође разматрана од стране многих истраживача, као што се наводи у наставку.

Liu et al. [179]представљају 'прошириви QoS модел' у којем су вредности атрибута за сваки од расположивих сервиса представљене у облику матрице. Над матрицом се врши неколико корака нормализације којима се на крају добијају вредности које представљају у којем степену је сваки од посматраних QoS атрибута присутан код датог сервиса. Предложени модел је проширив јер

[8]OWL-S: Web Ontology Language for Services. http://www.w3.org/Submission/2004/07/
[9]Semantic Web Services Ontology (SWSO), W3C Member Submission 9 September 2005. http://www.w3.org/Submission/SWSF-SWSO/

омогућава додавање доменских карактеристика и једноставне механизме за проширење конструисане матрице.

Wang et al. [179]имају сличан приступ који уводи у разматрање лингвистичку интерпретацију појединих опсега вредности атрибута, као што су 'ниска вредност', 'висока вредност', итд. Процедура нормализације која се примењује у овом присупу као резултат даје вредности које припадају фиксном интервалу [0, 1], што се није могло срести у разматрањима која су претходила.

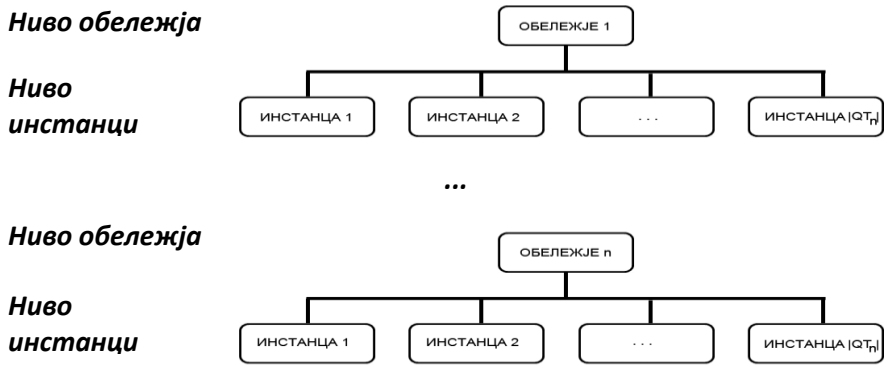


Слика 18. Онтологија за представљање QoS вредности [178]

С друге стране, приступ за представљање нефункционалних карактеристика који се користи у решењу које се предлаже овом дисертацијом је заснован на хијерархијској структури обележја и инстанци[105]која је перманентно развијена у области линија софтверских производа. Под скупом обележја $C = \{c_1,..., c_m\}$, се сматра билокоји скуп карактеристика који су од интереса за корисника, као на пример цена, расположивост, безбедност, време одзива. Под инстанцама (или могућим вредностима) се сматра скуп $QT_i = \{qt_1^i,...,qt_{|QT_i|}^i\}$ којег чине вредности сваког од обележја $c_i, 1 \le i \le m$ (нпр, инстанце обележја *цена* могу бити „ниска

цена“, „висока цена“, „прихватљива цена“). Обично се инстанцама додељују нека лексичка значења или ознаке;тако на пример за обележје *безбедност* се може навести да је високог, средњег или ниског нивоа, док се за обележје *цена* осим назива инстанце„ниска цена“, може навести и интервал [10, 100], представљену локалној валути, на који се значење ниске цене односи.

Структура обележја и инстанци има хијерархијски облик, с обзиром да се сва обележја сматрају једним нивоом, а инстанце као могуће вредности обележја, следећим нивоом. На тај начин је дефинисана двослојна структура представљена наСлици 19.



Слика 19. Хијерархијска структура обележја и инстанци

Дефинисана двослојна структура осим представљања вредности и/или могућих инстанци нефункционалних карактеристика омогућава и дефинисање захтева и ставова о вредностима датих обележја. На тај начин, уколико се расположиве опције означе у односу на развијену стуктуру, опција којој одговарају више префериране вредности обележја се може сматрати опцијом која у већој мери одговара дефинисаним преференцама. Како скуп инстанци у датој двослојној структури чине међусобно дисјунктни елементи, свака опција*о* се карактерише са највише једном инстанцом сваког од обележја, тј са уређеном *m*-торком $\left(qt_1^{i_1},..,qt_m^{i_m}\right)$, где је $i_j \in \left\{1,...,\left|QT_j\right|\right\}$. Уколико вредност неког обележја није позната или из неког разлога не може бити процењена, опција се не карактерише ни једном инстанцом датог обележја, и у том случају је $qt_j^{i_j} = \varnothing$.

Дата структураима могућност интеграције (на нивоу обележја) са онтологијом за представљање QoS карактеристика (које могу бити и доменски специфичне) и истовремено је аналогна хијерархијској структури успостављеној у области представљања и анализе захтева (заснованој на AHP алгоритму). Такође, ниво инстанци одговара приступу дефинисаном у [179] који обезбеђује већи степен персонализације (ка кориснику) самог QoS модела.

2.3.3.2.Оптимлана конфигурација SOA фамилије

У последњих неколико година, посебна пажња је посвећена проблему селекције сервиса у сервисно-оријентисаним архитектурама на основу QoS вредности[180]. Показано је да решења заснована на примени GA алгоритама обезбеђују довољно оптимално решење, тј решење које је довољно блиско оптималном.

Тако, Canfora et al. [137] предлажу приступ заснован на GA за селецију и композицију сервиса на основу QoS вредности. Предложени приступ користи једнодимензионе хромозоме за представљање комбинација сервиса, а функција циља садржи казнене факторе за оне геноме који не задовољавају дефинисане строге захтеве.

Такође, Gao et al. [181] су развили GA приступ заснован на кодирању помоћу структуре података стабла која се користи за представљање селекције и композиције сервиса. У раду[182]извршена је компаративна анализа приступа заснованих на примени GA и*Hill-Climbing* (HC) алгоритама. Њихови резултати показују да приступи засновани на GAимају боље перформансе од оних заснованих на HC у односу на целокупне QoS вредности које су досегнуте као и реализованом одстојању од оптималних решења.

С друге стране, Canfora et al.[137] представљају емпиријске податке за поређење метода заснованих на примени GA и ILP. Извршени експерименти показују да су GA алгоритми знатно скалабилнији, међутим спорији у конвергенцији у односу на ILP приступ. GA решења ипак могу да надмаше ILP у случајевима већег броја расположивих сервиса. Даље, аутори овог рада наводе да

многе хеуристичке технике за решавање NP сложених проблема (укључујући ILP) не могу бити директно примењене за оптималну селекцију одлика у SPL с обзиром да не могу подржати различита структурна и семантичка ограничења интегрисана у моделу одлика FM. Дати проблем постаје знатно сложенији с обзиром да дефинисани задатак конфигурације SOA фамилије садржи додатни задатак конфигурације одлика (тј инстанце фамилије) коју је потребно претходно селектовати како би се одредила оптимална финална конфигурација.

Као шо је већ представљено у 2.2.3.2,SAW метод се обично користи за представљање општих преференци о монотоним тенденцијама QoS вредности, па се у наведеним примерима примене техника за решавање задатака оптимизације, функција циља управо дефинише сходно SAW методу. Јасно је да се на тај начин не могу интегрисати стварне преференце корисника и да, стога, разијена решења нису персонализована ка корисницима и њиховим захтевима [180].

Примена MMKP приступа за решавање проблема оптималне селекције сервиса уз ограничења QoS вредности је анализирана у [163]. Као и у претходним разматрањима, функција корисности обухвата једноставно сумирање вредности посматраних QoS атрибута; док је акценат постављен на компаративну анализу алгоритама за решавањеоптималних задатака која је резултирала идентификацијом случајева када употреба одређене методе има боље перформансе. Такође, у литератури је препознато да се проблем композиције сервиса не може једноставно представити помоћу линеарне функције, шта више, линеаризација је могућа али захтева изузетно велики број помоћних бинарних променљивих, чиме се рапидно повећава величина проблема који се решава [164]. На тај начин је развијен хеуристички алгоритам *WFlow*[164] као и модел заснован на графовима *MCSP-K*[183], међутим и даље је присутан проблем повећања величине модела и последичног смањења перформанси.

3. Дефинисање проблема истраживања

Као што је представљено у претходном поглављу, ефикасан и флексибилан развој SOA заснованих софтверских решења треба да обезбеди испуњење скупа захтева који дефинишу функционалне карактеристике како појединих делова, тако и система у целини. Међутим, како је за сваку компоненту дате функционалности расположив скуп сервиса различитих нефункционалних карактеристика, као додатни захтев се намеће одабир сервиса који оптимално одговарају свеукупним очекивањима корисника о нефункционалним карактеристикама.

3.1.Идентификација проблема

Досадашња истраживања у овој области су била фокусирана искључиво на испуњење функционалних захтева (различитих приоритета) дефинисаних од стране једног или више корисника, док се проблем задовољења нефункционалних захтева посматрао само на појединачним инстанцама SOA архитектура. У овој дисертацији се иде корак даље и дефинише проблем конфигурације целе SOA фамилије у циљу истовременог испуњења дефинисаних функционалних захтева, и оптималне селекције сервиса с аспекта максимизације задовољења корисничких захтева над нефункционалним карактеристика система.

Захтеви над нефункционалним каратеристикама могу бити дефинисани на различите начине, квалитативно или квантитативно, а могу осликавати и међусобну зависност тј. условљеност међу карактеристикама. Ако се осим стандардних карактеристика веб-сервиса дозволи дефинисање доменски специфичних карактеристика, добија се нови проблем интерграције домена, представљања знања и захтева над њима. Као потенцијално решење се намеће додела онтолошког значења нефункционалним карактеристикама и примена онтолошки заснованих техника за предствљање захтева и даље резоновање над њима.

Цели приступ треба да води ка аутоматизацији процеса конфигурације, који је у теорији познат као NP-тежак задатак. Задатак постаје још комплекснији уколико се има у виду да осим великог броја сервиса који нуде исту функционалност и величине саме SOA архитектуре одређене бројем атомичних активности, SOA фамилија поседује и додатни степен варијабилности дефинисан опсегом архитектура које испуњавају тражене функционалне захтеве. Стога је неопходно развити интегрално решење које ће омогућити довољно ефикасно решење проблема. Искуства у истраживањима из сродних области су показала обећавајућим примену техника вештачке интелигенције.

Из наведеног, проблем који се намеће за предмет истраживања се може дефинисати на следећи начин: *примена семантичких техника за представљање разних врста захтева над нефункционалних карактеристикама SOA фамилија као и појединих њених делова, уз употребу метода вештачке интелигенције за решење проблема конструкције оптималне инстанце дате фамилије (тј. оптималне SOA архитектуре) и њене конфигурације, који свеукупно максимизују дефинисане функционалне и нефункционалне захтеве корисника.*

3.2.Анализа проблема

Као што се из саме дефниције проблема може уочити, јасно се намећу две целине које је могуће независно решавати, а само међусобно интегрисане, представљају решење дефинисаног проблема, а то су: *(i)* представљање нефункционалних карактеристика, разних врста захтева над њима и њихово рангирање, и *(ii)* развој метода за оптималну конфигурацију SOA фамилија у циљу максимизације испуњења дефинисаних захтева о нефункционалним карактеристикама.

Проблем представљања нефункционалних карактеристика је у литератури представљен од посебног значаја за омогућавање поређења функционално истоветних сервиса. Такође, развијене су разне семантичке структуре за њихово представљање, а приступ заснован на додели одређеног онтолошког значења и одговарајућих инстанци (као могућих вредности) нефункционалним

карактеристикама се већ показао добро применљивим у области линија софтверских производа (енгл. *software product lines*)[105] која је значајно повезана са SOA фамилијама [184]. Стога се каопрви циљ истраживања дефинише *развој семантичке структуре засноване на обележјима и инстанцама у циљу представљања нефункционалних карактеристикама целе SOA фамилије*.

С друге стране, као што је у претходном поглављу приказано, постојећи приступи за проблем конфигурисања сервиса не омогућавају дефинисање разних врста захтева над нефункционалним карактеристикама, већ су они усмерени ка максимизацији монотоно растућих карактеристика (као што су доступнос и, расположивост) уз истовремену минимизацију монотоно опадајућих карактеристика (као што су цена и време одзива). Међутим, у реалним сценаријима, корисници јако често морају да праве компромисе, па се сматра од изузетног значаја дозволити експлицитно дефинисање врста компромиса које су спремни прихватити (нпр, у случају изузетно кратког времена одзива, корисник може бити спреман платити и већи износ цене). У литератури, дефинисање оваквих врста захтева је омогућено условно дефинисаним захтевима, а за њихово представљање, *if-then*правила су добро успостављена у интелигенттим и експертним системима. Иако корисници могу бити спремни на компромисе, такође могу постојати и карактеристике које су од пресудног значаја и чије незадовољење води ка крајњем неприхватању производа (нпр. ниска цена је елиминаторни фактор у случају јако ограниченог буџета и сл). Овакви захтеви су у литератури познати као захтеви о лексикографском поретку, па је потребно омогућити и њихово дефинисање уколико се жели развити опште интегрално решење окренуто ка корисницима и њиховим захтевима. Упоредна анализа познатих метода и техника за анализу захтева (Табела 6) показује да у литератури није познато опште решење које омогућава ефикасну анализу свих врста захтева, већ се одабир технике врши у складу са доменом примене. Стога, следећи циљ у анализи се дефинише као *индентификација и формална спецификација могућих врста захтева над семантичком структуром за представљање нефункционалних карактеристика и анализа могућности адаптације постојећих алгоритама за њихово рангирање*.

Проблем оптималне конфигурације сервиса је такође познат у литератури и за његово решавање се традиционално користи метод целобројног линеарног програмирања [185]. Као што је наведено у претходној секцији, скорија истраживања идентификују скалабилност као један од лимитирајућих фактора за примену метода линеарног програмирања у области селеције сервиса [][][]. Такође, интеракција међу сложеним нефункционалним захтевима која одговара примерима из праксе није линеарне природе, тако да методе линеарног програмирања нису адекватне за примену у решавању дефинисаног задатка конфигурације SOA фамилија. *За решење се морају анализирати неке од осталих техника вештачке интелигенције заснованих на методама за решавање оптималних задатака*, што представља трећи задатак за истраживање.

На крају, целокупно решење се добија интегрисањем метода и техника развијених посебно за сваки од дефинисаних задатака. Развијене методе и технике треба да се односе на претходно развијену семантичку структуру за представљање нефункционалних карактеристика, а ефикасност, ефективност и скалабилност целог приступа је неопходно формално показати и/или симулационо тестирати.

4. Предлог решења

У овој секцији се представља решење проблема аутоматизације процеса оптималне конфигурације SOA фамилије, које обухвата: *(i)* конструкцију двослојне семантичке структуре за представљање нефункционалних карактеристика и разних врста захтева над њима, *(ii)* адаптацију AHP-алгоритма за анализу дефинисаних захтева у двослојној структури, и *(iii)* развој интегралног решења које омогућава оптимално конфигурисање SOA фамилије на основу дефинисаних нефункционалних захтева.

4.1.Представљање нефункционалних карактеристика

У овој секцији се представља сематичка структура која је развијена у облику двослојне хијерархијске структуре и омогућава представљање нефункционалних карактеристика и захтева над њима. Хијерархијска структура је прво развијена независно од домена примене [105] и доказана је њена експресивност за представљање захтева [95] [112], након чега је адаптирана за представљање нефункционалних карактеристика SOA фамилије.

4.1.1. Двослојна структура за представљање нефункционалних каракетристика

Приступ за представљање нефункционалних карактеристика који ће се користити у решењу које се предлаже је заснован на хијерархијској структури обележја и инстанци[105]представљеној у Секцији 2.3.3.1. У циљу илустрације, уведен је следећи једноставан пример.

Пример 1.

Нека су за електронско плаћање у он-лине продавници расположиви следећи сервиси:*VeriSign, PayPal*и*CyberSource*.Одабир сервиса који ће се користити зависи од вредности следећих нефункционалних карактеристика (које у

двослојној структури представљају *обележја*): *безбедност (Sec), једноставност коришћења (CustEase), могућност међународног плаћања (ISale)* и *цена (Cost)*. За сваку од каракетристика се разликују *ниска*, *средња* и *висока* вредност (које у двослојној структури представљају *инстанце*). Сервис *PayPal* је означен инстанцама„C“, „D“, „G“што значи да је окарактерисан са ниским нивоом сигурности (C), високом једноставношћу употребе (D) и високом могућношћу за међународно плаћање (G). Легенда ознака и примери сервиса су представљени на Слици 21.

Concern	High	Medium	Low
Security	**A**	**B**	**C**
Customer ease	**D**	**E**	**F**
International Sale	**G**	**H**	**I**
Cost	**J**	**K**	**L**

Tag legend

Слика 20. Сервиси за електорнско плаћање окаракетрисани инстанцама одговарајућих обележја [112]

Одабир сервиса за плаћање зависи од личних захтева и преференци корисника. Тако, за корисника за којег је висок ниво сигурности најважнији критеријум за одабир, сервис који испуњава његове захтеве је *VeriSign* (јер је означен инстанцом „A“ која представља висок ниво сигурности). Међутим, за другог корисника за којег је је висок ниво једноставности употребе најважнији захтев, сервис *PayPal*одговара дефинисаним захтевима јер је означен са инстанцом „D“, док за остале сервисе нису познате вредности обележја *ISale*.Захтеви корисника често нису овако једноставно дефинисани, већ укључују разне врсте компромиса које је корисник спреман прихватити, с обзиром да је захтеве у општем случају тешко испунити. На пример, корисник може дефинисати, да је спреман платити средњу висину цене иако сигурност није на највишем нивоу, и у том случају сервис *CyberSource*задовољава његове захтеве (јер је означен са инстанцама „B“ и „K“).■

Дати пример осликава начин на који корисник може дефинисати захтеве уколико се двослојна структура обележја и инстанци користи за представљање нефункционалних карактеристика. У поглављу 4.2. се формално дефинишу врсте

захтева над датом структуром као и алгоритам који омогућава квантитативно рангирање расположивих опција у односу на дефинисане захтеве.

4.1.2. Семантичко описивање нефунцкионалних карактеристика SOA фамилија

У овој секцији ће се представити и формално дефинисати начин употребе двослојне структуре обележја и инстанци у области SOA фамилија. Нека је за сваку атомичну активност у фамилији дат коначан скуп сервиса различитих нефункционалних карактеристика који је имплементирају. На основу нефукционалних карактеристика сваког од сервиса, могуће је проценити интервале очекиваних вредности сваке од нефункционалних каракетристика за целу фамилију, као и поједине делове фамилије (сходно [12]).

Расположиве информације о нефункционалним карактеристикама сервиса и интервали могућих вредности на нивоу целе SOA фамилије се користи за креирање двослојне структуре обележја и инстанци у следећа два корака:

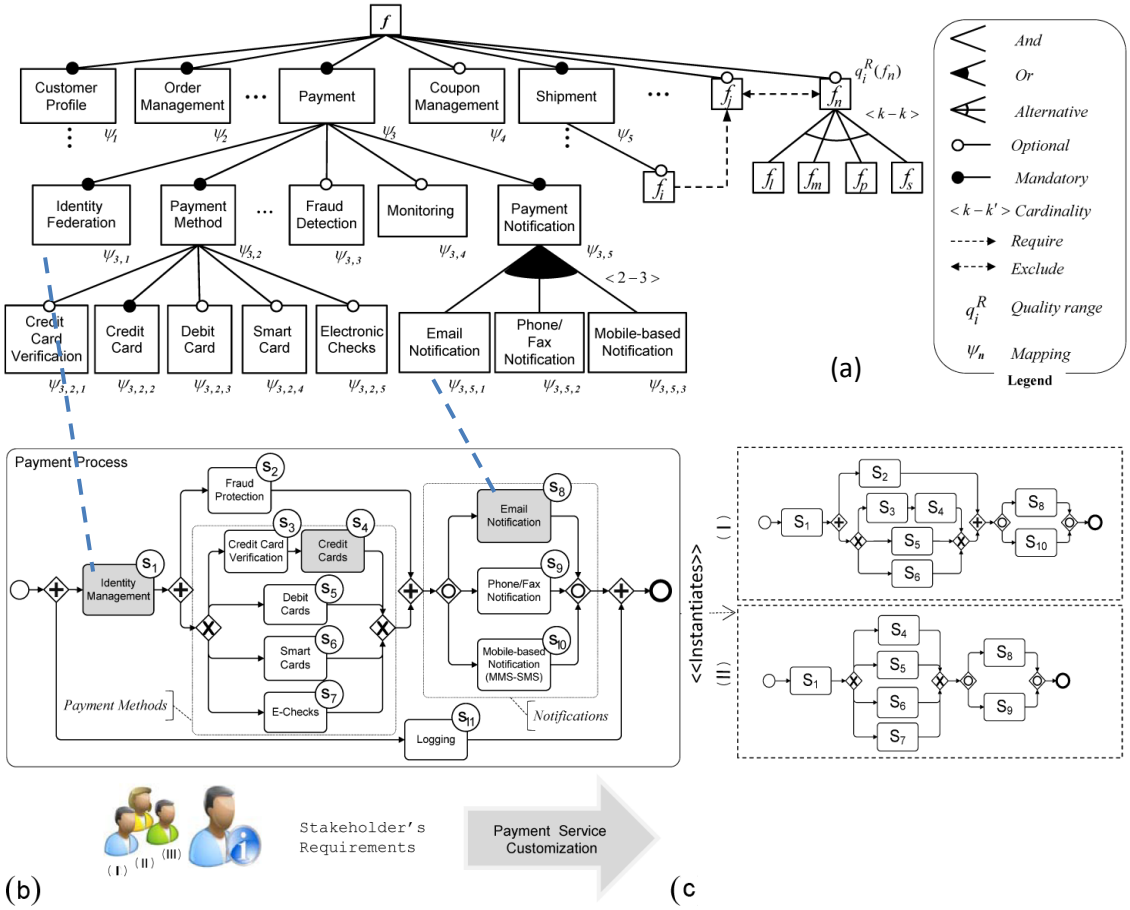
Корак 1 (Креирање нивоа обележја). Скупом обележја се сматра скуп свих нефункционалних карактеристика сервиса који су од интереса кориснику за одабир конфигурације SOA фамилије.

Корак 2 (Креирање нивоа инстанци). С обзиром да сервиси карактеришу само атомичне активности у фамилији, агрегирањем нефункционалних карактеристика сервиса и пропагацијом агрегираних вредности, у складу са бијективним мапирањем модела карактеристика и модела пословног процеса [12], добијају се интервали могућих вредности нефункционалних карактеристика целе фамилије. Добијени интервали $\left(Q_{agg}^{LB}, Q_{agg}^{UB}\right) = \left(\left\langle Q_1^{LB},..,Q_k^{LB}\right\rangle, \left\langle Q_1^{UB},..,Q_k^{UB}\right\rangle\right) \in R^k \times R^k$ представљају границе вредности сваког од обележја које је могуће достићи одговарајућом комбинацијом сервиса. Јасно је да се на овакав начин не добија процена вредности које се достижу комбинацијама сервиса, већ се само добија процена интервала којима те вредности припадају. На основу добијених интервала, корисник је у могућности да дефинише своје захтеве и ставове, поделом интервала на дисјунктне подинтервале који га покривају. На тај начин, уколико се

сваком подинтервалу додели одговарајуће лексичко значење, дефинишу се инстанце обележја.

У циљу илустрације објашњења процеса креирања структуре обележја и инстанци, посматрајмо следећи пример.

Пример 2. Посматрајмо део студије сучаја фамилије е-портала за трговину, као што је нпр *eBay*. На Слици 22bje представљена декомпозиција процеса за е-плаћање који је део велике фамилије заједничког оквира за е-плаћање предвиђено на е-тржишту. Ради једноставности, детаљи су изостављени, а приказан је поглед на процес плаћања од интереса за предмет истраживања.



Слика 21. а)Модел-карактеристика е-продавнице;б) Део одговарајућег модела фамилије пословних процеса која представља реализацију и композицију активности плаћања и доставе; с) Примери процеса е-плаћања који одговарају датом моделу пословних процеса

За различите инстанце производа из представљене фамилије е-портала за трговину, могу се посматрати различити начини е-плаћања, чиме се добија одређени број варијација у референтном процесу. Стога се одабир одговарајућег врши у складу са захтевима заинтересованих страна и пословних циљева. Неки сервиси за плаћање су неопходни и предуслов су процесу плаћања (нпр. кредитна картица је опште прихваћени начин е-плаћања), који би требало бути укључени у свим инстанцама из дате фамилије. С друге стране, неки функционални сервиси (нпр. *Smart Card e-Check*и*Debit Card*) или екстра-функционални сервиси (нпр. *Logging*и*Monitoring*) могу бити предстаљени као опционални чије укључење или искључење зависи директно од потреба заинтересованих страна (видети Слику 22c).

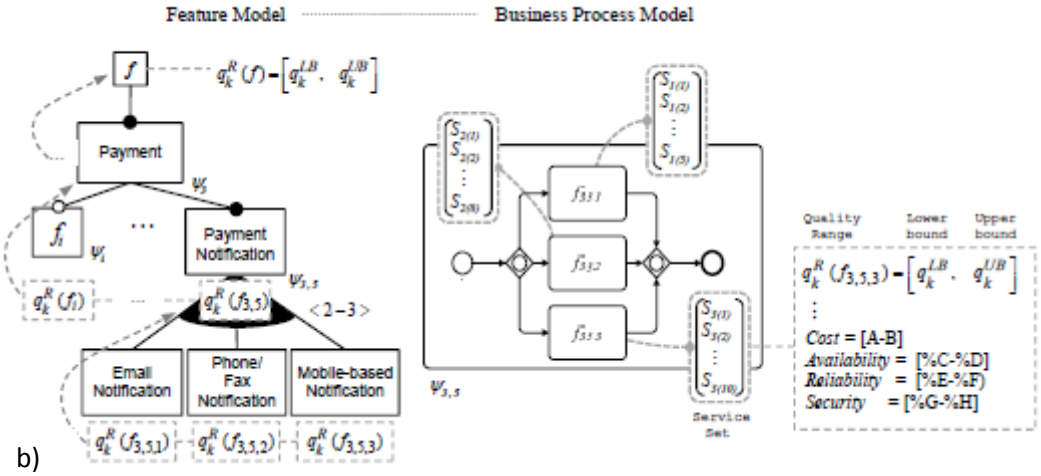
На пример, Корисник I може да захтева додатну функцију која обезбеђује висок ниво безбедности плаћања, укључујући посебну заштиту од превара, док ова функција није захтевана од стране Корисника II. Након испуњења функционалних захтева, нефункционалне карактеристике директно утичу на избор одговарајућих сервиса из скупа кандидата са различитим вредностима нефункционалних каракетристика (које се сматрају њиховим спецификацијама квалитета). Поред тога, функционални захтеви могу бити недефинисани за неке опционе функције и њихов избор / неизбор директно диктира оптималну комбинацију сервиса сходно нефункционалним карактеристикама и захтевима над њима.

На Слици 23**Ошибка! Источник ссылки не найден.**а је приказан скуп расположивих сервиса са вредностима нефункционалних карактеристика q_{pr}, ..., q_{tp}који имплементирају атомичне активности *EmailNotification, Phone/Fax*и*Notification*и*Mobile-based*и*Notification.* На основу вредности карактеристика сервиса, прво су процењени интервали вредности сваке од нефункционалних карактеристика за атомичне активности (Слика 23a), на основу којих су процењене вредности целе фамилије (као што је предсатвљено у Сексији 2.1.6.2.)(**Ошибка! Источник ссылки не найден.**Слика 23b). Сада корисник може да дефинише своје ставове и однос према процењеним вредностима. Нека је процењени интервал цене (карактеристика q_{pr}) за дати пример интервал [8, 48]. Корисник I може дефинисати да, сходно својој финансијској ситуацији, вредности

из интервала [8, 15) одговарају ниској цени, док вредности из интервала [15, 30) и[30, 48] одговарају редом средњим и високим вредностима. Такође, нека је укупно време одзива (карактеристика q_{tp}) процењено на интервал [20, 67].

	EmailNotification(S_1)	*Phone/FaxNotification*(S_2)	*Mobile-basedNotification*(S_3)
Range Values	$(Q_1^{LB}, Q_1^{UB}) = (\langle 8,\ldots,5\rangle, \langle 20,\ldots,31\rangle)$	$(Q_2^{LB}, Q_2^{UB}) = (\langle 17,\ldots,15\rangle, \langle 48,\ldots,67\rangle)$	$(Q_3^{LB}, Q_3^{UB}) = (\langle 9,\ldots,17\rangle, \langle 25,\ldots,45\rangle)$
Available Services	$\begin{array}{c} \\ S_{1(1)} \\ S_{1(2)} \\ S_{1(3)} \\ S_{1(4)} \end{array} \begin{array}{ccc} q_{pr} & q_i & q_{tp} \\ 8 & \ldots & 43 \\ 14 & \ldots & 31 \\ 20 & \ldots & 20 \\ 11 & \ldots & 5 \end{array}$	$\begin{array}{c} \\ S_{2(1)} \\ S_{2(2)} \\ S_{2(3)} \end{array} \begin{array}{ccc} q_{pr} & q_i & q_{tp} \\ 27 & \ldots & 32 \\ 48 & \ldots & 67 \\ 17 & \ldots & 15 \end{array}$	$\begin{array}{c} \\ S_{3(1)} \\ S_{3(2)} \\ S_{3(3)} \end{array} \begin{array}{ccc} q_{pr} & q_i & q_{tp} \\ 9 & \ldots & 25 \\ 11 & \ldots & 37 \\ 17 & \ldots & 45 \end{array}$

a)



b)

Слика 22. Дио SOA фамилије представљене помоћу модела карактеристика и одговарајућег пословног процеса; и а) нефункционалне карактеристике расположивих сервиса; и b) агрегација интервала нефункционалних карактеристика у складу са дефинисаном архитектуром

Поделом интервала могућих вредности нефункционалних карактеристика на дисјунктне подинтервале, креирана је структура обележја и инстанци, након чега корисник дефинише своје захтеве о префериранim карактеристикама за које је

задатак одредити конфигурацију која их у што већој мери задовољава. На пример, корисник I је спреман платити високу цену само у случају да је време одзива система мало; док у супротном жели платити само ниску цену. С друге стране, корисник II може дефинисати да је ниска цена најважнији критеријум за његов избор, с тим да при дефинисању може задржати исте интервале ниске цене као и претходни корисник, или их променити.

Нека је, сада, потребно урадити селекцију сервиса на основу дефинисаних захтева корисника. За корисника А најбоља конфигурација обухвата одабир сервиса $S_{1(1)}$и$S_{2(3)}$редом за интерфејсе *EmailNotification* и *Phone/FaxNotification*јер они дају укупну цену 25 и време одзива 58. С друге стране, оптимална конфигурација за корисника II подразумева одабир сервиса $S_{1(1)}$ који имплементира *EmailNotification* а најмање је цене (која износи 8) док се не врши одабир за опционе интерфејсе,*Phone/FaxNotification*и*Mobile-basedNotification*,јер ће укупна цена конфигурације бити повећана, а то је корисников најважнији критеријум за селецкију. ■

Формално, структура обележја и инстанци креирана за представљање нефункционалних карактеристика SOA фамилије се дефинише на следећи начин:

Дефиниција 5. (Двослојна структура нефункционалних караткеристика над SOA фамилијом). За дати скуп нефункционалних карактеристика $C = \{c_1,..., c_n\}$и дату SOA фамилију $G_{PM}^{FM} = (\nu, \varepsilon)$ у којој је за сваку атомичну активност позната горња и доња граница вредности сваке нефункционалне карактеристике, у ознаци $\left(Q_a^{LB}, Q_a^{UB}\right) = \left(\left\langle Q_1^{LB},..,Q_n^{LB}\right\rangle, \left\langle Q_1^{UB},..,Q_n^{UB}\right\rangle\right) \in R^n \times R^n$, ***двослојна структура нефункционалних карактеристика*** се дефинише као уређена четврока (C, $Q_{agg}^{LB}, Q_{agg}^{UB}$, *QT),* где $\left(Q_{agg}^{LB}, Q_{agg}^{UB}\right) = \left(\left\langle Q_1^{LB},..,Q_n^{LB}\right\rangle, \left\langle Q_1^{UB},..,Q_n^{UB}\right\rangle\right) \in R^n \times R^n$ представља агрегиране интервале за сваку од *n* нефункционалних карактеристика, а $QT = \left\langle\left\langle QT_1^1,...,QT_{k_1}^1\right\rangle,...,\left\langle QT_1^n,...,QT_{k_n}^n\right\rangle\right\rangle$је колекција подинтервала који покривају агрегиране интервале. Сваки од подинтервала QT_i^j је *отворени*, *полуотворени* или

затворени интервал $(a_i^j, b_i^j), (a_i^j, b_i^j], [a_i^j, b_i^j), [a_i^j, b_i^j]$ који задаовољава услове: $\bigcup_{i=1}^{k_j} QT_i^j = [Q_j^{LB}, Q_j^{UB}]$ и $QT_i^j \cap QT_l^j = \varnothing, 1 \le i < l \le k_j$, за свако фиксирано $j \in \{1,.., n\}$.

Стога, у дефинисаној структури су интенгрисане информације о нефункционалим карактеристикама целе фамилије, као и корисников став о њима, што се може сматрати једном врстом захтева које корисник дефинише у току самог креирања структуре. Креирана стуктура описује нефункционалне карактеристике фамилије у целости, или појединих њених делова тј потпроцеса, у случају да је потребно или корисник жели посебно дефинисати своје ставове о њима. Међутим, за дефинисани проблем конфигурације фамилије пословних процеса, намеће се додатна потреба за карактеризацијом нефункционалних карактеристка појединачне комбинације сервиса у циљу њиховог даљег поређења. На основу[185], свака комбинација сервиса $\langle s_1,\ldots,s_k \rangle$ се карактерише агрегираном вредношћу нефунционалних карактеристика $Q_{agg} = \langle q_1^{agg},..,q_n^{agg} \rangle \in R^n$, која припада тачно једној комбинацији подинтервала који покривају интервале могућих вредности нефункционалних карактеристика. Прецизније, у складу са терминологијом креиране двослојне струкутре, свакој комбинацији сервиса одговара тачно једна комбинација инстанци $QT_{i_1}^1 \times \ldots \times QT_{i_n}^n$.

Сада је могуће интуитиво представити и начине за поређење и селекцију комбинација сервиса на основу нефункционалних карактеристика и захтева корисника. Уколико корисник преферира комбинацију инстанци која одговара датој конфигурацији, може се сматрати да она задовољава дефинисани проблем; у супротном је неопходно пронаћи комбинацију сервиса која боље одговара преференцама корисника. Такође, уколико су идентификоване две различте комбинације сервиса које одговарају истој комбинацији инстанци, која је с друге стране прихватљива од стране корисика, потребно је омогућити начин за њихово поређење и одређивање више преферираних међу њима.

Наведени примери (представљени у општем домену као и у домену SOA фамилија) уводе проблем дефинисања захтева над двослојном структуром као и потребу за развојем алгоритама који ће омогућити квантитативно поређење

опција (у домену SOA фамилија, комбинација сервиса), сходно дефинисаним захтевима.

У наставку се прво наводи алгоритам за рангирање захтева над двослојном структуром обележја и инстанци, након чега се резултати алгоритма користе за развој мера квалитета појединих комбинација сервиса.

4.2.Рагирање безусловних захтева у двослојној структури (S-AHP алгоритам)

Као што је у претходним разматрањима наведено, корисници могу дефинисати разне врсте захтева, а најједноставнија форма обухвата безусловно дефинисане захтеве. У овој секцији се представља формализација безусловно дефинисаних захтева над двослојном структуром обележја и инстанци, као и алгоритам за њихово рангирање, који ће касније бити проширен у циљу рангирања осталих врста захтева које је могуће дефинисати над датом структуром.

За дефинисану структуру, сходно анализи алгоритама познатих у литератури (Секција 2.2), поређење парова на оба нивоа структуре се намеће као начин представљања и квантификације захтева. Стога, релативни приоритет међу обележјима и могућим инстанцама у двослојној структури се дефинише на следећи начин:

Дефиниција 6 (Релативни приоритет).

Релативни приоритет међу обележјима (или инстанцама) *a* и *b* има вредност α, у ознаци $a \overset{\alpha}{\succ} b$, аккообележје (или инстанца) *a* има већи приоритет у односу на обележје (или инстанцу) *b*, са коефицијентом α, α> 0. Основне карактеристике дефинисане релације су:

1. *рефлексивност*: $a \overset{1}{\succ} a$
2. *α^{-1}-симетричност* : $a \overset{\alpha}{\succ} b \Rightarrow b \overset{1/\alpha}{\succ} a$.

Својство *транзитивности* у општем случају не важи, што значи да се на основу релација $a \overset{\alpha}{\succ} b \wedge b \overset{\beta}{\succ} c$, не може извести закључак о односу међу обележјима (или инстанцама) *а*и*с.*

Традиционално, вредности 1, 3, 5, 7 и 9 представљају степене приоритета међу различитим опцијама у AHP методу [77]. Оне представљају редом, једнакост или индиферентност, слабу вредност, јаку вредност, врло јаку и екстремно јаку. Једнакост $a \overset{1}{\succ} b$ показује да опција (инстанца) *a* има исти релативни однос као и опција (инстанца) *b,* или да је корисник индиферентан око релативног односа међу опцијама (инстанцама) *а*и *b.*

На основу релације релативног приоритета на оба нивоа: нивоу обележја и нивоу инстанци, креирају се матрице за рачунање релативних приоритета, формално дефинисане на следећи начин:

Дефиниција 7(Матрица релативног приоритета међу обележјима).
Матрица релативног приоритета међу обележјима из скупа $C = \{c_1, ..., c_n\}$ у односу на релацију $\overset{\alpha}{\succ}$ је дефинисана са: $R^{\succ}_{n \times n}(C) = \{R^{\succ}[i, j] = \alpha \mid 1 \le i, j \le n, c_i \overset{\alpha}{\succ} c_j\}$. ■

Дефиниција 8(Матрица релативног приоритета за обележје *c*).
Матрица релативног приоритета за обележје скоје има скуп инстанци $QT = \{qt_1, ..., qt_{|QT|}\}$ у односу на релацију $\overset{\alpha}{\succ}$ је дефинисана са : $R^{\succ}_{|QT| \times |QT|}(c, QT) = \{R^{\succ}[i, j] = \alpha \mid 1 \le i, j \le |QT|, qt_i \overset{\alpha}{\succ} qt_j\}$. ■

Како је релација $\overset{\alpha}{\succ}$ рефлексивна и α^{-1}-симетрична, за матрице $R^{\succ}(C)$ и $R^{\succ}(c,QT)$ важи: $R^{\succ}(C)[i,i] = 1$, тј., $R^{\succ}(c, QT)[i,i] = 1$ и $R^{\succ}(C)[i, j] = \frac{1}{R^{\succ}(C)[j, i]}$, тј., $R^{\succ}(c,QT)[i, j] = \frac{1}{R^{\succ}(c,QT)[j, i]}$; па су матрице јединствено одређене скуповима релативних приоритета $\{R^{\succ}(C)[i, j], i < j\}$, тј., $\{R^{\succ}(c, QT)[i, j], i < j\}$.

