UNIVERZITET U BEOGRADU

FAKULTET ORGANIZACIONIH NAUKA

Nina S. Turajlić

NOVI MODELI I METODE ZA SELEKCIJU I KOMPOZICIJU WEB SERVISA NA OSNOVU NEFUNKCIONALNIH KARAKTERISTIKA

doktorska disertacija

Beograd, 2014.

UNIVERSITY OF BELGRADE

FACULTY OF ORGANIZATIONAL SCIENCES

Nina S. Turajlić

NOVEL MODELS AND METHODS FOR THE SELECTION AND COMPOSITION OF WEB SERVICES BASED ON NON-FUNCTIONAL REQUIREMENTS

Doctoral Dissertation

Belgrade, 2014.

Mentor:

_____
dr Nenad Mladenović,
redovni profesor, Fakultet Organizacionih Nauka, Univerzitet u Beogradu

Članovi komisije:

_____
dr Milica Vučković,
vanredni profesor, Fakultet Organizacionih Nauka, Univerzitet u Beogradu

_____
dr Milan Stanojević,
vanredni profesor, Fakultet Organizacionih Nauka, Univerzitet u Beogradu

_____
dr Slađan Babarogić,
docent, Fakultet Organizacionih Nauka, Univerzitet u Beogradu

_____
dr Dragan Radojević,
naučni savetnik, Institut „Mihajlo Pupin, Beograd

Datum odbrane: _____

Mom tati Stevi koji bi se radovao

Uobličavanju ideja izloženih u ovoj disertaciji značajno su doprineli brojni razgovori sa mentorom prof. dr Nenadom Mladenovićem, kao i prof. dr Draganom Radojevićem, prof. dr Milanom Stanojevićem i prof. dr Bogdanom Stanojević. Najtoplije se zahvaljujem na poverenju, strpljenju, savetima i podršci. Veliku zahvalnost dugujem i prof. dr Milici Vučković i prof. dr Slađanu Babarogiću kako na savetima i pomoći tokom izrade disertacije tako i na izuzetnoj saradnji i podršci tokom svih godina koje sam provela studirajući i radeći na fakultetu.

Neizmerno se zahvaljujem mnogobrojnim prijateljima sa fakulteta bez čije izuzetne podrške, kako logističke tako i moralne, izrada ove disertacije ne bi bila moguća.

Dugogodišnjim prijateljima Ivani Dragović, Maši Bratuši, Marku Petroviću, Ani Mišić, Mileni Damjanović i Jugoslavu Dragoviću, kao i Kolji, Peđi, Saši, Bojani, Mileni, Radmili i Nikoli Lazarević, dugujem posebnu zahvalnost na nestvarnom strpljenju, razumevanju i podršci tokom svih ovih godina.

Na kraju ne postoje reči kojima bih mogla izraziti zahvalnost svojim roditeljima Srbijanki i Stevanu i sestri Mili.

Novi modeli i metode za selekciju i kompoziciju web servisa na osnovu nefunkcionalnih karakteristika

Rezime:

Ovaj rad je zamišljen da poveže oblast razvoja informacionih sistema sa oblašću operacionih istraživanja i to kroz primenu različitih tehnika mekog računarstva i optimizacije u jednom od savremenih pristupa razvoju aplikacija – servisno-orijentisanom pristupu.

Servisno-orijentisani pristup pripada novoj generaciji pristupa za razvoj distribuiranih aplikacija i automatizaciju poslovanja preduzeća koji vodi unapređenju efikasnosti, agilnosti i produktivnosti preduzeća i stoga se danas sve više primenjuje. Osnovna ideja je da se omogući fleksibilno programsko povezivanje nezavisno razvijenih softverskih komponenti – servisa (koje mogu biti realizovane na različitim platformama i u različitim programskim jezicima). Primena principa servisno-orijentisane paradigme vodi ka razvoju servisa čija je funkcionalnost nezavisna od bilo kog konkretnog poslovnog procesa i koji se samim tim mogu koristiti na različite načine u različitim aplikacijama.

Web servisi predstavljaju fizički nezavisne softverske komponente, dostupne na internetu, koje pružaju određenu funkcionalnost. Budući da se zasnivaju na skupu široko prihvaćenih standarda oni su danas dominantni način realizacije servisno-orijentisanih softverskih rešenja.

S obzirom na to da se web servisi nezavisno razvijaju i objavljuju na internetu svakim danom sve je veći broj dostupnih servisa koji pružaju istu funkcionalnost. Očigledno je da je problem selekcije najboljeg, iz skupa servisa koji pružaju istu funkcionalnost, sve aktuelniji. Nefunkcionalne karakteristike servisa (*eng. Quality of Service – QoS*) tada postaju sve značajnije. Pri tome se selekcija zapravo najčešće vrši na osnovu većeg broja nefunkcionalnih karakteristika koje su po svojoj prirodi veoma heterogene.

Sa druge strane, kada zahtevanu funkcionalnost ne može obezbediti pojedinačni servis ali bi se mogla realizovati izvršavanjem skupa različitih servisa, od kojih svaki obezbeđuje deo zahtevane funkcionalnosti, problem selekcije postaje još kompleksniji. Kompozicija servisa predstavlja agregaciju servisa koji su komponovani na takav način

da zajedno automatizuju određeni zadatak ili poslovni proces. Prednost ovakvog pristupa razvoju aplikacija je u tome što se za ispunjenje novih poslovnih zahteva i automatizaciju novih poslovnih procesa mogu koristiti postojeći servisi (bilo proširenjem postojećih kompozicija, bilo kreiranjem novih) uz manji napor i manje troškove.

Problem selekcije web servisa za kompoziciju se sastoji u selekciji konkretnih servisa, za svaku od komponenti definisane kompozicije, na takav način da rezultujuća kompozicija ne samo zadovolji postavljene funkcionalne zahteve već i je i optimalna u pogledu relevantnih QoS zahteva. Kada se razmatra selekcija servisa za određenu kompoziciju izbor bi trebalo zasnivati na nefunkcionalnim karakteristikama celokupne kompozicije. Određivanje QoS za kompoziciju postavlja sledeći izazov: potrebno je na neki način agregirati QoS vrednosti pojedinačnih servisa u kompoziciji, pri čemu način agregacije zavisi od prirode samih QoS atributa ali i od strukture kompozicije. Pored toga, često se nameću i određena ograničenja u pogledu minimalnih QoS karakteristika koje kompozicija mora da zadovolji.

Dat je kritički osvrt na relevantne postojeće pristupe modelovanju problema i ukazano je na neadekvatnost mnogih pristupa koji podrazumevaju određena pojednostavljenja problema u smislu da omogućavaju razmatranje samo kvantitativnih QoS atributa, zahtevaju linearnost funkcija za agregiranje QoS vrednosti prilikom određivanja QoS vrednosti celokupne kompozicije, zahtevaju dodeljivanje arbitrarnih težina i normalizaciju samih vrednosti.

U cilju prevazilaženja ovih nedostataka predlažu se dva nova pristupa za modelovanje problema koji ne zahtevaju dodeljivanje arbitrarnih težina niti normalizaciju vrednosti nefunkcionalnih karakteristika. Istovremeno cilj je bio i da se omogući da se na adekvatniji način izraze zahtevi korisnika u pogledu kompromisa koje je spreman da prihvati a koji se često mogu izraziti samo pomoću složenih verbalnih iskaza.

Stoga se prvo predlaže primena konzistentne fazi logike koja omogućava da se složeni zahtevi korisnika izraze kao logički zahtevi koji su definisani u Bulovom okviru. Šta više, primena logičke agregacije rezultuje novom strukturom komponenti za razliku od težinske sume. Konzistentna fazi logika je izabrana imajući u vidu da primena konvencionalne fazi logike ne omogućava da se uzme u obzir da definisani zahtev može

uključivati negaciju koja u klasičnoj fazi logici narušava princip kontradikcije i isključenja trećeg.