Например, у случају три обележја $\{c_1, c_2, c_3\}$, њихова матрица релативних приоритета је јединствено одређена са $\{c_1 \overset{\alpha_{12}}{\succ} c_2, c_1 \overset{\alpha_{13}}{\succ} c_3, c_2 \overset{\alpha_{23}}{\succ} c_3\}$. Јединствена одређеност матрице једног обележја (на нивоу инстанци) се аналогно дефинише.

Вредности у матрицама дефинисаним Дефиницијама2-3 представљају релативне приоритете између сваког пара обележја и пара инстанци сваког обележја. Вредности су дефинисане безусловно, па се у наставку под *безусловно дефинисаним захтевом* сматра било који скуп релативних приоритета међу обележјима и/или њиховим инстанцама.

За анализу дефинисаних безусловних захтева над двослојном структуром обележја и интанци у литератури је познат S-AHP алгоритам [105] који представља адаптацију добро познатог AHP алгоритма[77]. Овим алгоритмом се, у следећа два корака једноставним рачунањима одређују вредности за рангирање расположивих опција (означених у односу на дату структуру).

Корак 1 (Рангирање обележја). У овом кораку се стандарди AHP алгоритам користи за поређење скупа обележја *C*. Значи, на основу релативних приоритета између сваког пара обележја, креира се матрица $R^{\succ}(C)$, а на основу стандардног AHP алгоритма се добија скуп вредности $\{r_1,..., r_n\}$, $0 \le r_i \le 1, \sum_{i=1}^{n} r_i = 1$ које представљају релативни ранг за сваки од датих обележја. Обележја са већим рангом се користе за касније рангирање доступних опција као и за филтрирање осталих обележја. Обележја мањег ранга се даље не разматрају, па, како би се задржало основно својство AHP метода [77] о јединичној суми вредности на сваком нивоу, користи се пропорционална прерасподела суме рангова филтрираних обележја, којом се истовремено чувају релативни односипретходно добијених рангова обележја.

Након редукције скупа обележја $C = \{c_1,..., c_n\}$ на $m \le n$ обележја, ново-добијени рангови $\{r'_{i_1}, \ldots, r'_{i_m}\}$ се рачунају по следећој формули:

$$r'_{i_k} = r_{i_k}\left(1 + \Delta r \Big/ \sum_{j=1}^{m} r_{i_j}\right), k \in \{1,\ldots,m\} \quad \textbf{(2)}$$

где је $\Delta r = \sum_{i \in \{1,\ldots,n\} \setminus \{i_1,\ldots,i_m\}} r_i$ сума рангова филтрираних обележја.

Корак 2 (Рангирање инстанци). У овом кораку се на основу стандрадног AHP алгоритма одређују локалне вредности за инстанце сваког обележја $c_{i_k}, k \in \{1,\ldots, m\}$, на основу матрица $R^{\succ}(c_{i_k}, QT_{i_k})$. На тај начин се добијају локални рангови $r_1^{i_k},\ldots,r_{|QT_{i_k}|}^{i_k}$, на основу којих се множењем са глобалним рангом одговарајућег обележја, добијају и глобалнирангови инстанци $r_{i_k}' \cdot r_1^{i_k},\ldots,r_{i_k}' \cdot r_{|QT_{i_k}|}^{i_k}, k \in \{1,\ldots,m\}$. На крају, ранг расположиве опције се рачуна на основу добијених рангова инстанце сваког од обележја које карактеришу дату опцију. У случају да вредност обележја није позната или да она не карактерише дату опцију, за одговарајући ранг се сматра нулта вредност (нула), у складу са главним циљем овог корака: одређивање и додела већег ранга опцији која више одговара обележјима од интереса за корисника који је дефинисао захтеве.

Ранг опције се добија применом унапред дефинисане функције над ранговима инстанци обележја који карактеришу опцију. Обично се узимају функције средња вредност, минимум или максимум, које се дефинишу при моделовању проблема у складу са доменом примене. Дефинисана функција омогућава једноставно рачунање приоритета опције које су означене са више различитих обележја. Формално, уколико са qt_j^i означимо *j*-ту инстанцу обележја c_i, крајњи ранг опције означене са скупом инстанци $\{qt_{j_1}^{i_1},\ldots,qt_{j_k}^{i_k}\}$ је одређен са

$$r(qt_{j_1}^{i_1},\ldots,qt_{j_k}^{i_k}) = f(r_{i_1}' \cdot r_{j_1}^{i_1},\ldots,r_{i_k}' \cdot r_{j_k}^{i_k}). \quad \textbf{(3)}$$

У циљу илустрације наведеног, проширимо претходно уведени Пример 1.

Пример 1 (Проширење 1). Претпоставимо да је потребно рангирати расположиве сервисе за електорнско плаћање на основу следећих захтева:

Neka је корисник на нивоу обележја дефинисано односе међу обележјима, формално записане на следећи начин:

$Sec \overset{3}{\succ} Cost$, $Sec \overset{5}{\succ} ISale$, $Sec \overset{5}{\succ} CustEase$, $Cost \overset{3}{\succ} ISale$, $Cost \overset{3}{\succ} CustEase$, $CustEase \overset{1/3}{\succ} ISale$.

На основу Satty-евог AHP алгоритма[100], рагови за обележја $Sec, Cost, ISale, CustEase$ су редом $0.54, 0.24, 0.14$, и 0.08. На основу добијених вредности се закључује да су обележја *ISale* и *CustEase* од мањег значаја за доношење одлуке о одабиру одговарајућег сервиса, што се у S-AHP алгоритму користи за уклањање из даљег процеса рангирања и прерасподелу њихових рангова на преостала обележја. Сума рангова ова два обележја је $0.14+0.08=0.22$, а након пропорционалане прерасподеле ове вредности на преостала обележја *Cost* и *Sec*, њихови нови рагови су $0.54+0.54/(0.54+0.24)*0.22=0.69$, тј. $0.24+0.24/(0.54+0.24)*0.22=0.31$.

У следећем кораку се врши рангирање опција на основу рангова инстанци којима су опције означене. Аналогно првом кораку, AHP алгоритам се користи за рачунање локалних рангова инстанци сваког обележја. Претпоставимо да су након рачунања, добијене следеће локалне вредности: за висок, средњи и ниски ниво обележја *Sec* редом 0.6, 0.3 и 0.1; а за високу, средњу и ниску вредност обележја *Cost* редом 0.2, 0.3, 0.5. Глобални рангови инстанци се добијају једноставним множењем локалних рангова и рангова одговарајућег обележја.
Како је сервис *VeriSign*означен инстанцама „A“, „I“, „L“ (Слика 2), које одговарају редом, високом нивоу обележја*Sec*, ниском нивоу обележја*ISale* и ниском нивоу обележја *Cost*, његов ранг се рачуна као просечна вредност глобалних рангова датих инстанци. Како је претходно обележје *ISale* проглашено од мањег значаја и више није разматрано, просечна вредност се рачуна као $(0.69*0.6+0.31*0.5)/2=0.285$. С друге стране, ни једна инстанца обележја *Cost* не означава сервис *PayPal* па се његов ранг рачуна са: $(0.69*0+0.31*0.1)/2=0.015$.■

4.3.Формална дефиниција захтева у двослојној структури

У овој секцији се идентификују разне врсте захтева релевантних за двослојну структуру обележја и инстанци. Прво, у секцији 4.3.1. даје се идентификација и формална спецификација условно и безусловно дефинисаних захтева, док се у 4.3.2. даје формализација захтева о доминантним релативним приоритетима, као специјалној врсти захтева.

4.3.1. Формална дефиниција условних и безусловних захтева

У овој секцији се помоћу елемената Пропозиционе Логике проширује релација релативног приоритета за представљање условних захтева.Пре навођења формалних концепата, наводи сејош једно проширење примера 1 које објашњава природу условних захтева о релативним приоритетима међу обележјима и њиховим инстанцама.

Пример 1 (Проширење 2). У претходно дефинисаном сценарију, претпоставимо да корисник дефинише захтеве о релативним приоритетима међу обележјима на следећи начин:

Безусловни релативни приоритет:$Cost \overset{3}{\succ} ISale$ **(4)**

Условни приоритет: у случају високе могућности интернационализације продаје(*ISale. High)* тада је$Sec \overset{5}{\succ} CustEase$, у супротном $Sec \overset{3}{\succ} CustEase$, $Cost \overset{3}{\succ} Sec$. **(5)**

На другом нивоу, тј нивоу инстанци, корисник може условно дефинисати приоритетемеђу инстанцама обележја *ISale*, на следећи начин:

У случају високог степена безбедности (обележје *Sec. High),* мале вредностиобележја *ISale* су знатно мање прихватљиве у односу на средње и високе: (тј., $ISale.High \overset{5}{\succ} ISale.Low$, $ISale.Medium \overset{5}{\succ} ISale.Low$); у супротном, мале вредности су изузетно мало прихватљиве у односу на средње и високе вредности (тј.$ISale.High \overset{1}{\succ} ISale.Medium$, $ISale.High \overset{3}{\succ} ISale.Low$, $ISale.Medium \overset{3}{\succ} ISale.Low$). **(6)**

Важно је истаћи да релативни приоритети међу инстанцама једног обележја могу бити дефинисани са више од једног захтева, тј кориснику је дозвољено да дефинише више условних/безусловних захтева о релативном односу међу инстанцама обележја *ISale,* као у следећем примеру.

У случају мале вредности цене *Cost,* тада је незнатно већа погодност средње вредности обележја *ISale* у односу на високу вредност, а средња вредност је знатно погоднија у односу на мале вредности (тј. $ISale.Medium \overset{3}{\succ} ISale.High$, $ISale.Medium \overset{5}{\succ} ISale.Low$); док у супротном, високе вредности обележја *ISale* имају слаби и јаки приоритет у односу на средње и мале вриједнсоти(тј. $ISale.High \overset{3}{\succ} ISale.Medium$, $ISale.High \overset{5}{\succ} ISale.Low$). **(7)** ■

Из примера се може закључити да релативни приоритетизмеђу два обележја може зависити од инстанци других обележја. Такође, релативни однос међу инстацама једног обележја може зависити само од инстанци неког (неких) других обележја. У наставку се сматра да било који други облик зависности у међусобним приоритетима међу обележјима и инстанцама није релевантан за дефинисану двослојну структуру обележја и инстанци.

У циљу формализације условних захтева, уводе се следеће дефиниције:

Дефиниција 9(Условни приоритет међу обележјима).

За дати скуп обележја $C = \{c_1,..., c_n\}$ и скуп инстанци $QT = \bigcup_{i=1}^{n} QT_i$, условни приоритетподскупа обележја $\{c_{j_1},.., c_{j_k}\}$, $j_l \in \{1,.., n\}, l \in \{1,..., k\}$ у односу на релацију $\overset{\alpha}{\succ}$ се дефинише као уређена петорка $(\Psi, C, QT, R^{\succ}(C)_1, R^{\succ}(C)_2)$, где је:

1. Ψ - пропозициона логичка формула над променљивима из скупа инстанци $\bigcup QT_j, QT_j \in QT / \bigcup_{l=1,k} QT_{j_l}$ и логичким операторима конјункције, дисјункције и негације, $\wedge$, $\vee$, и $\neg$
2. $QT_i \in QT$ је скуп инстанце обележја c_i;
3. $R^{\succ}(C)_1$ је матрица релативног приоритета међу обележјима која се примењује у случају да је услов дефинисан формулом Ψ задовољен;

4. $R^{\succ}(C)_2$ је матрица релативног приоритета међу обележјима која се примењује у случају да услов дефинисан формулом Ψ није задовољен.

Истинитоносна вредност лиогичке формуле Ψ одређује матрицу релативних приоритета која ће бити примењена, што се може представити помоћу следећег псеудо-логичког израза: $\Psi : R^{\succ}(C)_1 \mid R^{\succ}(C)_2 \blacksquare$

Неформално интерпретирано, матрица релативних приоритета $R^{\succ}(C)_1$ представља *then*-деоусловног захтева, а матрица $R^{\succ}(C)_2$ представљањен *else* –део.

На пример, условни захтев о приоритетима међу обележјима $\{c_1, c_2, c_3\}$, може бити дефинисана са: $\Psi : c_1 \overset{\alpha_{12}}{\succ} c_2, c_1 \overset{\alpha_{13}}{\succ} c_3, c_2 \overset{\alpha_{23}}{\succ} c_3 \mid c_1 \overset{\beta_{12}}{\succ} c_2, c_1 \overset{\beta_{13}}{\succ} c_3, c_2 \overset{\beta_{23}}{\succ} c_3$. Значи, уколико је услов Ψ задовољен, матрица релативних приоритета је дефинисана са $\{c_1 \overset{\alpha_{12}}{\succ} c_2, c_1 \overset{\alpha_{13}}{\succ} c_3, c_2 \overset{\alpha_{23}}{\succ} c_3\}$, у супротном са $\{c_1 \overset{\beta_{12}}{\succ} c_2, c_1 \overset{\beta_{13}}{\succ} c_3, c_2 \overset{\beta_{23}}{\succ} c_3\}$. Неопходно је нагласити да наведена формална дефиниција 4 садржи ограничење да логички израз Ψ који представља услов у условно дефинисаном захтеву о релативним приоритетима обележја $\{c_{j_1}, .., c_{j_k}\}$, не може садржати променљиву која се односи на неку од инстанци подскупа обележја $\{c_{j_1}, .., c_{j_k}\}$. На овај начин се спречава дефиниција захтева у којем однос међу обележјима зависи од њих самих.

На основу уведене дефиниције, претходно дефинисани захтев **(5)** се формално може представити у следећем облику: *ISale.High: Sec* $\overset{5}{\succ}$ *CustEase | Sec* $\overset{3}{\succ}$ *CustEase, Cost* $\overset{3}{\succ}$ *Sec.*

Дефиниција 10(Условни приоритети за обележје c_i).

За дати скуп обележја C и инстанци $QT = \bigcup_{i=1}^{n} QT_i$ условни приоритети обележја $c_i \in C$ у односу на релацију $\overset{\alpha}{\succ}$ се дефинишу као уређена петрока $(\Psi, c_i, QT_i, R^{\succ}(c_i, QT_i)_1, R^{\succ}(c_i, QT_i)_2)$ у којој је:

5. Ψ - пропозициона логичка формула над променљивима из скупа инстанци $\bigcup_{QT_j \in QT \setminus QT_i} QT_j$ и логичким операторима конјункције, дисјункције и негације, $\wedge$, $\vee$, и $\neg$

1. c_i је обележје за које се захтев дефинише;
2. $QT_i \in QT$ је скуп инстанци обележја c_i;
3. $R^{\succ}(c_i, QT_i)_1$ је матрица релативних приоритета обележја c_i у случају да је услов дефинисан формулом Ψ задовољен;
4. $R^{\succ}(c_i, QT_i)_2$ је матрица релативних приоритета обележја c_i у случају да услов дефинисан формулом Ψ није задовољен.

Истинитоносна вредност логичког израза Ψ одређује која матрица релативних приоритета ће бити примењена, што се може представити у облику псеудо-логичког исказа у следећем облику: $\Psi : R^{\succ}(c_i, QT_i)_1 \mid R^{\succ}(c_i, QT_i)_2$. ■

На пример, условни захтев за обележје које има три инстанце $\{qt_1, qt_2, qt_3\}$, се може дефинисати са $\Psi : qt_1 \overset{\alpha_{12}}{\succ} qt_2, qt_1 \overset{\alpha_{13}}{\succ} qt_3, qt_2 \overset{\alpha_{23}}{\succ} qt_3 \mid qt_1 \overset{\beta_{12}}{\succ} qt_2, qt_1 \overset{\beta_{13}}{\succ} qt_3, qt_2 \overset{\beta_{23}}{\succ} qt_3$; што значи да, у случају задовољена услова дефинисаног са Ψ, матрица релативних приоритета је одређена са $\{qt_1 \overset{\alpha_{12}}{\succ} qt_2, qt_1 \overset{\alpha_{13}}{\succ} qt_3, qt_2 \overset{\alpha_{23}}{\succ} qt_3\}$, у супротном са $\{qt_1 \overset{\beta_{12}}{\succ} qt_2, qt_1 \overset{\beta_{13}}{\succ} qt_3, qt_2 \overset{\beta_{23}}{\succ} qt_3\}$.

Аналогно претходној дефиницији, и дефиниција 5 садржи ограничење да, у случају дефинисања захтева за обележје c_i, скуп променљивих у логичкој формули Ψ не може садржати инстанце обележја c_i. На тај начин није дозвољено дефинисање захтева о односима међу инстанцама које зависе од самих себе.

Како су наведене дефиниције уведене у циљу формализације захтева за поређење опција, случајеви када опције нису означене са инстанцама које су садржане у логичким изразима су посебно анализиране касније у секцији 4.3.3.

На основу уведене дефиниције, претходно дефинисани захтев **(6)** се формално може представити у следећем облику:

Sec.High: ISale.High $\overset{5}{\succ}$ *ISale.Low, ISale.Medium* $\overset{5}{\succ}$ *ISale.Low | ISale.High* $\overset{1}{\succ}$ *ISale.Medium, ISale.High* $\overset{3}{\succ}$ *ISale.Low, ISale.Medium* $\overset{3}{\succ}$ *ISale.Low*.

4.3.2. Формална спецификација захтева о доминантним приоритетима

У овом поглављу се уводи дефиниција доминантног релативног приоритета која одговара дефиницији добро познатог лексикографског поретка[187]. Такође, након увођена дефиниције, анализира се могућност комбиновања тако дефинисаног захтева са претходно уведеним условним и безусловним захтевима. Лексикографски поредак је специјалан врста сортирања, која се често среће у литератури у различитим контекстима и значењима [113]. Она представља генерализацију алфабетског поретка међу речима над датим алфабетом, а може се интерпретирати као сортирање на основу односа вредности променљивих истог нивоа, без обзира на вредности променљивих нижег нивоа [113]. Оваква врста захтева има широку могућност примене у реалним проблемима и свакодневно се среће у животу.

У циљу илустрације наводи се пример корисника у сценарију из Примера 1 који има ограничен буџет и дефинише своје захтеве за одабир сервиса за електронско плаћање.

Пример 1 (Проширење 3).Ограниченост буџета је елиминаторни критеријум за корисника, али и поред тога он жели висок или средњи ниво безбедности одабраног сервиса. То значи да расположиве сревисетреба рангиратипрво на основу најмање ценекао приоритетног критеријума, без обзира на остале карактеристике,**(8)**; док се разлике међу сервисима са истом ценом одређују на основу средњег или високог нивоа безбедности **(9).** Даље, природносе намеће да се кориснику дозволи да дефинише условљеностиу захтевима о релативним односима међу осталим обележјима,*ISale* и *CustEase*. При дефинисању оваквих врста захтева, сматра се да, уколико је већ дефинисан доминантни приоритет обележја *Cost*, као и значај обележја *Sec*, над њима касније није могуће

дефинисати било какве условне захтеве. Овакво ограничење је дефинисано у складу са интерпретацијом доминантности приоритета обележја[112].■

Формално, доминантни релативни приоритет у двослојној структури обележја и инастанци се дефинише са:

Дефиниција 11(Доминантни релативни приоритет обележја c_i).

За дати скуп обележја $C = \{c_1,..., c_n\}$ и скуп инстанци $QT = \bigcup_{i=1}^{n} QT_i$, обележјеc_iима ***доминантни релативни приоритет*** у односу на скуп обележја $C/\{c_i\}$(у ознаци $\overset{D}{\succ}(c_i)$) уколико је индуковано рангирање над скупом опција дефинисано на следећи начин:

Опција o_1, која је означена са скупом инстанци $\{qt_{j_1}^{i_1},...,qt_{j_k}^{i_1}\}, k \le m$, има већи приоритет у поређењу са опцијом o_2 која је означена са скупом инстанци $\{qt_{j_1}^{i_1^2},...,qt_{j_l}^{i_l^2}\}, l \le m$, akko је један од следећих услова испуњен:

ц1) Инстанца из скупаQT_iозначава само опцијуo_1;

ц2) Ни једна инстанца из скупаQT_i не означава ни једну опцију (o_1илиo_2) а опцијаo_1има већи ранг на основу вредности инстанци осталих обележја којима је означена;

*ц3)*Ако су обе опције означене неком од инстанце из скупа QT_i, разликују се следећи случајеви:

ц3.a) обе опције су означене истом инстанцом из скупа QT_iи опција o_1 има већи ранг на основу вредности инстанци осталих обележја којима је означена;

*ц3.b)*опцијаo_1 је означена инстанцом из скупа QT_i која је већег ранга у односу на инстанце из истог скупа које означавају опцију o_2.

У свим осталим случајевима, опција o_2 има већи или једнак ранг као и опција o_1. ■

Може се уочити да интуитиван опис захтева **(8)** из уводног примера може бити формално представљен у облику *(ц1)* а опис **(9)** са *(ц3),* док се *(ц2)* односи на поређење опција које нису означене обележјима на које се односи дефинисани доминантни приоритет (тј поређење без доминантног приоритета).

У циљу илустрације, уведи се следећи пример.

Пример 3. Посматрајмо следећих пет опција:o_1*={Cost.Low, CustEase.Low, ISale.High},* o_2*={Cost.High, CustEase.Low, ISale.High},* o_3*={Cost.High, CustEase.High, ISale.High},* o_4*={CustEase.Low, ISale.High},* o_5*={CustEase.High, ISale.High},* **(10)**; и захтев о доминантном приоритету обележја *Cost* са највећим приоритетом малих вредности и мањих приоритета за средње и високе вредности, респективно. Такође, претпоставимо да комбинација инстанци *{CustEase.High, ISale.High}* има већи приоритет у односу на *{CustEase.Low, ISale.High}*.**(11)**

Сада, применом Дефиниције 7 и дефинисаних захтева, рангирајмо опције дефинисане са **(10)**. На основу *(ц1)*, прве три опције o_1, o_2, o_3 имају већи ранг у односу на опције o_4и o_5. Такође, на основу *(ц2)* и **(11),** опција o_5 је већег ранга него o_4. Услов *(ц3)* даје рангирање међу опцијама o_1, o_2, o_3 на следећи начин: *(ц3.а)* и **(11)** дају већи ранг за o_3 у односу на o_2; док *(ц3.б)* даје већи ранг опцији o_1 у односу на опције o_2, o_3. На крају, опције рангиране у опадајућем поретку по приоритетима су: o_1, o_3, o_2, o_5, o_4. ■

На основу дефиниције доминантног релативног приоритета једног обележја, може се дефинисати доминантни релативни приоритет уређене *к*-торке обележја, на следећи начин:

Дефиниција 12 (Доминантни релативни приоритет уређене *к*-торке обележја $(c_{i_1},..., c_{i_k})$).

За дати скуп обележја $C = \{c_1,..., c_n\}$ и скуп инстанци $QT = \bigcup_{i=1}^{n} QT_i$, уређена*k*-торка $(c_{i_1},..., c_{i_k})$ има ***доминантни релативни приоритет*** (у ознаци $\overset{D}{\succ}(c_{i_1},..., c_{i_k})$) акко обележје c_{i_1} има доминантни релативни приоритет у односу на облежја из скупа $C/\{c_{i_1}\}$, а за свако $j \in \{2,...,k\}$ обележје c_{i_j} има доминантни релативни приоритет у односу на обележја из скупа $C \setminus \{c_{i_1},...,c_{i_{j-1}}\}$. ■

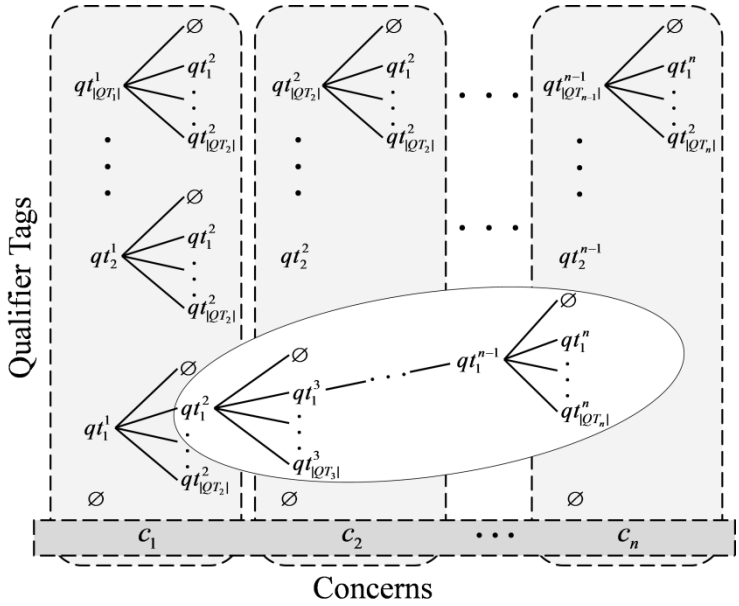
Поновним разматрањемПримера 1 (Проширење 3), може се уочити да дефинисани захтеви представљају интерпретацију доминантног релативног приоритета пара (*Cost, Sec*). Додатно, ниска вредност обележја *Cost* је значајнија у односу на средње и високе вредности, а висок ниво обележја *Sec* има већи приоритет у односу на ниже вредности. Важно је уочити да, дефинисањем доминантног релативног приоритета за обележје *Cost* са приоритетним ниским вредностима, престаје потреба и значај дефинисања квантитативних вредости приоритета међу инстанцама које представљају ниске, средње и високе вредности тог обележја. У ствари, информација да ли је ниска вредност обележја *Cost* знатно или изузетно много већег приоритета не представља податак од значаја. Она неће утицати на крајње рангирање расположивих опција, јер захтев о доминантном релативном значају обележја са ниском приоритетном вредношћу, као резултат даје боље рангирану било коју опцију окарактерисану инстанцом која представља ниску вредност обележја *Cost* од оних окарактерисанихинстанцама које представљају средње или високе вредности.

Стога, од корисника се очекује да у случају дефинисања обележја са доминантним приоритетом, квалитативно дефинише редоследњегових инстанци. У наставку се подразумева да су инстанце обележја са доминантним релативним приоритетом распоређене у растућем редоследу по приоритетима. Са $r_D(qt_j^i)$, означимо позицију *j*-те инстанце обележја c_i у растуће уређеном низу његових инстанци.

Дакле, за пример из претходног параграфа важи:
$r_D(Cost.High) = 1, r_D(Cost.Medium) = 2, r_D(Cost.Low) = 3.$ **(12)**

Дефинисањем доминантних релативних приоритета над целим скупом обележја *C*, дефинише се лексикографски поредак над двослојном структуром обележја и њихових инстанци. Без губитка општости, претпоставимо да уређени скуп обележја $\{c_1,...,c_n\}$ има доминантни релативни приоритет и да су скупови инстанци сваког обележја уређени у растућем поретку, тј qt_1^i је инстанца обележјаc_i са најмањим приоритетом, и тако редом. Резултат крајњег рангирања скупа свих могућих опција за дати скуп обележја (concerns) *C* и инстанци

(qualifier tags) $QT = \bigcup_{i=1}^{n} QT_i$, под дефинисаним захтевима доминантног релативног приоритета целог скупа обележја *C* је илустрован на Слици 23**Ошибка! Источник ссылки не найден.**. Опција која је најмањег ранга је означена само са инстанцом најмањег приоритета од обележја које такође има најмањи доминантни приоритет. С друге стране, опција највећег ранга је она која је означена са инстанцама највећих приоритета свих обележја.



Слика 23. Шематски приказ лексикографског поретка над двослојном структуром обележја и инстанци [112]

4.4.Интегрално решење за оптималну конфигурацију SOA фамилије

У овој секцији сена основу развијене двослојне структуре за представљање нефункционалних карактеристика SOA фамилије и идентификованих врста захтева, развија CS-AHP алгоритам за њихову анализу и рангирање. Развијени алгоритам се даље користи за увођење мера квалитета појединих комбинација сервиса и формално дефинише крајњи проблем оптималне конфигурације. Такође, за решење дефинисаног проблема се предлаже употреба хеуристичког

приступа чије се карактеристике ефикасностии ефективности посебно анализирају. На тај начин се добија *OptConfSOAF*приступ који омогућава аутоматизацију процеса конфигурације SOA фамилије на основу дефинисаних захтева о нефункционалним карактеристикама. Као што је наведено у поглављу 2.2.3.2. , на основу приступа представљеног у [189], обезбеђено је да крајња конфигурација максимално испуњава дефинисане функционалне захтеве над моделом карактеристика. Тиме решење *OptConfSOAF*омогућује конструкцију оптималне инстанце SOA фамилије уз истовремену максимизацију испуњења дефинисаних функционалних и нефунционалних захтева.

Остатак овог поглавља је организован на следећи начин. Развој алгоритма за рангирање корисничких захтева је дат у 4.4.1.3. (рангирање условно и безусловно дефинисаних захтева у 4.4.1.1, док је рагурање захтева о доминантним релативним приоритетима дато у 4.4.1.2.). Теоретска анализа предложеног алгоритма је дата у 4.4.1.4. Даље, развој мера квалитета комбинација сервиса у складу са варијабилностима присутним у моделу је представљен у секцији 4.4.2.1., а њихова анализа је дата у секцији 4.4.2.2. Сходно дефинисаним мерама и њиховим карактеристикама, задатак оптимизације је дефинисан у секцији 4.4.3., док је његово решење дато у секцији 4.4.4. Као што је представљено у уводним поглављима, реч је о NP-сложеном проблему, па се његовом решавању приступило секвенцијално: у 4.4.4.1. је представљен хеуристички приступ који у унапред дефинисаном броју корака даје довољно ефикасно решење, а додатна оптимизација целог приступа је дата у 4.4.4.2.

4.4.1. Метод за рангирање захтева у двослојној структури (CS-AHP алгоритам)

У овој секцији је дата анализа утицаја разних врста захтева на сами процес рангирања као и идентификација конзистентних захтева; након тога је представљен CS-AHP алгоритам, конструисан као проширење претходно представљеног S-AHP алгоритма, за рангирање безусловних и условних захтева, и захтева о доминантним приоритетима над двослојном структуром обележја и инстанци.

4.4.1.1.Адресирање условних захтева

У проширењу 2 Примера 1 су наведени примери захтева који не дефинишу у потпуности релативне приоритете између свака два пара обележја, тј сваки дефинисани захтев само осликава релативне приоритете међу обележјима која зависе од дефинисаног услова, остављајући остале приоритете недефинисаним. Тачније, захтев **(4)** дефинише релативни приоритет само између обележја *Cost*и*ISale*, док, ако се посматра заједно са захтевом **(5),** релативни приоритети између обележја *ISale*и*Sec* и даље остају недефинисани. С друге стране, релативни односи за обележје*ISale*је дефинисан са два условна захтева **(6)** и **(7).** За сва остала обележја, захтеви су дефинисани условно или безусловно, и само њиховим спајањем се могу добити све информације које се односе на ниво обележја.

Формално, захтеви корисника су представљени у облику делова матрица дефинисаних са Дефиницијама 2-5, $\Re = \left\{ \begin{array}{l} \left\{R^{\succ}(\overline{C})\right\}_{\overline{C}\subset \mathrm{P}_C}, \left\{R^{\succ}(c_i, QT_i)\right\}_{i\in\{1,\ldots,n\}}, \left\{\left(\Psi, \overline{\overline{C}}, QT, R^{\succ}(\overline{\overline{C}})_1, R^{\succ}(\overline{\overline{C}})_2\right)\right\}_{\overline{\overline{C}}\subset \mathrm{P}_C} \\ , \left\{\left(\Psi, c_j, QT_j, R^{\succ}(c_j, QT_j)_1, R^{\succ}(c_j, QT_j)_2\right)\right\}_{j\in\{1,\ldots,n\}} \end{array} \right\}$ где је P_C скуп свих подскупова скупа обележја C. Основни циљ процеса обраде дефинисаних захтева се заснива на попуњавању целокупне матрице дефинисане на нивоу обележја (Дефиниција 2) и n матрица на нивоу инстанци (Дефиниција 3: n матрица димензија $|QT_i|\times|QT_i|$ гдје је $|QT_i|$ број инстанци j-тог обележја). Рачунање локалних и глобалних рангова је могуће само уколико су матрице на оба нивоа комплетно попуњене.

У наставку се даје даља разрада Проширења 2 Примера 1 и приказује попуњавање матрица на основу дефинисаних захтева.

Пример 1 (Проширење 2- наставак).У датом примеру се посматрају четири обележја која је потребно рангирати. Као што је већ наведено, захтев **(4)** безусловно дефинише релативни приоритет између обележја *Cost*и*ISale*; док захтев**(5)**у свом *then*-делу дефинише релативне приоритете између *Sec*и*CustEase*, а

у *else*-делу приоритете између два пара обележја: *Sec*и*CustEase*, и*Cost*и*Sec*. Сада се могу закључити две ствари: прво, не може се донети одлука који део (*then*- или *else*-) да се примени при условно дефинисаним захтевима; друго, који год део да се примени, матрица на нивоу обележја неће бити комплетно попуњена (шест елемената у матрици треба да буде попуњено, а у примеру су дефинисане само две или три, зависно од дела захтева који се примењује).

Одлука о делу захтева који се примењује не може бити донесена без конкретне опције за коју се анализирају захтеви. Посматрајмо две опције расположиве у моделу, сервисе *VeriSign*и*CyberSource* (Слика 2). За обе опције локални и глобални рангови се посебно рачунају, на основу чега се касније доноси одлука о опцији која више одговара дефинисаним захтевима.

Посматрајмо прву опцију. Ниска вредност обележја *ISale* чини да *else* део захтева **(5)** буде задовољен. Спајањем захтева **(5)** са беусловно дефинисаним захтевом **(4),** добија се: $Cost \overset{3}{\succ} ISale, Sec \overset{3}{\succ} CustEase, Cost \overset{3}{\succ} Sec$, докврединости о приоритетима међу осталим паровима обележја остају недефинисане. У том случају, подразумевана вредност за недефинисане релативне приоритете је један, сходно коментарима датим уз Дефиницију 1 и чињеници да све што се не дефинише се може сматрати да није од значаја за корисника. Тако, на нивоу обележја се добија матрица попуњена следећим вредностима:

$$Cost \overset{3}{\succ} ISale, Sec \overset{3}{\succ} CustEase, Cost \overset{3}{\succ} Sec, Cost \overset{1}{\succ} CustEase, ISale \overset{1}{\succ} CustEase, ISale \overset{1}{\succ} Sec .$$

Након примене стандрадног Saaty-јевог AHP алгоритма, добијају се следеће вредности за рангове обележја *Sec, CusmEase, ISale* и *Cost*, редом 0.25, 0.19, 0.16 и 0.40, чиме су рачунања на нивоу обележја завршена. **(8)**

Следећи корак обухвата рангирање над скупом инстанци сваког обележја. Аналогно нивоу обележја, потребно је одредити задовољене делове захтева **(6)** и **(7),** попунити матрице на нивоу инстанци и применити стандардни AHP алгоритам. У случају прве опције, у оба захтева **(6)** и **(7)** су задовољени *then*-делови, чијим се спајањем добија: *ISale.Medium* $\overset{3}{\succ}$ *ISale.High, ISale.High* $\overset{5}{\succ}$ *ISale.Low, ISale.Medium* $\overset{5}{\succ}$ *ISale.Low*. Дакле, локални рангови за инстанце које представљају високу, средњу и ниску вредност обележја *ISale* су редом: 0.30, 0.60 и 0.10. Коначно, одговарајуће глобалне вредности су:

$ISale.High: 0.16 * 0.30 = 0.05, ISale.Meidum: 0.16 * 0.60 = 0.10, ISale.Low: 0.16 * 0.10 = 0.016$
.

Сада посматрајмо следећу опцију, сервис *Cyber Source*. Исто као и у случају прве опције, на нивоу обележја имамо следеће рангове обележја *Sec, CusmEase, ISale* и *Cost*, редом 0.25, 0.19, 0.16 и 0.40. На нивоу инстанци, *else*-део захтева **(6)** је задовољен, што даје: ($ISale.High \overset{1}{\succ} ISale.Medium$, $ISale.High \overset{3}{\succ} ISale.Low$, $ISale.Medium \overset{3}{\succ} ISale.Low$), као и *else*-део захтева **(7),** на основу чега је ($ISale.High \overset{3}{\succ} ISale.Medium$, $ISale.High \overset{5}{\succ} ISale.Low$). Међутим, ова два скупа релативних приоритета не могу бити спојена јер релативни приоритет међу инсанцама *ISale.High*и*ISale.Low*није јединствено одређен, што рачунање локалног ранга инстанци обележја *ISale* чини немогућим. ■

Ситуације као управо наведену је потребно детектовати и препознати као неконзистенције у захтевима корисника, и у том циљу се уводи следећа дефиниција (Дефиниција 8); док су неконзистенције и њихов утицај на алгоритам детаљније анализирани у секцији 4.1.2.3.

Дефиниција 13(Конзистентан скуп корисничких захтева).
Скуп корисничких захтева је ***конзистентан*** уколико су задовољени следећи услови:

- Сваки захтев је дефинисан у формама наведеним у дефиницијама 2-5;
- Свака вредност матрице на нивоу обележја или није дефинисана или је јединствено одређена на основу безусловно дефинисаних захтева и задовољених делова (*then* или *else*) условно дефинисаних захтева;
- Свака вредност матрице сваког обележја на нивоу инстанци или није дефинисана или је јединствено одређена на основу безусловно дефинисаних захтева и задовољених делова (*then* или *else*) условно дефинисаних захтева.