Drugi pristup, koji takođe do sada nije predlagan za modelovanje problema, namenjen je slučajevima kada ne postoje logički zahtevi vezani za kriterijume ali su kriterijumi heterogeni i izraženi pomoću nekoliko različitih jedinica mere. U tom slučaju se predlaže da se problem modeluje kao diskretan problem višekriterijumskog razlomljenog programiranja (*eng. Multi-Objective Fractional Programming - MOFP*). Relevantni kriterijumi bi se sparivali uzimajući u obzir njihovu prirodu (jedinicu mere i tip) te bi svi formirani količnici imali istu jedinicu mere i optimizovali bi se u istom smeru.

Za selekciju pojedinačnih servisa predloženo je inovativno proširenje AHP (*eng. Analytic Hierarchy Process*) metode primenom konzistentne fazi logike kako bi se omogućila selekcija na osnovu logičkih zahteva vezanih za njihove nefunkcionalne karakteristike. Naime, imajući u vidu da kriterijumi mogu biti međusobno uslovljeni ili da njihov značaj može zavisiti od ostvarenih vrednosti drugih kriterijuma, a kako klasična AHP metoda ne obezbeđuje podršku za takvu vrstu problema, predloženo je uvođenje logičkih funkcija koje omogućavaju da se iskažu različiti odnosi između kriterijuma. Na ilustrativnim primerima je pokazano da primena konvencionalne fazi logike i konzistentne fazi logike ne moraju voditi istom izboru.

Za selekciju servisa za kompoziciju predlaže se nekoliko metoda. Prvo se, za egzaktno rešavanje MOFP problema, predlaže primena jedna postojeće metode koja je u ovom radu prilagođena prirodi postavljenog problema. Kada je reč o većim problemima kod kojih je vreme izvršavanja od kritičnog značaja, kao što je dinamička selekcija servisa (kada se selekcija vrši neposredno pred ili čak i u toku izvršavanja), prednost se daje heurističkim metodama. Shodno tome, najpre se predlaže, po prvi put, primena poznate metaheuristike metode promene okolina (*eng. Variable Neighborhood Search – VNS*). Zatim je izložena jedna nova metaheuristika zasnovana na tabu pretraživanju i VNS metodi. Pokazano je da se obe metode mogu koristiti za rešavanje kako MMKP (*eng. Multi-dimensional Multiple-choice 0-1 Knapsack Problem*) modela problema (budući da je ovaj pristup modelovanju u literaturi najzastupljeniji) tako i problema modelovanih pomoću predloženog pristupa zasnovanog na konzistentnoj fazi logici. Pored toga, predložena i je i jedna nova metoda za pronalaženje početnog dopustivog

rešenja MMKP problema. Valjanost predloženih pristupa izvršena je eksperimentalnom proverom, koja je podržana razvijenim softverskim rešenjem.

Konačno, razmatra se i mogućnost šire primene predloženih pristup.

Ključne reči: servisno-orijentisani pristup – SOA, selekcija servisa, kompozicija servisa, web servisi, nefunkcionalne karakteristike – QoS, konzistentna fazi logika, logička agregacija, višekriterijumsko razlomljeno programiranje – MOFP, metoda analitičkog hijerarhijskog procesa – AHP, metoda promene okolina – VNS, tabu pretraživanje.

Naučna oblast: Tehničke nauke

Uža naučna oblast: Operaciona istraživanja i informacioni sistemi

UDK broj: 004:519.8(043.3)

Novel models and methods for the selection and composition of web services based on non-functional requirements

Abstract:

This thesis proposes to combine two different fields of study, that of information systems and that of operational research, through the application of various soft computing and optimization techniques in the service-oriented approach to modern software development.

Service-orientation represents a new generation approach for developing distributed applications and business process automation which can increase business efficiency, agility and productivity, and as such has grown to be widely adopted. It is based on the idea that distributed applications can be flexibly and dynamically composed by integrating independently developed software components – services (which may be developed on different platforms and in different programming languages). The application of service-oriented paradigm principles leads to the development of services whose functionality is independent of any specific business process and which can therefore be used in various applications in different ways.

Web services are physically independent software components, accessible on the Internet, which provide certain functionality. Due to the fact that they are based on a set of widely adopted standards they are the predominant approach for implementing modern service-oriented software solutions.

With the proliferation of web services offering similar functionality (due to the fact that they are independently developed and published on the Internet) the problem of how to choose the best one, from a vast collection of suitable services, is becoming increasingly important. The non-functional properties of services (Quality of Service – QoS) thus play an important role in the process. Furthermore, the selection is usually based on multiple non-functional characteristics which are very heterogeneous in nature.

On the other hand, when the desired functionality cannot be provided by a single service but might be achieved by executing a set of services, each providing part of the desired functionality, the task becomes even more complex. A service composition represents an aggregation of services collaborating collectively to automate a specific

task or business process. The main benefit of this approach to application development is that existing services can be used for achieving new business demands or automating new business processes (either by expanding existing compositions or by creating new ones) with less effort and in a more cost effective way.

The web service selection problem for a composition consists of choosing actual services, for each component of the established composition, in such a manner that the resulting composition not only fulfills the desired functional requirements but is also optimal in terms of the relevant QoS requirements. Furthermore, when selecting services for a composition the basis for the selection should be the performances of the composite service as a whole. The main challenge, which arises when computing the overall QoS for a web service composition, is that the QoS values need to be aggregated in some way and that the nature of the aggregation depends on the chosen QoS attributes as well as the actual structure of the composition. In addition, certain constraints regarding the required QoS levels of the composition are also often given.

A critical review of the existing approaches for modeling these problems is given and the drawbacks of many approaches are discussed. Namely, most of the existing approaches only allow the selection of services based on quantitative QoS attributes, require linear functions for the aggregation of QoS values when determining the QoS of the composition and presume the assignment of arbitrary weights and normalization of the actual QoS values.

In order to overcome these issues two new models are proposed which do not require the assignment of arbitrary preferences or the normalization QoS values. Furthermore the aim was also to allow the users to more adequately specify the trade-offs they might be willing to make which can usually only be formulated as complex verbal statements.

Thus first the of application of Boolean consistent fuzzy logic is proposed since it enables expressing complex verbal statements in the form of logical demands which are defined in the Boolean frame. Moreover logical aggregation results in a new structure of the components as opposed to the weighted sum approach. The application of Boolean consistent fuzzy logic is preferred since conventional fuzzy logic does not allow for taking into account statements which include negation, i.e. in such cases

conventional fuzzy logic does not satisfy the axioms of excluded middle and contradiction.

A second novel approach is proposed for situations when there are no logical demands regarding the relevant criteria but the criteria are heterogeneous and expressed using several measurement units. In such cases the problem can be modeled as a discrete multi-objective fractional programming problem (MOFP). The initial criteria would be paired taking into account the nature (unit measure and type) so that all of the fractional objectives have the same unit and can be optimized in the same direction.

For the selection of services for a particular functionality, an innovative approach based on the combination of the Analytical Hierarchy Process method (AHP) and Boolean consistent fuzzy logic is proposed to allow for the selection of services on the basis of logical demands regarding the QoS criteria. Given that certain criteria may influence or be influenced by other criteria, as well as the fact that the importance of the criteria may vary based upon the demonstrated level of other criteria, on the one hand, and the fact the traditional AHP method lacks support for this type of problem on the other, it is proposed that logical functions could be used to take into account these relationships. The illustrative examples point out that the conventional and consistent fuzzy approaches can lead to the choice of different alternatives.

For the selection of services for a composition several methods are proposed. First, an adaptation of an existing exact method for solving the posed MOFP problem is presented. For larger, time-critical problems, such as the dynamic selection of services (when the services are to be selected just prior or even during the execution of the composition) the use of heuristic methods is advocated. To this end, the use the well-known Variable Neighborhood Search (VNS) metaheuristic is applied, for the first time, for the selection of services. Then a novel metaheuristic based on tabu search and VNS is presented. It is shown that both methods can be used for solving both the MMKP (Multi-dimensional Multiple-choice 0-1 Knapsack Problem) model (as one of the most common approaches in literature) and the proposed model based on consistent fuzzy logic. In addition novel method is proposed for generating an initial feasible solution of the MMKP problem. The validation of the proposed approaches is supported by a specially developed software solution.