Ако бар један од дефинисаних захтева није задовољен, скуп захтева се сматра *неконзистентним*. ■

На крају, као што је већ наведено у претходном примеру, расположиве опције могу бити рангиране на основу захтева дефинисаних на нивоима обележја и

инстанци само у случају да је реч о конзистентним захтевима, што је формално доказано у секцији 4.1.3.

4.4.1.2.Адресирање захтева о доминантним приоритетима

Као увод у разматрање рангирања на основу захтева о доминантним приоритетима, наводи се опште познат пример о позиционим бројним системима, развијен између 1. и 5. века од стране Индијских математичара [188] који ће послужити за инспирацију решења постављног проблема. Позициони бројни систем је систем бројева који су представљени помоћу цифара које се налазе на одређеним позиционим местима, и вредност цифре директно зависи од позиционог места на којем се налази. На сваком позиционом месту се налази тачно једна цифра, а позиционо место се дефинише као степен базе система (нпр, у декадном бројном систему, позициона места су 1, 10, 10^2, ...). На крају, вредност тако представљеног броја се одређује сумирањем вредности цифара помножених са позиционим местима на којима се налазе [188].

Овако дефинисано поређење бројева се може интерпретирати као доминанти релативни приоритет већих позиционих места, па се вредност броја $a_n...a_1a_0$ записаног у b-базном позиционом систему добија као $a_n \cdot b^n + ... + a_1 b + a_0$, где је $a_i \in \{0,..., b-1\}, i \in \{0,..., n\}$.

Са датим примером је пронађена аналогија између позиционих места и цифара са редом обележјима са доминантним приоритетом и инстанцама у двослојној структури обележја и инстанци. Такође, по аналогији са мешовито-базним позиционим системима [188], дефинише се *к*-елементни низ база одређен *к*-торком обележја са доминантним релативним приоритетима, на следећи начин: Нека је у датом скупу обележја $C = \{c_1,..., c_n\}$ и одговарајућих инстанци $QT = \bigcup_{i=1}^{n} QT_i$, уређена k-торка обележја $(c_{i_1},..., c_{i_k})$ са релативним доминантним приоритетима. ***Функција агрегирања*** опције која је означена скупом инстанци

$\{qt_{j_1}^{i_1},\ldots,qt_{j_s}^{i_s}\}$, се дефинише у односу на *базу*

$(\prod_{j=2}^{k}(|QT_{i_j}|+1), \ \ldots, \ (|QT_{i_{k-1}}|+1)(|QT_{i_k}|+1), \ |QT_{i_k}|+1, \ 1)$ са:

$$F(qt_{j_1}^{i_1},...,qt_{j_s}^{i_s}) = \sum_{l=1}^{k} b_l \cdot r_D(qt_{j_l}^{i_l}) + r(\{qt_{j_1}^{i_1},...,qt_{j_s}^{i_s}\} \setminus \{qt_{j_1}^{i_1},...,qt_{j_k}^{i_k}\}), s \le m,$$

$$\text{где је } b_l = \begin{cases} 1, & l = k \\ \prod_{j=l+1}^{k}(|QT_{i_j}|+1), & 1 \le l < k \end{cases} \quad \textbf{(13)}$$

а$r()$и$r_D()$ су раније дегинисани редом са**(3)**и**(12).**

На основу функције агрегирања *F*, дефинише се ранг опције означене скупом инстанци $\{qt_{j_1}^{i_1},...,qt_{j_s}^{i_s}\}$ у случају домнантног релативног приоритета *k*-торке обележја $(c_{i_1},...,c_{i_k})$, на следећи начин:

$$\bar{r}(qt_{j_1}^{i_1},\ldots,qt_{j_s}^{i_s}) = F(qt_{j_1}^{i_1},\ldots,qt_{j_s}^{i_s}) \Big/ \prod_{j=1}^{k}(|QT_{i_j}|+1) \quad \textbf{(14)}$$

Коректност уведених формула је доказан у секцији 4.1.3. а начин њихове примене ће се илустровати на Примеру 1 (Проширење 3) уведеном раније у секцији 4.1.1.

Пример 1 (Проширење 3-наставак).Претпоставимо да је корисник осим захтева **(9)** и **(10)** дефинисао и следеће захтеве:

на нивоу обележја: $Sec.High : ISale \overset{5}{\succ} CustEase \mid ISale \overset{1}{\succ} CustEase$;

и на нивоу инстанци:

$CustEase.High \overset{3}{\succ} CustEase.Low, CustEase.Medium \overset{3}{\succ} CustEase.Low$ и

$ISale.High \overset{3}{\succ} ISale.Low, ISale.High \overset{3}{\succ} ISale.Medium$.

На основу квалитативних рангова инстанци обележја са доминантним релативним приоритетима, дефинисаним са **(12),**имамо $r_D(Cost.High) = 1, r_D(Cost.Medium) = 2, r_D(Cost.Low) = 3$.

Такође, корисник је дефинисао да жели висок или средњи ниво обележја Sec, што значи да инсатнце *Sec.High*и*Sec.Medium*имају исти релативни приоритет у доношењу одлука, па су њихови рангови дефинисани са: $r_D(Sec.High) = 2, r_D(Sec.Medium) = 2, r_D(Sec.Low) = 1$. Значи, база креирана на основу пара обележја са релативним доминантним приоритетима *(Cost, Sec)*је (3,1).

С друге стране, на основу стандардног AHP алгоритма, глобални рангови обележја*CustEase*и*ISale*су, у случају да јеуслов *Sec.High* задовољен, 0.167 и 0.833; док у супротном, оба ранга имају једнаку вредност 0.5. Локални рангови инстанци које представљају високе, средње и ниске вредности обележја*CustEase*су редом: 0.43, 0.43, 0.14. Такође, локални рангови инстанци које представљају високе, средње и ниске вредности обележја*ISale*су редом 0.6, 0.2, 0.2.

На крају, сервис*VeriSign*је означен са инстанцама које представљајувисоку вредност обележја*Sec*, ниску обележја *ISale*и ниску обележја*Cost* (Слика 2), па се његов ранг на основу формуле **(14)**рачуна као:

$\bar{r}(Cost.Low, Sec.High, ISale.Low) = (9 + 2 + 0.833 \cdot 0.2)/(9 + 2 + 1) = 0.93$. ■

4.4.1.3.CS-AHP алгоритам за адресирање разних врста захтева над двослојном структуром

У овој секцији се на основу анализа и резултата из претходне две врши адаптација S-AHP алгоритма у циљу омогућавања рангирања опција на основу безусловно и условно дефинисаних захтева као и захтева о доминантним релативним приоритетима. Алгоритам који се предлаже је задржао постојеће кораке у S-AHP процесу предстаљеном у секцији 4.2., с додатком неколико нових и модификацијом неких од постојећих. Тако, алгоритам који омогућава анализу разних врста захтева дефинисаних над двослојном структуром обележја и инстанци, под називом ***CS-AHP*** (енгл. *Conditional S-AHP*) се састоји од следећих корака:

Корак 1 (Дефиниција захтева). У овом кораку корисници дефинишу захтеве на оба нивоа, нивоу обележја и нивоу инстанци. Укупан број захтева које корисник може дефинисати није ограничен, а могуће је дефинисати више различитих условних захтева о релативним приоритетима између истог пара обележја (инстанци), или, их пак, оставити недефинисаним. У случају дефинисања захтева о доминантним приоритетима, дефинишу се релативни рангови, у складу са **(12).**
У овој фази се врши провера првог услова из Дефиниције 6, тј. кориснику није дозвољено да дефинише захтеве о релативним приоритетима између обележја (тј инстанци) који зависе од њих самих.

Корак 2 (Подела скупа обележја). На основу дефинисаног скупа захтева, одваја се скуп обележја за које је дефинисано да имају доминантни релативни приоритет.

Даље се претпоставља да је дефинисан и скуп опција означен са највише једном инстанцом сваког од обележја датог скупа, а следећи алгоритамски кораци се односе на њихово рангирање и извршавају се посебно за сваку од расположивих опција:

Корак 3 (Одређивање захтева на нивоу обележја).За сваку опцију се проверавају услови дефинисани у сваком од условних затева. Уколико је задовољен услов, узима се *then*-део захтева, у супротном *else*-део. Важно је нагласити да све расположиве опције нису означене инстанцама сваког од обележја, тако да су могуће ситуације када услов не може бити проверен за дату опцију. У том случају, сматраће се да је логичка вредност целог израза нетачна, тј да опција не задовољава дефинисани услов.

Корак 4 (Одређивање рангова на нивоу обележја). На основу безусловно дефинисаних захтева и задовољених делова условно дефинисаних, попуњава се матрица на нивоу обележја. У случају да матрица не може бити јединствено попуњена, скуп захтева се проглашава неконзистентним и кориснику може бити дозвољено да донесе одлуку у случајевима вишеструких вредности. Након тога се рачунају рангови на нивоу обележја и врши филтрирање мање значајних обележја

и пропорционална прерасподела рангова, по процедури представљеној у Секцији 4.2.

Корак 5 (Одређивање захтева на нивоу инстанци). Аналогно нивоу обележја, сваки условно дефинисани захтев о релативним приоритетима на нивоу инстанци се проверава и ситуације где опција није означена са неком од инстанци из захтева се адресирају на исти начин. Уколико је корисник у претходном кораку филтрирао неко од обележја, условни захтеви који у услову садрже инстанце филтрираног обележја, морају се модификовати да не зависе од обележја мањег приоритета. Филтрирање се ради на следећи начин:

Означимо са *qt* било којуинстанцу обележја мањег приоритета и сваки условно дефинисани захтев трансформишимо итеративно по следећим правилима:

*Правило 1:*Условни захтев у облику $\Psi \wedge qt : R^{\succ}(c_i, QT_i)_1 \mid R^{\succ}(c_i, QT_i)_2$ или $\Psi \wedge \neg qt : R^{\succ}(c_i, QT_i)_1 \mid R^{\succ}(c_i, QT_i)_2$ се трансформише у $\boldsymbol{\Psi : R^{\succ}(c_i, QT_i)_1 \mid R^{\succ}(c_i, QT_i)_2}$;

*Правило 2:*Условни захтев у облику $\Psi \vee qt : R^{\succ}(c_i, QT_i)_1 \mid R^{\succ}(c_i, QT_i)_2$ или $\Psi \vee \neg qt : R^{\succ}(c_i, QT_i)_1 \mid R^{\succ}(c_i, QT_i)_2$ се трансформише у $\Psi : R^{\succ}(c_i, QT_i)_1 \mid R^{\succ}(c_i, QT_i)_2$;

*Правило 3:*Условни захтев у облику $qt : R^{\succ}(c_i, QT_i)_1 \mid \varnothing$ или $\neg qt : R^{\succ}(c_i, QT_i)_1 \mid \varnothing$ се сматра неадекватним и не узима се у разматрање;

*Правило 4:*Условни захтев у облику $qt : R^{\succ}(c_i, QT_i)_1 \mid R^{\succ}(c_i, QT_i)_2$ или $\neg qt : R^{\succ}(c_i, QT_i)_1 \mid R^{\succ}(c_i, QT_i)_2$ се сматра неконзистентним па се процес завршава.

Правила 1 и 2 се понављају све док се не задовољи правило 3 или 4, или се добије захтев који више не садржи инстанце филтерисаних обележја.

Корак 6 (Одређивање рангова на нивоу инстанци). На основу безусловно дефинисаних захтева и задовољених делова условно дефинисаних, попуњавају се матрице на нивоу инстанци сваког од обележја који није филтриран. У случају да матрица не може бити јединствено попуњена, скуп захтева се проглашава неконзистентним и кориснику може бити дозвољено да донесе одлуку у случајевима вишеструких вредности. Након тога се рачунају локални рангови на

нивоу инстанци, а на основу претходно добијених рангова на нивоу обележја, рачунају се глобалне вредности рангова.

Корак 7 (Рангирање опција). Крајњи ранг сваке од опција се рачуна по формули **(14).**

4.1.1.4. Теоретска анализа (коректност, комплетност и ефикасност)

Анализа предложеног алгоритма за рангирање обухвата анализу коректности, комплетности, временске сложености и утицаја цикличности у условним захтевима као и идентификованих неконзистенција, на процес рангирања.

Како предложени CS-AHP метод представља проширење постојећег, деценијама добро утемељеног AHP алгоритма, у циљу доказа коректности потребно је показати да основна својства алгоритма нису нарушена. Предложено проширење је направљено с циљевима: *(i)* анализе условно дефинисаних захтева, и*(ii)*анализе захтева о доминантним приоритетима. У анализи условно дефинисаних захтева је стандардном процесу додата провера задовољености услова дефинисаних у облику логичких израза, што не утиче на коректност примене AHP метода. С друге стране, за анализу захтева о доминантним релативним приоритетима, уведене су формуле **(13)** и **(14)**, за које је потребно додатно доказати да:

1) Функција агрегације дефинисана са**(13)**дефинише рангирање опција у складу са дефиницијама 7 и 8 о доминантним релативним приоритетима;
2) Рангирање дефинисано са **(14)** дефинише вредности из интервала [0, 1];
3) У случају недефинисања обележја са доминантним релативним приоритетима, формула **(14)**се своди на формулу **(3)**.

У том циљу се дефинишу следећа тврђења;

Лема. За сваку комбинацију опција $\{qt^{i_1}_{j_1},..., qt^{i_s}_{j_s}\}, s \le m$ у случају *к*-торке обележја $(c_{i_1},..., c_{i_k})$ са доминантним релативним приоритетима, за свако $1 \le u \le k$ важи неједнакост:

$$\sum_{l=u}^{k} b_u r_D(qt_{j_l}^{i_l}) + r(\{qt_{j_1}^{i_1},..., qt_{j_s}^{i_s}\} \setminus \{qt_{j_1}^{i_1},..., qt_{j_k}^{i_k}\}) \le b_{u-1}, \text{ где је } b_0 = \prod_{j=1}^{k} \left(| QT_{i_j} | +1 \right).$$

Доказ је дат у Додатку А.

Пропозиција 1. Функција агрегирања дефинисана са **(14)** дефинише рангирање над опцијама на основу безусловно и условно дефинисаних захтева, као и захтева о доминантим релативним приоритетима, све у складу са дефиницијама 7 и 8.

Доказ је дат у Додатку А.

Као директна последица Леме (за случај када је *u=1*) и чињенице да у случају недефинисања доминантног релативног приоритета, сума $\sum_{l=1}^{k} b_l r_D(qt_{j_l}^{i_l})$ је једнака нули, добија се следеће тврђење.

Пропозиција 2.За сваку опцију*o*означену скупом инстанци $\{qt_{j_1}^{i_1},...,qt_{j_s}^{i_s}\}, s \le m$, важи:

1. $0 \le F(qt_{j_1}^{i_1},...,qt_{j_s}^{i_s}) \le \prod_{j=1}^{k} \left(| QT_{i_j} | +1 \right)$, независно од*s;*
2. У случају недефинисања захтева о доминантном релативном приоритету обележја, важи: $\bar{r}(qt_{j_1}^{i_1},...,qt_{j_s}^{i_s}) = r(qt_{j_1}^{i_1},...,qt_{j_s}^{i_s})$. ■

Као закључак дефинисаних тврђења имамо да се помоћу рангирања *r* (дефинисаним са **(14)**) дефинише ранг опција на основу скупа безусловно и условно дефинисаних захтева, као и захтева са доминантним релативним приоритетима, добијеним проширењем стандардног AHP метода (односно, S-AHP метода) над двослојном структуром обележја и инастанци.

Улазни модел CS-AHP алгоритмаје четворка *(O, C, QT,* $\mathfrak{R}$ *),* који у односу на претходно развијени S-AHP метод садржи додатно димензију $\mathfrak{R}$ која представља

скуп дефинисаних захтева. Као резултат, метод даје рангирање скупа расположивих опција *O*. Важи и следеће јаче тврђење:

Пропозиција 3. За сваки скуп захтева и сваки коначан скуп опција, CS-AHP алгоритам у коначном броју корака одређује јединствено рангирање над скупом расположивих опција, или закључује да скуп захтева није конзистентан.

На основу алгоритамских корака представљених у претходној секцији, може се уочити да је алгоритам јасно подељен на два нивоа: ниво обележја и ниво инстанци. За сваку опцију, одговарајуће матрице на оба нивоа су јединствено попуњене. Такође, при идентификацији првог захтева који нарушава јединственост, алгоритам завршава са извршавањем обавештењем о неконзистенцији у захтевима. Иако алгоритам врши детекцију нарушавања дефинисаних правила конзистенције, кориснику се може дозволити да донесе одлуку о вишезначности појединих елемената у матрицама, тј он може дефинисати коју вредност треба сматрати одговарајућом, тако да може да се настави процес рангирања. С друге стране, уколико се корисник не жели укључити у цели процес, за разрешавање уочених неконзистенција се може користити принцип прве вредности, у смислу прихватања вредности која је прво добијена на основу дефинисаних захтева. На тај начин, уколико се захтеви анализирају редоследом којим су дефинисани, сваки захтев који противречи вредностима дефинисаним на основу захтева који му претходе се сматра неконзистентним. При њиховој идентификацији, вредности у матрицама се неће променити и алгоритам наставља са извршавањем. Наведене две могућности (када је корисник укључен у разрешавање неконзистенција и принцип прво дефинисане вредности) омогућавају да за сваки скуп захтева алгоритам као резултат генерише рангирање расположивих опција. Ипак, алгоритам у општем случају не даје јединствено решење јер зависи од одлуке корисника о разрешењу неконзистенција или о редоследу дефинисања захтева.

Значи, у случају неконзистентних захтева, рангирање опција није јединствено дефинисано од начина за разрешавање уочених неконзистенција, па се оно сматра од директног утицаја на крајње рангирање. С друге стране, како се сваки од дефинисаних захтева посебно проверава за сваку опцију, онемогућене су појаве

циклуса и зависности међу захтевима. Такође је важно нагласити, да у случају дефинисања захтева о доминантном приоритету, не може доћи до нарушења конзистентности јер се они дефинишу безусловно у односу на остале [113].

У области анализе захтева, разматрају се разне врсте упита над дефинисаним захтевима и опцијама: *(i)**упит доминације***, који се односи на поређење две опције, *(ii)**упит сортирања***, којим се дати коначан скуп опција уређује у растући/опадајући поредак сходно генерисаним приоритетима на основу дефинисаних захтева, и *(iii)**упит оптимизације***, који одређује теоријски најбољу опцију [95]. У области анализе захтева, показано је да су сва три упита у општем случају NP-сложена.

Пропозиција 4. За дати CS-AHP модел *(O, C, QT,* $\Re$*)*, упити доминације и сортирања имају полиномијалну временску сложеност, док је упит оптимизације у најгорем случају NP -сложен.

Означимо са $\Re_C$ скуп захтева дефинисаних на нивоу обележја, $\Re_c^D$ скуп захтева о доминантном релативном приоритету, и са $\Re_{QT}$ скуп захтева на нивоу инстанци, где важи: $\Re = \Re_C \cup \Re_{QT}, |\Re_C| = n_C, |\Re_c^D| = n_c^D, |\Re_{QT}| = n_{QT}$. Као што је раније наведено, за сваку опцију, сваки захтев се анализира и попуњава се једна матрица на нивоу обележја и $n_C - n_C^D$ матрица на нивоу инстанци, на основу којих се рачунају локални рангови. Претходно, услови у сваком условно дефинисаном захтеву се проверавају, што је укупно $O(n_R + (n_C - n_C^D)^2 + (n_C - n_C^D)n_{QT}^2)$ операција. Ове операције представљају основну цену метода заснованог на поређењу парова, а оне се извршавају за сваку од n_O расположивих опција, што је укупно $O(n_O(n_R + (n_C - n_C^D)^2 + (n_C - n_C^D)n_{QT}^2))$.

У случајевима упита доминације и сортирања, скуп расположивих опција је унапред познат, што укупно даје полиномијалну сложеност. С друге стране, упит оптимизације се решава једино генерисањем свих могућих комбинација инстанци и њиховим рагирањем, што је експоненцијална временска сложеност.

Важно је истаћи да, у случају доминантног релативног приоритета целог скупа обележја (тј, лексикографског поретка), упит оптимизације се може решити једноставним обиласком скупа свих обележја и одабиром инстанци највећег приоритета. У том случају, имамо линеарну сложеност. Такође, у случају дефинисања само безусловних захтева, упит оптимизације захтева полиномијално време за локално рангирање инстанци сваког обележја и генерисања оптималне опције одабиром инстанци највећег ранга у оквиру сваког од обележја.

У горњем параграфу су идентификована два случаја у којима упит оптимизације има линеарну, односно полиномијалну временску сложеност, на основу чега се закључује да поједине струкутре захтева оптимизују упит оптимизације, што овај проблем поставља предметом будућег истраживања.

4.4.2. Захтеви над дослојном структуром нефункционалних каракетристика SOA фамилија

Као што је претходно предстаљено у секцији 4.1.2., двослојна структура нефункционалних захтева над SOA фамилијом се креира уз корисниково дефинисање преференци и личних ставова над могућим вредностима сваке од нефункционалних карактеристика. Сходно резултатима представљеним у секцији 4.3., над датом структуром је могуће дефинисати разне врсте захтева о приоритетним односима међу нефункционалним карактеристикама (које представљају ниво обележја у структури) и њиховим могућим вредностима (које представљају ниво инстанци). Стога, корисник може представити своја очекивања и ставове о нефункционалним карактеристикама целе фамилије (или појединих њених делова) у облику безусловно или условно дефинисаних захтева, или пак захтева о доминантним приоритетима. За сваку од могућих комбинација сервиса неопходно је обезбедити мерење квалитета (представљено степеном испуњења дефинисаних захтева), па се избором комбинације највећег квалитета добија конфигурација целе фамилије која у највећој мери одговара захтевима корисника.

4.4.2.1.Мере испуњења дефинисаних захтева над нефункционалним карактеристикама

Као што је раније представљено, улазни модел CS-AHP алгоритма је четворка *(O, C, QT,* $\Re$ *),* гдје је *O*-скуп опција, *C*-скуп обележја, *QT*-скуп инстанци, $\Re$ -скуп захтева; а као резултат алгоритам даје нумеричке вредности рангова сваке од расположивих опција сходно степену испуњења дефинисаних захтева. Рангови су генерисани на основу претходно добијених ранг вредности за оба нивоа у структури, тако да ће се у наставку управо ове вредности (које одговарају ранг вредностима сваке од нефункционалних карактеристика и њених могућих вредности инстанци) користити за дефинисање мера квалитета комбинације сервиса. Сходно резултатима секције 4.1.2.,свака комбинација сервиса $\langle s_1,\ldots,s_k \rangle$ се карактерише тачно једном комбинацијом инстанци $QT_{i_1}^1 \times \ldots \times QT_{i_n}^n$, на основу агрегираних вредности нефункционалних карактеристика $Q_{agg} = \langle q_1^{agg},..,q_n^{agg} \rangle \in R^n$, па вредности рангова над инстанцама дефинишу следеће мере квалитета.

Дефиниција 14 (CS-AHP мере квалитета над SOA фамилијом).
За дату SOA фамилију $G_{PM}^{FM} = (\nu, \varepsilon)$, CS-AHP метод за улазне аргументе има модел (*C,* $Q_{agg}^{LB}, Q_{agg}^{UB}$, *QT, R*), који у односу на двослојну *структуру (C,* $Q_{agg}^{LB}, Q_{agg}^{UB}$, *QT)* додатно садржи скуп захтева*R* дефинисаних од стране корисника о префериранim вредностима нефункциоанлних каракетристика. Ранг вредности скупа нефункционалних карактеристика (које у моделу одговарају скупу обележја; означено са $r_1^c,..., r_n^c$) и колекције покривајућих подинтервала (која у моделу одговара скупу инстанци; означено са $\langle r_1^1,\ldots,r_{k_1}^1 \rangle,\ldots,\langle r_1^n,\ldots,r_{k_n}^n \rangle$) добијене на основу CS-AHP алгоритма дефинишу следеће мере над комбинацијама сервиса:

(1') ***1-димензиона мера квалитета покривајућег подинтервала*** QT_j^i**:**
$r^{QT}\left(QT_{ij}^i\right) = r_i^c \cdot r_j^i$

(1) ***Мера квалитета комбинације покривајућих подинтервала*** $QT_{i_1}^1 \times \ldots \times QT_{i_n}^n$**:**

$$r^{QT}\left(QT_{i_1}^{1}\times\ldots\times QT_{i_n}^{n}\right)=\frac{1}{n}\sum_{k=1}^{n}r^{QT}\left(QT_{i_k}^{k}\right),\ i_l\in\{1,\ldots,k_l\},l\in\{1,..,n\}$$

(2') ***Мера квалитета конфигурацијеза i-ту нефункционалну карактеристику*:**

$$r_i^S\left(\langle s_1,\ldots,s_k\rangle\right)=\begin{cases}r_i^c\cdot\left[r_j^i+\dfrac{m_i^j-q_i^{agg}}{m_i^j-a_i^j}\left(r_j^i-r_{j-1}^i\right)\right], & q_i^{agg}\le m_i^j\\ r_i^c\cdot\left[r_j^i+\dfrac{q_i^{agg}-m_i^j}{b_i^j-m_i^j}\left(r_{j+1}^i-r_j^i\right)\right] & q_i^{agg}\ge m_i^j\end{cases},\quad \text{где је } q_i^{agg}\in QT_j^i$$

агрегирана вредност i-те нефункционалне карактеристике, $m_i^j=\dfrac{a_i^j+b_i^j}{2}$-средина j-тог покривајућег подинтервала QT_j^i, $r_0^i=r_1^i+\dfrac{r_2^i-r_1^i}{m_2^i-m_1^i}\left(a_1^i-m_1^i\right)$и

$$r_{k_i+1}^i=r_k^i+\frac{r_{k-1}^i-r_k^i}{m_{k-1}^i-m_k^i}\left(b_k^i-m_k^i\right)$$

(2)*Мера квалитета конфигурације*: за комбинацију сервиса$\langle s_1,\ldots,s_k\rangle$ чије агрегиране вредности нефункционалних карактеристика $Q_{agg}=\langle q_1^{agg},..,q_n^{agg}\rangle\in R^n$ припадају комбинацији покривајућих подинтервала$QT_{i_1}^1\times\ldots\times QT_{i_n}^n$, мера квалитета се дефинише са:

$$r^S\left(\langle s_1,\ldots,s_k\rangle\right)=\frac{1}{n}\sum_{i=1}^{n}r_i^S\left(\langle s_1,\ldots,s_k\rangle\right).$$

У циљу једноставности, мере су прво дефинисане за једну нефункционалну карактеристику (*1'* и*2'*), и даље генерализоване за случај*n*нефункционалних карактеристика (*1*и*2*). Мере су добијене сумирањем свих мера нефункционалних карактеристика тежинским функцијама у$r^{QT}()$и$r^S()$,на основу хипотезе да агрегиране вредности квалитета могу бити евалуиране као просечне вредности одговарајућих мера квалитета компоненти карактеристика [185] [190].

У циљу илустрације, наводи се следећи пример.

Пример 2Ошибка! Источник ссылки не найден. **(Проширење 1).**

Претпоставимо да су интервали могућих вредности за нефункционалне карактеристике цена (обележје *Price*), време одзива (обележје *ResponseTime*),

доступност (обележје *Availability*) и поузданост (обележје *Reliability*) за целу фамилију представљену у примеру 2 (Слика 3), редом [100, 350], [15,55], [0.05, 0.2] и [0.07, 0.15].

Такође, претпоставимо да је корисник I дефинисао следеће истанце:

- ниска цена (*Price.Low*) којој одговарају вредности мање од 200, средња цена (*Price.Med*) којој одговарају вредности из интервала [200, 300], и висока цена (*Price.High*) са вредностима које су веће од 300;
- мало време одзива (*RespTime.Low*)којем одговарају вредности мање од 20, средње време одзива (*RespTime.Medium*) са вредностима из интервала [20, 40], и велико време одзива (*RespTime.High*) са вредностима које су веће од 40;
- ниски ниво доступности (*Availability.Low*)са вредностима не већим од0.12 и висок ниво доступности (*Availability.High*) са вредностима већим од 0.12;
- ниски ниво поузданости (*Reliability.Low*)са вредностима не већим од0.10 и висок ниво поузданости (*Reliability.High*) са вредностима већим од 0.10.

Даље, посматрајмо комбинацију сервиса која даје следеће агрегиране вредности $Q^1_{aqq} = <100,\ 19,\ 0.15,\ 0.09>$ редом за нефункционалне карактеристике тј обележја у моделу *Price*, *RespTime*, *Availability* и*Reliability*. Дате вредности припадају интервалима дефинисаним као *Price.Low*, *RespTime.Low*, *Availability.High* и *Reliability.Low*.

Корисник I може дефинисати и додатне захтеве (у односу на оне претходно дефинисане у Секцији 4.3), и претпоставимо да након процесирања свих дефинисаних захтева, применом CS-AHP алгоритма одређен је ранг вредности за обележје *ResponseTime* $r^C_{\mathrm{Re}\,spTime} = 0{,}54$ и његове инстанце*RespTime.Low*, $r^{\mathrm{Re}\,spTime}_{\mathrm{Re}\,spTime.Low} = 0{,}50$ и *RespTime.Medium*, $r^{\mathrm{Re}\,spTime}_{\mathrm{Re}\,spTime.Medium} = 0{,}40$. Иако је ранг 0.40 дефинисан за цели интервал [15, 20] који представља вредност средњег времена одзива (сходно преференцама Корисника А дефинисаним при самом креирању двослојне структуре), вредност 0.40 се додељује средини интервала (тј. вредности 17.5), и конструише се униформна расподела вредности рангова од 0.40 до 0.50 (вредност ранга инстанце*RespTime.Low*) над вредностима обележја *RespTime*,

чиме се добија: $0.40+\frac{19-17.5}{20-17.5}(0.50-0.40)=0.46$. На крају, мера квалитета комбинације сервиса чија је агрегирана вредност обележја *RespTime*једнака 19 је дата са: $0.54*0.46=0.25$.

Додатно, претпоставимо да су израчунате вредности мера квалитета осталих обележја (*Price, Availability*и*Reliability*) редом 0.12, 0.22 и 0.12, па се свеукупна мера квалитета посматране комбинације сервиса дефинише као: $\frac{0.12+0.25+0.22+0.12}{4}=0.18$. ■

4.4.2.2.Анализа мера $r^{C}()$и $r^{QT}()$

У претходној дефиницији, ранг вредности над инстанцама и обележјима у двослојној структури, генерисан на основу CS-AHP алгоритма, је прво додељен подинтервалима које представљају могуће инстанце нефункционалних карактеристика, чиме је добијена мера $r^{QT}()$. Сходно основном својствуCS-AHP алгоритма, веће вредности дефинисане мере одговарају комбинацијама подинтервала могућих вредности нефункционалних карактеристика, које у већој мери одговарају дефинисаним захтевима корисника.

Такође, исте ранг вредности је додељен средњим вредностима подинтервала са униформним дистрибуцијама ка ранговима првих суседних подинтервала (исте нефункционалне карактеристике), чиме је генерисана мера квалитета комбинације сервиса $r^{S}()$. Овако дефинисаном мером, такође на основу основног својстваCS-AHP алгоритма, већа вредност мере одговара комбинацији сервиса која у већем степену испуњава дефинисане захтеве корисника[180].

Јасно је да обе дефинисане мере одговарају захтевима корисника, и представљају њихову квантификацију на нивоу инстанци (тј подинтервала могућих вредности нефункционалних карактеристика) мером $r^{QT}()$, односно на нивоу комбинација сервиса (сходно агрегираним вредностима нефункционалних карактеристика) мером $r^{S}()$. Како су обе мере дефинисане на основу истих резултата CS-AHP алгоритма, за анализу се намеће: *(i)* анализа наслеђених карактеристика од самог CS-AHP алгоритма, као и *(ii)* анализа међусобне

повезаности дефинисаних мера. Тиме ће се омогућити да се *(i)* изврши анализа својстава дефинисаних мера, и *(ii)* одреди минималан скуп мера које једнозначно карактеришу SOA фамилију на основу нефункционалних карактеристикама и захтева корисника над њима.

Осим поменутог својства монотоности, за дефинисане мере квалитета $r^{QT}()$и $r^{S}()$ важе и следећа својства:

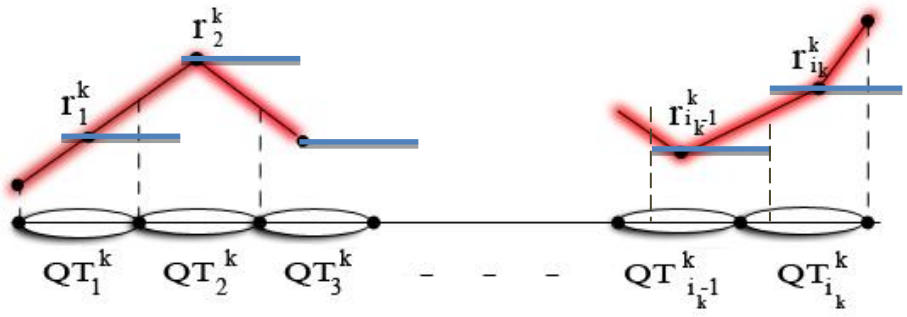
- Обе мересу јединичне, тј важи $0 \le r^{QT}(), r^{S}() \le 1$, сходно основним карактеристикама CS-AHP алгоритма [112]
- Мера квалитета $r^{QT}()$, у општем случају, не одговара монотоним тенденцијама нефункционалних карактеристика (тј. монотоно растућим и опадајућим карактеристикама) дефинисаним у[185]. CS-AHP алгоритам као улазне параметре има скуп захтева корисника о префериранимвредностима представљеним у облику релативних приоритета између парова обележја и њихових инстанци. За дефинисане захтеве се не уводи услов мотоности јер би у том случају нестао смисао дефинисања препознатих врста захтева чиме би се знатно умањила и експресивност самог алгоритма. Стога, генерисане мере па ни мере квлитета комбинација инстанци, не одговарају монотоним функцијама.
- Израчунавање мера $r^{QT}()$ и$r^{S}()$захтева полиномијално време у односу на скуп расположивих комбинација инстанци тј сервиса, док упит о одређивању највише префериране комбинације инстанци тј сервиса је у најгорем случају NP-сложен задатак (последица Пропозиције 4).

Из наведног се може закључити да развијене мере имају исти недостатак као и представљени CS-AHP алгоритам, који се огледа у временској сложености одређивања комбинације инстанци тј сервиса са оптималном вредношћу мере квалитета. Међутим, оне су изузетно ефикасан алат за мерење разлике у квалитету између расположиве две комбинације инстанци тј сервиса. Стога, уколико се оне прихвате за мере квалитета у SOA фамилији, за оптималну селекцију ће бити неопходно развити приступ који може у одређеној мери савладати идентификоване недостатке датих мера.

Међутим, пре него што се препоручи њихова примена, потребно је анализирати међусобни однос дефинисаних мера чиме ће се добити одговор на

питање да ли је могуће само једну од њих одабрати за меру квалитета над SOA фамилијом, а да су при томе садржане све информације које одговарају претходно дефинисаним захтевима корисника.

Ради једноставности, шематски приказ дефинисаних мера и њихов међусобни однос за једнодимензиони случај је представљен на Слици 24.Иако је реч о најједноставнијем случају, јасно је да не постоји јасна интерпретација о односу међу дефинисаним мерама. Конкретно, над инстанцама QT_2^k и $QT_{i_k}^k$ важи релација $r_2^k > r_{i_k}^k$, али не постоји јединствена интерпретација односа међу мерама комбинације сервиса чије каракетристике припадају датим инстанцама, с обзиром да је на једном делу већа мера над истанцом $QT_{i_k}^k$, док је на остатку обрнуто.



Слика 24. Шематски приказ мера r^{QT} и r^{S} над нефункционалним каракетристикама

Следеће проширење Примера 2 показује комбинаторну лакоћу генерисања неодређености односа међу мерама у вишедимензионом случају.

Пример 2 (Проширење 2). У претходно дефинисаном примеру е-продавнице додатно посматрајмо:

- Комбинацију сервиса која има агрегиране вредности $Q_{aqq}^2 =< 250,\ 30,\ 0.15,\ 0.09 >$ редом за нефункционалне карактеристике *Price, ResponseTime, Availability и Reliability.*
- CS-AHP ранг вредности за обележје *Price* $r_{\text{Price}}^{C} = 0{,}16$ и његове инстанце: *Price.Low*, $r_{\text{Price.Low}}^{\text{Price}} = 0{,}60$ и *Price.Medium*, $r_{\text{Price.Medium}}^{\text{Price}} = 0{,}30$.

Као што је наведено у Проширењу 1 Примера 2, вредности Q^1_{aqq} прве комбинације сервиса припадају редом инстанцама: *Price.Medium, RespTime.Medium, Availability.High*и *Reliability.Low*. С друге стране, вредности Q^2_{aqq} припадају инстанцама дефинисаним као *Price.Low, RespTime.Low, Availability.High* и*Reliability.Low*.

Сходно Дефиницији 6.1, разлика међу мерама квалитета комбинација инстанци (*Price.Low, RespTime.Low, Availability.High, Reliability.Low*) и (*Price.Medium, RespTime.Medium, Availability.High, Reliability.Low*) се може израчунати као $\frac{(0.16*0.6+0.54*0.4)-(0.16*0.3+0.54*0.5)}{4}=-0.0015$. Значи, комбинација инстанци дефинисаних као средња вредност обележја *Price*(*Price.Medium*), средња вредност обележја*Time* (*RespTime.Medium*), висока вредност обележја*Availability* (*Availability.High*)и ниска вредност обележја*Reliability* (*Reliability.Low*)у већем степену одговара дефинисаним захтевима корисника, тј представљају више преферирану комбинацију инстанци.

С друге стране, на основу резултата Проширења 1 Примера 1, Дефиниције 6.2 и Дефиниције 6.2', разлика међу мерама квалитета дате две комбинације сервиса је: $\frac{(0{,}16*0{,}75+0{,}54*0{,}46)-(0{,}16*0{,}3+0{,}54*0{,}5)}{4}=0{,}0126$, на основу чега се може закључити да прва комбинација сервиса више одговара дефинисаним захтевима иако припада комбинацији мање префериpаних инстанци. ■

На основу датог примера се може закључити да, припадање највише префериpаној комбинацији инстанци у општем случају не мора да води ка комбинацији сервиса са највише префериpаним нефункционалним карактеристикама. Према томе, дефинисане мере, r^{QT} иr^{S}, само у случају истовременог разматрања, у потпуности осликавају захтеве које је корисник дефинисао. Овакав резултат може бити и оправдан уколико се имају у виду све варијабилности које постоје у моделу: *(i)*варијабилност при дизајну и извршавању дефинисан самом SOA фамилијом, *(ii)*варијабилност нефункционалних

карактеристика расположивих сервиса исте функционалности, и*(iii)*варијабилност индукована условно дефинисаним захтевима насталих као последица чињенице да јединствени рангови вредности нису дефинисани (тј. они зависе од задовољења услова у условно дефинисаним захтевима). Додатно, разне врсте неконзистенција могу постојати услед дефинисања приоритета међу паровима обележја и инстаци (што је карактеристика целе фамилије AHP метода) [93] као и нове врсте неконзистенција које су претходно идентификоване због дефиниције условљености у захтевима (карактеристика CS-AHP метода - Дефиниција 8). Као што је у претходним анализама показано за CS-AHP алгоритам (Поглавље 4.3.) и у литератури познато за AHP фамилију [192] [193] [194], дефинисане врсте неконзистенција за последицу имају неконзистенцију индукованих мера.

Према томе, дефинисана двослојна структура, насупрот експресивности која се огледа у дефинисању разних врста захтева, и дефинисаним мерама, посједује недостатке у: *(i)*неодређености (i.e. *неизвесности*) односа међу мерама квалитета разних комбинација сервиса које припадају истим/различитим комбинацијама инстанци нефункционалних карактеристика; и*(ii)*временској сложености израчунавања дефинисаних мера квалитета.

У циљу даље анализе уочених недостатака развијеног модела, идентификују се услови под којима их је могуће елиминисати, или пак редуковати, и врши се анализа утицаја идентификованих услова (тј ограничења) на цели модел. Пример такве врсте захтева је идентификован по аналогији са приступом представљеним у [163] (у којем се сви захтеви корисника представљају као субјективни, насупрот објективним достављеним од стране провајдера сервиса) и монотоним тенденцијама (растућим или опадајућим) нефункционалних карактеристика[185]. Препознати скуп захтева ће бити означен као скуп „објективних информација о нефункционалним карактеристикама“ с обзиром да ови захтеви објективно деле цели интервал могућих вредности на једнаке величине и тако дефинише инстанце обележја. Са друге стране, претпоставка о постојању инстанци једнаке величине за сваку од карактерисика, у већини случајева не одговара ставовима и односу корисника SOA фамилије, ка нефункционалним каракетристикама.

Дефиниција 15 (Објективни захтеви над SOA фамилијом)

За дату SOA фамилију $G_{PM}^{FM} = (\nu, \varepsilon)$, ***скуп објективних захтева*** $\boldsymbol{R_{OB}}$над двослојном структуром нефункционалних захтева се карактерише са:

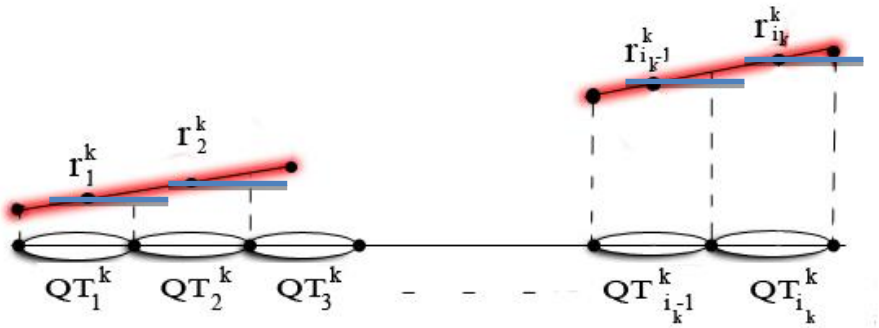
(i) Инстанце сваког обележја су једнаке величине

(ii) Једно-димензиона мера над комбинацијом инстанци r^{QT}представља јединствену монотоно растућу/опадајућу дистрибуцију над средним вредностима сваке од инстанци

(iii) Једно-димензиона мера конфигурација r^{C}је јединствена монотоно растућа/опадајућеа линеарна диференцијабилна функција.

Уколико бар један од дефинисаних услова није задовољен, скуп захтева представља***субјективан скуп захтева*** $\boldsymbol{R_{SB}}$.

Услов *(i)* се односи на могућу енумерацију сваке од посматраних нефункционалних каракетристика, док услов*(ii)*одговара монотоним тенденцијама нефункционалних карактеристика над инстанцама. Исти услов је дефинисан и у *(iii)*за нефункционалне карактеристике комбинација сервиса,док додатни услови линеарности и диференцијабилности обезбеђују униформну дистрибуцију мера квалитета са подједнаким утицајима на диференцирање вредности у оквиру инстанци. Шематска презентација мера r^{QT}иr^{C}које одговарају обективно дефинисаним захтевима је представљена на Слици 25.



Слика 25. Шематкси приказ мера r^{QT} иr^{S}у случају објективно дефинисаног скупа захтева R_{OB}

Оправданост овако дефинисаног скупа захтева је представљен у облику следеће пропозиције о међусобном односу мера над двослојним моделом нефункционалних карактеристика и обективног скупа захтева.

Пропозиција 5. *За дату SOA фамилију* $G_{PM}^{FM} = (\nu, \varepsilon)$ *и двослојни модел нефункционалих карактеристика са CS-AHP алгоритмом (C,* $Q_{agg}^{LB}, Q_{agg}^{UB}$*, QT,* R_{OB}*), важи:*

$$\text{ако је } r^{QT}\left(QT_{i_1}^1 \times \ldots \times QT_{i_n}^n\right) > r^{QT}\left(QT_{j_1}^1 \times \ldots \times QT_{j_n}^n\right) \text{ и}$$

$$\sum_{k \in I} r_k^C \left(r_{\max}^{QT}(QT_{i_k}^k) - r_{\min}^{QT}(QT_{i_k}^k)\right) < \sum_{l \in J} r_l^C \left(r_{\max}^{QT}(QT_{i_l}^l) - r_{\min}^{QT}(QT_{i_l}^l)\right), \quad (***)$$

$$\text{онда је } r^S\left(\left\langle s_1^1, \ldots, s_k^1 \right\rangle\right) > r^S\left(\left\langle s_1^2, \ldots, s_k^2 \right\rangle\right),$$

где су I и J скупови монотоно растућих и опадајућих нефункционалних карактеристика; а агрегиране вредности нефункционалних карактеристика за комбинације сервиса $\left\langle s_1^1, \ldots, s_k^1 \right\rangle$ *и* $\left\langle s_1^2, \ldots, s_k^2 \right\rangle$ *припадају редом покривањима комбинацијама подинтервала* $QT_{i_1}^1 \times \ldots \times QT_{i_n}^n$ *и* $QT_{j_1}^1 \times \ldots \times QT_{j_n}^n$.

Доказ пропозиције је дат у Додатку А.

Додатно ограничење (***) дефинисано у услову теореме, захтева мање могућности у смањењу вредности рангова над инстанцама нефункционих захтева које имају монотоно растућу тенденцију, у односу на повећање вредности рангова над инстанцама нефункционалних карактеристика са монотно опадајућом тенденцијом. Општији закључак се може формулисати у облику следеће последице.

Последица. *За дату SOA фамилију* $G_{PM}^{FM} = (\nu, \varepsilon)$ *и двослојни модел нефункционалних карактеристика са CS-AHP алгроитмом (C,* $Q_{agg}^{LB}, Q_{agg}^{UB}$*, QT,* R_{OB}*), комбинација сервиса са највећом мером квалитета која истовремено припада комбинацији инстанци са највише префериранim карактеристикама уз*

задовољење услова (***), представља комбинацију сервиса која је највише преферирана за целу SOA фамилију.

Значи, у случају идентификованог скупа објективних захтева са додатним захтевом (***)о двослојној структури нефункционалних карактеристика, проблем одређивања оптималне конфигурације је редуциран на простор највише фериране комбинације инстанци. То значи да, у случају објективног скупа захтева и услова (***), мера $r^{C}()$ садржи све информације које се односе на дефинисане захтеве корисника. Међутим, дефинисани скуп објективних захтева, уколико се посматра изван двослојне структуре, у потпуности одговара захтевима о монотоно растућим/опадајућим нефункционалним карактеристикама опште коришћеним у литератури [137] [195][163] за које је очигледан недостатак експресивности и дефинисања могућности прављења компромиса од стране крисника, што се сматра од посебног значаја за развој дефинисаног модела [180].

Резултати предстаљених анализа непротивречно указују да креирање модела који је подређен кориснику и његовим захтевима, захтева компромис између лошијих перформанси и експресивности представљања корисничких захтева и очекивања.

4.4.3. Формална дефиниција задатка оптимизације

Резултати представљени у претходној секцији јасно показују да је за квантификацију корисничких захтева неопходно посматрати обе конструисане мере, јер мера $r^{QT}()$ осликава преференце над инстанцама могућих вредности нефункционалних захтева, док се због високог степена неодређености у фамилији, мера $r^{C}()$ додатно дефинише над конкретним вредностима из интервала који представљају инстанце. Стога ће се и задатак оптимизације дефинисати као двоструки задатак максимизације обе дефинисане мере. Пре његовог формалног увођења, дефинишу се ограничења конфигурације, која се односе на ограничења индукована варијабилностима у оба модела и бијективном везом међу њима.

Комбинација сервиса је ***валидна*** уколико су њоме задовољена сва ограничења SOA фамилије индукована ограничењима оба модела, тј бијективном везом међу потпроцесима и атомичним активностима (у SOA архитектури) и додатним ограничењима опционалности и интегритета (у моделу карактерстика фамилије). Комбинација која није валидна у односу на дефинисана ограничења не може бити даље разматрана у процесу конфигурације.

Додатно, корисник може ограничити вредности појединих нефункционалних карактеристика над целим процесом или појединим потпроцесима. Оваква врста ограничења представља ***строга ограничења*** (енгл. *hard constraints*) чије нарушавање одмах елиминише дату комбинацију сервиса, без обзира на испуњење осталих захтева о нефункционалним карактеристикама. Нпр, корисник може дефинисати да укупно време одзива целог процеса плаћања мора бити мање од 20 секунди.

На крају, задатак оптималне конфигурације SOA фамилије се дефинише као одређивање валидне комбинације сервиса која истовремено у односу на обе мере у највећем степену задовољава дефинисане захтеве корисника о нефункционалним карактеристикама; што се формално дефинише на следећи начин:

Дефиниција 16 (Оптимална конфигурација SOA фамилије).
За дату SOA фамилију и конструисану двослојну структуру за представљање нефункционаних карактеристика *(C,* $Q_{agg}^{LB}, Q_{agg}^{UB}$*, QT, R),*као и дати скуп строгих ограничења и скуп расположивих сервиса за сваку од акивности у фамилији, ***задатак оптималне конфигурације SOA фамилије*** се дефинише као конструкција валидне комбинације сервиса која максимизује укупну меру квалитета комбинације сервиса $r^S\left(\langle s_1,\ldots,s_k\rangle\right)$, под условом припадања дате комбинације сервиса комбинацији највише префериранх инстанци нефункционалних карактеристика и додатно задовољење дефинисаних строгих ограничења.

Важно је приметити да, у поређењу са стандардним задатком оптимизације који се најчешће користе у литератури [137, 195, 163], а дефинише се као

максимизација свеукупне мере квалитета, дата дефиниција оптималне конфигурације SOA фамилије захтева додатни услов припадања највише преферираним инстанцама нефункционалних карактеристика.

На основу Пропозиције 5 и њене Последице, задатак оптимизације дефинисан Дефиницијом 11 је еквивалентан стандардном задатку оптимизације у случају идентификованог скупа објективних захтева у моделу. С друге стране, Проширење 2 Примера 2 показује да општем случају постоји ненулто растојање међу решењима дефинисаних оптималних задатака, чиме се добија ненулто растојање међу дефинисаним задацима конфигурације.

Неопходно је уочити да је Дефиницијом 11 задатак оптималне конфигурације SOA фамилије дефинисан као двоструки задатак оптимизације, тј максимизација се врши по обе дефинисане мере, чиме је проблем постао знатно комплекснији са становишта примене техника за решавање оптималних задатака. Додатно, као што је и раније наведено, уколико се жели обезбедити приступ оријентисан кориснику и његовим захтевима, неоподно је обезбедити приступ који ће брзом конвергенцијом савладати недостатке предложеног модела (идентификованих као неодређеност односа међу мерама и временска сложеност израчунавања).

4.4.4. Интегрално решење

У овој секцији се представља интегрално решење за конфигурацију SOA фамилија, под називом *OptConfSAOF*, које на основу нефункционалних карактеристика и захтева представљених над развијеном двослојном структуром обележја и инстанци, применом метахеурустичког приступа заснованог на генетичком алгоритму, решава дефинисани задатак оптимизације. При развоју интегралног решења је, сходно дискусији из претходне секције, било неопходно обезбедити: *(i)* оптимално решење двоструког задатка оптимизације, и *(ii)* превазићи идентификоване недостатке модела, тј развити метод који неће бити директно угрожен идентификованим недостацима модела.

Стога, у наставку се прво представља *ConfSAOF* решење које оптимално решава дати проблем употребом генетичког алгоритма (који ће омогућити

трансформацију дефинисаног задатка у једноструки оптимални задатак); док се оптимизацијом тако развијеног приступа (увођењем динамички креираних елемената модела) добија крајње оптимално решење проблема, *OptConfSAOF*.

4.4.4.1.Метахеуристички приступ за оптималну кофнигурацију SOA фамилија

Као што је већ представљено у поглављу 2.3. генетички алгоритми су адаптивни алгоритми претраге који симулирају основна начела природне селекције и генетике [196]. У циљу примене алгоритма овог типа за решавање конкретног проблема из посматраног домена, потребно је адаптирати његове основне елементе: хромозоме, почетну популацију, функцију циља (енгл. *fitness function*), операторе мутације (енгл. *mutationoperator*) и рекомбинације(енгл. *crossover operator*). У свакој генерацији се креирају нови елементи популације и за даљу репродукцију се бирају само они који у већој мери одговарају дефинисаном циљу. За примену оваквог приступа за решавање проблема оптималне SOA конфигурације, осим адаптације дефинисаних елемената, неопходно је и обезбедити да сваки елемент популације буде валидан на основу дефиниције из Поглавља 4.4.2 тј да задовољава ограничења и правила дефинисана моделима из SOA фамилије.

Кодирање хромозома за представљање комбинације сервиса. Потенцијално решење (тј. једна комбинација сервиса) се у облику хромозома представља помоћу низа кодних вредности $\langle e_1, e_2, ..., e_k \rangle$. Могуће вредности за *i*-ти елемент низа e_i чини скуп расположивих сервиса S_{a_i}. У случају да је *i*-та активност или нека од њених родитељских активности (посматрано у стуктури стабла модела)опциона, скуп могућих вредности *i*-тог елемента (e_i) додатно садржи нулти елемент (што представља да елемент није укључен у инстанци SOA фамилије, па самим тим није ни конфигурисан).

Почетна популација и *serviceTransform*алгоритам.За почетак процеса претраге за генетички алгоритам је неопходно дефинисати иницијалну популацију. Устаљени приступ је у њеном насумичном генерисању, како процес не би био диктиран (у позитивном или негативном смислу), па почетна популација представља насумице креиране комбинације сервиса, које последично, могу бити и невалиде. У циљу решења проблема генерисања невалидних елемената популације, користи се*servicesTransform*алгоритам [180]за трансформисање насумице креираног скупа сервиса у валидан скуп. На основу улазне комбинације сервиса $\langle e_1, e_2, ..., e_k \rangle$ алгоритам генерише валидан скуп сервиса, а главни кораци трансформације су следећи:

Корак 1. На основу селектованог скупа сервиса, одређује се одговарајући скуп карактеристика са нивоа листова F_L. Почевши од скупа F_L, скуп карактеристика који може бити досегнут на путу ка корену модела карактеристика је селектован и означен као F_1.

Корак 2. Почевши од скупа F_1у унији са кореним елементом (уколико није претходно досегнут и самим тиме укључен), валидан скуп карактеристика F_2за цели модел је одређен обиласком до нивоа листова.

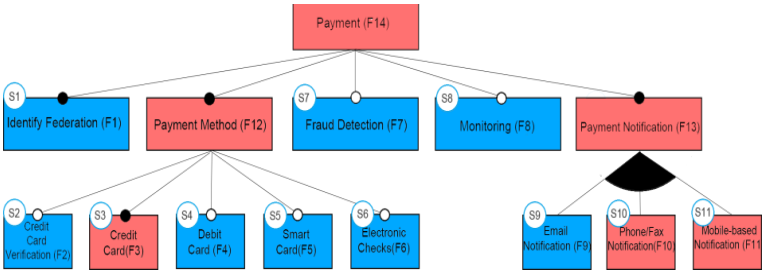
Корак 3. На основу скупа F_2, валидан скуп сервиса *Sconf*је селектован и он представља валидну конфигурацију SOA фамилије.

Како задовољење модела карактеристика са ограничењима интегритета може бити посматрано као комбинаторни проблем задовољења (енгл. *satisfiabilityproblem*), није увек могуће на ефикасан начин одредити валидан скуп карактеристика [180]. Стога, у циљу обезбеђења ефикасности, *serviceTransform*алгоритам ограничава број покушаја којим се врши корекција полазног скупа карактеристика на нивоу листова, неком унапред задатом фиксном вредношћу.

a) Initial chromosome:

$(s_{1(1)}, s_{2(1)}, 0, s_{4(1)}, s_{5(1)}, 0, 0, 0, s_{9(1)}, 0, 0)$

b)



serviceTransform.STEP1		serviceTransform.STEP2		serviceTransform.STEP3	
$F_L = \{A_1, ..., A_8\}$ $F1 = F_L$		$F_1 = \{A_1, ..., A_4, A_6, A_8, A_{11}, A_9, A_{12}, A_{13}\}$ $F_2 = F_1 \cup \{root\}$		$F_1 = \{A_1, ..., A_4, A_6, A_8, A_{11}, A_9, A_{12}, A_{13}, A_{14}, A_{15}, A_{10}\}$ $S_{conf} = \oslash$	
s1.3a	$F_1 = F_1 \cup \{A_{11}\}$	s2.1	$F_2 = F_2 \cup \{A_{14}\}$	s3.1 A_1	$S_{conf} = S_{conf} \cup \{s_{1(1)}\}$
s1.3a	$F_1 = F_1 \cup \{A_9\}$	s2.1	$F_2 = F_2 \cup \{A_{15}\}$	s3.1 A_2	$S_{conf} = S_{conf} \cup \{s_{2(1)}\}$
s1.2	$F_1 = F_1 \cup \{A_{12}\}$	s2.1	$F_2 = F_2 \cup \{A_{10}\}$	s3.1 A_3	$S_{conf} = S_{conf} \cup \{s_{3(1)}\}$
s1.2	$F_1 = F_1 \setminus \{A_5\}$			s3.1 A_4	$S_{conf} = S_{conf} \cup \{s_{5(1)}\}$
s1.2	$F_1 = F_1 \setminus \{A_7\}$			s3.1 A_6	$S_{conf} = S_{conf} \cup \{s_{7(1)}\}$
s1.3a	$F_1 = F_1 \cup \{A13\}$			s3.2 A_9	$S_{conf} = S_{conf} \cup \{s_{4(2)}\}$
				s3.2 A_{10}	$S_{conf} = S_{conf} \cup \{s_{12(2)}\}$

Слика 26. а) Невалидни хромозом и одговарајући дио FM-аза активност плаћања (плава и црвена боја одговарају каракетристикама за које ограничења јесу тј нису задовољена датим хромозомом) и б) корак*serviceTransform*алгоритма [180]

Алгоритам *serviceTransform*гарантује да коректност комплетног приступа није нарушена, тј. он чува коректност сваке инстанце SOA архитектуре која може бити изведена из целе фамилије.

У циљу илустрације описаног процеса трансформације, посматрајмо почетни невалидни хромозом представљен на Слици 26а који одговара делу SOA фамилије предстаљеном у десном делу исте слике. Дати хромозом представља низ сервиса *($s_{1(1)}$; $s_{2(1)}$; $s_{3(1)}$; 0; $s_{5(1)}$; $s_{6(1)}$; $s_{7(1)}$; $s_{8(1)}$; $s_{9(1)}$; 0; :::; 0)*селектован редом за активности *IdentifyFederation* (A_1), *GatewayInterface* (A_2),*CreditCardPayment* (A_3), *DebitCardPayment* (A_4), *Email-Voicemail* (A_5), *PhoneFax* (A_6), *Mobile-basedNotification*(A_7), и*Shipment* (A_8). Кораци индуковани *serviceTransform* алгоритмом за креирање валидне конфигурације су представљени на Слици 26b. На крају, алгортимом се добија следећа конфигурација *Sconf* =<$s_{1(1)}$; $s_{2(1)}$; $s_{3(1)}$; $s_{5(1)}$; $s_{7(1)}$; $s_{4(2)}$; $s_{12(2)}$>, која у односу на иницијално селектоване сервисе за активности *IdentifyFederation*(A_1), *GatewayInterface* (A_2), *CreditCardPayment* (A_3), *DebitCardPayment* (A_4), и *PhoneFax* (A_6) садржи и насумично одабране сервисе из скупа расположивих сервиса за активности *CreditCardValidation* (A_9) и*SMS* (A_{10}).

Функција циља. Како би се извршиле евалуације могућих решења и одабрало најбоље, дефинише се функција циља која осликава перформансе сваке индивидуе [135]. Комбинација сервиса са најбољим перформансама се добија оптимизацијом тако дефинисане функције. У литератури, многа истраживања указују на начине дефинисања функције циља за разне класе проблема [197, 198]и у већини се препоручује дефинисање пенала за све индивидуе које не задовољавају дефинисана ограничења [135]. Главна предност оваквог приступа је што омогућава да разне врсте ограничења буду интегрисане у саму функцију уз одговарајуће пенале за њихову меру, чиме се од самог генетичког алгоритма очекује да обезбеди испуњење ограничења уколико је то могуће.

За решавање дефинисаног двоструког задатка оптимизације, казнени пенали се дефинишу на основу једног од два критеријума оптимизације, чиме се задатак своди на решавање једноструког оптималног задатка (у односу на преостали критеријум). Стога, казнени пенали су дефинисани на следећи начин:

1) Главни циљ оптимизације је дефинисан као проналажење комбинације сервиса коју карактерише највише преферирана комбинација инстанци над нефункционалним карактеристикама. Како би се овакав захтев уврстио у функцију циља, дефинише се казнени фактор (пенал) као релативно растојање у односу на комбинацију инстанци над нефункционалним карактеристикама која је највише преферирана. То значи да, уколико је посматрана комбинација сервиса окарактерисана као највише преферирана комбинација инстанци, тада нема пенала; казнени фактор постаје све већи како комбинација инстанци све мање одговара захтевима корисника, тј све више је удаљена од највише префериране комбинације. Да би растојање било независно од самих вредности, користи се релативна вредност у односу између мере квалитета највише префериране комбинације инстанци $r^{[\,]}_{c_1,\ldots,c_n}$ и мере квалитета комбинације инстанци која се достиже посматраном комбинацијом сервиса $r^{[\,]}_{c_1,\ldots,c_n}\left(e_1,\ldots,e_{|A|}\right)$.

2) Додатни казнени фактор се дефинише за стуктурна ограничења, односно ограничења вредности појединих нефункционалних карактеристика над целим процесом или појединим потпроцесима, претходно дефинисаним као строга ограничења. Њихово нарушавање одмах елиминише дату комбинацију сервиса из

даљих разматрања. Растојање у односу на овако дефинисано ограничење се дефнише као: $D(e_1,...,e_{|A|}) = \sum_{i=1}^{l} cl_i(e_1,...,e_{|A|}) \cdot y_i$, где је $y_i = \begin{cases} 0, & cl_i(e_1,\ldots,e_{|A|}) \le u_i \\ 1, & cl_i(e_1,\ldots,e_{|A|}) > u_i \end{cases}$.

У случају да се вредност пенала дефинише као исувише мала, постоји опасност да индивидуа која нарушава ограничења неће бити елиминисана [135]. Зато, како би се генерисала прихватљива комбинација сервиса (у односу на захтеве и ограничења модела), неопходно је за строга ограничења одредити веће казнене факторе него за захтеве корисника. У литератури [199] је опште прихваћен приступ динамичког генерисања казнених фактора у односу на број генерација у популацији, чиме се повећањем вредности казненог фактора за нове генерације обезбеђује и већи степен заводољења ограничења на који се односи.

На крају, како би се сви казнени фактори могли интегрисати у функцију циља и при томе задовољити захтев њене ненегативности, задатак максимизације мере квалитета комбинације сервиса $R(v(e_1,\ldots,e_{|A|}))$ се трансформише у задатак минимизације реципрочне вредности $\frac{1}{R(v(e_1,\ldots,e_{|A|}))}$, и фукција циља се дефинише са:

$$Fitness(e_1,..., e_{|A|}) = \frac{1}{R(v(e_1,..., e_{|A|}))} + w_1(gen)\frac{r^{[\,]}_{c_1,...,c_n} - r^{[\,]}_{c_1,...,c_n}(e_1,..., e_{|A|})}{r^{[\,]}_{c_1,...,c_n}} + w_2(gen)D(e_1,..., e_{|A|}) \quad \textbf{(15)}$$

Разлози за промену природе задатка који се решава (од задатка максимизације на реципрочни задатак минимизације) су оправдани чињеницом да су вредности мере квалитета сервиса из интервала [0,1] и да би она требала бити умањена вредностима пенала. Како мера квалитета комбинације сервиса може бити изузетно мала (вредности близу 0), вредности пенала би биле још мање, што је у контрадикцији са захтевом ненегативности и препорукама за дефинисање вредности пенала и обезбеђење конвергенције целог процеса.

На крају је неопходно дефинисати и критеријуме заустављања. Један од могућих начина је дефинисањем фиксног броја итерација, док с друге стране, критеријуми могу дефинисани и на следећи начин:

1) Итерације се настављају све док се не испуне обе врсте захтева, тј. док се не испуне дефинисани строги захтеви $D(e_1,...,e_{|A|})$ и док комбинација сервиса $(e_1,...,e_{|A|})$ не буде одговарала највише префериранim комбинацијама инстанци нефункционалних карактеристика. У колико се они не испуне за унапред дефинисани фиксни број пролаза, онда се поступа на следећи начин:

   а) уколико строги захтеви нису испуњени, решење није нађено;

   б) уколико строги захтеви јесу испуњени, али не и захтеви корисника о нефункциоанлним карактеристикама, тада се комбинација сервиса којој одговара најмања вредност функције циља узима као генерисано приближно решење.

2) Када се први пут генерише комбинација која задовољава строге захтеве и чије вредности нефункционалних карактеристика одговарају највише префериранoj комбинацији инстанци, процес се наставља фиксан и унапред дефинисан број пута како би се покушала генерисати комбинација сервиса која и даље задовољава строге захтеве и припада истој комбинацији инстанци, али има мању вредност функције циља.

Може се уочити да услов 1) обезбеђује задовољавање строгих захтева и, уколико је могуће, захтеве корисника о префериранim нефункционалним карактеристикама. Услов 2) омогућава да се одреди највише преферирана комбинација сервиса која одговара највише префериранoj комбинацији инстанци над функционалним карактеристикама.

На крају, као саставни дио процеса генетичке евалуације се дефинишу и **оператори мутације и рекомбинације** којима се одређују начини креирања нових елемената популације. Како је начин представљања хромозома небинаран, користи се *n*-тачкасти оператор рекомбинације. Овај оператор насумице одабере *n* тачака у хромозому којима подели дати хромозом на мање делове и њиховом рекомбнацијом генерише нови хромозом. С друге стране, користи се оператор мутације који насумице одабере једну тачку у хромозому и замени њену вредност са новом насумице одабраном из скупа могућих. Оператор мутације се примењује након оператора рекомбинације и као разултат примене оба оператора може се добити невалидна комбинација сервиса. Стога се, у циљу добијање валидне комбинације, примењује раније дефинисани *serviceTransform*алгоритам.

Algorithm. *ConfSOAF*

Input: FM, SOAF, set of available services S_{a_k}, set of requirements $\Re_{EN}$ about enumerations of the range values of QoS dimensions, set of requirements $\Re$ about preferable characteristics

Output. Configured SOA

Begin

$[q_{c_i}^{LB}, \quad q_{c_i}^{UB}] \leftarrow$ estimate ranges of QoS (FM, SOAF, S_{a_k}); //1

$Q_{agg}^{LB}, Q_{agg}^{UB}$ - estimate ranges of QoS for the SOAF; //2

$r_{c_1,...,c_n}^{[\ ]}$ - determine the most preferable combination of subintervals//3

Genetic algorithm (FM, SOAF, S_{a_k});

Begin
initialize population; //4
evaluate population with calculation of QoS measurements; //5
while *TerminationCriteriaNotSatisfied*
do
select parents for reproduction; //6
perform crossover and mutation; //7
evaluate population with calculation of QoS measurements; //8
end while;
end

End.

Слика 27. Кораци интергалног решења *ConfSAOF*

На крају, целокупно решење задатка оптималне конфигурације SOA фамилије, под називом *ConfSAOF*обухвата прво конструкцију двослојне структуре за представљање нефункционалних каректеристика целе фамилије и каснију примену представљеног метахеуристичког приступа за решавање оптималног задатка. Сумирани кораци у тако добијеном интегралном приступу су представљени на Слици 27.

4.4.4.2.Анализа решења

С обзиром да решење које се предлаже интегрише претходно развијени алгоритам за рангирање захтева и метахеуристички приступ за решавање оптималног задатка, при чему је за сваки понаособ идентификован скуп предности и недостатака, добијено интегрално решење је неопходно додатно

анализирати и утврдити интеграцијом индуковане карактеристике. У овој секцији се представља анализа ефикасности и временске сложености целог приступа.

4.4.4.2.1. Анализа ефикасности решења

Како је предложено решење засновано на употреби генетичког алгоритма, неопходно је анализирати да ли се на тај начин обезбеђује довољно ефективно и ефикасно решење проблема оптималне конфигурације SOA фамилије. Као критеријум ефикасности се узима мера *растојања од оптималног решења*, што је уобичајен приступ при коришћењу техника за приближно решавање оптималних задатака [137].

Стога, следећа анализа се врши у циљу процене очекиване вредности удаљености решења добијеног предложеним приступом у односу на оптимално решење. Процена се заснива на вредностима које се добијају довољно великим бројем независних генерисања SOA фамилија. За сваку генерисану SOA фамилију, одређује се тачност решења добијеног применом *ConfSOA* приступа (његовим поређењем са оптималним решењем) а просечна вредност тако добијених вредности се узима за процену ефикасности предложеног приступа, као што се наводи у наставку.

Експерименталне поставке. Улазни параметри *ConfSOA*решења су редом: 1) модел карактеристика, 2) сервисно-оријентисани дијаграм тока, 3) скуп расположивих сервиса за сваку од активности у SOA фамилији, и 4) скуп захтева о префериранim карактеристикама и скуп строгих ограничења. За независне варијабле које карактеришу дати модел узимају се редом: 1а) број карактеристика у FM, 1б) процентуална заступљеност опционих елемената у моделу, 2а) број активности у дијаграму тока, 2б) процентуални удио стандардних патерна композиције у дијаграму тока, 3а) број расположивих сервиса за сваку од активности, 3б) вредности нефункционалних карактеристика за сваки од расположивих сервиса, 4а) захтеви корисника о нефункционалним

карактеристикама система као и строга ограничења о њиховим вредностима над појединим деловима или целој SOA архитектури (уколико се желе дефинисати).
Као што је у уводном поглављу наведено, SOA архитектуре које се разматрају подразумевају ограничења над топологијом модела пословних процеса [12] као и бијективно мапирање између FM и дијаграма тока фамилије. Наведена ограничења за последицу имају бијективно мапирање међу процентуалним заступљеностима ограничењима интегритета у FM и композиционих патерна у дијаграму тока фамилије као и једнакост вредности варијабли 1a и 2a.

На основу препознатих дескриптивиних карактеристика модела, могуће је генерисати разне фамилије модела. У том циљу, следећи генератори су развијени: 1) генератор FMкоји као улазне параметре прима број карактеристика и процентуалну заступљеност опционих елемената и ограничења интенгритета у моделу; 2) генератор SOA фамилије који као улазне параметре прима параметре FM, дистрибуцију одговарајућих патерна у дијаграму тока фамилије и захтеве корисника о структури обележја и инстанци у моделу. У литератури је већ имплементиран FMгенератор [135], генератор BPMF модела је претходно развијен у [180] на основу којих је *ConfSOA* приступ имплементиран као Eclipse plugin. Такође, да би се одредило оптимално решење постављеног задатка конфигурације, имплементиран је *brute force*алгоритам чији ће резултати бити упоређени са резултатима развијеног *ConfSOA*.

У претходним истраживањима о моделима карактеристика [57] је показано да укупан број карактеристика у моделу припада интервалу [14, 287], при чему је просечна вредност (М) = 76.86 и стандардна девијација (СД) = 96.25. Стога, генерисање модела FM и SOA фамилије се односи управо на дату просечну величину која је добијена емпиријским путем. У претходним истраживањима је такође наведено да величина двослојне струкутуре се састоји од највише 7 инстанци, вредности која је разумљива и прихватљива за људски ум [112]. Максималан број расположивих сервиса за сваку активност је постављен на 100 а вредности њихових нефункционалних карактеристика су насумично генерисане.

Резултати екперимента. За процену просечног растојања међу решењима, сваки од дефинисаних улазних параметара је насумице генерисан. За сваки модел и

дефинисани задатак оптималне конфигурације, решења су генерисана применом *brute force*алгоритма и *ConfSOA*приступа, а растојање међу њима је одређено као релативно растојање међу њиховим мерама квалитета. Експеримент је поновљен 1000 пута и просечна вредност релативног растојања међу решењима је 10.23% (СД=9.87%).

Добијени емпиријски резултати показују да је оптималност предложеног решења приближно 90%, тј. резултат добијен применом предложеног приступа је за приближно 10% мањи у односу на оптимални.

4.4.4.2.2. Анализа временске сложености

Како је симулационо показано да предложено решење има висок степен ефикасности, а реч је о комбинаторном проблему опималног испуњења дефинисаног двоструког задатка оптимизације, неопходно је анализирати временску сложеност целог модела.

Прво, уведимо следеће ознаке: *n*-број нефункционалних карактеристика у моделу, *m*- број апстрактних сервиса (тј. активности), *s*-највећи број расположивих сервиса за сваку од активности, *q*-највећи број покривајућих подинтервала дефинисан у моделу, *r*-број захтева о префериранim вредностима над двослојном структуром.

Сложеност *ConfSOA*алгоритма се може разложити на следеће елементе:
Корак //1 захтева $O(nms)$операција за процену интервала вредности сваке од нефункционалних карактеристика у SOA архитектури. Пропагација интервала (корак //2) нефункционалних карактерстика кошта *O(n*logm)* , сходно [12]. Корак //3 захтева генерисање свих могућих комбинација инстанци нефункционалних карактеристика и одређивање њихове мере квалитета, што захтева $O(m^n \cdot T(CS-AHP))$ операција, где је *T(CS-AHP)* време потребно за израчунавање мере квалитета комбинције инстанци/сервиса. На основу коментара уз Пропозицију 4, *T(CS-AHP)*је процењено на $O\left(r + n^2/2 + n \cdot q^2/2\right)$операција.

Сложеност операција генетичког алгоритма (који обухвата кораке //4-//8 од интегралног решења *ConfSOA*) се декомпонује на следећи начин. Корак //4 захтева $O(\|P\| \cdot m \cdot T(serviceTransform))$ операција, где је*T(serviceTransform)*време потребно за*serviceTransform*алгоритам. У раду [135] је оно процењено на $O(cmn * \log^2 n)$, где је*c*највећи број ограничења у моделу карактеристика. Рачунање дефинисаних мера за целу популацију (корак //5) захтева $O(\|P\| \cdot T(CS - AHP))$операција, сходно [112].

Следећи кораци се понављају $\|G\|$ пута:

- корак //6 -одабир родитеља за репродукцију захтева *O(1)*операција
- корак //7 - оператор рекомбинације захтева *O(m)* операција, док оператор мутације захтева*O(1)*
- корак //8 -оператор замене кошта $O(\|P\|)$; док се валидност сваког елемента популације проверава помоћу *serviceTransform*алгоритма, што захтева *T(serviceTransform)*. Над сваким елементом популације је неопходно израчунати вредност мера квалитета, чија је сложеност $T(CS - AHP)$

Значи, итеративни кораци генетичког алгоритма захтевају $O_{GA} = O\left(\|G\|\left(r + n^2/2 + n \cdot q^2/2 + m + \|P\| + cmn * \log^2 n\right)\right)$опеарција,што представља полиномијалну сложеност. Међутим, укупна сложеност *ConfSOA* алгоритма је $O\left(nms + n * \log m + \left(\|P\| + m^n\right) \cdot \left(r + n^2/2 + n \cdot q^2/2\right) + O_{GA}\right)$што је експоненцијална сложеност због фактора m^n који дефинише величину свих комбинација модела двослојне структуре обележја и инсатнци.

Иако су претходно при симулационим анализама дефинисана ограничења која се односе на број могућих инстанци (као вредности које су прихватљиве за људски ум), број обележја није био ограничен. То је последица чињенице да корисник може дефинисати нефункциоанлне каракетристике специфичне за дати домен примене, који може бити јако велики (корисник може дефинисати на десетине нефункционалних карактеристика). Тиме би сложеност креирања самог двослојног модела била реда $c \cdot 10^k, k > 7$, где је број нефункционалних

карактеристика $m = 10 \cdot \sqrt[7]{c}$, c=конст. Значи, како је величина популације код примене генетичког алгоритма ограничена на вредност $\|G\|$, а мере квалитета за поједине елементе популације захтевају полиномијалну времеску сложеност, фактор који утиче на експоненцијалну сложеност се односи на одређивање највише префериране комбинације инстанци нефункционалних карактеристика (што је претходно дефинисано као један од два критеријума двоструког задатка оптимизације).

Значи, уколико се претпостави познатим решење датог критеријума оптимизације, представљеном адаптацијом генетичког алгоритма се добија довољно ефикасно решење за проблем оптималне конфигурације SOA фамилије. Добијени резултат је од посебног значаја уколико се посматрају домени примене у којима нема захтева за скалабилношћу (у односу на број дефинисаних нефункционалних карактеристика). Нпр, у областима пословних процеса, уколико се изузме дефинисање доменских карактеристика, сходно [17], све нефункционалне карактеристике се могу категорисати у четири групе, па је представљено решење довољно ефикасно (фактор m^n је највише реда 4^7).