Finally, the possibility of a wider application of the proposed approaches is discussed.

Key words: service-oriented approach – SOA, service selection, service composition, web services, non-functional requirements – QoS, Boolean consistent fuzzy logic, logical aggregation, multi-objective fractional programming – MOFP, analytic hierarchy process – AHP, variable neighborhood search – VNS, tabu search.

Scientific field: Technical sciences

Specific scientific field: Operational research and information systems

UDK number: 004:519.8(043.3)

Sadržaj

1. UVOD

1.1. Problem, predmet i cilj istraživanja

Ovaj rad je zamišljen da poveže oblast razvoja informacionih sistema sa oblašću operacionih istraživanja i to kroz primenu različitih tehnika mekog računarstva i optimizacije u jednom od savremenih pristupa razvoju aplikacija – servisno-orijentisanom pristupu.

1.1.1. Problem

U okviru ovog rada razmatra se problem selekcije web servisa kao jedan od značajnih elemenata servisno-orijentisanog pristupa razvoju aplikacija. Servisno-orijentisani pristup pripada novoj generaciji pristupa za razvoj distribuiranih aplikacija i automatizaciju poslovanja preduzeća koji vodi unapređenju efikasnosti, agilnosti i produktivnosti preduzeća i stoga se danas sve više primenjuje (*Erl, 2008*). Osnovna ideja ovog pristupa je da se omogući fleksibilno programsko povezivanje nezavisno razvijenih softverskih komponenti koje mogu biti realizovane na različitim platformama i u različitim programskim jezicima. Osnovni elementi softverskih rešenja zasnovanih na ovom pristupu su servisi. Web servisi, dakle, predstavljaju fizički nezavisne softverske komponente, dostupne na Internetu, koje pružaju određenu funkcionalnost. Primena principa servisno-orijentisane paradigme vodi ka razvoju servisa čija je funkcionalnost nezavisna od bilo kog konkretnog poslovnog procesa i koji se samim tim mogu koristiti na različite načine u različitim aplikacijama.

Budući da se softverske komponente nezavisno razvijaju i objavljuju na Internetu svakim danom sve je veći broj dostupnih web servisa, a kako je servisno-orijentisani pristup razvoju aplikacija sve prisutniji neminovna posledica je i porast broja web servisa koji pružaju istu funkcionalnost. Očigledno je da je problem selekcije najboljeg servisa iz skupa servisa koji pružaju istu funkcionalnost sve aktuelniji. Nefunkcionalne karakteristike servisa (*eng. Quality of Service* – QoS) tada postaju sve značajnije. U literaturi je prisutno mnoštvo QoS mera kojima se opisuju različiti kvantitativni i kvalitativni aspekti servisa kao što su performanse, dostupnost, pouzdanost, sigurnost, interoperabilnost itd. (*W3C, 2003; ISO 9126, 2001*). Pored ovih opštih, u literaturi prihvaćenih, QoS mera takođe postoje i domensko-specifične mere. Jedan od ključnih problema prilikom selekcije servisa na osnovu nefunkcionalnih

karakteristika je heterogenost QoS mera koje mogu biti kvantitativne i kvalitativne i izražene pomoću različitih neuporedivih jedinica mere. Pored toga treba uzeti u obzir i činjenicu da je za neke QoS atribute poželjna maksimalna vrednost dok je za druge poželjna minimalna vrednost. Takođe treba imati u vidu da različiti QoS atributi mogu biti relevantni prilikom selekcije servisa kao i da njihov značaj može varirati u zavisnosti od potreba korisnika. Dakle, **problem selekcije pojedinačnih web servisa** bi se mogao definisati na sledeći način: *Potrebno je za dati funkcionalni zahtev izabrati konkretan servis, iz skupa raspoloživih servisa koji obezbeđuju datu funkcionalnost, koji je optimalan u pogledu relevantnih nefunkcionalnih karakteristika.*

Sa druge strane, kada zahtevanu funkcionalnost ne može obezbediti pojedinačni servis ali bi se ona mogla realizovati izvršavanjem skupa različitih servisa, od kojih svaki obezbeđuje deo zahtevane funkcionalnosti, problem selekcije postaje još kompleksniji. Kompozicija servisa predstavlja agregaciju servisa koji su komponovani na takav način da zajedno automatizuju određeni zadatak ili poslovni proces. Jedan od osnovnih zahteva servisno-orijentisane paradigme je razvoj servisa koji su interoperabilni i koji se mogu uključivati u različite kompozicije. Prednost ovakvog pristupa je u tome što se za ispunjenje novih poslovnih zahteva i automatizaciju novih poslovnih procesa mogu koristiti postojeći servisi (bilo proširenjem postojećih kompozicija, bilo kreiranjem novih) uz manji napor i manje troškove. Prilikom definisanja kompozicije za realizaciju nekog složenog funkcionalnog zahteva prvo je neophodno odrediti koji su to delovi funkcionalnosti koje svaka od komponenti treba da pruži kao i redosled njihovog izvršavanja. U ovom koraku razmatraju se samo funkcionalne karakteristike servisa. Kada se kompozicija definiše sledeći korak je izbor konkretnih servisa koji će se uključiti u kompoziciju i obaviti definisane zadatke. U skladu sa time prvi korak je planiranje čiji je rezultat apstraktna kompozicija servisa, a drugi korak je selekcija u okviru koje se kompoziciji dodeljuju konkretni servisi.

Problem selekcije web servisa za definisanu kompoziciju se sastoji u selekciji konkretnih servisa, za svaku od komponenti kompozicije, na takav način da rezultujuća kompozicija ne samo zadovolji postavljene funkcionalne zahteve već je i optimalna u pogledu relevantnih nefunkcionalnih karakteristika. Kada se razmatra selekcija servisa za određenu kompoziciju izbor bi trebalo zasnivati na nefunkcionalnim karakteristikama celokupne kompozicije. Određivanje QoS za kompoziciju postavlja sledeći izazov:

potrebno je na neki način agregirati QoS vrednosti pojedinačnih servisa u kompoziciji, pri čemu način agregacije (sumiranje, množenje, minimum itd.) zavisi od prirode samih QoS atributa. Pored toga, često se nameću i određena ograničenja u pogledu minimalnih QoS karakteristika koje kompozicija mora da zadovolji, a kako broj relevantnih QoS karakteristika može biti proizvoljan, selekcija web servisa zapravo predstavlja višekriterijumski optimizacioni problem. Imajući u vidu navedeno, **problem selekcije web servisa za kompoziciju** bi se mogao definisati na sledeći način: *Potrebno je za svaku od komponenti kompozicije izabrati konkretan servis, iz skupa raspoloživih servisa koji obezbeđuju zahtevanu funkcionalnost, kako bi se dobila kompozicija koja je optimalna u pogledu relevantnih nefunkcionalnih karakteristika a koja istovremeno zadovoljava sva postavljena QoS ograničenja.*

1.1.2. Predmet

Predmet ovog rada je ispitivanje mogućnosti primene različitih savremenih pristupa mekog računarstva za modelovanje i rešavanje problema selekcije i kompozicije web servisa zasnovane na njihovim nefunkcionalnim karakteristikama.

Modelovanje

U literaturi postoje različiti pristupi za modelovanje datog problema koji su u najvećoj meri uvedeni kako bi se olakšalo rešavanje navedenog problema. U velikom broju radova se navodi da se problem selekcije servisa za kompoziciju može posmatrati kao specijalna vrsta problema ranca – MMKP (*Multi-dimensional Multiple-choice 0-1 Knapsack Problem*) (npr. *Ardagna & Pernici, 2005; Jaeger & al, 2005; Yu & Lin, Jul, 2005; Cao & al, 2007; Yu & al, 2007; Alrifai & Risse, 2009; Liu & al, 2009; Luo & al, 2011; Sun & Zhao , 2012*). Drugi pristupi predlažu modelovanje problema pomoću grafova (*Jaeger & al, 2005; Yu & Lin, Jul, 2005; Yu & Lin, Decembar, 2005; Gao & al, 2006; Jaeger, 2006; Yu & al, 2007*). Većina ovih pristupa predlaže metodu jednostavnih aditivnih težina (SAW) za agregaciju datih QoS kriterijuma u jedinstvenu funkciju cilja.