Ипак, у циљу обезбеђења општег, доменски независног решења, неопходно је обезбедити оптимизацију процеса примене CS-AHP алгоритма у случајевима када они имају експоненцијалну сложеност. У теорији не постоји општи начин оптимизације [112], већ се он дефинише у зависности од проблема који се решава. Како се решење употребом генетичког алгоритма показало довољно ефикасним, за истраживање се намеће анализа могућности његове примене за истовремено решавање проблема сложености примене CS-AHP алгоритма над конструисаном двослојном структуром за представљање нефункционалних карактеристика.

4.4.4.2.3. Оптимизација интегралног решења

При конструкцији решења *ConfSOA* извршена је трансформација двоструког задатка оптимизације у једноструки који подразумева оптимално решавање једног од критеријума који има мању сложеност (с обзиром да је број нефункционалних карактеристика знатно мањи од укупног броја расположивих сервиса). Међутим, како је цели приступ заснован на употреби генетичког алгоритма, за који је устаљена пракса динамичког генерисања његових елемената [198] [196] [199], намеће се управо могућност динамичког решавања оптималног задатка по једном критеријуму (уз истовремено решавање преосталог задатка оптимиазције).

Algorithm. *OptConfSOA*

Input: FM, SOAF, set of available services S_{a_k}, set of requirements $\Re_{EN}$ about enumerations of the range values of QoS dimensions, set of requirements $\Re$ about preferable characteristics

Output. Configured SOA

Begin

$[q_{c_i}^{LB}, \quad q_{c_i}^{UB}] \leftarrow$ estimate ranges of QoS (FM, SOAF, S_{a_k}); //1'

$Q_{agg}^{LB}, Q_{agg}^{UB}$ - estimate ranges of QoS for the SOAF; //2'

Genetic algorithm (FM, SOAF, S_{a_k});

Begin

initialize population; //3'

evaluate population with calculation of QoS measurements; //4'

while *TerminationCriteriaNotSatisfied*

do

select parents for reproduction; //5'

perform crossover and mutation; //6'

evaluate population with calculation of QoS measurements; //7'

end while;

end

End.

Слика 28. Кораци интергалног решења *OptConfSAOF*

У решењу *ConfSOA,* први казнени пенал који се односио на припадање дате комбинације сервиса највише преферираној комбинацији инстанци, је имао динамичку природу због динамички (у односу на број генерације) креираног мултипликативног фактора $w_1(gen)$. Међутим, исти фактор може постати динамички уколико се, уместо растојања у односу на највише преферирану

комбинацију инстанци посматра растојање у односу на највише преферирану комбинацију инстанци али досегнуту претходним генерацијама популације, означено са $r^{QT}_{MAX-gen}()$. Казнени пенал се на тај начин ажурира након сваке ново креиране генерације у популацији, и то на основу постојећих елемената у популацији, и ова врста динамичког пенала је у литератури позната под називом *адаптивни пенали*[200]. Увођењем дефинисаног пенала се постиже да при свакој новој генерацији, вредности пенала постају мање уколико елементи припадају више преферираним комбинацијама инстанци, чиме се обезбеђује решавање задатка оптимизације по датом критеријуму.

Према томе, наведеном корекцијом казненог пенала, функција циља **(15)** се трансформише у:

$$Fitness(e_1, e_2, ..., e_k) = \frac{1}{r^S(\langle e_1, ..., e_k \rangle)} + \frac{\left| r^{QT}_{MAX-gen}() - r^{QT}\left(QT^1_{i_1} \times ... \times QT^n_{i_n}\right) \right|}{r^{QT}_{MAX-gen}()} + w(gen)D(\langle e_1, ..., e_k \rangle)$$

Како вредност највише префериране комбинације инстанци није позната пре употребе генетичког алгоритма, неопходне су и измене у претходно дефинисаним критеријумима заустављања, тако да се у овом случају дефинишу на следећи начин: када се задовоље строга ограничења у моделу (тј $D(g) = 0$), процес се наставља фиксан број итерација у циљу одређивања ниже вредности функције циља (уколико је могуће). У супротном, уколико строга ограничења нису задовољена, итерације се настављају док вредност функције циља постане непроменљива након одређеног броја итерација.

Сви остали елементи генетичког алгоритма (генерисање почетне популације, оператори мутације и рекомбинације) су непромењени у односу на претходно решење, тако да је ново добијени алгоритам, под називом *OptConfSOA* представљен на Слици 28.

4.4.4.2.5. **Анализа ефикасности оптимизованог решења**

Конструкција *OptConfSOA* решења има јасне бенефите у домену временске сложености комплетног приступа, с обзиром да кораци //1'-//6' имају исту времеску сложеност као претходни //1,//2,//4-//7. Временска сложеност корака //7' сада додатно (у односу на сложеност корака //8) обухвата одабир највише префериране комбинације инстанци из скупа досегнутих, која се реализује једноставним поређењем претходно највише префериране (досегнуте са *gen-1*елемената популације) и нове комбинације, што је сложеност *O(1).* Према томе, укупна временска сложеност *OptConfSOA* решења је $O\left(nms + n * \log m + \|P\| \cdot \left(r + n^2/2 + n \cdot q^2/2 \right) + O_{GA} \right)$, што је полиномијална сложеност.

Важно је напоменути да се, представљеним приступом, применом генетичког алгоритма врши истовремено решавање двоструког опималног задатка, које, за разлику од претходног, као резултат даје само приближно највише преферирану комбинацију инстанци (а не експлицитно одређену). Међутим, овакав приступ, осим оптималности извршавања има још једну потврду која се огледа у чињеници да инстанце нефункционалних карактеристика представљају само интервале могућих вредности који могу бити досегнути неком комбинацијом расположивих сервиса, али у општем случају не морају. Теоријски, коментар постаје заснован уколико се има у виду да је укупан број комбинација сервиса коначан, а могућих вредности из интервала (тј инстанци) је бесконачно непребројиво много, па је и реч о скупу веће димензије и не може се успоставити бијективно пресликавање (којим би се за сваку од могућих комбинација инстанци гарантовала егистенција комбинације сервиса која достиже вредности из ње, самим тиме и највише преферирану).

На крају, уколико се има у виду проблем неодређености односа међу дефинисаним мерама (дефинисаних у Секцији 4.4.2.1.) и чињенице да се применом генетичког решења обезбеђује само приближно решавање задатка оптимизације, неопходно је испитати да ли је предложеном модификацијом угрожена конвергенција целог приступа. Такође, интересантно је утврдити да ли

уведена динамичност решавања оба критеријума (двоструког) задатка има другачије карактеристике конвергенције, тј да ли постоје разлике у решењима генерисаним за задатке оптималне конфигурације истих SOA фамилија. У том циљу, дефинише се следећа хипотеза:

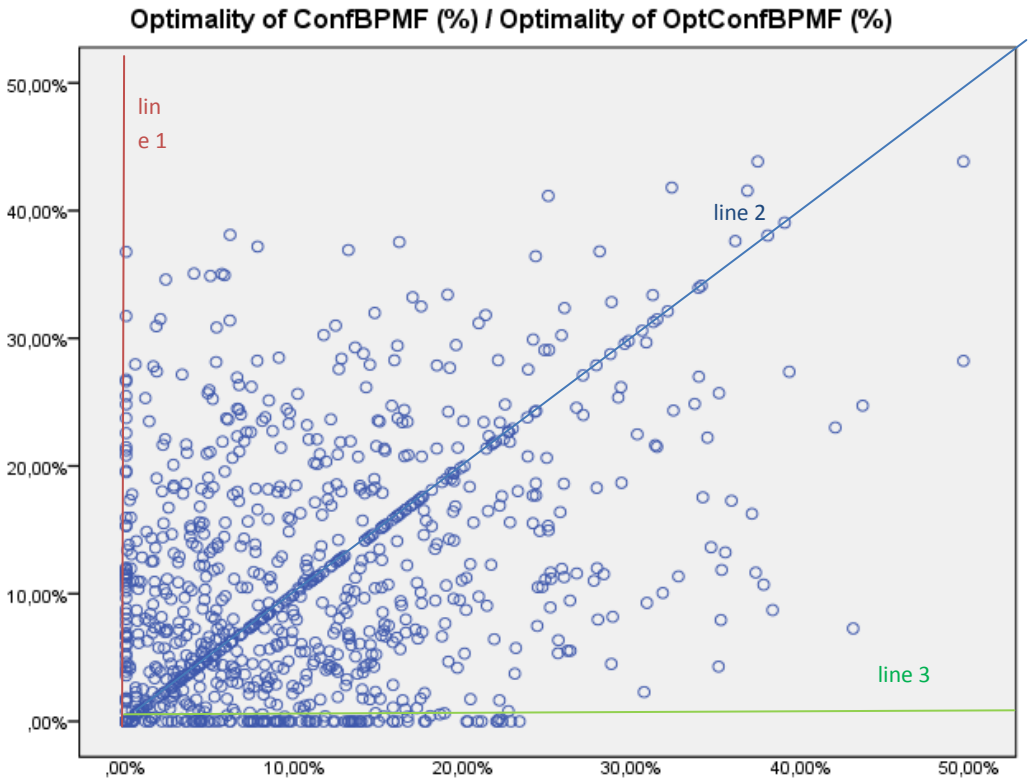
Хипотеза 4.1.Не постоји статистички значајан разлика у ефикасности оптимизованог и неоптимизованог приступа за решење дефинисаног задатка оптималне конфигурације SOA фамилије.

Аналогно анализи представљеној у секцији 4.4.3.2., релативни однос растојања међу мерама решењахеуристичког приступа у односу на оптимално решењесе користи за процену растојања међу решењима. Применом претходно развијеног FM генератора [135], BPMF генератора [180] и имплементираног *ConfSOA* приступа, имплементиран је и *OptConfSOA* приступ. Насумично генерисане SOA фамилије (са вредностима броја карактеристика у FM и величине модела обележја и инстанци, дискутованим у поглављу 4.4.1.2.) су оптимално конфигурисане истовремено са оба хеуристичка приступа (*ConfSOA*и*OptConfSOA*) као и са *brute force*алгоритмом помоћу којег је добијено оптимално решење. На тај начин су добијене вредности растојања решења оба приступа у односу на оптимално, па је за тестирање хипотезе коришћен ANOVA тест за поређење просечних вредности добијених растојања као и за процене да ли међу њима постоји статистички значајна разлика (Хипотеза 4.1).

Резултати. Средња вредност растојања решења добијеног помоћу*OptConfSOA*приступа у односу на оптимално решење износи 10.41% (Ст.Дев.=0.964%). Значи, оптималност приступа је приближно 89.5%, тј. резултат добијен оптимизованим приступом је приближно 10.5% мањег квалитета у односу на оптимално решење.

С друге стране, средња вредност растојања решења проблема конфигурације истих SOA фамилија, добијена помоћу*ConfSOA*приступа износи 10.20% (Ст.Дев.=0.928%).

Иако су просечне вредности оптималности оба приступа приближно једнаке, графичка презентација (представљена на Слици 29) показује јасну неповезаност међу оптималностима оба приступа. Само у 23.00%случајева (line 2) оба приступа имају једнаку оптималност, док у 13.4% (line 1) тј 17.2% (line 3) резултати добијени једним од *OptConfSOA*тј*ConfSOA* приступа су оптимални, док са другим имају ненулто растојање од оптималног.



Слика 29. Дијаграм односа међу растојањима решења добијених примерном алгоритама*OptConfBPMF*и*ConfBPMF*у односу на оптимално решење

Како подаци о вредностима растојања решења оба приступа нису нормално дистрибуирани, једнофакторска ANOVA се користи над логаритамски трансформисаним подацима у циљу поређења добијених просечних вредности .

Резултати показују да нема статистички значајне разлике у тачности оба приступа $F(1,782)=11.814$, $p=0.229$. Значи, хипотеза 4.1. је прихваћена чиме се закључује да представљена оптимизација целог приступа не утиче на оптималност добијених решења, што је резултат од изузетног значаја с обзиром да је сложеност целог приступа редукована до полиномијалне, уз употребу адаптивних казнених пенала

који имају у литератури познати ризик од конвергенције ка локалној оптималној вредности, а не траженој глобалној [200].

5. Имплементација

У овом поглављу се представљају детаљи развоја и имплементације окружења под називом *cs-ahp-fmp*које је развијено у области линија софтверских производа и омогућава корисницима да дефинишу различите врсте захтева на основу којих се врши рангирање одлика у датом моделу одликаи одређивање оптималне конфигурације.

5.1.Коришћени софтверски алати и библиотеке

*Cs-ahp-fmp*је развијен као проширење (енгл. *plug-in*)Eclipse развојном окружењу са циљем да омогући корисницима опис проблема конфигурације и дефинисања структуре обележја и инстанци релевантних за дати проблем. Такође, корисник може дефинисати ограничења вредности појединих обележја, тј., за монотоно растућа обележја могуће је дефинисати највећу вредност, а за монотоно опадајућа најмању вредност.

За имплементацију модела одлика (*feature model*) којим се представљају карактеристике линија софтверских производа користи се *feature model plug-in (fmp)* [218]. Овај plug-in обезбеђује интерактивно окружење за едитовање и конфигурисање модела одлика. Осим употребе у Eclipse развојном окружењу, могуће је, уз примену *fmp2rsm*[219], *fmp plug-in* користити и у Rational Software Modeler (RSM) и Rational Software Architect (RSA) окружењима која интегришу *fmp* и омогућавају UML моделовање линија производа.

Да би се корисницима омогућило дефинисање двослојне структуре обележја и инстанци, као и каснији захтеви над њима, проширен је *Visual-feature model plug-in (vis-fmp)*алат представљен у[220]. Овим алатом је омогућено интерактивно дефинисање модела одлика и инстанци као и означавање одлика у односу на дату структуру. У те сврхе алат обезбеђује визуелизацију модела одлика и структуре за представљање нефункционалних карактеристика, као и директну интеракцију са датим моделом и структуром. Према томе, *vis-fmp*алат омогућава:

- Интеграцију модела одлика са структуром обележја и инстанци за представљање нефункционалних карактеристика.
- Аутоматизацију процеса конфигурације модела одлика на основу нефункционалних карактеристика и основних врста захтева који се односе на монотоне тенденције нефункционалних карактеристика.
- Подршку техникама визуелизације и интеракције које обезбеђују јасан поглед на генерисану конфигурацију.

Задатак имплементације у овом раду се стога огледа у проширењу *vis-fmp*алата и омогућавању:

- Дефинисања разних врста комплекснијих захтева над нефункционалним карактеристикама.
- Рангирања одлика на основу дефинисаних захтева и одабира одговарајуће конфигурације.

За имплементацију је коришћен *Prefuse*[221],графичка библиотека имплементирана у програмском језику Java која садржи различите начине за графичку презентацију стабала и графова. Један од начина представљања модела одлика је 2D стабло за које је у [220] показано да је због величине модела одлика најпогоднији начин за њихово представљање. Ово стабло се користи као иницијална репрезентација која је проширена у циљу представљања модела одлика, нефункционалних карактеристика и захтева, као и крајње оптималне конфигурације. У креираном стаблу за представљање модела одлика, чворови представљају одлике, а гране међу њима композиционе релације и релације варијабилности међу одликама.

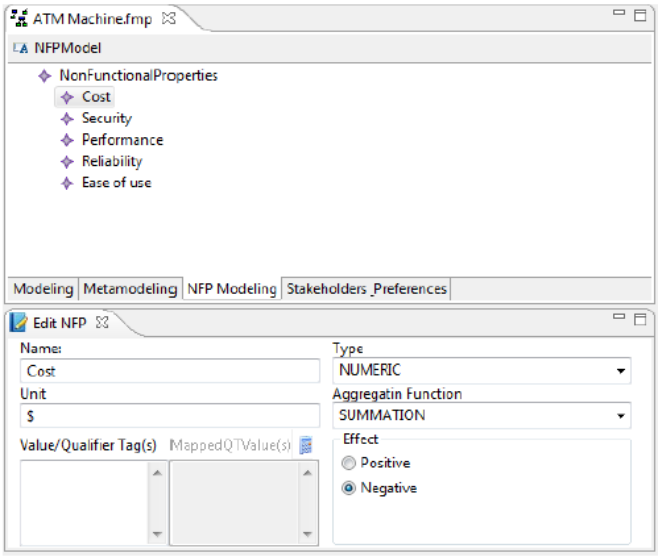
За представљање условних захтева, логички израз који дефинише услов се представља у облику стабла у којем постоји неколико врста чворова: чворови за представљање логичких променљивих, помоћни чворови за представљање сложенијих логичких израза, као и чворови који одговарају логичким операторима. У овим стаблима гранама се дефинише хијерархија међу чворовима, а *preorder* обиласком датог стабла се добија дефинисани логички израз.

CS-AHP алгоритам за анализу захтева корисника је претходно имплементиран у [203], док се за имплементацију приступа заснованог на адаптацији генетичког алгоритма користи *Java Genetic Algorithms Package* (JGAP)[10] пакет.

5.2.Детаљи корисничког окружења

У циљу дефинисања захтева над нефункционалним карактеристикама и омогућавања оптималне конфигурације, неопходно је дефинисати одговарајућу структуру обележја и инстанци, као и омогућити одређивање вредности сваке од карактеристика за одређену комбинацију одлика.

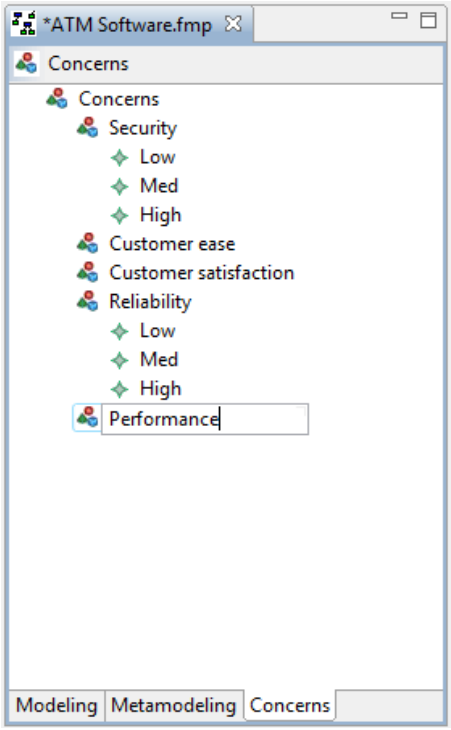
У развијеном алату, као што је показано на Слици 30, корисник може да дефинише нефункционалне карактеристике од интереса, које, као што је дефинисано у Секцији 2.3.3.1., представљају скуп обележја за дати модел. У складу са анализом датом у Секцији 2.1.4., за свако од обележја је могуће дефинисати основне карактеристике, као што су тип, јединична мера, функција агрегације, подразумевана вредност итд. Након креирања скупа одлика, за сваку од одлика се врши процена интервала могућих вредности на основу скупа расположивих сервиса који је имплементирају.



Слика 30. Дефинисање скупа обележја

[10] http://jgap.sf.net

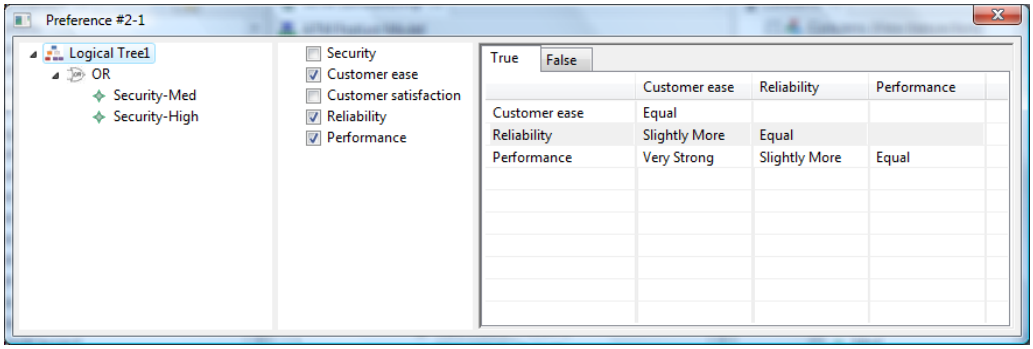
На основу процењених интервала, корисник може дефинисати скуп инстанци за свако од обележја (Слика 31) као скуп могућих вредности са одговарајућим лексикографским значењем. Подразумевана подела интервала могућих вредности је на једнаке подинтревале за сваку од инстанци, док је кориснику остављена могућност њихове модификације уз аутоматско обезбеђење дисјунктности подинтервала и потпуног покривања интервала датог обележја.



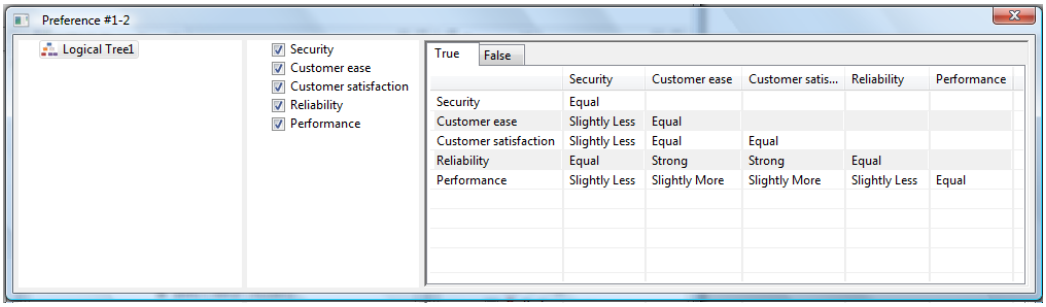
Слика 31. Дефиниција скупа инстанци за дефинисана обележја

Након креирања структуре обележја и инстанци, кориснику је омогућено да дефинише захтеве на оба нивоа у хијерархији, нивоу обележја и нивоу инстанци. На нивоу обележја, корисник може дефинисати: *(i)* која обележја имају доминантни релативни приоритет и квалитативни редослед међу њиховим инстанцама, *(ii)* условне захтеве међу одговарајућим (под)скупом обележја (Слика 32 *i*), као и*(iii)* безусловне захтеве међу одговарајућим (под)скупом обележја (Слика 32 *ii*). Код условно и безусловно дефинисаних захтева, релативни приоритети се дефинишу помоћу матрице која је реципрочно симетрична у односу на јединичну дијагоналу, а одговарајућа поља имају унапред дефинисане могуће вредности: 1-*equal (indifferent*), 3-*slight value*, 5-*strong value*, 7-*very strong*и9-*extreme value*.

(i)



(ii)

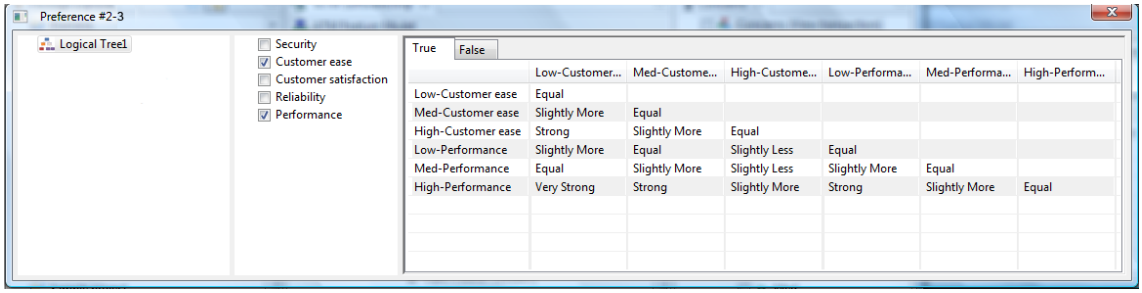


Слика 32. Дефинисање различитих врста захтева на нивоу обележја: *(i)* условно дефинисани захтеви; *(ii)* безусловно дефинисани захтеви

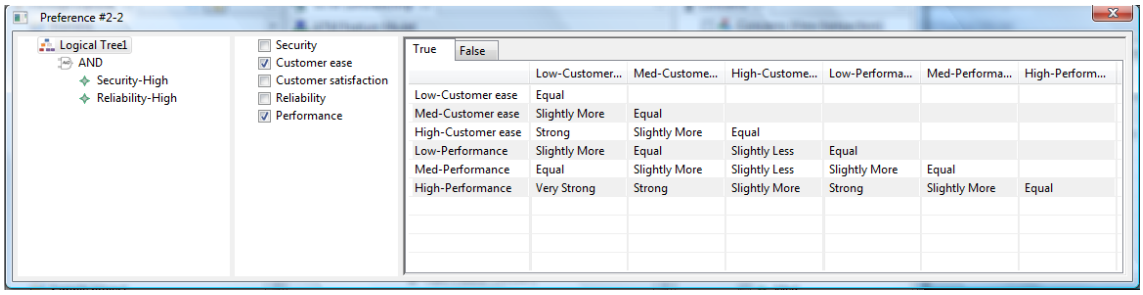
Аналогно, на нивоу инстанци је могуће дефинисати условне (Слика 33ii) и безусловне захтеве о релативним односима инстанци обележја(Слика 33ii) за која претходно није дефинисано да имају доминантни релативни приоритет. Форме за унос су креиране на аналоган начин, при чему се прво врши одабир обележја за чије инстанце се дефинише приоритет. У услову (уколико је реч о условно дефинисаном захтеву) се не могу садржати инстанце обележја за који се дефинише релативни приоритет.

При дефинисању захтева на оба нивоа у хијерархији (ниво обележја и ниво инстанци), део форме за дефинисање матрице о релативним односима се састоји од два таба која одговарају редом *true-* и*false-* делу захтева. Недефинисањем неког од делова, дефинише се захтев који се састоји само од true- или false- дела.

(i)

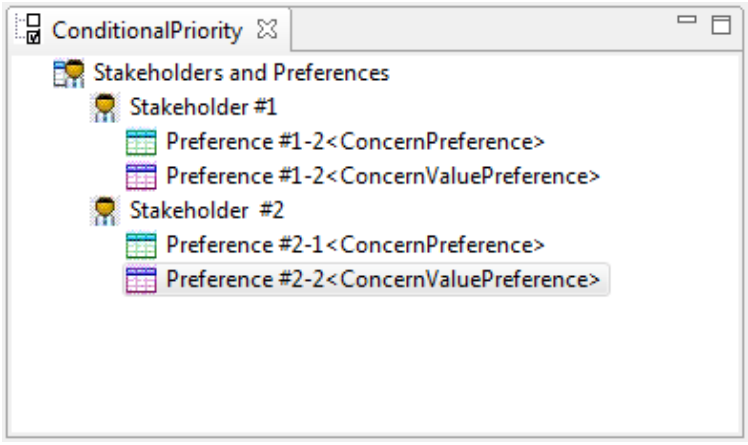


(ii)



Слика 33. Дефинисање различитих врста захтева на нивоу инстанци: *(i)* условно дефинисани захтеви; *(ii)* безусловно дефинисани захтеви

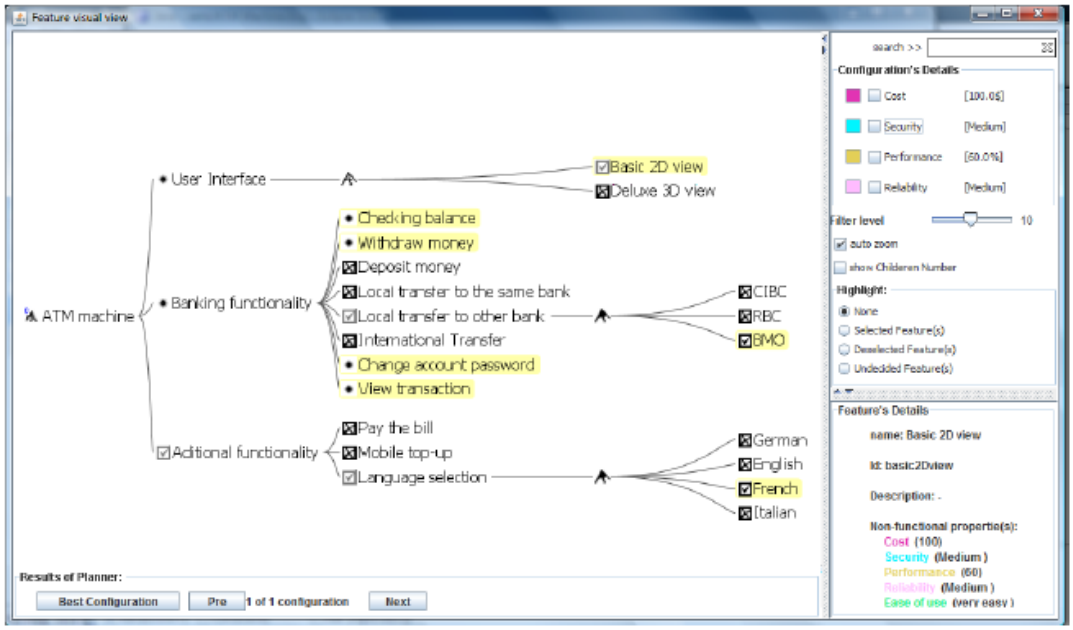
На тај начин се добија цели скуп захтева дефинисан од стране датог корисника. Развијено решење омогућава да више корисника независно дефинишу своје захтеве (Слика 34), при чему се за одређивање крајње конфигурације сви дефинисани захтеви разматрају независно тј., оптимална конфигурација се одређује независно сходно захтевима сваког од корисника.



Слика 34. Дефинисање захтева од стране више корисника

Након дефинисања захтева, оптимална конфигурација се одређује позивом генетичког алгоритма, тј. *OptConfSAOF*решења. Оптималну конфигурацију (скуп

одлика са селектованим сервисима) је могуће експортовати, а додатно се даје и графичка репрезентација која видно означава одабране одлике и приказује остварене вредности нефункционалних карактеристика. Помоћу Visual View, претходно развијен у [220] за проблем конфигурације одлика, корисник је у могућности да врши измене у добијеном решењу и на тај начин проверава како дате измене утичу на укупне нефункционалне карактеристике конфигурације. На Слици 35 је приказан пример генерисане конфигурације који помаже кориснику да разуме генерисану конфигурацију и начине на које је могуће извршити измене.



Слика 35. Пример оптималне конфигурације (visual view)

6. Анализа перформанси

У овом поглављу се даје анализа развијених метода као и резултати њихове примене у области фамилије пословних процеса.

6.1.Симулациона анализа CS-AHP алгоритма

У овој секцији се, на основу закључака Пропозиције 1 (из Секције 4.4.3.), врши карактеризација модела ***(O, C, QT, ℜ)***са становишта употребе CS-AHP алгоритма и дају препоруке за пожељну структуру модела која значајно смањује неконзистенције.

6.1.1. Дескриптивни параметри модела (O, C, QT, ℜ)

Неконзистености у захтевима могу бити идентификоване независно на оба нивоа двослојне структуре, тако да са становишта употребе CS-AHP алгоритма, посматрају се редом *број неконзистентности на нивоу обележја* и *број неконзистности на нивоу инстанци*.

За дати CS-AHP модел ***(O, C, QT, ℜ),*** скуп расположивих опција *O* се сматра интрументом за тестирање да ли је почетни скуп захтева конзистентан, а идентификовани број неконзистентности се сумира на оба нивоа за комплетан скуп расположивих опција. У циљу постављања израчунатих вредности независним од броја опција, користе се просечне вредности, тако да се у даљим разматрањима употреба CS-AHP алгоритма анализира на основу *просечног броја неконзистентности у затхевима* идентификованих посебно на оба нивоа.

Дескриптивни параметри модела ***(O, C, QT, ℜ)***се идентификују на следећи начин:

- Двослојна хијерархијска структура је представљена скуповима обележја и инстанци, па број обележја и просечан број инстанци по сваком обележју представљају пар вредности за карактеризацију двослојне структуре. Карактеризација је јединствена до на скуп обележја и њихових инстанци.

- С друге стране, скуп захтева може имати различите структуре које директно зависе од начин на који корисник дефинише захтеве и сходно могућности за дефинисањем непотпуних захтева и више различитих условних захтева који се односе на исти пар обележја (или инстанци). Стога, следећи параметри се препознају као дескриптивни за скуп захтева, који су у сагласности са корисниковим могућностима за дефинисањем разних врста захтева који могу имати директне импликације на сами алгоритам:
    - *Структура условних захтева* која се односи на форму захтева, тј да ли се условно дефинисани захтев састоји само из *then*-дела или садржи и *else*- део
    - Како укупан број захтева у моделу није ограничен и у општем случају не зависи од структуре обележја и инстанци, укупан број захтева се не може сматрати параметром модела. На нивоу обележја, сваки захтев дефинише релативни однос између два или више обележја, тако да се на овом нивоу, *условни пар обележја* дефинише као пар обележја чији је релативни приоритетни однос дефинисан са најмање једним условно дефинисаним захтевом. Аналогно, на нивоу инстанци, сваки захтев дефинише релативни однос међу инстанцама датог обележја, тако да се *условним обележјем* сматра обележје код којег је релативни припритетни однос између бар једног пара његових инстанци дефинисан бар једним условним захтевом. На крају, *број условних парова обележја на нивоу обележја* и *број условних обележја на нивоу инстанци* се сматрају параметрима који описују концепт условљености у захтевима у двослојној структури.

У наставку се врше анализе у циљу идентификације потенцијалне везе између идентификованих дескриптивних параметара с једне стране, и броја неконзисентности на оба нивоа, с друге стране. Такође, врши се процена просечног броја неконзистентности који се појављују у захтевима на оба нивоа.

6.1.2. Хипотезе

Главни циљ, дефинисан као статистички значајна карактеризација броја неконзистенција у захтевима и процена њихових просечних вредности, се може разложити на неколико појединачних хипотеза:

Хипотеза 7.1. Промене вредности једног дескриптивног параметра, при фиксним вредностима осталих параметара резултирају статистички значајним променама броја неконзистенција идентификованих на оба нивоа модела ***(O, C, QT,*** $\Re$ ***).***

Хипотеза 7.2. Просечан број неконзистенција идентифкованих на оба нивоа модела ***(O, C, QT,*** $\Re$ ***)***се статистички значајно може окарактерисати бројем условних парова обележја (на нивоу обележја) и бројем условних обележја (на нивоу инстанци).

Хипотеза 1 даје информацију о појединачном утицају сваког дескриптивног параметра на цели модел, док Хипотеза 2 идентификује најбољи скуп предиктора из скупа дескриптивних параметара за број неконзистенција у моделу, посебно за сваки ниво модела.

С друге стране, како условљеност у дефинисаним захтевима корисника одређује различите дистрибуције скупа захтева над скупом обележја, оне се намећу интересантним за разматрање као додатне дексриптивне карактеристике модела. Тешко је очекивати да се у реалним применама дистрибуције сматрају стварним дескриптивним параметром, посебно због дискретних вредности и потешкоћа у процени, али се с теоретског становишта намеће анализа утицаја оваквог параметра на цели модел. С тим у вези, дефинише се следећа хипотеза.

Хипотеза 7.3. Модели ***(O, C, QT,*** $\Re$ ***)***са истим дескриптивним параметрима имају статистички значајне разлике у броју неконзистенција при различитим дистрибуцијама условних захтева.

За тестирање наведених хипотеза се користе разне статистичке меотде којима је могуће анализирати зависности међу дескриптивним параметрима модела и идентификованим неконзистенцијама. Регресиони модел и корелациона анализа се користе за проверу да ли дескриптивни параметри статистички значајно одређују број неконзистенција у моделу (Хипотеза 2). ANOVA тест за поређење средњих вредности над више независних популација се користи за тестирање статистичке значајности у броју неконзистенција у случају промена вредности дексриптивних параметара (Хипотеза 1), као и у случају промена дистрибуција условних захтева (Хипотеза 3).

Као крајњи резултат анализе, даје се процена просечних вредности неконзистенција за групе модела са истим дескриптивним каракетристикама, које су добијене као просечна вредност броја неконзистенција добијена на довољно великом независном узорку.

6.1.3. Експерименталне поставке

Као што је раније наведено, дескриптивни параметри се могу посматрати са становишта карактеризације модела за употребу CS-AHP алгоритма. У том циљу насумичним (енгл. *random*) генерисањем се креирају различити модели са истим вредностима дескриптивих параметара,па се за сваки од тако креираних модела рачунају неконзистенције на оба нивоа и добијене вредности се користе за даљу анализу.

Максималан број обележја и инстанци при генерисању модела је ограничен редом на 7 и 10, с обзиром да се сматрају разумним и могућим за разликовање људском уму [112]. За сваки пар обележја, укупан број захтева је насумично генерисан, од којег било који број може бити условно дефинисан са насумично дефинисаним условљеностима. Аналогно насумично генерисање, без било каквих ограничења је примењено на ниво инстанци. С друге стране, као што је наведено у коментару Пропозиције 1 (Секција 4.4.3.), захтеви дефинисани о доминантним релативним приоритетима не утичу на број неконзистенција у моделу, тако да симулације обухватају само моделе са условно и безусловно дефинисаним

захтевама, а резултати анализе ће касније бити генерализовани за општи случај.Укупан број условних парова обележја на нивоу обележја је у интервалу *[0, n(n-1)/2*], а сваки пар обележја може зависити од највише *n-2* других обележја, где је *n*-укупан број обележја у моделу . *Т*акође, укупан број условних обележја на нивоу инстанци је из интервала *[0, n*], а свако обележје може зависити од највише *n-1* других обележја, гдје је *n* укупан број обележја у моделу. Као што је наведено у Секцији 4.1.2.3, у 4. кораку алгоритма (под називом *Одређивање рангова на нивоу обележја*) се врши филтрирање обележја мањег приоритета. Током симулација, одлуке о броју обележја мањег приоритета који ће бити одстрањени из даљих разматрања се доносе такође насумично. На крају, симулација обухвата 1,000 насумице генерисаних модела за сваку могућу комбинацију дескриптивних параметара.

Током генерисања захтева корисника, задовољени су услови о формама захтева из Дефиниција 9-12 (Секција 4.4.1.1.). У претходним истраживањима из области Линија софтверских производа резултати су показали да је укупан број опција у моделу у опсегу од 10 до 192, са просечном вредношћу 51.71 (СД=64.24). Према томе, скуп захтева у сваком моделу је тестиран са укупним бројем опција који одговара датој средњој вредности (тј. 52), при чему су карактеристике сваке опције насумично генерисане.