Primena SAW metode podrazumeva da korisnik svakom od kriterijuma dodeli težinu izražavajući na taj način koliko je dati kriterijum za njega značajan. Međutim, dodeljivanje težina je arbitrarno i u krajnjoj liniji ne mora adekvatno odražavati korisnikove preference. Ukoliko su dva kriterijuma podjednako značajna dodeljivanje

jednakih težina svakom od njih neće uvek obezbediti da oni ravnopravno učestvuju u selekciji. Na primer, neka postoje dve alternative od kojih je prva mnogo dominantnija od druge po jednom od kriterijuma a značajno lošija po drugom, dok je druga alternativa prosečna po oba kriterijuma. U slučaju prve alternative usled dominantnosti prvog kriterijuma formiranjem težinske sume se potire uticaj drugog kriterijuma što može voditi izboru pogrešne alternative. Budući da je zahtev korisnika bio da oba kriterijuma ravnopravno učestvuju u selekciji imalo bi više smisla izabrati alternativu koja je prosečna po oba kriterijuma ali čija je ukupna težinska suma manja. Ovakve zahteve bi bilo adekvatnije izraziti pomoću logičkih funkcija. Sa druge strane, navedeni pristupi ne uzimaju u obzir činjenicu da QoS kriterijumi na osnovu kojih se vrši selekcija mogu uticati jedni na druge, odnosno biti u korelaciji i/ili kontradikciji. Dodatni izazov predstavlja činjenica da su u realnosti zahtevi korisnika često dosta složeniji. Na primer korisnik može imati potrebu da definiše zahteve u kojima se lošija vrednost po nekom od kriterijuma može nadomestiti nekim drugim kriterijumom, odnosno u kojima značaj kriterijuma zavisi od ostvarenih vrednosti drugih kriterijuma. U većini realnih slučajeva, ovakvi zahtevi se mogu izraziti samo pomoću složenih verbalnih iskaza.

Fazi logika se nameće kao prirodno rešenje za specifikaciju zahteva korisnika budući da je usklađena sa kvalitativnim opisima koje ljudi koriste u svakodnevnoj komunikaciji. Teorija fazi skupova omogućava prevođenje ovakvih iskaza u odgovarajuće matematičke izraze. Druga prednost primene fazi logike se ogleda u tome što omogućava uzimanje u obzir korelacije koja može postojati među različitim QoS atributima. Iako u literaturi postoje pristupi koji razmatraju upotrebu fazi logike prilikom definisanja zahteva vezanih za nefunkcionalne karakteristike servisa (npr. *Agarwal & Lamparter, 2005; Lin & al, 2005; Zhang & al, 2006; Herssens & al, Januar 2008; Sora & al, 2009; Almulla & al, 2011*) ne postoji pristup koji razmatra mogućnost primene konzistentne fazi logike predstavljene u (*Radojević, 2008b*). Logičke funkcije kojima se predstavljaju zahtevi korisnika bi trebalo da zadovoljavaju sve zakone klasične logike, odnosno da budu definisane u Bulovom okviru, a budući da nijedna konvencionalna teorija fazi skupova nije u Bulovom okviru (*Radojević, 2008a*) prednost se daje konzistentnom pristupu nad konvencionalnim. U konvencionalnom pristupu posmatraju se vrednosti a ne struktura promenljivih. Nasuprot tome, u konzistentnom

pristupu transformacija se vrši najpre na strukturnom nivou, da bi se tek onda uvele i izračunale vrednosti (zahvaljujući tome svi Bulovi postulati važe). Ova značajna razlika između ova dva pristupa u određenim slučajevima vodi i do razlike u rezultatima. Logički iskazan zahtev se može razložiti korišćenjem pravila konzistentne fazi logike. Najpre se razmatra struktura. Tek nakon što se izvrše transformacije i uspostavi konačna struktura prelazi se na vrednosni nivo i vrše izračunavanja. Stoga, logička agregacija za rezultat ima novu strukturu komponenti što nije slučaj kada je reč o SAW pristupu (*Radojević, 2010*). Namera rada je da se pokaže da se opisani pristup može primeniti za definisanje složenih korisničkih zahteva vezanih za relevantne QoS atribute prilikom modelovanja problema selekcije web servisa kao i da njegova primena ne mora rezultovati istim izborom servisa kao primena konvencionalne fazi logike.

Drugi pristup koji se po prvi put predlaže u slučajevima kada ne postoje logički zahtevi vezani za QoS kriterijume je modelovanje problema kao diskretnog problema razlomljenog programiranja MOFPP (*Multi-Objective Fractional Programming Problem*) u kome se QoS kriterijumi sparuju sa ciljem da se optimizuju njihovi količnici. QoS kriterijumi bi se sparivali uzimajući u obzir njihovu prirodu (jedinicu mere i vrstu optimizacije - minimizacija ili maksimizacija) čime bi se dobile razlomljene funkcije cilja koje imaju iste jedinice mere i koje se optimizuju u istom smeru. Da bi se u SAW pristupu omogućilo agregiranje vrednosti različitih QoS atributa u jedinstvenu funkciju cilja neophodno ih je prethodno normalizovati budući da mogu biti izraženi pomoću različitih jedinica mera ili čak i istih ali sa različitim rasponima vrednosti. Prednosti predloženog pristupa nad SAW pristupom proizilaze iz činjenice da MOLFPP ne podrazumeva formiranje jedinstvene funkcije cilja zbog čega se ne zahteva normalizacija podataka niti dodeljivanje arbitrarnih težina QoS kriterijumima.

Rešavanje

S obzirom na sve veću aktuelnost izloženog problema predloženi su brojni pristupi za njegovo rešavanje koji zavise pre svega od prirode problema odnosno cilja koji se želi postići selekcijom. U literaturi se predlaže korišćenje dobro poznatih metoda višekriterijumskog odlučivanja (MCDM) kao što je AHP (*Analytic Hierarchy Process*) (npr. *Godse & al, 2008; Tran & al, 2009; Garg & al, 2013*), kao i različitih metoda višekriterijumske optimizacije, egzaktnih i heurističkih. Za egzaktno rešavanje predlažu

se između ostalog celobrojno linearno programiranje (npr. *Zeng & al,2003; Aggarwal & al, 2004; Ardagna & Pernici, 2005; Yu & al, 2007; Kattepur & al, 2011*), metode grananja i ograničavanja (npr. *Lin & al, 2005;Yu & Lin, Decembar, 2005; Yu & al, 2007*), dinamičko programiranje (npr. *Gao & al, 2006; Huang & al, 2009; Yu & Lin, Jul, 2005*), mada neki od navedenih pristupa podrazumevaju odsustvo QoS ograničenja što je suviše restriktivno za većinu realnih problema. Od heuristika, budući da je poznato da je MMKP problem NP težak, predlaže se upotreba genetskih algoritama (npr. *Canfora & al, 2004; Claro & al, 2005; Jaeger & Muhl, 2007; Parejo & al, 2008; Wada & al, 2008; Ai, 2011*), tabu pretraživanja (npr. *Parejo & al 2008; Pop & al, 2011*), simuliranog kaljenja (npr. *Berbner & al, 2006; Gao & al, 2009*), PSO – Partical Swarm Optimization (npr. *Ming & Zhen-wu, 2007; Fan & al, 2009; Wang & al, 2010*) kao i specijalno razvijenih heuristika. Većina predloženih pristupa koristi SAW metodu za agregaciju kriterijuma u jedinstvenu funkciju cilja.

Za **problem selekcije pojedinačnih servisa** u radu se predlaže proširenje AHP metode, kao jedne od najčešće korišćenih metoda višekriterijumskog odlučivanja (*Saaty, 1988*) koje omogućava rešavanje problema modelovanog pomoću predloženog pristupa za agregiranje višestrukih korelisanih kriterijuma u jedinstvenu funkciju cilja na osnovu složenih zahteva korisnika pomoću konzistentne fazi logike.

Kada je u pitanju **problem selekcije servisa za kompoziciju** predlaže se nekoliko pristupa. Najpre se za predloženi način modelovanja problema u slučajevima kada ne postoje logički zahtevi vezani za QoS kriterijume – MOLFPP, predlaže, po prvi put, primena novije tehnike (*Stanojević & Stanojević, 2013*) za generisanje jako efikasnih rešenja koji je u ovom radu prilagođena za diskretan MOLFPP. Namera je da se pokaže da primena ove tehnike omogućava egzaktno rešavanje problema u veoma kratkom vremenskom roku čak i za probleme većih dimenzija.

Zatim se pažnja usmerava na primenu heuristika za rešavanje vremenski kritičnih problema većih dimenzija. Prednost se daje heurističkim metodama imajući u vidu činjenicu da je jedan od zahteva, koji je u poslednje vreme sve aktuelniji, obezbeđivanje mogućnosti dinamičke kompozicije servisa (kada se selekcija servisa vrši neposredno pre izvršavanja kompozicije ili čak u toku njenog izvršavanja). Za razliku od toga statički pristup podrazumeva da se unapred izaberu servisi i permanentno dodele kompoziciji. Međutim, skup raspoloživih servisa ne mora u svakom trenutku biti

isti, odnosno u međuvremenu se mogu pojaviti novi servisi sa boljim QoS karakteristikama ili se može desiti da u vreme izvršavanja kompozicije predviđeni servis bude nedostupan ili da se njegovi QoS nivoi značajno promene. Sa druge strane i zahtevi poslovanja se često menjaju što može iziskivati i promenu zahteva vezanih za QoS. Stoga mogućnost brze i dinamičke kompozicije servisa postaje imperativ. U svetlu navedenog, egzaktne metode za rešavanje problema mogu biti neadekvatne (u pogledu računarskih resursa i vremena) za dinamičku selekciju servisa s obzirom na to da broj mogućih kombinacija servisa za datu kompoziciju eksponencijalno raste sa porastom broja raspoloživih servisa.

Najpre se razmatra rešavanje problema modelovanog kao MMKP budući da je ovaj pristup modelovanju problema selekcije u literaturi najzastupljeniji. Kako je MMKP problem NP težak za njegovo rešavanje se predlaže primena poznate metaheurističke metode, koja nije do sada korišćena za selekciju servisa, Variable Neighborhood Search – VNS (*Mladenović & Hansen, 1997; Hansen & Mladenović, 2001; Hansen & Mladenović, 2005*). Namera je da se pokaže da njena primena omogućava efikasno rešavanje problema selekcije servisa za kompoziciju. Uz to se predlaže i nova metoda koja bi se mogla koristiti za efikasno pronalaženje početnog dopustivog rešenja. Pored toga je, za ovako modelovan problem, razvijena i nova hibridna metaheuristika zasnovana na VNS i tabu pretraživanju. Dalje, namera je da se pokaže da se predložene metode mogu koristiti i za rešavanje problema formulisanog pomoću predloženog modela zasnovanog na primeni konzistentne fazi logike.

Treba imati u vidu da većina postojećih pristupa zahteva linearnu funkciju cilja a da određivanje QoS za kompoziciju u opštem slučaju može rezultovati nelinearnim funkcijama (što zavisi od načina na koji se pojedinačni QoS atributi agregiraju). Primena ovih pristupa dakle podrazumeva svođenje nelinearnih funkcija na linearne (na primer primenom logaritamske funkcije na proizvode kako bi se preveli u sumu logaritama). Stoga se većina autora vezuje za određeni predefinisani skup QoS mera koje se agregiraju pomoću sume ili proizvoda dok se ostali načini agregacije uglavnom ne razmatraju. Namera je i da se pokaže da bi se predloženi pristupi mogli iskoristiti za rešavanje problema bez nametanja ovakvog ograničenja. Na ovaj način bi se omogućilo da se selekcija vrši na osnovu bilo kog skupa QoS mera uključujući i domensko-specifične QoS mere koje se agregiraju pomoću korisnički definisanih funkcija.

1.1.3. Cilj

Osnovni cilj ovog rada je definisanje novih pristupa za modelovanje i rešavanje problema selekcije web servisa na osnovu nefunkcionalnih karakteristika.

Posebni ciljevi:

- Definisanje novih pristupa za modelovanje problema selekcije web servisa na osnovu nefunkcionalnih karakteristika koji će omogućiti da se na adekvatan način uzmu u obzir zahtevi korisnika koji su često izraženi pomoću složenih verbalnih iskaza, različiti odnosi koji postoje između relevantnih nefunkcionalnih karakteristika kao i specifičnosti samih mera.
- Razvoj nove metode koja bi unapredila rešavanje problema selekcija pojedinačnih servisa modelovanog pomoću predloženog novog modela.
- Utvrđivanje mogućnosti efikasne primene postojećih metoda višekriterijumske optimizacije za rešavanje identifikovanog problema, koje nisu do sada primenjivane za problem selekcije web servisa na osnovu nefunkcionalnih karakteristika, kako za postojeće modele problema tako i za predložene nove modele.
- Razvoj nove heuristike koja bi unapredila rešavanje problema selekcije kada je neophodna brza i dinamička selekcija servisa za kompoziciju.
- Razvoj prototipskog softverskog rešenje kojim bi se obezbedila implementacija predloženih metoda.
- Razmatranje mogućnosti primene predloženih pristupa u drugim oblastima.

1.2. Polazne hipoteze

Opšta hipoteza: Primenom različitih savremenih pristupa mekog računarstva i optimizacije moguće je unaprediti modelovanje i rešavanje problema selekcije i kompozicije web servisa zasnovane na njihovim nefunkcionalnim karakteristikama.

Pomoćne hipoteze:

- Moguće je definisati nove modele za formulisanje problema selekcije web servisa zasnovane na njihovim nefunkcionalnim karakteristikama koji će na adekvatan način uzeti u obzir složene verbalne zahteve korisnika, različite odnose koji postoje između relevantnih nefunkcionalnih karakteristika kao i specifičnosti samih mera.

- Za predstavljanje preferenci korisnika vezanih za nefunkcionalne karakteristike adekvatnija je primena fazi logike od uobičajenog arbitrarnog dodeljivanja težina.
- Dodeljivanjem težina nefunkcionalnim karakteristikama nije moguće predstaviti složene verbalne zahteve korisnika ali se oni mogu adekvatno izraziti pomoću logičkih funkcija.
- Primenom konzistentne fazi logike moguće je složene verbalne zahteve korisnika prevesti u logičke funkcije, definisane u Bulovom okviru, koje će uzeti u obzir i korelaciju koja može postojati među nefunkcionalnim karakteristika.
- Primena konzistentne fazi logike fazi ne mora rezultovati istim izborom servisa kao primena konvencionalne fazi logike.
- U slučajevima kada ne postoje logički zahtevi vezani za nefunkcionalne kriterijume ali su dati kriterijumi heterogeni problem se može modelovati kao diskretni problem linearnog razlomljenog programiranja MOLFPP (Multi-Objective Linear Fractional Programming Problem).
- Modelovanje problema kao diskretnog MOLFPP je adekvatnije od formiranja jedinstvene funkcije cilja pomoću metode jednostavnih aditivnih težina (SAW).

- Metode višekriterijumskog odlučivanja i optimizacije se mogu efektivno i efikasno koristiti za rešavanje problema selekcije, kako pojedinačnih web servisa tako i servisa za kompoziciju, na osnovu nefunkcionalnih karakteristika.
  - Moguće je proširiti AHP (Analytic Hierarchy Process) metodu tako da omogući selekciju pojedinačnih servisa na osnovu logičkih zahteva vezanih za nefunkcionalne karakteristike servisa.
  - Moguće je primeniti noviju tehniku za generisanje efikasnih rešenja za egzaktno rešavanje problema selekcije servisa za kompoziciju modelovanog kao diskretni MOLFPP.
  - Poznate metaheuristike kao što su VNS (Variable Neighborhood Search), i tabu pretraživanje mogu se uspešno primeniti za rešavanje problema selekcije servisa za kompoziciju, modelovanog kako pomoću postojećih

pristupa kao što je MMKP (Multi-dimensional Multiple-choice 0-1 Knapsack Problem) tako i pomoću predloženih modela.