За тестирање хипотеза 1 и 2, мотивисани илустративним примером 1 и његовим проширењима, следеће три симулације су извршене. У првој симулацији, након насумице генерисаног *then*-дела условног захтева, одлука о дефинисању *else*-дела је насумице генерисана. У другој симулацији, захтеви су креирани само са *then*-делом, тј у случају захтева које би требале да садрже и *else*-део, креира се нови захтев који се састоји од негације услова (из претходног захтева) и *then*-дела. Трећа симулација обухвата захтеве који обавезно садрже и *then*- и *else*-део. Током насумичног генерисања захтева, могуће је генерисати и празан else-део у захтеву, што одговара дефинисању захтева само са then-делом.

За тестирање хипотезе 3, насумичне симулације су креиране са три различите дистрибуције условних парова обележја (на нивоу обележја) и условних обележја (на нивоу инстанци), у односу на укупан број обележја у моделу, од којег дистрибуције зависе. Три дистрибуције које имају опште познате интерпретације

које одговарају реалним ситуацијама из праксе су инверзна експоненцијална, нормална и потпуно насумична дистрибуција. У нашим симулацијама, број обележја, број условних парова обележја (на нивоу обележја) и број условних обележја (на нивоу инстанци) су цели бројеви, па су наведене дистрибуције генерисане са целобројним вредностима над целим бројевима. Насумично генерисање осталих параметара модела је урађено на исти начин као што је претходно наведено за хипотезе 1 и 2.

6.1.4. Технике за анализу података

Сходно врстама података који су генерисани током симулација, интерпретирани су елементима стандардне дескриптивне статистике (што се на основу [201] сматра устаљеном праксом) укључујући средњу вредност и стандардну девијацију. Параметри који представљају укупан број обележја, број условних парова обележја (на нивоу обележја) и број условних обележја (на нивоу иснтанци) припадају групи интервалних података (енгл. *interval data*). У складу са тим, анализе постојања разлика међу групама, корелације и регресиона анализа се врше употребом параметарских тестова: ANOVA, Pearson-ов коефицијент корелације, вишеструка регресија и вишеструка регресија са додатним-променљивима[11] (енгл. *dummy-variables*), редом). У случају променљивих које нису нормално расподељене, параметарски тестови се користе над логаритамски трансформисаним подацима[12].

Параметар модела који се односи на врсте условно дефинисаних захтева може имати три вредности: само *then*-део, опциони *else*-део и *then*- са *else*-делом. Могуће вредности су кодиране редом са 0, 1 и 2, при чему тако дефинисани параметар представља категоријски параметар за анализу (енгл. *categorical data*). Подаци који се односе на овај параметар су подељени у три групе и регресиони модел је креиран посебно за сваку групу, након чега су резултати упоређени.

[11]Употреба помоћних-променљивих се односи на квантитативно предстаљање припадности групи при чему се превазилази проблем фаворизовања групе [66]

[12]Овакав начин трансформисања података је устаљен и потиче из дисциплина, као што је медицина, које заснивају доказе на чињеницама о подацима [67]. Штавише, овакав приступ је конзистентан са приступима својственим за психолошки заснованих мерења [64], а циљ нашег истраживања је управо анализа захтева и преференци корисника.

Такође, у циљу проналажења најбољег регресионог модела, креирано је и међусобно упоређено неколико различитих модела.

Као што ће касније током описа резултата анализа бити показано, дескриптивни параметар који представља број условних парова обележја на нивоу обележја ће бити трансформисан у нову категоријску променљиву која се добија груписањем укупног броја условних парова обележја.

6.1.5. Резултати анализа

У овој секцији се наводе коришћене статистичке техинке, као и добијени резултати и њихова тумачења.

Као први корак у анализи, тестирано је да ли постоји статистички значајна корелација између дефинисаних дескриптивних параметара и броја неконзистенција у захтевима на оба нивоа. Табела 9 садржи сумиране резултате анализе корелације за оба нивоа. Вредности коефицијента корелације који су мањи од 0.5 се обично сматрају незначајним [201], па се стога, параметар који представља просечан број инстанци по обележјима у структури неће разматрати у даљим анализама јер је у слабој корелацији са бројем неконзистенција (на нивоу обележја: $0.20<0.5$, а на нивоу инстанци $0.192<0.5$).

У циљу анализе утицаја дескриптивних параметара који предстаљају број обележја при фиксираним вредностима осталих дескриптивних параметара, полазни скуп података је даље посматран по групама формираним на основу броју обележја (тј вредности параметра чији утицај се анализира). Како подаци нису били нормално расподељени, једнофакторски ANOVA тест је примењен над логаристамски транфорсмисаним подацима у циљу поређења средње вредности зависне променљиве (*број неконзистенција у захтевима на оба нивоа*) за различите групе дефинисане на основу броја обележја. Резултати показују статистички значајну разлику у броју неконзистанција за различите вредности броја обележја на оба нивоа: на нивоу обележја је $F(7,2460)=69.526,\ p=0.000$, а на нивоу инстанци $F(7,772)=93.328,\ p=0.000$; средње вредности броја неконзистенција за сваку групу су наведене у Табели 10. *Tukey post-hoc* тест

открива да не постоји статистички значајна разлика међу групама код којих се број обележја разликује за 1.

Табела 9. Резултати корелационе анализе

Дескрипивни параметар	Број неконзистенција у захтевима (ниво обележја)	Број неконзистенција у захтевима (ниво инстанци)
	Корелација*ρ*-коефицијент; *p* вредност	Корелација*ρ*-коефицијент; *p* вредност
Број обележја	0.515;< 0.005	0.516; < 0.005
Број инстанци	-0.20; < 0.005	0.192; < 0.005
Број условних парова обележја / условних обележја	0.672; < 0.005	0.650; < 0.005
Структура условних захтева	0.646; < 0.005	0.595; < 0.005

Табела 10.Средња вредност и стандардна девијација за број неконзистенција у захтевима за различите вредности броја обележја у моделу

Број обележја	Број неконзистенција у захтевима		Број обележја	Број неконзистенција у захтевима	
	Ниво обележја	Ниво инстанци		Ниво обележја	Ниво инстанци
	Ср. вредност, Станд. Дев			Ср. вредност, Станд. Дев	
3	0.497, 0.384	0.208, 0.194	7	2.658, 2.391	2.062, 1.789
4	0.859, 0.722	0.449, 0.425	8	3.492, 3.165	3.021, 2.641
5	1.336, 1.172	0.781, 0.641	9	4.455, 4.062	4.293, 3.755
6	1.944, 1.743	1.324, 1.138	10	5.610, 5.024	5.885, 5.216

Табела 11.Средња вредност и стандардна девијација за број неконзистенција у захтевима за различите стуктуре условних захтева

Структура условних захтева	Број неконзистенција у захтевима		Структура условних захтева	Број неконзистенција у захтевима	
	Ниво инстанци	Ниво инстанци		Ниво обележја	Ниво инстанци
	Ср. вредност, Станд. Дев			Ср. вредност, Станд. Дев	
then-део	0.879, 0.644	0. 861, 0.839	Then-и else-део	6.573, 4.793	4.889, 4.673
else-део опциони	3.797, 2.675	2.925, 2.938			

Аналогна анализа је извршена и за остале дескриптивне параметре. Резултати показују да статистички значајна разлика у броју неконзистенција постоји међу захтевима различите структуре: *F(2,2465)=936.888, p=0.000*, на нивоу обележја

и*F(2, 777)=150.227, p=0.000*на нивоу инстанци. *Tukey post-hoc* тест је показао да статистички значајна разлика постоји између сваког пара група. Средње вредности неконзистенција по групама су представљене у Табели 11.

Табела 12.Средња вредност и стандардна девијација за број неконзистенција у захтевима за различите вредности броја условних парова обележја

Број условних парова обележја	Ср. вредност, Станд. Дев	Број условних парова обележја	Ср. вредност, Станд. Дев	Број условних парова обележја	Ср. вредност, Станд. Дев	Број условних парова обележја	Ср. вредност, Станд. Дев
1-3	0.565, 0.488	7-10	1.985, 1.311	16-21	4.501, 2.889	29-36	7.665, 5.068
4-6	1.159, 0.763	11-15	3.208, 2.059	22-28	5.998, 3.889	37-45	9.836, 6.184

Резултати показују да стастистички значајна разлика у броју неконзистенција у захтевима постоји у случају различитих вредности броја условних парова обележја на нивоу обележја, *F(45,2422)=58.446, p=0.000.* У овом случају *Tukey post-hoc* тест је показао да не постоји статистички значајна разлика између сваке две групе. Зато су креиране подгрупе тако да међу елементима у подгрупама не постоји статистичка значајна разлика. Узимајући у обзир чињеницу да за број обележја *n*, максималан број условних парова обележја је *n(n-1)/2* (претходно објашњено у 7.1.1.), подгрупе су креиране на следећи начин: прва подгрупа садржи до 3 условна пара обележја, друга подгрупа садржи од 4 до 6 условних парова, а остале редом, од 7 до 10, од 11 до 15, од 16 до 21, од 22 до 28, од 29 до 36 и од 37 до 45. Једнофакторски ANOVA тест је показао да не постоји статистички значајна разлика међу паровима унутар сваке од креираних подгрупа. Такође, исти тест показује да постоји статистички значајна разлика у броју неконзистенција међу креираним подгрупама, *F(7,2460)=345.93, p=0.000*. *Tukey post-hoc* тест, супротно случају када подргупе нису биле креиране, сада је показао статистички значајну разлику између свака два пара подгрупа. Средње вредности неконзистенција по креираним подгруама су представљене у табели 12.

С друге стране, резултати су показали да постоји статситички значајна разлика у броју неконзистенција међу различитим вредностима условних обележја дефинисаних на нивоу инстанци, *F(9,770)=79.923, p=0.000*. Слично претходној анализи, *Tukey post-hoc* тест је показао да не постоји статистички

значајна разлика међу групама код којих се број условних обележја разликује за 1. Средње вредности неконзистенција по групама су представљене у табели 13.

Табела 13.Средња вредност и стандардна девијација за број неконзистенција у захтевима за различите вредности броја условних обележја

Број условних обележја	Ср. вредност, Станд. Дев	Број условних обележја	Ср. вредност, Станд. Дев	Број условних обележја	Ср. вредност, Станд. Дев
1	0.502, 0.504	4	2.262, 2.026	7	5.457, 3.876
2	1.002, 1.001	5	3.139, 2.586	8	6.881, 4.703
3	1.513, 1.597	6	4.216, 3.187	9	8.633, 5.609
				10	10.743, 6.956

На крају се може закључити да је прва хипотеза доказана у случају следећих дескриптивних параметара: броја обележја, структуре условних захтева, броја условних парова обележја (на нивоу обележја) и броју условних обележја (на нивоу инстанци); док је оповргнута у случају дескриптивног параметра који представља просечан број инстанци по обележјима. За даље анализе се користе подгрупе креиране од дескриптивног параметра који представља условне парове обележја на нивоу инстанци.

Дискусија. Као што се може видети, током тестирања Хипотезе 1, показано је да сваки од дефинисаних дескриптивних параметара модела осим просечног броја инстанци по обележјима, појединачно има статистички значајан утицај на број неконзистенција, под претпоставком фиксираних вредности осталих параметара модела.

Табеле 6-8 сумирају резултате регресионе анализе за Хипотезу 2. Као што је раније наведено, анализа је подељена на три дела на основу стуктуре условних захтева. Стога, табеле су организоване на следећи начин: свака табела одговара једној од три структуре захтева, а у свакој од њих су представљени подаци посебно за оба нивоа (ниво обележја и ниво инстанци). Такође, током тестирања Хипотезе 1, дескриптивни параметар који одговара броју условних парова обележја је трансформисан у категоријску променљиву са осам могућих вредности. Како се ова променљива користи при регресионој анализа, креирано је седам помоћних варијабли, као што је представљено на Слици 31. У анализи су

разматрана два регресиона модела: први се састоји само од помоћних варијабли (на нивоу обележја) и условних обележја (на нивоу инстанци), као параметра који је највише корелисан са бројем неконзистенција у захтевима; други модел који као додатни предиктор има број обележја у моделу.

Number of pairs of conditional concerns	Dummy variable 1	Dummy variable 2	Dummy variable 3	Dummy variable 4	Dummy variable 5	Dummy variable 6	Dummy variable 7
1-3	0	0	0	0	0	0	0
4-6	1	0	0	0	0	0	0
7-10	0	1	0	0	0	0	0
11-15	0	0	1	0	0	0	0
16-21	0	0	0	1	0	0	0
22-28	0	0	0	0	1	0	0
29-36	0	0	0	0	0	1	0
36-45	0	0	0	0	0	0	1

Слика 36. Помоћне варијабле за регрециону анализу

Табела 14.Резултати регресионе анализе за условне захтеве који се састоје само од *then*-дела

Променљива		Нестандардизовани коефицијент		Стандардизовани коефицијент	Резиме Модела
		B	SE B	B	r^2, F(df1,df2), p
Модел 1:	Константа	0.205	0.011	0.043	0.957, F(7,812)=2606.442, 0.00<0.001
	Пом. пром.1	0.082	0.017		
	Пом. пром.2	0.276	0.016	0.152	
	Пом. пром.3	0.562	0.017	0.286	
	Пом. пром.4	0.844	0.016	0.465	
	Пом. пром.5	1,212	0.017	0.631	
	Пом. пром.6	1,645	0.019	0.760	
	Пом. пром.7	2,099	0.023	0.745	
Модел 2:	Константа	0.187	0.023	0.042	0.957, F(1,811)=2280.297, 0.00<0.001
	Пом. пром.1	0.081	0.017		
	Пом. пром.2	0.273	0.017	0.151	
	Пом. пром.3	0.559	0.018	0.285	
	Пом. пром.4	,839	0.017	0.462	
	Пом. пром.5	1,206	0.018	0.628	
	Пом. пром.6	1,637	0.020	0.756	
	Пом. пром.7	2,090	0.025	0.741	
	Бр обележја	0.003	0.003	0.008	
Модел 1:	Константа	-0.335	0.057	0.834	0.880, F(1,258)=214.009, 0.00<0.001
	Бр условних обележја	0.288	0.012		
Модел 2:	Константа	-1.100	0.085	0.682	0.919, F(1,257)=5.226, 0.00<0.001
	Бр условних обележја	0.236	0.011		
	Бр обележја	0.134	0.012	0.344	

Табела 15.Резултати регресионе анализе за условне захтеве који се састоје од *then*-дела и опционог *else*-дела

Променљива		Нестандардизовани коефицијент		Стандардизовани коефицијент	Резиме Модела
		B	SE B	B	r^2, F(df1,df2), p
Модел 1:	Константа	1.081	0.091	0.042	0.937, F(7,820)=603.418, 0.00<0.001
	Пом. пром.1	0.345	0.142		
	Пом. пром.2	1.185	0.137	0.151	
	Пом. пром.3	2.358	0.144	0.278	
	Пом. пром.4	3.502	0.137	0.446	
	Пом. пром.5	4.894	0.142	0.589	
	Пом. пром.6	6.282	0.155	0.671	
	Пом. пром.7	8.855	0.190	0.726	
Модел 2:	Константа	0.014	0.188	0.037	0.986, F(1,819)=559.180, 0.00<0.001
	Пом. пром.1	0.304	0.139		
	Пом. пром.2	1.066	0.135	0.136	
	Пом. пром.3	2.159	0.144	0.255	
	Пом. пром.4	3.224	0.140	0.411	
	Пом. пром.5	4.537	0.149	0.546	
	Пом. пром.6	5.846	0.165	0.625	
	Пом. пром.7	8.340	0.202	0.684	
	Бр обележја	0.158	0.025	0.106	
Модел 1:	Константа	-1.157	0.210	0.814	0.895, F(1,258)=198.023, 0.00<0.001
	Бр условних обележја	0.983	0.044		
Модел 2:	Константа	-3.950	0.315	0.656	0.922, F(1,257)=9.321, 0.00<0.001
	Бр условних обележја	0.791	0.041		
	Бр обележја	0.491	0.046	0.359	

Дискусија. У сва три случаја различитих структура условних захтева, оба модела довољно прецизно дају оцену очекиваног броја неконзистенција у моделу, на оба нивоа, нивоу обележја и нивоу инстанци (табеле 14-16). Може се уочити да, регресиони модел који се састоји од помоћних варијабли (на нивоу обележја) и броја условних обележја (на нивоу инстанци) може предвидети преко 90% варијабилности у броју неконзистенција у моделу. То значи да дескриптивни параметри који описују скуп захтева $\Re$ статситички значајно карактеришу модел *(O, C, QT,* $\Re$*)*са становишта употребе CS-AHP алгоритма, чиме је Хипотеза 2 доказана.

Табела 16.Резултати регресионе анализе за условне захтеве који се састоје од *then*- и *else*- дела

Променљива		Нестандардизовани коефицијент		Стандардизовани коефицијент	Резиме Модела
		B	SE B	B	r^2, F(df1,df2), p
Модел 1:	Константа	1.516	0.077	0.043	0.963, F(7,812)=1928.460, 0.00<0.001
	Пом. пром.1	0.613	0.119		
	Пом. пром.2	2.097	0.115	0.155	
	Пом. пром.3	4.242	0.121	0.290	
	Пом. пром.4	6.357	0.115	0.469	
	Пом. пром.5	9.086	0.119	0.634	
	Пом. пром.6	12.266	0.130	0.760	
	Пом. пром.7	15.752	0.159	0.749	
Модел 2:	Константа	1.118	0.162	0.042	0.963, F(1,811)=2631.574, 0.00<0.001
	Пом. пром.1	0.599	0.119		
	Пом. пром.2	2.054	0.116	0.152	
	Пом. пром.3	4.169	0.123	0.285	
	Пом. пром.4	6.254	0.120	0.461	
	Пом. пром.5	8.954	0.128	0.625	
	Пом. пром.6	12.105	0.142	0.750	
	Пом. пром.7	15.562	0.172	0.740	
	Бр обележја	0.059	0.021	0.023	
Модел 1:	Константа	-1.877	0.305	0.848	0.923, F(1,258)=175.564, 0.00<0.001
	Бр условних обележја	1.629	0.063		
Модел 2:	Константа	-6.524	0.423	0.683	0.952, F(1,257)=4.112, 0.00<0.001
	Бр условних обележја	1.311	0.055		
	Бр обележја	0.817	0.062	0.375	

Како вредност R^2 мери колики дио варијабилности зависне променљиве је описан предикторима, иста ће се користити као критеријум за одабир бољег модела за оба нивоа, посебно. Може се уочити да модел који се састоји од броја обележја и броја помоћних варијабли на нивоу обележја, у сва три случаја различитих структура у условним захтевима има исту или већу вредност R^2него модел који се састоји само од помоћних варијабли (Табела14: 0.957=0.957, Табела15: 0.986>0.937, Табела16: 0.963=0.963). Исти закључак се може извести и на нивоу инстанци, гдје модел који се састоји од броја обележја и броја условних обележја је бољи него модел који се састоји само од условних обележја (Табела 14: 0.919>0.880, Табела 15: 0.922>0.895, Табела 16: 0.952>0.923). Овај резултат само потврђује оно што је и очекивано, да се главни узрочници неконзистенција у условним захтева односе на број условно дефинисаних парова обележја (тј

обележја на нивоу инстанци), структуре условних захтева, и броја обележја у моделу.

На основу претходне хипотезе закључено је да структура условних захтева статистички значајно утиче на број неконзистенција у захтевима и да је најмања вредност у случају захтева који се састоје само од *then*-дела, па ће се Хипотеза 3 тестирати само у том случају.

Резултати показују да постоји статистички значајна разлика у броју неконзистенција при различитим дистирбуцијама условних захтева над обележјима: на нивоу обележја $F(2,2465)=44.507,\ p=0.000$; на нивоу инстанци $F(2,777)=40.535,\ p=0.000$. Средње вредности неконзистенција за различите врсте дистрибуција су представљене у Табели 17. *Tukey post-hoc* тест је показао да не постоји статистички значајна разлика између комплетно насумице генерисане дистрибуције и нормалне дистрибуције, али постоји међу осталим паровима дистрибуција.

Табела 17.Средња вредност и стандардна девијација за број неконзистенција у захтевима за различите врсте дистибуција

Дистрибуција	**Ниво обележја**	**Ниво инстанци**
	Ср. вредност, Станд. Дев	
Нормална	1.123, 0.768	2.234, 2.005
Инверзна експоненцијална	0.876, 0.123	1.612, 1.749
Насумице генерисана	1.234, 0.567	2.192, 3.016

Дискусија. Добијени резултати потврђују очекивану интепретацију наведене три дистрибуције. Нормална дистирбуција форсира зависности од средњег броја обележја, док инверзна експоненцијана форсира зависности од мањег броја обележја, што последично смањује укупан број неконзистенција у захтевима.

6.1.6. Критички осврт и препоруке

Резултати хипотеза откривају утицаје сваког од идентификованих параметара на број неконзистенција у захтевима на оба нивоа у моделу. Анализа Хипотезе 2 показује да број неконзистенција може бити с највећом прецизношћу окарактерисан следећим параметрима: број обележја и број условних парова обележја (на нивоу обележја), и број условних обележја (на нивоу инстанци).

Како је симулација обухватала генерисање 1,000 различитих модела за сваку комбинацију параметара, а неконзистенције су тестиране на 52 насумице генерисане опције, добијени резултати се сматрају довољно великим независним узорком за процену очекиваног бројанеконзистенција у моделу. Резултати су представљени у табелама Б.1. и Б.2. у Додатку Б.

Вредности из табела Б.1. и Б.2. се предлажу за процену вредности добијених применом CS-AHP алгоритма над датим моделом **(O, C, QT, ℜ)**, на следећи начн:

1. Просечан број неконзистенција добијен на оба нивоа у моделу треба упоредити са одговарајућим вредностима у табелама. Вредности у табелама су одређене на основу дескриптивних карактеристика модела: броја обележја (умањеног за број обележја са доминантним релативним приоритетом), броја условних парова обележја (на нивоу обележја) и броја условних обележја (на нивоу инстанци);
2. Ако су обе добијене вредности мање или једнаке у односу на вредности у табелама, значи да је број неконзистенција испод просека па се може занемарити
3. Ако је било која од добијених вредности неконзистенција већа од вредности у табелама, значи да је број неконзистенција изнад просека. Корисник треба да одлучи о прихватању вредности донијених као резултат CS-AHP алгоритма, јер се на основу тако великог броја неконзистенција може закључити да је модел неадекватан за употребу CS-AHP алгоритма. То значи да рангиране вредности добијене овим алгоритмом могу бити јако удаљене у односу на почетни скуп захтева а све проузроковано бројем неконзистенција који је изнад просечног.

На основу резултата и анализа симулација, намећу се следеће препоруке за употребу предложеног алгоритма и начина за најадекватнију конструкцију модела и скупа захтева, са становишта употребе алгоритма.

Препорука 1. Најмањи број неконзистенција у моделу је у случају дефинисања захтева који садрже само *then*-део, тј. мањи је број конфликата за задовољење дефинисаних услова у захетвима.

Препорука 2. Модел са мањим бројем обележја и већим бројем инстанци је погоднији у односу на модел са више обележја, с обзиром да је доказано да не постоји статистички значајно повећање неконзистенција повећањем броја инстанци по обележјима.

Препорука 3. Мањи број обележја који се појављују у захтевима не гарантују мањи број неконзистенција у захтевима. То значи да је неконзистентност у захтевима условљена искључиво ситуацијама када је више различиих захтева истовремено задовољено, а дефинишу приоритетне односе међу истим паровима обележја или инстанци.

6.2.Каракетристике примене развијеног системау области фамилије пословних процеса

Као што је представљено у претходним секцијама, област фамилије пословних процеса представља главни пример примене развијеног приступа. Како су веб сервиси и BPEL процеси су постали де-факто стандарди за имплементацију пословних процеса, у литератури је највећа пажња поклоњена управо анализи веб сервиса и њихових нефункционалних карактеристика. Стога, и поред генерализације развијеног приступа и општости његове примене, анализа перформанси се врши над фамилијама пословних процеса, а добијени резултати се касније могу генерализовати и поредити са резултатима у осталим областима примене.

6.2.1. Фамилије пословних процеса

Предузеће које је испоручилац софтвера употребом веб сервиса нуди, посредством Интернета, софтвер енкапсулиран унутар веб сервиса. Број расположивих сервиса рапидно расте, а предузеће и поред сопствених ограничења (цена улагања, итд) мора задовољити и захтеве клијената који се тичу квалитета [223].Наведени аргументи представљају јасне разлоге за популарност анализе проблема конфигурације у области фамилија пословних процеса уз оптимално испуњење различитих врста захтева који одговарају реалним сценаријима примене.

Стога се, фамилија пословних процеса дефинише, у складу са Дефиницијом 4 (Секција 2.1.6.), као пар $BPMT^{FM} = \left(G_{PM}^{FM}, \{S_a\}_{a \in A}\right)$, где G_{PM}^{FM} представља шаблон пословних процеса, а $\{S_a\}_{a \in A}$ скуп веб сервиса расположив за сваку од атомичних активности $a \in A$ са одговарајућим нефункционалним карактеристикама.

6.2.2. Симулациона анализа оптималне конфигурације фамилије пословних процеса

За анализу развијених приступа са становишта употребе у области фамилија пословних процеса врши се симулација и поређење перформанси оба развијена приступа, тј оптимизованог *OptConfSOAF* и неоптимизованог*ConfSOAF*. Осим анализа оптималности, врши се и анализа зависности у односу на основне карактеристике модела фамилија пословних процеса. Тиме се добијају закључци о могућем утицају предложеног начина за моделовање фамилија пословних процеса (секција 2.1.6.1.) и карактеристика структуре за представљање нефункционалних захтева на оптималност предложеног решења.

6.2.2.1.Дескриптивни параметри модела

Као што је претходно наведено у поглављу 4.4. параметри који карактеришу посматрани модел су: 1а) број карактеристика у моделу одлика (*Feature Model - FM*), 1б) процентуална заступљеност опционих елемената у моделу, 2а) број

активности у дијаграму тока, 2б) процентуални удио стандардних образаца композиције у дијаграму тока, 3а) број расположивих сервиса за сваку од активности, 3б) вредности нефункционалних карактеристика за сваки од расположивих сервиса, 4а) захтеви корисника о нефункционалним карактеристикама система као и строга ограничења о њиховим вредностима над појединим деловима или целој SOA архитектури (уколико се желе дефинисати). Такође, због ограничења над топологијом модела пословних процеса [201] као и бијективног мапирања између FM модела и дијаграма тока фамилије, вредности варијабли 1а и 2а су исте.

За генерисање модела одговарајућих карактеристика, користе си FM генератор претходно развијен у [144] и генератор за насумично генерисање QoS модела (сходно карактеризацији карактеристика веб сервиса из поглавља 2.1.4) [202]. За анализу оптималности предложених решења користе се brute-force алгоритам, неоптимизовани GA приступ (*ConfSOAF*) [202] и оптимизовани приступ *OptConfSOAF*, након чега се врши поређење добијених резултата.

Претходна истраживања су показала да је за анализе довољно посматрати само четири наведене групе QoS карактеристика [209], а да се број различитих карактеристика у свакој групи [210][211], као и њихове вредности насумично генеришу. До сада није креиран систематски приступ одређивања вредности GA параметара, па се у експериментима користе следеће вредности: величина популације $\|P\| = 50$, највећа генерација $\|G\| = 200$, вероватноћа рекомбианције = 1 (тј. увек се примењује), стопа мутације = 0.1, динамички казнени фактор $w(gen) = C * gen$, C=*0.5*[212] .

6.2.2.2.Експеримент 1: Поређење оптималног решења и хеуристичих приступа

Главни циљ овог експеримента је процена растојања међу вредностима карактеристика квалитета (QoS) оптималног решења и решења добијених применом предложених хеуристичких приступа (оптимизованог и неоптимизованог). На тај начин је могуће извршити анализу да ли примењени

начини оптимизације имају статистички значајан утицај на оптималност целог приступа. У складу са тим дефинисана је следећа хипотеза.

Хипотеза 6.4. Решења добијена применом оптимизованог и неоптимизованог приступа се не разлику (статистички) значајно у свом растојању у односу на оптимално решење.

Апроксимациони однос (хеуристички приступ у односу на оптимално решење) се користи као метрика за процену растојања у односу на оптимално решење. Насумично генерисање BPMFs и одговарајућих QoS модела се извршава 1000 пута; задаци оптимизације се решавају истовремено применом оба развијена хеуристичка приступа, као и *brute-force* алгоритмом помоћу којег се добија оптимално решење.

Подаци добијени симулацијама су анализирани стандардним елементима дескриптивне статистике, средња вредност (*Ср.вредност*) и стандардна девијација (*Станд.Дев*). Наведена хипотеза је тестирана помоћу ANOVA теста. Овим тестом су поређене средње вредности независно генерисаних популација које одговарају резултатима примене наведених присупа и алгоритама, а које је потребно упоредити како би се проверило постојање статистички значајне разлике међу њима.

Експеримент 1: Резултати.Просечна вредност релативног растојања вредности квалитета конфигурације генерисане помоћу *OptConfSOAF* приступа и оптималног решења износи 10.41% (SD=0.964%). Значи, оптималност развијеног приступа је приближно 89.5%, тј резултати добијени применом оптимизованог хеуристичког приступа су приближно са 10.5% мањим вредностима квалитета у односу на оптимално решење.

Као што је наведено у претходним анализама (поглавље 4), оптималност *ConfSOAF* је процењена на 10.20% (SD=0.928%).

Како подаци добијени датим експериментом не одговарају нормалној расподели, једнофакторски ANOVA тест је примењен над логаритамски трансформисаним подацима како би се извршило поређење просечних вредности

релативних растојања хеуристичких приступа у односу на оптимално.Резултати показују да разлика међу приступима није статистички значајна, $F(1,782)=11.814$, $p=0.229$. Значи, Хипотеза 6.4 је доказана па се закључује да извршена оптимизација нема статистички значајан утицај на оптималност предложеног приступа.

Дати резултат је од посебног значаја, с обзиром да је сложеност целог приступа редукована до полиномијалне, док употреба адаптивних казнених фактора може бити од великог ризика за локалну конвергенцију, као што је показано у [207].

6.2.2.3.Експеримент 2: Анализа перформанси *ConfSOAF* присупа за разне вредности улазних параметара модела

Као што је раније наведено (у Секцији 4.4.3.), *serviceTransform* алгоритам је један од централних делова применеGA алгоритма за решавања проблема оптималне конфигурације SOA фамилија. Алгоритам*serviceTransform* се користи за разрешавање проблема генерисања комбинација сервиса који нису валидни у односу на ограничења дефинисана моделом. Како се, додатно, ограничења у оба модела генеришу насумично, за главни циљ анализе се намеће утврђивање да ли развијени приступ статистички значајно зависи од различитих образаца у FM и BPMF моделима. У том циљу, дефинишу се следеће хипотезе [202]:

Хипотеза 6.5. У случају различитих дистрибуција опционих одлика у моделу одлика FM, не постоји статистички значајна разлика оптималности *ConfSOAF* решења.

Хипотеза 6.6. У случају различитих дистрибуција образаца у моделу одлика FM, постоји статистички значајна разлика оптималности *ConfSOAF* решења.

Хипотеза 6.7. У случају различитих дистрибуција образаца у моделу BPMF, не постоји статистички значајна разлика оптималности *ConfSOAF* решења.

У циљу тестирања наведених хипотеза, извршено је неколико симулација са различитим дистрибуцијама патерна у оба модела. Дистрибуције опционих

одлика су одређене процентуалним односом у односу на укупни број одлика у моделу, а у експериментима се разматрају следеће вредности: 25%, 50%, 75% и 100% (које креирају редом групе 1-4 које се касније пореде).

Аналогно, разматрају се следеће вредности процентуалне заступљености патерна у моделу одлика FM: 33% (приближно једнака заступљеност AND, OR и XOR група одлика – што представља групу 5 у експерименту), насумична процентуална заступљеност тачно два патерна за сваку од могућих комбинација (AND-OR, AND-XOR, OR-XOR – групе 6-8), и 100% (тачно један патерн у моделу – групе 9-11).

На основу бијективног мапирања међу FM и BPMF моделима, наведене дистрибуције патерна у моделу одлика дефинишу и заступљености патерна у BPMF моделу, осим дистрибуције секвенцијалних и паралелених AND патерна, који одговарају AND групи одлика у моделу. Стога, у симулацијама се посматрају следеће дистрибуције секвенцијалних и паралелних AND образаца у BPMF моделу: 25%, 50%, 75% и 100% (групе 12-15).

Свака симулација се генерише 1000 пута, а добијени подаци се користе за тестирање дефинисаних хипотеза помоћу ANOVA теста за утврђивање да ли постоји статистички значајна разлика у односу на оптимално решење у случајевима различитих дистрибуција опционих одлика (Хипотеза 6.5), као и у случајевима различитих дистрибуција образаца у оба модела (Хипотеза 6.6 и Хипотеза 6.7).

Експеримент 2: Резултати.У циљу анализе утицаја дистрибуције опционих одлика у моделу одлика FM, сматра се да су добијени подаци подељени у четири групе на основу процентуалног удела опционих и обавезних одлика у моделу. Како подаци не одговарају нормалној дистрибуцији, једнофакторски ANOVAтест се користи над логаритамски трансформисаним подацима како би се употребиле просечне вредности зависне променљиве.

Резултати показују да нема статистички значајне разлике међу групама $F(3,584)=23.328$, $p=0.634$; просечне вредности за сваку од група су представљене у Табели 18.

Табела 18. Просечне вредности релативног растојања решења *ConfSOAF* у односу на оптимално решење за различите дистрибуције опционих одлика у FM моделу

Процентуални удио опционих одлика у FM моделу	Релативно растојање у односу на оптимално решење	Процентуални удио опционих одлика у FM моделу	Релативно растојање у односу на оптимално решење
	Ср. вредност, Станд. Дев		Ср. вредност, Станд. Дев
25%	9.87% ; 0.644	75%	9.98% ; 0.912
50%	10.13% ; 0.724	100%	10.35% ; 0.928

Аналогно претходном, у циљу анализе утицаја патерна варијабилности и композиције, сматра се да су добијени подаци подељени у седам група сходно процентуалном уделу патерна у моделу FM и четири групе сходно процентуалном уделу секвенцијалних и паралелних AND патерна у BPFM моделу. Како подаци не одговарају нормалној дистрибуцији, једнофакторски ANOVA тест се користи над логаритамски трансформисаним подацима како би се употребиле просечне вредности зависне променљиве.

Табела 19. Просечне вредности релативног растојања решења *ConfSOAF* у односу на оптимално решење за различите дистрибуције патерна у FM моделу

Дистрибуције патерна у FM моделу	Релативно растојање у односу на оптимално решење	Дистрибуције патерна у FM моделу	Релативно растојање у односу на оптимално решење
	Ср. вредност, Станд. Дев		Ср. вредност, Станд. Дев
33% (AND), 33% (OR), 33% (XOR)	9.98%, 0.956	100% (AND)	10.27%, 0.963
(random) AND-OR	10.25%, 0.923	100% (OR)	10.37%, 0.927
(random) AND-XOR	10.51%, 0.978	100% (XOR)	10.35%, 0.959
(random) OR-XOR	10.43% , 0.945		

Резултати показују да постоји статистички значајна разлика међу групама сходно дистрибуцијама патерна у FM моделу: $F(6,2260)=62.526$, $p=0.000$; просечне вредности за сваку од група су представљене у Табели 19. Такође, Tukey post-hoc тест је идентификовао да статистички значајна разлика постоји само

између група са једнаким процентуалним уделима обрасца AND, OR и XOR и група са тачно једним патерном у целом моделу.

С друге стране, резултати показују да нема статистички значајне разлике међу групама са различитим дистрибуцијама патерна у BPMF моделу: F(3,572)=33.328, p=0.580; просечне вредности за сваку од група су представљене у Табели 20.

Табела 20. Просечне вредности релативног растојања решења *ConfSOAF* у односу на оптимално решење за различите дистрибуције секвенцијалних и паралелних AND патерна у BPMF

Дистрибуције секвенцијалних патерна	Релативно растојање у односу на оптимално решење	Дистрибуције секвенцијалних патерна	Релативно растојање у односу на оптимално решење
	Ср. вредност, Станд. Дев		Ср. вредност, Станд. Дев
25%	9.91%, 0.934	75%	10.75%, 0.974
50%	10.56, 0.988	100%	10.32%, 1.034

Експеримент 2: Закључак. На основу представљених резултата извршених експеримената, може се закључити да патерни варијабилности који су представљени у моделу одлика FM су главни извор за статистички значајан утицај на оптималност добијених резултата и стога афирмишу само хипотезу 6.6. док у исто време остале хипотезе чине негативно дефинисаним. Главни разлог се може пронаћи у чињеници да су одлике у моделу одлика FM главни извор варијабилности, тј FM је примарно креиран за моделовање варијабилности, па се и очекује да покаже такав утицај у експериментима. Стога се, варијабилност у FM моделу може третирати као главни извор девијације добијеног решења у односу на оптимално.

Такође, како ефективност SPL приступа директно зависи од начина како су варијабилности у моделу одлика имплементиране и представљене у току развојног животног циклуса, добијена процена просечне вредности оптималности од 90% показује да је представљени хеуристички приступ довољно ефективан и ефикасан у свим случајевима и успешно може превазићи зависности у односу на карактеристике начина моделовања.

6.2.2.4.Експеримент3: Анализа перформанси *OptConfSOAF* присупа за различите вредности улазних параметара модела

Аналогно претходном експерименту, каоглавни циљ анализе се намеће утврђивање да ли развијени оптимизовани приступ статистички значајно зависи од различитих образаца у FM и BPMF моделима. У том циљу, дефинишу се следеће хипотезе:

Хипотеза 6.8. У случају различитих дистрибуција опционих одлика у моделу одлика FM, не постоји статистички значајна разлика оптималности *OptConfBPMF*решења.
Хипотеза 6.9. У случају различитих дистрибуција патерна у моделу одлика FM, постоји статистички значајна разлика оптималности *OptConfBPMF*решења.
Хипотеза 6.10. У случају различитих дистрибуција патерна у моделу BPMF, не постоји статистички значајна разлика оптималности *OptConfBPMF*решења.

Евалуација дефинисаних хипотеза је реализована на начин аналоган претходно описаном експерименту. Такође су коришћене исте поделе добијених резултата по групама.