- Moguće je razviti nove heuristike koje će unaprediti rešavanje problema kada je neophodna brza i dinamička selekcija servisa za kompoziciju.
- Predloženi pristupi se mogu primeniti i u različitim oblastima.

1.3. Očekivani rezultati i doprinosi

Očekivani rezultati istraživanja u predmetnoj oblasti će sadržati veći broj doprinosa, među kojima se izdvajaju:

- Primena različitih pristupa mekog računarstva i optimizacije u oblasti razvoja informacionih sistema čime se obezbeđuje multidisciplinarnost;
- Pregled dosadašnjih istraživanja u predmetnoj oblasti i kritički osvrt na relevantne pristupe;
- Definisanje novih modela za formulisanje problema selekcije web servisa zasnovane na njihovim nefunkcionalnim karakteristikama koji će na adekvatan način uzeti u obzir složene verbalne zahteve korisnika, različite odnose koji postoje između relevantnih nefunkcionalnih karakteristika kao i specifičnosti samih mera [*novi modeli*];
- Inovativno proširenje AHP metode (koja je jedna od najčešće korišćenih metoda višekriterijumskog odlučivanja) konzistentnom fazi logikom koje bi se moglo koristiti, ne samo za rešavanje problema pojedinačne selekcije web servisa već i u bilo kom drugom domenu višekriterijumskog odlučivanja u kome je potrebno doneti odluku na osnovu složenih verbalnih zahteva korisnika. [*nova metoda*];
- Ukazivanje na mogućnost efikasne primene postojećih metoda višekriterijumske optimizacije koje nisu do sada korišćene za problem selekcije web servisa za kompoziciju na osnovu nefunkcionalnih karakteristika, kako za postojeće modele problema tako i za predložene nove modele [*nova primena*];
- Razvoj novih heuristika koje će unaprediti rešavanje problema kada je neophodna brza i dinamička selekcija servisa za kompoziciju[*nova metoda*];
- Prototipsko softversko rešenje kojim bi se obezbedila implementacija predloženih metoda [*novi softver*];

Društveni doprinos se ogleda u mogućnosti primene predloženih pristupa u raznim domenima a ne samo za razvoj aplikacija i automatizaciju poslovanja. Pre svega treba imati u vidu da servisno-orijentisani pristup razvoju aplikacija predstavlja

opštu paradigmu za obezbeđivanje složene funkcionalnosti povezivanjem skupa komponenti, koji se ne mora nužno zasnivati na web servisima, iako se u praksi najčešće na ovaj način realizuje. Dakle predloženi pristupi bi se mogli primeniti i za selekciju komponenti realizovanih na druge načine. Šta više predloženi pristupi bi se mogli koristiti i u bilo kom drugom polju u kome se ideja povezivanja jedinica funkcionalnosti (koje ne moraju nužno biti softverske) može primeniti.

1.4. Struktura disertacije

U drugom poglavlju su predstavljena osnovna teorijska polazišta vezana za oblast istraživanja. Detaljno je izložen servisno-orijentisani pristup razvoju aplikacija, i objašnjen pojam kompozicije servisa. Zatim je akcenat stavljen na pregled nefunkcionalnih karakteristika servisa sa posebnim osvrtom na način njihove agregacije.

U trećem poglavlju najpre je detaljno definisan problem selekcije servisa zasnovane na nefunkcionalnim karakteristikama kao jedan od značajnih elemenata servisno-orijentisanog pristupa razvoju aplikacija. Zatim se daje opšta matematičku formulaciju problema. Nakon toga je dat osvrt na neke od relevantnih postojećih pristupa modelovanju i motivacija za uvođenje novih pristupa. Na kraju je predloženo rešenje za modelovanje problema selekcije servisa korišćenjem konzistentne fazi logike koja omogućava da se uzmu u obzir specifičnosti nefunkcionalnih karakteristika na osnovu kojih se vrši selekcija, kao i složeni verbalni zahtevi korisnika. Pored toga predlaže se i rešenje za modelovanje problema kao diskretnog problema linearnog razlomljenog programiranja MOLFPP (Multi-Objective Linear Fractional Programming Problem) u slučajevima kada ne postoje logički zahtevi vezani za QoS kriterijume.

U četvrtom i petom poglavlju su izložene mogućnosti primene postojećih metoda iz oblasti optimizacije (koje do sada nisu primenjivane na ovaj problem) ali i predložene nove metode za selekciju kako pojedinačnih servisa tako i za selekciju servisa za zadatu kompoziciju.

U šestom poglavlju su izložene mogućnosti šire primene predloženih rešenja, dok je u sedmom poglavlju dat zaključak i pregled doprinosa disertacije kao i mogući pravci daljeg istraživanja. Na kraju rada je navedena literatura koja je korišćena prilikom izrade disertacije.

2. OSNOVNA TEORIJSKA POLAZIŠTA

Sve veća prisutnost informacionih tehnologija (IT) u svakodnevnom životu neminovno je dovela i do toga da je danas praktično nezamisliv opstanak preduzeća na tržištu bez adekvatne IT podrške poslovanju. U (*Finkelstein, 2004*) autor navodi: „Sredinom XX veka poslovanje industrijskih preduzeća je evoluiralo u kompleksne sekvence manuelnih procesa. Stopa progresa uzrokovala je da preduzeća koriste sve složenije poslovne procese. Krajem 1950-tih procesi se sve više automatizuju pomoću računara. Postojeći procesi su se uglavnom automatizovali takvi kakvi su bez značajnih izmena. Sa sve većom prisutnošću Interneta u drugoj polovini 1990-tih način poslovanja se drastično menja. Preduzeća su sada dostupna na klik mišem, međutim ukoliko im procesi ne obezbeđuju ono što im je potrebno klijenti isto tako jednim klikom mogu napustiti preduzeće.“ U (*Harmon, 2010*) se dalje navodi: „Za samo dve decenije računari su u potpunosti izmenili način poslovanja preduzeća. Već 1980-tih godina postalo je jasno da će u budućnosti skoro svaki proces u svakoj organizaciji ili biti u potpunosti automatizovan ili će ga obavljati ljudi ali uz podršku računara i softvera. Pojava Interneta i web-a je 1995. godine radikalno izmenila način na koji kupci komuniciraju sa preduzećima. Za samo dve godine prešlo se od posmatranja računara kao alata za automatizaciju internih poslovnih procesa ka posmatranju računara kao sredstva za komunikaciju koje omogućava radikalno nove poslovne modele. Internet danas sve više vodi ka ekstenzivnom „outsourcing-u“ procesa i globalnoj integraciji poslovnih aktivnosti.“ Automatizacija poslovanja podrazumeva razvoj softverskih rešenja kojima se pruža podrška obavljanju delatnosti preduzeća i ispunjenju njegovih osnovnih ciljeva.

Međutim, preduzeća su takođe prinuđena da stalno prate nova dostignuća u IT oblasti i u skladu sa njima unapređuju postojeća i uvode nova rešenja kako ne bi izgubila trku sa tehnološki razvijenijom konkurencijom. Savremeni uslovi u sve dinamičnijem poslovnom okruženju kao i povećanje složenosti samog poslovanja nameću potrebu za razvojem rešenja koja su:

- Fleksibilna i Agilna – omogućavaju lako i brzo prilagođavanje novim zahtevima koji su posledica promena u okruženju (tržištu, konkurenciji, zakonskoj regulativi, potrebama korisnika, novim tehnologijama, itd.),

- Skalabilna – omogućavaju opsluživanje rastućeg broj simultanih korisnika kroz jednostavno dodavanje novih resursa,
- Distribuirana – omogućavaju da se delovi funkcionalnosti izvršavaju na različitim računarima (koji se pri tome mogu nalaziti i na geografski veoma udaljenim lokacijama) umesto da se ogromna aplikacija izvršava na jednom jedinom računaru koji u slučaju velikog broja korisnika postaje usko grlo,
- Stabilna i Pouzdana – obezbeđuju da prestanak rada jednog računara ne utiče na celu aplikaciju,
- Prilagođena heterogenom okruženju – u smislu različitih platformi, operativnih sistema, programskih jezika, itd.,
- Interoperabilna – omogućavaju zajedničko funkcionisanje heterogenih aplikacija (eksternih aplikacija, nasleđenih rešenja (*eng. legacy solutions*) itd.) u distribuiranom okruženju.
- i koja se mogu razviti i uvesti u kratkom vremenskom roku.

Prema važećem međunarodnom standardu (*ISO/IEC 12207*) jedan od prvih koraka u procesu razvoja softvera je prikupljanje i analiza zahteva svih zainteresovanih strana (*eng. stakeholders*), vezanih za ponašanje budućeg sistema uzimajući u obzir poslovne ciljeve, na osnovu kojih se zatim definišu odgovarajući funkcionalni zahtevi. Na osnovu izvršene analize definišu se odgovarajući funkcionalni zahtevi, kojima se opisuje očekivano ponašanje budućeg sistema.

Sledeći korak je izbor odgovarajuće softverske arhitekture kojom se zapravo uspostavlja struktura budućeg rešenja i to na najvišem nivou apstrakcije odnosno bez ulaženja u pojedinosti vezane za način implementacije. Neke od najčešće citiranih definicija softverske arhitekture su:

> „Arhitektura predstavlja fundamentalnu organizaciju sistema otelotvorenu u njegovim komponentama, njihovim međusobnim odnosima, i odnosima prema okruženju, kao i principe koji upravljaju njegovim projektovanjem i evolucijom.“ (*IEEE, 2000*).

„Softverska arhitektura obuhvata skup značajnih odluka vezanih za organizaciju softverskog sistema uključujući:

- izbor strukturnih elemenata (i njihovih interfejsa) pomoću kojih će se sistem komponovati,
- ponašanje koje se specificira kolaboracijom tih elemenata,
- način kompozicije elemenata strukture i ponašanja u veće podsisteme i
- arhitektonski stil koji usmerava proces organizovanja elemenata i njihovih interfejsa, kolaboracija i kompozicija.

Međutim, softverska arhitektura nije orijentisana samo na strukturu i ponašanje, već i na upotrebu, funkcionalnost, performanse, otpornost, mogućnost ponovnog korišćenja, razumljivost, ekonomska i tehnološka ograničenja i kompromise, i estetiku.“ (*Grady Booch, Philippe Kruchten, Rich Reitman i Kurt Bittner ,1995* kao proširenje *Shaw & Garlan, 1996*)

„Softverska arhitektura jednog sistema predstavlja strukturu ili strukture tog sistema, sastavljene od softverskih elemenata, eksterno vidljivih svojstava tih elemenata i veza između njih.“ (*Bass, Clements, & Kazman, 2003*)

Dakle, arhitekturom se uspostavlja struktura odnosno organizacija budućeg rešenja kroz definisanje potrebnih elemenata (modula odnosno komponenti) i njihovih međusobnih veza (pomoću kojih se obezbeđuje zahtevano ponašanje sistema) i to na visokom nivou apstrakcije, odnosno fokus je samo na značajnim elementima bez ulaženja u detalje vezane za način njihove realizacije. Kao što je napomenuto u (*Eeles, 2006*) treba imati u vidu da se svaki sistem zasniva na nekoj arhitekturi, čak i kada je reč o veoma jednostavnom sistemu koji se na primer sastoji od jednog jedinog elementa.

Jedan od osnovnih principa koji se primenjuje prilikom izbora načina organizacije aplikacije je princip jasnog razdvajanja nadležnosti (*eng. separation of concerns*) čija je osnovna ideja da se složeni problem može lakše razumeti pa samim tim i rešiti raščlanjavanjem na skup manjih problema (*Dijkstra, 1982*). Cilj je dakle, smanjiti kompleksnost sistema te se u skladu sa ovom principom razvoj aplikacija zapravo zasniva na razvoju skupa softverskih komponenti koje obuhvataju manje celine

poslovne logike. Treba napomenuti da će se pojam komponente u daljem tekstu koristiti za označavanje jedinice funkcionalnosti, a ne u smislu načina implementacije te funkcionalnosti kao grupe povezanih klasa o kojima će biti reč u sledećem poglavlju.

Funkcionalna dekompozicija sistema treba da se zasniva na dva osnovna principa dobrog programiranja definisana u (*Stevens, Myers, & Constantine, 1974*):

- Povezanost (*eng. coupling*) ukazuje na stepen zavisnosti između komponenti. Visok stepen povezanosti ukazuje na to da su komponente međusobno zavisne, dok slaba povezanost (*eng. loose coupling*) ukazuje na to da su komponente potpuno ili skoro nezavisne. Stoga izmene u jednoj komponenti neće uzrokovati promene, neočekivane efekte ili greške u drugim komponentama. Slaba povezanost omogućava bolje razumevanje, lakše održavanje i mogućnost višestrukog korišćenja komponenti.
- Kohezija (*eng. cohesion*) ukazuje na stepen povezanosti funkcija unutar komponente. Drugim rečima, stepen kohezije ukazuje na to koliko su zadaci koji se izvršavaju unutar komponente funkcionalno povezani. Svi elementi unutar komponente treba da se odnose na isti zadatak, stoga u dobro projektovanom sistemu postoji visok stepen kohezije između elemenata komponente. Nizak stepen kohezije ukazuje na to da modul obavlja zadatke koji nisu međusobno veoma povezani.

Drugim rečima, dok stepen povezanosti ukazuje na odnose između komponenti, kohezija ukazuje na odnose unutar pojedinačnih komponenti. Treba napomenuti da je slaba povezanost obično u korelaciji sa visokim stepenom kohezije i obrnuto.

Prilikom dekompozicije sistema na komponente trebalo bi težiti što većem stepenu kohezije elemenata unutar komponente i istovremeno što manjem stepenu povezanosti između samih komponenti. To znači da bi trebalo definisati komponente koje su odgovorne samo za jednu konkretnu funkcionalnost ili skup usko povezanih (visoko kohezivnih) funkcionalnosti odnosno, koje implementiraju jednu logičku celinu (pri čemu svaki deo komponente doprinosi datoj funkcionalnosti) i koje se, što je moguće manje, preklapaju. Naravno obezbeđivanje zahtevanog ponašanja sistema podrazumeva da je neophodno obezbediti i neki način komunikacije samih komponenti, pri čemu je poželjno da ova komunikacija ne zavisi od načina na koji su komponente realizovane.

Slaba povezanost komponenti postiže se apstrakcijom odnosno sakrivanjem detalja vezanih za interne mehanizme samih komponenti. Osnovna premisa ovog principa, predloženog po prvi put u (*Parnas, 1972*), se ogleda u tome da nije neophodno poznavanje svih detalja vezanih za funkcionisanje određene komponente da bi se ona mogla koristiti. Da bi se ovo postiglo potrebno je za svaku komponentu obezbediti interfejs (pomoću koga će ona izložiti svoju funkcionalnost) i koji će omogućiti povezivanje komponenti nezavisno od načina njihove implementacije. Interfejsi pružaju pogled, višeg nivoa apstrakcije, na operacije koje komponenta može da izvrši ali ne sadrže nikakve informacije o unutrašnjim karakteristikama i načinu funkcionisanja same komponente. Pored toga, oni treba da sadrže i informacije o tome kako se izloženoj funkcionalnosti može pristupiti kao i druge pojedinosti vezane za njeno korišćenje u smislu očekivanih ulaza i izlaza, preduslova i post-uslova, pratećih efekata, mogućih izuzetaka, karakteristika vezanih za performanse itd. Drugim rečima, interfejsima se jasno propisuje skup funkcionalnosti koje data komponenta pruža kao i način korišćenja izložene funkcionalnost dok sama implementacija (način na koji se funkcionalnost realizuje) ostaje sakrivena. Na ovaj način omogućava se lako menjanje same komponente s obzirom na to da izmene u načinu njene implementacije neće povlačiti promene u celom sistemu, dokle god ta promena ne zahteva i promenu interfejsa date komponente. Takođe je moguće i celu komponentu zameniti drugom komponentom koja obezbeđuje istu funkcionalnost (odnosno, pruža isti interfejs) ali je implementirana na drugačiji način. Komunikacija između komponenti postiže se pozivanjem odgovarajuće funkcionalnosti na propisani način.