Експеримент 3: Резултати.Резултати показују да постоји статистички значајна разлика међу групама сходно дистрибуцијама опционих патерна у FM моделу: $F(3,584)=15.489$, $p=0.001$; просечне вредности за сваку од група су представљене у Табели 21. Такође, Tukey post-hoc тест је идентификовао да статистички значајна разлика постоји међу свим групама у моделу.

Резултати показују да постоји статистички значајна разлика међу групама сходно дистрибуцијама патерна у FM моделу: $F(6,2260)=34.4789$, $p=0.000$; просечне вредности за сваку од група су представљене у Табели 22. Такође, Tukey post-hoc тест је идентификовао да статистички значајна разлика постоји само између група са једнаким процентуалним уделима обрасца AND, OR и XOR и група са тачно једним патерном у целом моделу.

Табела 21. Просечне вредности релативног растојања решења *OptConfSOAF* у односу на оптимално решење за различите дистрибуције опционих одлика у FM моделу

Процентуални удио опционих одлика у FM моделу	Релативно растојање у односу на оптимално решење	Процентуални удио опционих одлика у FM моделу	Релативно растојање у односу на оптимално решење
	Ср. вредност, Станд. Дев		Ср. вредност, Станд. Дев
25%	9.67%, 0.548	75%	10.58%, 0.736
50%	10.24%, 0.294	100%	10.76%, 0.341

Табела 22. Просечне вредности релативног растојања решења *OptConfSOAF* у односу на оптимално решење за различите дистрибуције патерна у FM моделу

Дистрибуције обрасца у FM моделу	Релативно растојање у односу на оптимално решење	Дистрибуције обрасца у FM моделу	Релативно растојање у односу на оптимално решење
	Ср. вредност, Станд. Дев		Ср. вредност, Станд. Дев
33% (AND), 33% (OR), 33% (XOR)	9.65%, 1.423	100% (AND)	10.69%, 0.913
(random) AND-OR	10.25%, 0.944	100% (OR)	10.77%, 0.941
(random) AND-XOR	10.48%, 0.961	100% (XOR)	10.65%, 1.743
(random) OR-XOR	10.52% , 1.216		

С друге стране, резултати показују да постоји статистички значајне разлике међу групама са различитим дистрибуцијама патерна у BPMF моделу: $F(3,572)=19.457$, $p=0.000$; просечне вредности за сваку од група су представљене у Табели 23.Такође, Tukey post-hoc тест је идентификовао да статистички значајна разлика постоји само између група са само једним патерном, секвенцијалним или паралелним -AND.

Табела 23. Просечне вредности релативног растојањарешења *OptConfSOAF* у односу на оптимално решење за различите дистрибуције секвенцијалних и паралелних -AND обрасца у BPMF

Дистрибуције секвенцијалних обрасца	Релативно растојање у односу на оптимално решење	Дистрибуције секвенцијалних обрасца	Релативно растојање у односу на оптимално решење
	Ср. вредност, Станд. Дев		Ср. вредност, Станд. Дев
25%	9.68%, 0.569	75%	10.78%, 1.024
50%	10.54, 1.762	100%	10.85%, 0.965

Експеримент 3: Закључак.На основу представљених резултата извршених експеримената, може се закључити да патерни варијабилности и композиција, извор за статистички значајан утицај на оптималност добијених резултата. У поређењу се претходним приступом, може се уочити да су варијабилности у моделу подржане представњним начином оптимизације.

6.2.2.5.Закључак симулационих анализа

Представљени резултати експеримената показују да иако су оба предложена приступа довољно оптимална (у односу на критеријум оптималног испуњења дефинисаних захтева), карактеришу их различити облици зависности од карактеристика модела фамилије пословних процеса. Као главни разлог се намећу различити степени варијабилности у целом моделу, не само у моделу одлика већ и велики број расположивих веб сервиса различитих карактеристика квалитета. Додатно, неодређеност модела захтева која карактерише условљеност у захтевима и могућност дефинисања непотпуних захтева даје додатну сложеност на основу које се врши предложена оптимизација целог приступа, што свакако води ка закључку о директном утицају на перформансе целог приступа.

Као општи закључак урађених анализа, осим препорука о начинима за дефинисање захтева који смањују могуће неконзистенције (резултати представљени у секцији 6.1.5.), намеће се неопходност анализа за конкретне моделе примене у којима би се представљени резултати детаљније анализира у циљу обезбеђења већег степена оптималности.

7. Закључак

У овом поглављу се наводе резултати остварени решењима предложеним овом дисертацијом. Такође се даје коментар око потенцијалних домена примене развијеног CS-AHP алгоритма и интегралног *OptConfSOA*решења, каои могући правци у даљем развоју.

7.1.Остварени допринос

У овој дисертацији су предложена иновативна решења проблема представљања и анализе нефункционалних захтева корисника, као и проблема оптималне конфигурације SOA архитектура. CS-AHP алгоритам разбијен је као проширење широко примењивог и познатог AHP алгоритма за потребе представљања и анализе условно дефинисаних захтева и захтева о лексикографском поретку као врсте захтева које имају могућност примене у реалним ситуацијама. Такође, за дати алгоритам су симулационо утврђене карактеристике примене на основу којих је могуће, у односу на карактеристике конкретног модела примене, утврдити и карактеристике конзистентности. Алгоритам је креиран за двослојну хијерархијску структуру нефункционалнаих карактеристика па се симулационо дају и препоруке око креирања саме структуру у циљу обезбеђења већег степена конзистенције у захтевима.

Интегрално решење*ConfSOA*је развијено применом генетичког алгоритма за решењавање задатка оптималне конфигурације који је дефинисан у складу са развијеном двослојном структуром за представљање нефункционалних карактерстика и одговарајућих захтева над њима, као и моделима за представљање SOA фамилија. За дато решење је симулационо показано да даје довољно ефикасно решење, међутим уочени су недостаци временске сложености. За оптимизацију су примењени приступи динамичког генерисања вредности појединих елемената генетичког алгоритма, чиме је добијено решење *OptConfSOA*. Дато решење је посебно анализирано са становишта примене у области пословних процеса, где је анализирана зависност оптималности у односу на карактеристике модела. Показано је да предложено решење обезбеђује

истовремено задовољење захтева о функционалностима система као и нефункционалних захтева који могу бити различитог нивоа приоритета, односити се на поједине делове или сервисно-оријентисане архитектуре у целости. Такође, предложено решење је опште и пременљиво у било ком домену у којем се SOA парадигма може применити посматрањем сервиса као било које компоненте (необавезно софтверске) дате функционалности.

На основу наведеног, допринос ове дисертације се може дефинисати као потпуно ново интегрално решење за аутоматизацију комплетног процеса спецификације, анализе и крајње конфигурације SOA фамилије на основу скупа расположивих сервиса.Појединачно, остварени су следећи доприноси:

- Преглед и анализа досадашњих достигнућа у областима развоја и конфигурације SOA и фамилија SOA, анализе и рангирања захтева корисника, као и областима представљања знања, интелигентног резоновања и решавања задатака оптимизације.
- Анализа недостатака постојећих метода/техника за спецификацију, рангирање и аутоматску конфигурацију SOA фамилија на основу различитих врста захтева корисника.
- Развој иновативног CS-AHP алгоритма за представљање и квантитативну анализу разних врста захтева (безусловних, условних и захтева о лексикографском поретку) који омогућава:
    - дефинисање разних врста захтева над двослојном семантичком структром за представљање карактерстика релевантних за процес одлучивања;
    - идентификацију и разрешавање разних врста неконзистенција које се могу појавити у дефинисаним захтевима;
    - проверу конзистентности у складу са карактеристикама конструисане двослојне структуре и захтевима над њом.
- Креирање иновативног модела за представљање нефункционалних карактеристика, заснованог на двослојној семантичкој структури и развијеном CS-AHP алгоритму за представљање захтева над њима, који омогућава:

    - представљање разних врста очекивања, ставова и захтева корисника о нефункционалним карактеристикама делова SOA-е као и SOA фамилије у целости;
    - дефиницију задатка оптималне конфигурације SOA фамилије што у основи представља задатак персоналне конфигурације у складу са разним врстама захтева дефинисаних од стране корисника.
- Креирање иновативног хеуристичког приступа за решење проблема оптималне конфигурације SOA фамилије које:
    - је засновано на претходно конструисаној двослојној структури за представљање нефункционалних карактеристика и CS-AHP алгоритма за анализу захтева над њом;
    - обезбеђује ефикасно и ефективно решење персонализовано ка кориснику и његовим захтевима;
    - обезбеђује конструкцију конфигурације SOA фамилје у односу на дефинисане функционалне и нефункционалне карактеристике.
- Примена развијеног интегралног решења у области пословних процеса и анализа карактеристика примене у зависности од модела пословних процеса тј.од карактеристика и услова окружења у којем се примењује.

Остварени доприниси са становишта практичне примене су:

- Препоруке тј. смернице за конструкцију двослојне семантичке структуре за представљање нефункционалних карактеристика и захтева над њима у циљу обезбеђења мањег броја неконзистенција;
- Проширење постојећих модела за представљање нефункционалних карактеристика развијеном двослојном семантичком структуром;
- Развијена конкретна апликација (прототип) који представља имплементацију развијеног решења за представљање и анализу захтева корисника;
- Анализа основних карактеристика развијеног интегралног решења као и развијеног алгоритма за анализу захтева.

7.2.Могућност коришћења и примене

Развијени CS-AHP алгоритам обезбеђује метод за представљање и анализу захтева над двослојном структуром обележја и инстанци независно од домена примене. У [208]је приказана примена датог приступа у области линија софтверских производа. С обзиром да је део добро познате AHP фамилије метода, развијени метод има велике потенцијале опште применљивости, које карактеришу све припаднике ове фамилије метода[102] [103] [104] [101].

Предложено интегрално решење *OptConfSOA* се односи на SOA фамилије и проблем њихове оптималне конфигурације. Представљеном компаративном анализом је показано да дато решење представља, у односу на постојећарешења представљена у литератури, најбоље решење са становишта персонализације ка кориснику и његовим захтевима. Домен примене се дефинише доменима креирања SOA фамилија, а уколико се ослободи услов посматрања веб сервиса, па се под сервисима посматра само једна компонента (која не мора бити софтверска) дате функционалности, добија се широк опсег могућности примене.Један од примера који потврђује наведено, односи се на домене персонализованог е-учења као и домен персонализованог одабира курсева на одређеном нивоу едукације.

У случају е-учења, уколико се разматра персонализација система е-учења, у којем се распоред и дужина трајања наставних јединица као и одабир наставних материјала за сваку од њих врши на основу каракетристика сваког студента појединачно (његовог претходног знања, преферираног стила учења, оствареног успеха), SOA фамилија се може дефинисати у складу са током наставног процеса, а сервисимасе могу представити расположиви скупови наставних материјала. Под нефункционалним карактеристикама се могу сматрати како карактеристике система (временска ограничења исл), тако и карактеристике које се односе на језик наставних материјала, типове провере знања, знања која се подстичу датим материјалом итд. Захтеви могу бити дефинисани од стране ученика (префериране врсте наставног материјала, темпо рада) као и предавача (крајњи циљ тј. ниво знања који је неопходно обезбедити, вештине којима студент мора овладати исл). На тај начин, проблем оптималне конфигурације наставних материјала и наставних јединица одговара конфигурацији предложене SOA архитектуре у

односу на функционалне захтеве (наставне цјелине) и захтеве корисника (ученика и професора) у складу са расположицим сервисима (наставним јединицама за сваку од наставних цјелина).

Услучају проблема персонализованог одабира курсева на одређеном нивоу едукације, SOA фамилија се може конструисати у односу на организацију процеса едукације у складу са његовим временским трајањем (нпр. додипломске студије трају 3-5 година, свака година се састоји од два семестра итд). Расположиви сервиси би били сви курсеви које студент може полагати; неки могу бити обавезни, неки су опциони, а међу некима од њих могу постојати релације искључења као и предуслови. Нефункционалне каракетристике могу бити дефинисане у складу са облашћу која је обухваћена курсом, оценама евалуације курса од стране претходних студената, оценама предавача курса, итд. Број курсева по семестрима, као и додатни захтеви могу бити дефинисани правилима организације и стога, укључени у сам модел SOA фамилије. На тај начин, оптимална конфигурације SOA фамилије дефинише распоред курсева по семестрима у складу са жељама и преференцама студента, уз поштовање свих правила и ограничења дефинсаних на датом нивоу едукације.

Наведени примери који показују могућности примене развијених решења у области едукативних система, само су неки од могућих примера примене и указују на широки опсег домена примене.

7.3. Могући даљи правци истраживања

С обзиром да су у овој дисертацији представљена иновативна решења у областима анализе захтева корисника и оптималне конфигурације SOA фамилије, могући правци даљих истраживања су дефинисани управо у тим областима. Конкретно, даља истраживања могу бити реализована у циљу:

- Аутоматизације процеса идентификације неконзистентности у захтевима и обезбеђења њихове динамичке идентификације и разрешавања;
- Развој метода за генерисање формалне спецификације захтева на основу захтева нижег нивоа структурираности, што за крајњи циљима развој

метода за директну трансформацијузахтева представљених у облику структурираног текста у формално представљене захтеве;

- Проширење CS-AHP алгоритма за анализу разних врста захтева, нпр динамички променљивих захтева, временских захтева итд;
- Примену развијених приступа у разним доменима у којима се могу идентификовати нове врсте захтева;
- Проширење развијеног приступа за адресирање проблема динамичких промена у доменима примене, нрп. неки од расположивих сервиса може постати недоступан; или се пак променити захтеви о преферираних карактеристикама система;
- Проширење развијеног приступа у циљу обезбеђења искуственог интелигентног учења, тј интеграције претходних понашања система и корисника на основу анализе стохастичких података и мониторинга претходних процеса конфигурације.

8. Литература

[1] M. Josuttis, SOA In Practice The Art of Distributd Syste, Design, Sebastopol, CA.: O'Reilly Media, Inc., 2007.

[2] W. T. Tsai, "Service-Oriented System Engineering: A New Paradigm," in *Proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering*, 2005.

[3] T. Erl, Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall PTR, 2005.

[4] N. Duane, L. Reitman, J. Ward and J. Wilber, "Service Oriented Architecture (SOA) and Specialized Messaging Patterns," 2007.

[5] T. Erl, SOA: Principles of Service Design, Prentice Hall, 2006.

[6] R. R. Kodali, "What is service-oriented architecture? An introduction to SOA," 2005.

[7] J. Martin, P. Tarr, A. Arsanjani and B. Hailpern, "Web Services: Promises and Compromises," *ACM-Queue,* vol. 1, no. 1, pp. 48-58, 2003.

[8] I. I. A. R. K. Lieberherr, "Object-Oriented Programming: An Objective Sense of Style," in *OOPSLA '88 Proceedings*, 1988.

[9] X. Chen, J. He, Z. Liu and N. Zhan, "Component-based programming," UNU-IIST Report No 350, 2007.

[10] H. Petritsch, "Service-Oriented Architecture (SOA) vs. Component Based Architecture," 2007.

[11] M. P. Papzoglou and W. J. den Heuvel, "Service oriented architecture: approaches, technologies and research issues," *VLDB Journal,* vol. 16, pp. 389-415, 2007.

[12] B. Mohabbati, D. Gašević, M. Hatala, M. Asadi, E. Bagheri and M. Bošković, "A quality aggregation model for service-oriented software product lines based on variability and composition patterns," in *Proceedings of the 10th International Conference on Service Oriented Computing*, 2011.

[13] D. T. Sanders, J. A. Hamilton and R. A. MacDonald, "Supporting A Service-

Oriented Archietcture," in *Proceedings of the 2008 Spring simulation multiconference*, 2008.

[14] M. Asadi, B. Mohabbati, N. Kaviani, D. Gašević, M. Bošković and M. Hatala, "Model-Driven Development of Families of Service-oriented Architecture," in *In Proceedings of the 1st International Workshop on Feature-Oriented Software Development*, USA, 2009.

[15] W. Van der Aalst and K. Van Hee, Workflow Management Models, Methods and Systems (Cooperative Information Systems series), The MIT Press, 2004.

[16] J. Cardoso, A. P. Sheth, J. A. Miller, J. Arnold and K. Kochut, "Quality of Service for Workflows and Web Service Processes," *Journal of Web Semantics*, vol. 1, no. 3, pp. 281-308, 2004.

[17] Q. Chen, U. Dayal, M. Hsu and M. L. Griss, "Dynamic-Agents, Workflow and XML for E-Commerce Automation," in *Electronic Commerce and Web Technologies*, Springer Berlin Heidelberg, 2000, pp. 314-323.

[18] G. Shegalov, M. Gillmann and G. Weikum, "XML-enabled workflow management for e-services across heterogeneous platforms," *The VLDB Journal*, vol. 10, no. 1, pp. 91-103, 2001.

[19] D. Riehle and H. Zullighoven, "Understanding and using patterns in software development," *Theory and Practice of Object Systems*, pp. 3-13, 1996.

[20] P. Wohed, W. M. P. van der Aals, M. Dumas, A. H. M. ter Hofstede and N. Russell, "Pattern-based Analysis of BPMN- an extensive evaluation of the Control-flow,the Data and the Resource Perspectives," *Swedish Research Council, "Analysis Patterns - An Approach for Flexible and Reusable Information Systems",* 2006.

[21] W. Van der Aalst, A. ter Hofstede, B. Kiepuszewski and A. P. Barros, "Workflow patterns," *Distributed and Parallel Databases*, vol. 14, no. 3, pp. 5-51, 2003.

[22] M. Jaeger, G. Rojec-Goldmann and G. Muhl, "QoS Aggregation for Web Service Composition using Workflow Patterns," in *Proceedings of the 8th IEEE Intl Enterprise Distributed Object Computing Conference*, 2004.

[23] Y. Yang, L. Garcia-Banuelos, A. Polyvyanyy, M. Dumas and L. Zhang,

"Aggregate Quality of Service Computation for Composite Services," in *Proceedings of the 6th International Conference on Collaborative Computing*, 2010.

[24] ISO9000, "ISO9000. International Organization for Standardization," 2002.

[25] G. Stalk and T. M. Hout, "Competing against time: how timebased competition is reshaping global markets.," Free Press, New York, 1990.

[26] G. Rommel and J. Kluqe, Simplicity Wins: How Germany's Mid-Sized Industrial Companies Succeed, Harvard Business Press, 1995.

[27] D. Garvin, Managing Quality: The Strategic and Competitive Edge, New York: Free Press, 1988.

[28] R. L. Cruz, "Quality of service guarantees in virtual circuit switched networks," *IEEE Journal on Selected Areas in Communications,* vol. 13, no. 6, pp. 1048-1056, 1995.

[29] L. Georgiadis, R. Guerin, V. Peris and K. Sivarajan, "Efficient Network QoS Provisioning Based on Per Node Traffic Shaping," *IEEE ACM Transactions on Networking,* vol. 4, no. 4, pp. 482-501, 1996.

[30] D. Clark, S. Shenker and L. Zhang, "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism," in *Proceedings of ACM SIGCOMM*, 1992.

[31] J. Zinky, D. Bakken and R. Schantz, "Architectural Support for Quality of Service for CORBA Objects," *Theory and Practice of Object Systems,* vol. 3, no. 1, pp. 1-20, 1997.

[32] M. A. Hiltunen, R. D. Schlichting, C. A. Ugarte and G. T. Wong, "Survivability through Customization and Adaptability: The Cactus Approach," in *DARPA Information Survivability Conference and Exposition (DISCEX 2000).*, 2000.

[33] S. Frolund and J. Koistinen, "Quality-of-Service Specification in Distributed Object Systems," *Distributed Systems Engineering Journal,* vol. 5, no. 4, pp. 179-202, 1998.

[34] K. Nahrstedt and J. M. Smith, "Design, Implementation and Experiences of the OMEGA End-point Architecture," *IEEE Journal on Selected Areas in*

Communications, vol. 14, no. 7, pp. 1263-1279, 1996.

[35] ISO9126, "International Standard ISO/IEC 9126: Information Technology-Software Product Evaluation," 1991.

[36] L. I. Fei, H. E. Yanxiang, H. U. Wensheng, W. U. Libing and W. E. N. Peng, "Web Service Selection Based on Fuzzy QoS Attributes," *Journal of Computational Information Systems*, vol. 7, no. 1, pp. 198-205, 2011.

[37] M. Hilari, "Quality of Service (QoS) in SOA Systems. A Systematic Review," Universitat Politecnica de Catalunya. Tesi de Master., 2009.

[38] J. Cardoso, "Stohastic Workflow Reduction Algorithm," LSDIS Lab, Department of Computer Science, University of Georgia, 2002.

[39] W. M. P. Aalst, "Generic Workflow Models: How to Handle Dynamic Change and Capture Management Information," in *Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems (CoopIS'99)*, Edinburgh, Scotland, 1999.

[40] B. Kiepuszewski, A. Hofstede and C. Bussler, "On structured workflow modelling," in *Proceedings of the 12th Int. Conf. on Advanced Information Systems Engineering*, 2000.

[41] M. Dumas, L. Garcia-Banuelos, A. Polyvyanyy, Y. Yang and L. Zhang, "Aggregate quality of service computation for composite services," in *Proceedings of ICSOC*, 2010.

[42] C. Ouyang and e. al, "From business process models to process-oriented software systems," *ACM Transactions on Software Engineering Methodology*, vol. 19, no. 2, pp. 1-37, 2009.

[43] K. Pohl, G. Bockle and F. J. van der Linden, Software Product Line Engineering: Foundations, Principles and Techniques, Heidelberg: Springer, 2005.

[44] J. McGregor, D. Muthig, K. Yoshimura and P. Jensen, "Successful Software Product Line Practices," *IEEE Software*, vol. 27, no. 3, p. 16–21, 2010.

[45] H. Gomaa, "Advances in Software Design Methods for Concurrent, Real-Time and Distributed Applications," in *Proceedings of the 3rd Int. Conf. on Software Engineering Advances*, 2008.

[46] I. Sommerville and P. Sawyer, Requirements engineering, London: Wiley , 1997.

[47] S. G. Cohen and R. Krut, "Managing variation in services in a software product line context," Technical Report SEI-2010-TN-007, CMU, 2010.

[48] R. W. Krut and S. G. Cohen, 3rd workshop on service-oriented architectures and software product lines: Enhancing variation, CMU, 2009.

[49] J. Lee, D. Muthig and M. Naab, "A feature-oriented approach for developing reusable product line assets of service-based systems," *Journal of Systems and Software,* vol. 83, no. 7, pp. 1123-1136, 2010.

[50] A. Helferich, G. Herzwurm, S. Jesse and M. Mikusz, "Software product lines, service-oriented architecture and frameworks: worlds apart or ideal partners?," in *Trends in Enterprise Application Architecture, 2nd Conf on,* 2007.

[51] B. Mohabbati, M. Hatala, D. Gašević, M. Asadi and M. Bošković, "Development and Configuration of Service-Oriented Systems Families," in *Proceedings of the 26th Symposium On Applied Computing (SAC'11),* Taiwan, 2011.

[52] K. Czarnecki and U. Eisenecker, Generative Programming: Methods, Tools, and Applications., New York: ACM Press, 2000.

[53] K. Lee, K. Kang and J. Lee, "Concepts and guidelines of feature modeling for product line software engineering," in *Lecture Notes in Computer Science*, 2002, p. 62–77.

[54] K. Czarnecki, . S. Helsen and U. Eisenecker, "Formalizing cardinality-based feature models and their specialization," *Software Process: Improvement and Practice,* vol. 10, no. 1, p. 7–29, 2005.

[55] M. Babar, L. Chen and F. Shull, "Managing variability in software product lines," *IEEE Software,* vol. 27, no. 3, p. 89–91, 2010.

[56] K. Kang, S. Cohen, J. Hess, W. Novak, A. Peterson and C.-M. U. P. P. S. E. INST, Feature-oriented domain analysis (FODA) feasibility study, Pittsburgh: PA: Carnegie Mellon University, Software Engineering Institute, 1990.

[57] E. Bagheri and D. Gašević, "Assessing the Maintainability of Software Product Line Feature Models using Structural Metrics," *Software Quality Journal,* vol. 19, pp. 579-612, 2010.

[58] H. Mukhtar, D. Belaïd and G. Bernard, “A quantitative model for user preferences based on qualitative specifications,” in *Proceedings of International Conference on Pervasive services*, New York, 2009.

[59] K. Czarnecki and M. Antkiewicz, “Mapping features to models: A template approach based on superimposed variants,” in *Proceedings of the 4th international conference on Generative Programming and Component Engineering*, 2005.

[60] G. Gröner, C. Wende, M. Bošković, F. Silva Parreiras, T. Walter, F. Heidenreich, D. Gaševиц and S. Staab, “Validation of Families of Business Processes,” in *Advanced Information Systems Engineering: Lecture Notes in Computer Science*, vol. 6741, Heidelberg, Springer Berlin, 2011, pp. 551-565.

[61] A. Aurum and C. Wohlin, Engineering and Managing Software Requirements, Berlin Heidelberg: Springer, 2005.

[62] IEEE-STD-610.12, “Standard Glossary of Software Engineering Technology,” Instutute of Electrical and Electronics Engineers, 1990.

[63] A. Davis, “System testing: Implications of requirements specifications,” *Information and Software Technology,* vol. 32, no. 6, pp. 407-414, 1990.

[64] M. Glinz, “On Non-Functional Requirements,” in *Proceedings of the 15th IEEE International Requirements Engineering Conference*, 2007.

[65] S. Robertson and J. Robertson, Mastering the Requirements Process, Addison-Wesley Professional, 1999.

[66] I. Sommerville, Software Engineering, 7th Edition, Pearson Education, 2004.

[67] A. Anton, Goal Identification and REfinment in the Specification of Information Systems, PhD Thesis, Georgia Institute of Technology, 1997.

[68] I. Jacobson, G. Booch and J. Rumbaugh, The Unified Software Development Process, Addison Wesley, 1999.

[69] G. Kotonya and I. Sommerville, Requirements engineering - Processes and techniques., UK: Wiley and Sons, 1998.

[70] A. Davis, Software Requirements: Objects, Functions and States, Prentice Hall, 1993.

[71] IEEE-STD-830-1998, “IEEE Recommended Practice for Software Requirements Specifications,” IEEE, 1998.

[72] J. Mylopoulos, L. Chung and B. Nixon, “Representing and Using Nonfunctional Requirements: A Process Oriented Approach,” *IEEE Transactions on Software Engineering,* vol. 18, pp. 483-497, 1992.

[73] C. Ncube, “A Requirements Engineering Method for COTS-Based Systems Development, PhD Thesis,” City University London, 2000.

[74] K. Wiegers, Software Requirements, 2nd edition., Microsoft Press, 2003.

[75] M. Glinz, “Rethinking the Notion of Non-Functional Requirements,” in *Proceedings of the 3rd World Congress for Software Quality,* Munich, Germany, 2005.

[76] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos and D. Poole, “CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements,” *Journal of Artificial Intelligence Research,* vol. 21, no. 1, pp. 135-191, 2004.

[77] T. L. Saaty, “The Analytic Hierarchy Process,” *McGraw-Hill, New York,* 1980.

[78] Z. Turski, “Multi-attribute contractors ranking method by applying ordering of feasible alternatives of solutions in terms of preferability technique,” *Technological and Economic Development of Economy,* vol. 14, no. 2, pp. 224-239, 2008.

[79] C. Domshlak, E. Hüllermeier, S. Kaci and H. Prade, “Preferences in AI: An overview,” *Artificial Intelligence, Elsevier,* vol. 175, no. 7-8, pp. 1037-1052, 2011.

[80] L. Chung and J. S. d. P. Leite, “On Non-Functional Requirements in Software Engineering,” in *CONCEPTUAL MODELING: FOUNDATIONS AND APPLICATIONS, Lecture Notes in Computer Science,* Springer, 2009, pp. 363-379.

[81] D. Berry and B. Lawrence, “Requirement Engineerng,” *IEEE Software,* vol. 25, no. 2, pp. 26-29, 1998.

[82] J. Goguen and C. Linde, “Techniques for Requirements Elicitation,” in

Proceedings of IEEE International Symposium on Requirment Engineering, 1993.

[83] B. Curtis, H. Krasner and N. Iscoe, "A field study of the software design process for large systems," *Communications of the ACM,* vol. 31, no. 11, pp. 1268-1287, 1988.

[84] B. Nuseibeh and S. Easterbrook, "Requirements Engineering: A Roadmap," in *Future of Sofware Engineering Limerick Ireland*, 2000.

[85] M. Christel, D. Wood and S. Stevens, "Applying Multimedia Technology to Requirements Engineering," in *Proceedings of the 6th Annual Software Technology Conference,* Salt Lake City, 1994.

[86] J. Leite and A. Gilvaz, "Requirements Elicitation Driven by Interviews: The Use of Viewpoints," in *Proceedings of the 8th Int.Workshop on Software Specification & Design,* 1996.

[87] A. Davis, "Operational Prototyping: A New Development Approach," *Software,* vol. 9, no. 5, pp. 70-78, 1992.

[88] M. Mannio and U. Nikula, "Requirements Elicitation Using a Combination of Prototypes and Scenarios," in *[available online] http://www.inf.puc-rio.br/wer/WERpapers/artigos/artigos_WER01/mannio.pdf,* 2001.

[89] R. M. Kimmond, "Survey into the acceptance of prototyping in software development," in *Proceedings from the IEEE 6th International Workshop on Rapid System Prototyping*, 1995.

[90] v. L. A., R. Darimont and E. Letier, "Managing conflicts in goal-driven requirements engineering," *IEEE Transactions on Software Engineering,* vol. 24, no. 11, pp. 908-926, 1998.

[91] N. Maiden, "CREWS-SAVRE: Scenarios for Acquiring and Validating Requirements," *Automated Software Engineering,* vol. 5, no. 4, pp. 419-446, 1998.

[92] M. Shaw and B. Gaines, "Requirements Acquisition," *Software Engineering Journal*, vol. 11, no. 9, pp. 149-165, 1996.

[93] R. I. Brafman and C. Domshlak, "Introducing variable importance tradeoffs into CP-nets," in *In The Proceedings of the Eighteenth Conference on Uncertainty in*

AI, Canada, 2002.

[94] P. Berander and A. Andrews, "Requirements Prioritization," *Engineering and Managing Software Requirements,* pp. 69-94, 2006.

[95] Z. Yu, Z. Yu, X. Zhou and Y. Nakamu, "Toward an Understanding of User-Defined Conditional Preferences," in *In Proceedings of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing*, Washinghton, USA, 2010.

[96] R. Keeny and H. Raiffa, Decisions with Multiple Objectives: Preferences and Value Tradeoffs, Cambridge: University Press, 1993.

[97] W. C. Stirling, R. Frost, M. Nokleby and Y. Luo, "Multicriterion Decision Making with Dependent Preferences," in *IEEE Symposium on Computational Intelligence in Multicriteria Decision Making*, 2007.

[98] C. Domshlak, "A Snapshot on Reasoning with Qualitative Preference Statements in AI," in *Preferences and similarities, CISM International Centre for Mechanical Sciences*, vol. 504, Springer Vienna, 2008, pp. 265-282.

[99] J. Goldsmith, J. Lang, M. Truszczynski and N. Wilson, "The computational complexity of dominance and consistency in CP-nets," *Journal of AI Research,* pp. 403-432, 2008.

[100] T. Satty, "The Analytic Hierarchy Process," *McGraw-Hill, New York,* 1980.

[101] G. C. Roper-Low, "The Analytic Hierarchy Process and its application to an information technology," *The Journal of Operational Research,* vol. 41, no. 1, pp. 49-59, 1990.

[102] G. Büyüközkan, G. Çifçi and S. Güleryüz , "Strategic analysis of healthcare service quality using fuzzy AHP methodology," *Expert Systems with Applications,* vol. 38, no. 8, p. 9407–9424, 2011.

[103] M. K. Chen and S. Wang, "The critical factors of success for information service industry in developing international market: Using analytic hierarchy process (AHP) approach," *Expert Systems with Applications,* vol. 37, no. 1, p. 694–704, 2010.

[104] E. H. Forman and S. I. Gass, "The analytical hierarchy process—an exposition,"

Operations Research, vol. 49, no. 4, pp. 469-487, 2001.

[105] E. Bagheri, M. Asadi, D. Gašević and S. Soltani, "Stratified analytic hierarchy process: prioritization and selection of software features," in *In Proceedings of the 14th International conference on Software product lines,* 2010.

[106] E. K. Zavadskas, A. Kaklauskas, F. Peldschus and Z. Turskis, "Multi-attribute assessment of road design solutions by using the COPRAS Method," *The Baltic Journal of Road and Bridge Engineering,* vol. 2, no. 4, pp. 195-203, 2007.

[107] K. R. MacCrimmon, "Decision-making among multiple-attribute alternatives: A survey and consolidated approach," in *Memorandum RM-4823-ARPA, The Rand Corporation,* Santa Monica, California, 1968.

[108] E. K. Zavadskas, Z. Turskis, T. Dejus and M. Viteikiene, "Sensitivity analysis of a simple additive weight method," *International Journal of Management and Decision Making,* vol. 8, no. 5-6, pp. 555-574, 2007.

[109] V. Srinivasan and A. D. Shocker, "Linear programming techniques for multidimensional analysis of privileged," *Psychometrika,* vol. 38, pp. 337-369, 1973.

[110] E. K. Zavadskas and A. Kaklauskas, "Determination of an efficient contractor by using the new method of multicriteria assessment," in *International Symposium for "The Organisation and Management of Construction"*, 1996.

[111] J. Figueira, V. Mousseau and B. Roy, "Electre methods, Multiple criteria decision analysis: State of the art surveys," *International Series in Operations Research & Management Science,* vol. 78, no. 3, pp. 133-153, 2005.

[112] I. Ognjanović, D. Gašević and E. Bagheri, "A Stratified Framework for Handling Conditional Preferences: an Extension of the Analytic Hierarchy Process," *Expert Systems with Applications,* in press.

[113] N. Wilson, "Computational techniques for a simple theory of conditional preferences," *Artificial Intelligence,* vol. 175, no. 7-8, pp. 1053-1091, 2011.

[114] C. Boutilier, R. I. Brafman, C. Domshlak and H. Holger , "CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements," *Journal of Artificial Intelligence Research,* vol. 21, no. 1, pp. 135-

191, 2004.

[115] M. McGeachie and J. Doyle, "The local geometry of multiattribute tradeoff preferences," *Artificial Intelligence,* vol. 175, no. 7-8, pp. 1122-1152, 2011.

[116] C. Boutilier, F. Bacchus and R. Brafman, "UCP-Networks: A Directed Graphical Representation of Conditional Utilities," in *In Proceedings of 17th Conference in Uncertainty in AI*, San Francisco, USA, 2001.

[117] S. Chen, S. Buffett and M. Fleming, "Reasoning with Conditional Preferences across Attributes," in *Proceedings of the 20th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in AI*, 2007.

[118] P. Berander and P. Jönsson, "Hierarchical Cumulative Voting (HCV) – Prioritization of Requirements in Hierarchies," *International Journal of Software Engineering & Knowledge Engineering,* vol. 16, pp. 819-849, 2006.

[119] J. Karlsson, S. Olsson and K. Ryan, "Improving Practical Support for Large-scale Requirements Prioritising," *Requirement Engineering,* vol. 2, no. 1, pp. 51-60, 1997.

[120] C. Domshlak and R. Brafman, "CP-nets- Reasoning and Consistency testing," in *In Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning*, France, 2002.

[121] C. Gonzales and P. Perny, "GAI networks for utility elicitation," in *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning*, Whistler, Canada, 2004.

[122] R. Brafman, C. Domshlak and T. Kogan, "Compact value-function representations for qualitative preferences," in *In Proceedings of the 20th Annual Conference on Uncertainty in AI*, Canada, 2004.

[123] A. G. Dahlstedt and A. Persson, "Requirements interdependencies - Molding the state of research into a research agenda," in *Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality*, Austria, 2003.

[124] K. Schmid, "A comprehensive product line scoping approach and its validation," in *Proceedings of 24th International conference on software engineering*

ICSE'02, USA, 2002.

[125] Å. Dahlstedt and A. Persson, "Requirements Interdependencies: State of the Art and Future Challenges," in *Engineering and Managing Software Requiremnts*, Springer, 2005, pp. 95-113.

[126] M. Earl and B. Khan, "E-commerce is changing the face of IT," MIT Sloan management Review, 2001.

[127] R. Dömges and K. Pohl, "Adapting traceability environment to project-specific needs.," Communication of the ACM, 1998.

[128] O. Gotel, "Contribution structures for requirements traceability, PhD Thesis," Department of Computing Imperial College of Science,Technology and Medicine, University of London, 1995.

[129] P. Grünbacher and N. Seyff, "Requirements Negotation," in *Engineering and Managing Software Requirements*, Springer, 2005, pp. 143-158.

[130] H. In and D. Olson, "Requirements Negotiation Using Multi-Criteria Preference Analysis," *Journal of Universal Computer Science*, vol. 10, no. 4, 2004.

[131] B. Boehm, P. Bose, E. Horowitz and M. J. Lee, "Software Requirements as Negotiated Win Conditions," in *Proceedings of the 1st International Conference on Requirements Engineering*, 1994.

[132] B. Boehm, P. Bose, E. Horowitz and M. J. Lee, "Software Requirements Negotiation and Renegotiation Aids: A Theory-W Based Spiral Approach," in *Proceedings of ICSE'95*, IEEE Computer Society Press, Seattle, 1995.

[133] B. Boehm and R. Ross, "Theory W Software Project Management:Principles and Examples," *IEEE Transactions on Software Engineering*, vol. 15, no. 7, pp. 902-916, 1989.

[134] C. Denger and T. Olsson, "Quality Assurance in Requirements Engineering," in *Engineering and Managing Software Requirements*, Springer, 2005, pp. 163-182.

[135] J. Guo, J. White, G. Wang, J. Li and Y. Wang, "A genetic algorithm for optimized feature selection with resource constraints in software product lines," *Journal on Systems and Software*, vol. 84, p. 2208–2221, 2011.

[136] A. Gao, D. Yang, S. Tang and M. Zhang, "QoS-Driven Web Service Composition

with Inter Service Conflicts," in *Frontiers of WWW Research and Development - APWeb 2006*, Springer, 2006, pp. 121-132.

[137] G. Canfora, M. Di Penta, R. Esposito and M. L. Villani, "An approach for qos-aware service composition based on genetic algorithms," in *Proceedings of Conference on Genetic and Evolutionary Computation*, 2005.

[138] W. M. P. van der Aalst, A. Dreiling, F. Gottschalk, M. Rosemann and M. H. Jansen-Vullers, "Configurable process models as a basis for reference modeling," in *Business Process Management Workshops*, 2005.

[139] M. Godse, R. Sonar and S. Mulik, "The Analytical Hierarchy Process Approach for Prioritizing Features in the Selection of Web Service," in *Proceedings of the 6th European Conference on Web Services*, 2008.

[140] A. Srivastava and P. G. Sorenson, "Service Selection based on Customer Rating of Quality of Service Attributes," in *2010 IEEE International Conference on Web Services*, 2010.

[141] R. Guha, R. McCool and E. Miller, "Semantic search," in *Proceedings of the 12th international conference on World Wide Web (WWW2003)*, 2003.

[142] V. Ramanujam, "Semantic Technologies in Integration and SOA," *The SOA magazine: Service Orientation, the Cloud and Beyond*, 2010.

[143] A. Newell, "The Knowledge Level," *Artificial Intelligence*, vol. 18, pp. 87-127, 1982.

[144] S. Grimm, P. Hitzler and A. Abecker, "Knowledge Representation and Ontologies," [available online] http://knoesis.org/pascal/resources/publications/kr-onto-07.pdf, 2007.

[145] K. C. M. T. W. Breitman, Semantic Web: Concepts, Technologies and Applications, London, UK.: Springer-Verlag, 2007.

[146] R. T. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, pp. 199-220, 1993.

[147] R. B. R. F. D. Studer, "Knowledge engineering: principles and methods," *IEEE Transactions and Data on Knowledge Engineering*, vol. 25, no. 1-2, p. 161–197., 1998.

[148] T. Gruber, “A Translation Approach to Portable Ontology Specifications,” *Knowledge Acquisitions,* vol. 6, no. 2, pp. 199-221, 1993.

[149] N. Guarino, “Formal Ontology and Information Systems,” in *Proceedings of the 1st International Conference on Formal Ontologies in Information Systems*, Trento, Italy, 1998.

[150] B. Chandrasekaran, J. Josepson and V. Benjamins, “What are ontologies, adn Why do we need them?,” *IEEE Intelligent Systems,* vol. 14, no. 1, pp. 20-26, 1999.

[151] H. F. R. N. N. M. M. Knublauch, “The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications,” in *International Semantic Web Conference 2004*, Hiroshima, Japan, 2004.

[152] J. Lloyd, Foundations of Logic Programming, Springer-Verlag, 1988.

[153] J. Minker, “Logic and Databases: Past, Present, and Future,” *Artificial Intelligence Magazine,* vol. 18, no. 3, pp. 21-47, 1997.

[154] J. Freeman-Hargis, “Rule-Based Systems and Identification Trees,” in *Introduction to Rule-Based Systems*, [available online] http://ai-depot.com/Tutorial/RuleBased.html, 2002.

[155] N. Chaudhari and A. Ghosh, “Feature Extraction using Fuzzy Rulebased system,” *International Journal of Computer Science and Applications*, vol. 5, no. 3, pp. 1-8, 2007.

[156] I. Ognjanović, D. Gašević, E. Bagheri and M. Asadi, “Conditional Preferences in Software Stakeholders' Judgments,” in *Proceedings of the 26th Annual ACM Symposium on Applied Computing*, Taichang, Taiwan, 2011.

[157] B. Schneier, “The Rete Matching Algorithm,” [available online] http://drdobbs.com/architecture-and-design/184405218, 2002.

[158] P. Lewis, “Knowledge Represetation,” in *Current Trends in Information Technology*, [available online] http://users.ecs.soton.ac.uk/phl/ctit/ho1/node3.html, 2003.

[159] G. Zielke, “Some remarks on matrix norms, condition numbers, and error estimates for linear equations,” *Linear Algebra Applications,* vol. 110, pp. 29-41,

1988.

[160] S. Martello and P. Toth, “Algorithms for Knapsack problems,” *Annals of Discrete Methematics,* vol. 31, pp. 70-79, 1987.

[161] S. Khan, “Quality Adaptation in a Multisession Multimedia System: Model, Algorithms and Architecture,” PhD thesis, University of Victoria, Canada, 1998.

[162] S. Khan, K. F. Li, E. G. Manning and M. M. Akbar, “Solving the knapsack problem for adaptive multimedia systems,” *Studia Informatica Universalis,* pp. 157-178, 2002.

[163] T. Yu and K. J. Lin, “Service Selection Algorithms for Composing Complex Servcies with Multiple QoS Constarints,” *Information Systems and e-business Management,* pp. 103-126, 2009.

[164] T. Yu, Y. Zhang and K.-J. Lin, “Efficinet Algorithms for Web Services Selection with End-to-End QoS Constraints,” *ACM Transactions on the Web,* 2007.

[165] H.-L. Fang and D. I. N. Bu, “Genetic Algorithms in Timetabling and Scheduling,” *PhD thesis, University of Edimburg,,* 1994.

[166] L. Jian-hua and e. al, “Application of genetic algorithm to QoS-aware Web Services composition,” in *Industrial Electronics and Applications, ICIEA 2008.*, 2008.

[167] C. Lopez-Pujalte, V. P. Guerrero-Bote and F. de Moya-Anegón, “Order-Based Fitness Functions for Genetic Algorithms Applied to Relevance Feedback,” *Journal of the American Soceity for Information Science and Technology,* vol. 54, no. 2, pp. 152-160, 2003.

[168] V. Petridis, S. Kazarlis and A. Bakirtzis, “Varying Fitness Functions in Genetic Algorithm Constrained Optimization: The Cutting STock and Unit Commitment Problems,” *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics,* pp. 629-640, 1998.

[169] C. Coello, “Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems,” *Computers in Industry,* vol. 41, no. 2, pp. 113-127, 2000.

[170] S. Bajaj, D. Box, D. Chappell, F. Curbera, G. Daniels, P. Hallam-Baker, M.

Hondo, C. Kaler, D. Langworthy, A. Nadalin, N. Nagaratnam, H. Prafullchandra, C. von Riegen, D. Roth, J. Schlimmer , C. Sharp, J. Shewchuk, A. Vedamuthu, U. Yalçinalp and D. Orchard, "Web Services Policy Framework," http://www.w3.org/Submission/WS-Policy/, 2006.

[171] S. Frolund and J. Koisten, "QML: A Language forQuality of Service Specification," http://www.hpl.hp.com/techreports/98/HPL-98-10.html, 1998.

[172] H. Ludwig, A. Keller, A. Dan, R. P. King and R. Franck, "Web Service Level Agreement (WSLA) Language Specification," http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf, 2003.

[173] V. Tošić, "Service Offerings for XML Web Services and Their Management Applications, PhD thesis," Department of Systems and Computer Engineering, Carleton University, Canada, 2004.

[174] J. O'Sullivan, D. Edmond and A. H. M. ter Hofstede, "Formal description of non-functional service properties.," Business Process Management Group, Centre for Information Technology Innovation, Australia, 2005.

[175] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M. T. Schmidt, A. Sheth and K. Verma, "Web Service Semantics," http://lsdis.cs.uga.edu /library/download/WSDL-S-V1.html, 2005.

[176] C. Zhou, L. T. Chia and B. S. Lee, "Web services discovery with DAML-QoS ontology," *International Journal of Web Services Research,* vol. 2, no. 2, p. 44–67, 2005.

[177] H. Jin and H. Wu, "Semantic-enabled specification for Web Services agreement," *International Journal of Web Services Practices,* vol. 1, pp. 13-20, 2005.

[178] L. H. Vu, F. Porto, K. Aberer and M. Hauswirth, "An Extensible and Personalized Approach to QoS-enabled Service Discovery," LSIR, 2006.

[179] X. Wang, T. Vitvar, M. Kerrigan and I. Toma, "A QoS-aware selection model for semantic web services," in *Proceedings of the 4th international conference on Service-Oriented Computing*, 2006.

[180] I. Ognjanović, B. Mohabbati, D. Gašević, E. Bagheri and M. Bošković, "A Metaheuristic Approach for the Configuration of Business Process Families," in

IEEE International Conference on service Computing (SCC2012), Hawaii, USA, 2012.

[181] C. Gao, M. Cai and H. Chen, "Qos-aware service composition based on tree-coded genetic algorithm," in *Proceedings of the 31st Annual International Computer Software and Applications Conference*, 2007.

[182] M. Jaeger and G. Muhl, "QoS-based selection of services: The implementation of a genetic algorithm," in *KiVS 2007Workshop: Service-Oriented Architectures und Service-Oriented Computing*, 2007.

[183] T. Yu, "Quality of Servcies (QoS) in web servcies: Model, architecture and algorithms, PhD Thesis," University of California, CA, 2006.

[184] S. Thiel and A. Hein, "Systematic integration of variability into product line architecture design," in *Proceedings of the 2nd International Conference on Software Product Lines*, 2002.

[185] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam and H. Chang, "QoS-aware Middleware for Web Services Composition," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp. 311-327, 2004.

[186] I. Stojmenović, "A simple systolic algorithm for generating combinations in lexicographic order," *Computers & Mathematics with Applications,* vol. 24, no. 4, pp. 61-64, 1992.

[187] A. Glaser, History of binary and other non-decimal numeration, Tomash Publishers, 1981.

[188] E. Bagheri, T. D. Noia, D. Gašević and A. Ragone, "Formalizing interactive staged feature model configuration," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 24, no. 4, pp. 375-400, 2011.

[189] D. Ardagna and B. Pernici, "Global and Local QoS Guarantee in Web Service Selection," in *BPM 2005 Workshops*, 2006.

[190] J. Aguarón and J. M. Moreno-Jiménez, "The geometric consistency index: Approximated thresholds," *European Journal of Operational Research*, vol. 147, no. 1, p. 137–145, 2003.

[191] E. Bulut, O. Duru, T. Keçeci and C. Yoshida, "Use of consistency index, expert

prioritization and direct numerical inputs for generic fuzzy-AHP modeling: A process model for shipping asset management," *Expert Systems with Applications,* vol. 39, no. 2, pp. 1911-1923, 2012.

[192] W. J. Xu, Y. C. Dong and W. L. Xiao, "Is It Reasonable for Saaty's Consistency Test in the Pairwise Comparison Method?," in *ISECS International Colloquium onComputing, Communication, Control, and Management.*, 2008.

[193] S.-P. Ma, J.-Y. Kuo, Y.-Y. Fanjang, C.-P. Tung and C.-Y. Huang, "Optimal Service Selection for Composition Based on Weighted Service Flow adn Genetic Algorithm," in *Proceedings of the 9th International Conference on Machine Learning and Cybernetics*, Qingdao, 2010.

[194] M. Srinivas and L. Patnaik, "Genetic algorithms: a survey," *Computer,* vol. 27, no. 6, pp. 17-26, 1994.

[195] U. Bhowan, M. Johnston and M. Zhang, "Developing new fitness functions in genetic programming for classification with unbalanced data," *IEEE Transactions on Systems, Man, and Cybernetics, Part B,* vol. 42, no. 2, pp. 406- 421 , 2012.

[196] W. Fan, E. A. Fox, P. Pathak and H. Wu, "The effects of fitness functions on genetic programming-based ranking discovery for web search: Research articles," *Journal of the American Society for Information Science and Technology,* vol. 55, pp. 628-636, 2004.

[197] M. H. Ghiasi, B. Dehghan-Manshadi and A. Abedian, "Effects of a linear-exponential penalty function on the gas efficiency in optimization of a laminated composite panel," *International Journal of Computational Intelligent,* vol. 2, pp. 5-11, 2005.

[198] A. B. Hadj-Alouane and J. C. Bean, "A Genetic algotihm for the multiple-choice integer program," *Operations Research,* vol. 45, pp. 92-101, 1997.

[199] N. Blaikie, Analyzing Quantitative Data, London: Sage, 2003.

[200] K. M. Khaled, Managing Web Service Quality: Measuring Outcomes and Effectiveness, New York: Information Science Reference, 2009.

[201] M. Hardy, Regression with Dummy Variables (Quantitative Applications in the Social Sciences), 1 ed., Sage Publications Inc., 1993, p. 96.

[202] O. Keene, "The Log-Transformation is Special," *Statistics in Medicine,* vol. 14, no. 8, pp. 811-819, 1995.

[203] J. L. Rasmussen and W. P. Dunlap, "Dealing with non-normal data: Parametric analysis of transformed data vs nonparametric analysis," *Educational and Psychological Measurement,* vol. 51, no. 4, pp. 809-820, 1991.

[204] J. Cardoso, A. P. Sheth, J. A. Miller, J. Arnold and K. J. Kochut, "Modeling quality of service for workflows and web service processes," *Web Semantics Journal: Science, Services and Agents on the World Wide Web Journal,* vol. 1, no. 3, p. 281–308, 2004.

[205] G. Canfora, M. Di Penta, R. Esposito, F. Perfetto and M. L. Villani, "Services composition (re)-binding driven by application-specific QoS," in *Proceedings of International Conference on Service Oriented Computing (ICSOC'06)*, Chacago, 2006.

Додатак А

Доказ (Лема 1):

На основу дефиниције $r_D()$као ондекса одговарајуће инстанце у неопадајуће уређеном низу инстанци датог обележја са доминантним релативним приоритетом, важи низ неједнакости:

$$\sum_{l=u}^{k} b_u r_D(qt_{j_l}^{i_l}) + r(\{qt_{j_1}^{i_1},\ldots,qt_{j_s}^{i_s}\}\setminus\{qt_{j_1}^{i_1},\ldots,qt_{j_k}^{i_k}\}) \le \sum_{l=u}^{k}\left(\prod_{j=l+1}^{k}\left(|QT_{i_j}|+1\right)\right)|QT_{i_l}|+1 = \prod_{j=u}^{k}\left(|QT_{i_j}|+1\right)$$

,

где се последња једнакост једноставно доказује математичком индукцијом. ■

Доказ (Пропозиција 1):

Доказ овог тврђења се изводи у два смера. У првом смеру, да би се доказало да, уколико опције o_1 и o_2 задовољавају један од услова *(c1)-(c3)* из Дефиниције 7, да је тада $F(o_1)>F(o_2)$, користи се математичка индукција по броју обележја са доминантним релативним приоритетима.

За базни дио индукције, претпоставимо да је само једно обележје c_i са доминантним релативним приоритетом, а да две произвољне опције, o_1, означена са $\{qt_{j_1}^{i_1},\ldots,qt_{j_k}^{i_k}\}, k \le m$ и o_2, означена са $\{qt_{j_1}^{i_2},\ldots,qt_{j_l}^{i_2}\}, l \le m$ задовољавају услове *(c1)-(c3)* из Дефиниције 7.

Сада претпоставимоо да је услов *(c1)* испуњен, нпр. опција o_1 је означена инстанцом qt_j^i обележја c_i (тј. $r_D(qt_j^i) \ge 1$). Тада важи:

$$F(\{qt_{j_1}^{i_1},\ldots,qt_{j_k}^{i_1}\}) = r_D(qt_j^i) + r(\{qt_{j_1}^{i_1},\ldots,qt_{j_k}^{i_1}\}/\{qt_j^i\}) > 1 \ge r(qt_{j_1}^{i_2},\ldots,qt_{j_l}^{i_2}) = F(qt_{j_1}^{i_2},\ldots,qt_{j_l}^{i_2})$$

.

У случају услова *(c2)* где опције o_1 и o_2 нису означене инстанцама обележја c_i и опција o_1 је веће рангирана него o_2 на основу захтева о осталим обележјима, тачност неједнакости је очигледна. Такође, у случају услова *(c3.a)* у Дефиницији 7 где су опције o_1 и o_2 означене истом инстанцом qt_j^i обележја c_i, очигладна је тачност неједнакости акко важи $r(\{qt_{j_1}^{i_1},\ldots,qt_{j_k}^{i_1}\}\setminus\{qt_j^i\}) > r(\{qt_{j_1}^{i_2},\ldots,qt_{j_l}^{i_2}\}\setminus\{qt_j^i\})$, тј, акко је

опција o_1веће рангирана у односу на опцијуo_2на основу захтева о осталим обележјима.

Сада претпоставимо да је задовољен услов*(c3.b)* из Дефиниције 7, тј.опцијеo_1иo_2су означене редом са инстанцама $qt^{i}_{j_i^1}$ и $qt^{i}_{j_i^2}$ обележјаc_i, при чему је инстанца $qt^{i}_{j_i^1}$ већег ранга у односу на $qt^{i}_{j_i^2}$. Тада важи следеће:

$$F(qt^{i_1^1}_{j_1^1},\ldots,qt^{i_k^1}_{j_k^1}) = r_D(qt^{i}_{j_i^1}) + r(\{qt^{i_1^1}_{j_1^1},\ldots,qt^{i_k^1}_{j_k^1}\}\setminus\{qt^{i}_{j_i^1}\}) \geq r_D(qt^{i}_{j_i^2}) + 1 \geq r_D(qt^{i}_{j_i^2}) + r(\{qt^{i_1^2}_{j_1^2},\ldots,qt^{i_l^2}_{j_l^2}\}\setminus\{qt^{i}_{j_i^2}\})$$
$$= F(qt^{i_1^2}_{j_1^2},\ldots,qt^{i_l^2}_{j_l^2})$$

чиме је доказан базни део индукције.

Сада претпоставимо да тврђење важи за било који број обележја који је мањи од*k*, и докажимо да оно важи за тачно*k*обележја са доминантним релативним приоритетом $\overset{D}{\succ}(c_{i_1},\ldots,c_{i_k})$. На основу Дефиниције 8, релација $\overset{D}{\succ}(c_{i_1},\ldots,c_{i_k})$ важи акко $\overset{D}{\succ}(c_{i_1})$ и $\overset{D}{\succ}(c_{i_2},\ldots,c_{i_k})$.

Сада имамо:

$$F(qt^{i_1}_{j_1},\ldots,qt^{i_k}_{j_l}) = \sum_{l=1}^{k} b_l \cdot r_D(qt^{i_l}_{j_l}) + r(\{qt^{i_1}_{j_1},\ldots,qt^{i_k}_{j_l}\}\setminus\{qt^{i_1}_{j_1},\ldots,qt^{i_k}_{j_k}\})$$
$$= b_1 \cdot r_D(qt^{i_1}_{j_1}) + F(\{qt^{i_1}_{j_1},\ldots,qt^{i_k}_{j_l}\}\setminus\{qt^{i_1}_{j_1}\})$$

У случајевима да је било који од услова*(c1), (c2) ,(c3a)* из Дефиниције 7 задовољен, по аналогији се доказује тражена неједнакост.

На крају, претпоставимо да су опцијеo_1иo_2означене редом са инстанцама $qt^{i_1}_{j_1^1}$ и $qt^{i_1}_{j_1^2}$ обележја c_{i_1}, где је инстанца $qt^{i_1}_{j_i^1}$ већег ранга у односу на $qt^{i_1}_{j_i^2}$. Следећи низ неједнакости важи као директна последица претходно дефинисане Леме:

$$F(qt^{i_1}_{j_1},\ldots,qt^{i_k}_{j_k}) = b_1 r_D(qt^{i_1}_{j_1^1}) + F(\{qt^{i_1}_{j_1},\ldots,qt^{i_k}_{j_k}\}\setminus\{qt^{i_1}_{j_1^1}\}) \geq$$
$$b_1\left(r_D(qt^{i_1}_{j_1^2})+1\right) + F(\{qt^{i_1}_{j_1},\ldots,qt^{i_k}_{j_k}\}\setminus\{qt^{i_1}_{j_1^1}\}) > b_1 r_D(qt^{i_1}_{j_1^2}) + F(\{qt^{i_1}_{j_1},\ldots,qt^{i_l}_{j_l}\}\setminus\{qt^{i_1}_{j_1^2}\}) = F(qt^{i_1}_{j_1},\ldots,qt^{i_l}_{j_l})$$

чиме је тврђење доказано у једном смеру.

У супротном смеру, да би се доказало да из неједнакости *F(o1)>F(o2)*следи испуњење једног од услова*(c1)-(c3)*, користи се свођење на апсурд и претходно доказани смер. ■

Доказ (Пропозиција 5).

Ради једноставности посматрајмо случај само две нефункционалне карактеристике и разликујмо два случаја: *(i)*карактеристике су различитих монотоности; и*(ii)*обе карактеристике су исте монотоности (растуће или опадајуће); и конструишимо одговарајући модел за оба случаја.

(i) Посматрајмо монотоно растућу нефункционалну карактеристику са процењеним интервалом могућих вредности $[a_L, a_U]$и монотоно опадајућу нефункционалну карактеристику чији је процењени интервал могућих вредности $[b_L, b_U]$. Даље, означимо да n_1иn_2број покривајучих подинтервала редом за обе карактеристике. На основу услова (i) из Дефиниције 7,покривајући подинтервали су редом: $[a_L, a_L + \Delta a], [a_L + \Delta a, a_L + 2\Delta a], ..., [a_L + (n_1 - 1)\Delta a, a_U]$за прву нефункционалну карактеристику, и $[b_L, b_L + \Delta b], [b_L + \Delta b, b_L + 2\Delta b], ..., [b_L + (n_2 - 1)\Delta b, b_U]$за другу. Такође, нека су мере r^Cдефинисане са: $r^C : [a_L, a_U] \to [\alpha_1, \alpha_2], 0 \le \alpha_1 \le \alpha_2 \le 1$и $r^C : [b_L, b_U] \to [\beta_1, \beta_2], 0 \le \beta_2 \le \beta_1 \le 1.$

Према томе, на основу (iii) из Дефиниције 7, функција$r^S()$ је облика:

$$r_1^S(s_1,...,s_k) = r_1^C \cdot \left[q_1^{agg} \cdot \frac{\alpha_2 - \alpha_1}{a_U - a_L} + \left[\alpha_1 - a_L \cdot \frac{\alpha_2 - \alpha_1}{a_U - a_L} \right] \right]$$ и

$$r_2^S(s_1,...,s_k) = r_2^C \cdot \left[q_2^{agg} \cdot \frac{\beta_2 - \beta_1}{b_U - b_L} + \left[\beta_1 - b_L \cdot \frac{\beta_2 - \beta_1}{b_U - b_L} \right] \right],$$ где су $\langle q_1^{agg}, q_2^{agg} \rangle$ агрегиране вредности нефункционалних карактеристика за комбинацију сервиса $(s_1,..., s_k)$, а r_1^C и r_2^C су константне вредности над скупомнефункционалних карактеристика (тј. услов јединствености из Дефиниције7(ii)). Стога, укупна вредност задовољења нефункционалних захтева је

$$r^S(s_1,...,s_k) = \frac{1}{2}\left[r_1^C \cdot r_1^S(s_1,...,s_k) + r_2^C \cdot r_2^S(s_1,...,s_k) \right].$$

Даље, вредност мере $r^{QT}()$се може добити као одговарајућа вредност мере $r^{C}()$агрегираних вредности која одговара средини покривајућег подинтервала, тј. $r^{QT}(QT_i^1) = r_1^C \cdot \left[\left(a_L + \frac{2i-1}{2}\Delta a\right)\cdot\frac{\alpha_2-\alpha_1}{a_U-a_L} + \left[\alpha_1 - a_L \cdot \frac{\alpha_2-\alpha_1}{a_U-a_L}\right]\right]$, где је $a_L + (i+\frac{1}{2})\Delta a$ средња вредностi-тог покривајућег подинтервала $\left[b_L + (i-1)\Delta b, b_L + i\Delta b\right]$.

Низом једноставних трансформација, услов $r^{QT}(QT_{N_1}^1 \times QT_{M_1}^2) > r^{QT}(QT_{N_2}^1 \times QT_{M_2}^2)$ се може трансформисати у следећи облик: $r_1^C(\alpha_2-\alpha_1)(N_2-N_1) > r_2^C(\beta_2-\beta_1)(M_2-M_1)$. (**)

Да би доказали да је мера задовољења нефукнционалних захтева од стране сваке комбинације сервиса из покривајуће комбинације подинтервала $QT_{N_1}^1 \times QT_{M_1}^2$ већа од мере било које комбинације сервиса из $QT_{N_2}^1 \times QT_{M_2}^2$, посматраћемо најгори случај, тј. комбинацију сервиса из прве покривајуће комбинације подинтервала са најмањом мером задовољења нефункционалних захтева, и комбинацију сериса са највећим степеном задовољења која припада другој комбинацији покривајућих подинтервала.

Сходно монотоним тенденцијама обе карактеристике, најмања вредност мере задовољења нефункционалних захтева из комбинације покривања $QT_{N_1}^1 \times QT_{M_1}^2$ одговара агрегираној вредности: $\langle a_L + (N_1-1)\Delta a, b_L + M_1\Delta b\rangle \in R^2$, док највећа мера задовољења из покривања $QT_{N_2}^1 \times QT_{M_2}^2$ одговара агрегираним вредностима: $\langle a_L + N_2\Delta a, b_L + (M_2-1)\Delta b\rangle \in R^2$. Након једноставних рачунања, разлика међу степенима задовољења нефункционалних захтева дате две комбинације сервиса је:

$$\Delta r^S = r_1^C(\alpha_2-\alpha_1)(N_2-N_1) - r_2^C(\beta_2-\beta_1)(M_2-M_1) - r_1^C(\alpha_2-\alpha_1) + r_2^C(\beta_2-\beta_1)$$

На основу (**) и (***) цели израз има позитивну вредност, чиме је доказ завршен.

(ii) У случају нефункционалних карактеристика са истом тенденцијом монотоности, аналогним рачунањима се добија да дата неједнакост увек важи, без потребе за испуњењем додатног захтева дефинисаног у (***).

Доказ у вишедимензионалном случају се изводи по аналогији груписањем каракатеристика у две групе сходно тенденцијама монотоности.

■

Додатак Б

Табела24.*Очекиване вредности (Ср. вредност, Станд. Дев) броја неконзистенција у условно дефинисаним захтевима на нивоу обележја*

Број бележја / *Бројусловних парова обележја*	**1-3**	**4-6**	**7-10**	**11-15**	**16-21**	**22-28**	**29-36**	**37-45**
3	**0.503**, 0.386	-	-	-	-	-	-	-
4	**0.524**, 0.401	**1.199**, 0.819	-	-	-	-	-	-
5	**0.530**, 0.419	**1.268**, 0.829	**2.002**, 1.374	-	-	-	-	-
6	**0.543**, 0.44	**1.266**, 0.818	**2.140**, 1.373	**3.049**, 2.116	-	-	-	-
7	**0.572**, 0.508	**1.283**, 0.853	**2.147**, 1.377	**3.223**, 2.029	**4.276**, 2.965	-	-	-
8	**0.578**, 0.513	**1.300**, 0.870	**2.145**, 1.37	**3.247**, 2.038	**4.562**, 2.867	**5.726**, 4.008	-	-
9	**0.576**, 0.538	**1.317**, 0.910	**2.138**, 1.376	**3.262**, 2.066	**4.551**, 2.852	**6.132**, 3.836	**7.474**, 5.211	-
10	**0.607**, 0.623	**1.330**, 0.933	**2.147**, 1.383	**3.261**, 2.094	**4.617**, 2.909	**6.135**, 3.844	**7.855**, 4.937	**9.836,** 6.184

Табела25.*Очекиване вредности (Ср. вредност, Станд. Дев) броја неконзистенција у условно дефинисаним захтевима на нивоу инстанци*

Број обележја / ***Број условних обележја***	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
3	**0.100**, 0.060	**0.195**, 0.135	**0.346**, 0.265	-	-	-	-	-	-	-
4	**0.162**, 0.121	**0.335**, 0.214	**0.502**, 0.333	**0.824**, 0.592	-	-	-	-	-	-
5	**0.261**, 0.173	**0.526**, 0.347	**0.784**, 0.510	**1.037**, 0.685	**1.324**, 0.756	-	-	-	-	-
6	**0.371**, 0.243	**0.758**, 0.496	**1.113**, 0.732	**1.497**, 0.983	**1.887**, 1.234	**2.356,** 1.432	-	-	-	-
7	**0.523**, 0.353	**1.025**, 0.667	**1.553**, 1.023	**2.066**, 1.351	**2.584**, 1.704	**3.095**, 2.027	**3.626**, 2.327	-	-	-
8	**0.682**, 0.441	**1.357**, 0.896	**2.022**, 1.335	**2.702**, 1.783	**3.402**, 2.221	**4.072**, 2.684	**4.743**, 3.110	**5.221**, 3.453	-	-
9	**0.865**, 0.572	**1.726**, 1.135	**2.596**, 1.702	**3.456**, 2.261	**4.325**, 2.842	**5.172**, 3.406	**6.010**, 3.943	**6.892**, 4.527	**7.634,** 4.801	-
10	**1.088**, 0.716	**2.158**, 1.411	**3.214**, 2.118	**4.294**, 2.828	**5.345**, 3.503	**6.415**, 4.208	**7.462**, 4.891	**8.532**, 5.607	**9.631**, 6.327	**10.746**, 6.960

Биографија аутора

Ивана Огњановић је рођена 20.06.1985. год у Подгорици. Похађала је Основну школу „Октоих“ у Подгорици коју је завршила као добитник дипломе „Луча“, а од стране Наставничког већа је проглашена за ђака генерације. Учествовала је на многим такмичењима из математике, а од резултата издваја треће место на регионалном као и треће место на републичком такмичењу ученика осмих разреда основних школа у Црној Гори. Уписала је Гимназију „Слободан Шкеровић“ у Подгорици - специјалистичку математичку гимназију коју је завршила као добитник дипломе „Луча I“, а од стране Наставничког већа је проглашена за ђака генерације. Учествовала је на многим такмичењима из физике, а од постигнутих резултата издваја: у првом разреду – друго место на републичком такмичењу и трећу награду на савезном такмичењу; у другом разреду – прво место на републичком такмичењу и трећу награду на савезном такмичењу; у трећем и четвртом разреду– прво место на републичком такмичењу.

Природно-математички факултет у Подгорици, смер: примењена математика и рачунарске науке је уписала 2003. год и дипломирала 2007. год прва у генерацији са просечном оценом 9,97. Током студија је била добитник већине престижних награда и признања: 2004. год – добитник Децембарске награде општине Подгорица, 2005. год – награда Црногорске академије наука и умјетности, 2006.год – добитник награде Универзитета Црне Горе, 2007. год – добитник Плакете Универзитета Црне Горе. У децембру 2007. год је уписала магистарске студије на Природно-математичком факултету у Подгорици, а мастер тезу под називом „Математички модели у управљању финансијским портфолиом“ под вођством ментора доц. др Владимира Јаћимовића је одбранила у јулу 2009.год.

Након завршених основних студија, на Природно-математичком факултету у Подгорици је била ангажована као сарадник у настави у периоду од новембра 2007. год до јуна 2008. год. Даље ангажовање је наставила на Факултету информационих технологија Универзитета „Медитеран“ у Подгорици где је тренутно асистент на предметима: Структуре података и алгоритми, XML технологије и Интелигентни системи; а од децембра 2010.год јој је додељена и

функција Продекана за наставу. Од септембра 2012.год је додатно ангажована као сарадник истраживач на Институту савремених технологија Црне Горе.

У оквиру учешћа на пројекту '*Model-Driven Development of Families of Semantically-enabled Service-oriented Architectures*' је у периодима од 01.07.2010.-01.08.2010.год и 01.05.2011.-01.08.2011.год боравила на *Simon Fraser University* у Канади где је као иностранu истраживач била ангажована на истраживањима у области анализе захтева у линијама софтверских производа (енгл. *software product line*) и фамилијама софтверских система. Такође, у периоду од јула-децембра 2012.год је ангажована на *Simon Fraser University* у Канади као сарадник истраживач на пројекту *'Adapting regression models with user preferences for recommendations in learning environments'* који се реализује у сарадњи са *University of British Columbia*из Канаде.

Објављени радови у међународним часописима:

- I.Ognjanović, D.Gašević, E.Bagheri-"***A Stratified Framework for Handling Conditional Preferences: an Extension of the Analytic Hierarchy Process***", Expert Systems with Applications, DOI: 10.1016/j.eswa.2012.08.026 (прихваћен за штампу)

Објављени радови на престижним конференцијама:

- I.Ognjanović, B.Mohabbati, D.Gašević, E.Bagheri, M.Bošković - "***A Metaheuristic Approach for the Configuration of Business Process Families***", IEEE International Conference on service Computing (SCC2012), Hawaii, USA, 2012
- T.Matijević, I.Ognjanović, R.Šendelj - "***Enhancement of Software Projects' Function Point Analysis Based on Non-functional Judgments***", IEEE Mediterranean Conference on Embedded Computing (MECO 2012), Montenegro 2012
- I.Ognjanović, R.Šendelj - "***Teachers' requirements in dynamically adaptive e-learning systems***", 4th International Conference on Education and new Learning Technologies (EDULEARN12), Barcelona, Spain, 2012

- I.Ognjanović, R.Šendelj - "***A Genetic Algorithm for Optimized Configuration of Business Process Families***", International Conference on Advance in Soft Computing (ICASC 2012), Thailand, 2012
- I.Ognjanovic, R.Sendelj- "***Making judgments and decisions about relevant learning resources***", Proc.of the 20th International Electrotechnical and Computer Science Conference, Portoroz, Slovenia (ERK 2011), B, pp.409-412
- R.Sendelj, A.Balota, I.Ognjanovic - "***Analyzes of the Information System of Primary Health Care of Montenegro***" OLS Journals - Software Engineering, Management & Application 1-2 , 2011, ISSN No: 2091-0266, pp. 124-130
- Ognjanović, I., Gašević, D., Bagheri, E., Asadi, M., "***Conditional preferences in software stakeholders' judgments,"*** The 26th Annual ACM Symposium on Applied Computing (SAC 2011), Tunghai University, TaiChung, Taiwan, 2011, pp. 683-690, ISBN: 978-1-4503-0113-8

Објављени радови на домаћим конференцијама:

- Т. Матијевић, И. Огњановић, Р. Шендељ - "***Развојopen-source решења за functional-point анализу софтверских пројеката***", IT Конференција, Жабљак 2012
- М. Букилић, И. Огњановић, Р. Шендељ - "***Одабир оптималнихсистема за учење помоћу online алата***", IT Конференција, Жабљак 2012
- Струјић Ц., Букилић М., Огњановић И. „***Агилни развој софтвера и екстремно програмирање*** “, ИТ Конференција, Жабљак 2011
- Шендељ Р., Огњановић И., Букилић М. „***Концептуални модел увођења ИКТ у образовни исстем***“, ИТ Конференција, Жабљак 2010
- Огњановић И., Шендељ Р. „***Електронско учење-подршка традиционалном учењу***“, INFOFEST 2009

Објављене књиге:

- Огњановић И., Милошевић Н., Раневски Љ., „***Алгоритми и програмирање***“-скрипта за обавезни изборни предмет за ученике III и IV разреда опште гимназије, Подгорица, 2011

Учешће на пројектимакоји су у директно спрези с истраживањима представљеним у дисертацији:

- ***'Model-Driven Development of Families of Semantically-enabled Service-oriented***

  Architectures', *Athabasca University&Simon Fraser University*, финансиран од стране *Alberta Innovates-Technology Futures, New Faculty Award*– пројекат за циљ има обезбеђење метода и технологија за развој семантички подржаних сервис оријентисаних архитектура. Конкретно, истражује се употреда добро успостављених линија софтверских производа (енгл. *software product line*) (SPL) које користе софтверске компоненте интерно развијене за домен SPL-а, док се, с друге стране, SOA заснива на идеји отворене интеграције пословних процеса помоћу заједничких услуга и сервиса.

  Учешће на овом пројекту је омогућило теоретско-аналитичко истраживање у области примене семантичких технологија за развој SOA и анализе функционаланих и нефункционалних корисничких захтева, као и конкретну примену на композицији веб сервиса. Рад на следећа два пројекта је дао практичну примену и елаборацију у областима развоја адаптивних система за учење:

- TEMPUS Project „***Enhancing the quality of distance learning at the western Balkan higher education institutions DL@WEB***“– пројекат за циљ има унапређење квалитета и релевантности учења на даљину (DL) у високошколским установама земаља Западног Балкана као и омогућавање њиховог лакшег и бржег укључења у Европски простор високог образовања (енгл.*European Higher Education Area*). Главни циљ пројекта је побољшање, развој и имплементација стандарда за акредитацију, као и развој смерница и процедура за обезбеђење квалитета на DL студијским програмима на националном нивоу система образовања земаља Западног Балкана.
- SEE-ERA.NET Plus Joint Call Ref. Nr. SEEERAPLUS-115 “***Online Presence for Learning (OP4L)“-*** пројекат за циљ има пружање подршке за напредни менаџмент процеса учења (енгл. *Learning Process Management*) у

персонализованим окружењима за учење (енгл. *Perslonal Learning Enviromets*) као све важније парадгме у процесу образовања. Пројекат се ослања на коришћење семантичких веб технологија за генерисање препорука за учење, на основу корисникових преференци и on-line статуса учесника у социјалним мрежама.

Прилог 1.

Изјава о ауторству

Потписани-а Ивана В. Огњановић

број уписа 20/2009

Изјављујем

да је докторска дисертација под насловом

Семантичке технологије за конфигурисање сервисно-оријентисаних архитектура на основу нефункционалних захтева

- резултат сопственог истраживачког рада,
- да предложена дисертација 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/ла ауторска права и користио интелектуалну својину других лица.

Потпис докторанда

У Београду, 05.12.2012.

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора ____Ивана Огњановић_____

Број уписа _____20/2009_____

Студијски програм ___Информациони системи и менаџмент_____

Наслов рада ____ Семантичке технологије за конфигурисање сервисно-оријентисаних архитектура на основу нефункционалних захтева_____

Ментор ___доц. др Јелена Јовановић_____

Потписани Ивана Огњановић

изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла за објављивање, на порталу **Дигиталног репозиторијума Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанда

У Београду, 05.12.2012

Ивана Огњановић

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Семантичке технологије за конфигурисање сервисно-оријентисаних архитектура на основу нефункционалних захтева

која је моје ауторско дело.

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

Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

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

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда

У Београду, 05.12.2012.

Ђурић

1. Ауторство - Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. Ауторство – некомерцијално. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. Ауторство - некомерцијално – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. Ауторство - некомерцијално – делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. Ауторство – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. Ауторство - делити под истим условима. Дозвољаватę умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.