Još jedan zahtev, koji se sve više postavlja, je mogućnost ponovnog korišćenja (*eng. reusability*) komponenti. U (*Brathwaite, 1990*) se navodi:

> „Mogućnost ponovnog korišćenja kôda je ključ za dramatično povećanje produktivnosti softvera budući da se vreme razvoja aplikacije može skratiti za nekoliko meseci ili čak godina (čime se smanjuju i troškovi razvoja), dok se sa druge strane poveća pouzdanost rešenja. Međutim, mogućnost ponovnog korišćenja softvera na nivou kôda je veoma ograničena jer zavisi od upotrebljenog jezika, operativnog sistema i same aplikacije, dok je ponovno korišćenje kôda u aplikacijama koje nisu slične često skoro nemoguće. Ponovno korišćenje kôda često zahteva da se u njemu izvrše

određene izmene, pre nego što se on može ponovo upotrebiti, a izmena nosi rizik uvođenja grešaka i nepredviđenih pratećih efekata. Stoga je akcenat na ponovnom korišćenju programskih specifikacija umesto samog kôda. Na nivou programske specifikacije nestaju problemi vezani za zavisnost od jezika, operativnog sistema i nedostaci vezani za sam kôd. Mogućnost ponovnog korišćenja uvodi revoluciju u proces razvoja softvera."

Mogućnost ponovnog korišćenja komponenti postiže se pre svega visokim stepenom kohezije unutar same komponente ali i istovremenom slabom povezanošću komponente sa drugim komponentama.

Dakle, prilikom projektovanja aplikacije najpre je potrebno utvrditi kako se funkcionalnost može logički podeliti na komponente. Zatim se identifikuju elementi unutar tih komponenti, definišu funkcionalni zahtevi za svaki od elemenata i propisuju metode koje će omogućiti komunikaciju između komponenti. Na kraju sledi interno projektovanje svakog od elemenata.

Konačno treba napomenuti da na izbor arhitekture utiču kako zainteresovani učesnici tako i okruženje. Okruženje zapravo nameće okvire unutar kojih sistema treba da funkcioniše što može imati velikog uticaj na izbor arhitekture. Najpre u pogledu internih ograničenja (organizacija preduzeća, IT kadrovi i njihove veštine, raspoloživa tehnologija, budžet itd.) a zatim i u pogledu eksternih ograničenja (kao što je, pre svega, potreba da se omogući interakcija sa drugim sistemima).

2.1. Pristupi razvoju aplikacija

Arhitektura može biti definisana u skladu sa nekim arhitektonskim pristupom (jednim ili više) koji zapravo predstavlja šablon (*eng. pattern*) odnosno skup smernica za organizovanje elemenata sistema. Kako u domenu softverskog inženjerstva, šablon predstavlja opšte rešenje za neki opšti problem u određenom datom kontekstu, tako i arhitektonski pristupi (*eng. architectural style*) zapravo predstavljaju skup praksi koje se koriste za izbor, definisanje i projektovanje softverske arhitekture. Odnosno:

„Arhitektonski stil predstavlja familiju sistema u pogledu obrazaca vezanih za način organizacije njihove strukture. Konkretnije rečeno, arhitektonski stil definiše rečnik komponenti i načina povezivanja tih komponenti, kao i skup ograničenja vezanih za moguće načine njihovog kombinovanja." (*Shaw & Garlan, 1996*).

Prema (*Microsoft Patterns & Practices Team, 2009*) arhitektonski stil obezbeđuje apstraktni okvir (*eng. framework*) za celu familiju sistema i zapravo predstavlja skup principa koji usmeravaju način strukturiranja sistema.

Konstantni napredak informacionih tehnologija povlači i potrebu za konstantnim preispitivanjem postojećih i definisanjem novih pristupa za razvoj softverskih rešenja. Stoga su danas prisutni mnogobrojni pristupi za razvoj aplikacija koji uzimaju u obzir principe i zahteve navedene u prethodnom potpoglavlju.

Objektno-orijentisani pristup je 1980-tih godina uveo revoluciju u načinu strukturiranja aplikacija budući da se u njemu sistem posmatra kao skup objekata (kojima se modeluju entiteti iz realnog sveta) koji su u međusobnoj interakciji umesto tradicionalnog načina posmatranja sistema kao skupa rutina ili proceduralnih instrukcija. Za razliku od tradicionalnog pristupa koji se zasnivao na dekompoziciji programa u **funkcije**, manje programe kojima se može pristupiti pomoću programskih interfejsa (*eng. Application Programming Interface* – API), objektno-orijentisani pristup se zasniva na podeli odgovornosti u jedinice funkcionalnosti – **objekte** kojima se modeluju entiteti iz realnog sveta (relevantni za dati sistem), i koje obuhvataju podatke i ponašanje datog entiteta. Objekti međusobno komuniciraju pozivanjem metoda ili pristupajući svojstvima drugih objekata, putem interfejsa, i razmenom poruka. Ovako razvijene jedinice funkcionalnosti su nezavisne, veoma kohezivne, lako proširive i mogu se ponovo koristiti. Sa druge strane, razvoj aplikacija na ovaj način mnogo je bliži načinu na koji realni svet funkcioniše, a rezultat primene takvog pristupa je da je i sam kôd mnogo razumljiviji.

Arhitekture zasnovane na komponentama su 1990-tih godina pažnju su usmerile na podelu sistema u jedinice funkcionalnosti – **komponente** koje imaju dobro definisane interfejse za komunikaciju (pomoću kojih izlažu svoju funkcionalnost). Komponente predstavljaju viši nivo apstrakcije u odnosu na objekte. Odnosno, prema (*MacDonald, 2003*) u objektno-orijentisanom pristupu, funkcionalnost programa se razbija na logičke jedinice (klase) kojima se modeluju entiteti iz realnog sveta, dok komponente predstavljaju klase koje imaju širu namenu odnosno, sa tehničkog aspekta jedna ili više klasa su učaurene u komponenti. Osnovna karakteristika ovog pristupa ogleda se u definisanju komponenti koje se mogu ponovo upotrebiti i koje su nezavisne

od drugih komponenti, kako bi se mogle koristiti kako u drugim okruženjima tako i u drugim aplikacijama. Ideja je da se omogući da se za razvoj novih aplikacija iskoriste i već postojeće komponente (bilo sopstvene bilo nabavljene), koje pri tome mogu biti razvijene i u drugim programskim jezicima, čime se značajno smanjuje vreme razvoja nove aplikacije a samim tim i troškovi. Pored toga, budući da je jedan od zahteva ovog pristupa i minimalna zavisnost između komponenti primena ovog principa obezbeđuje mogućnost lake izmene ili čak i zamene pojedinačnih komponenti a da to ne utiče na druge komponente u sistemu. Uz to komponente se projektuju tako da budu i proširive kako bi mogle obezbediti novo ponašanje. Međutim, ovaj pristup zahteva i postojanje mehanizma unutar platforme koji će upravljati lociranjem odgovarajućih komponenti i njihovih interfejsa kao i razmenom poruka između njih.

U višeslojnom pristupu za razvoj aplikacija namera je da se, pre svega, izvrši jasna logička podela funkcionalnosti aplikacije u **slojeve** (*eng. layer*), kako bi se jasno razdvojili delovi aplikacije koji su vezani za različite zadatke. Aplikacija se dakle, može posmatrati kao skup međusobno povezanih procesa, gde je svaki proces implementiran u odvojenom sloju. U tom smislu klasična troslojna arhitektura podrazumeva podelu aplikacije na tri dela: prezentacionu logiku, poslovnu logiku i logiku vezanu za pristup podacima. Pri čemu je, u zavisnosti od vrste aplikacije, moguće dodati i više slojeva na primer, kreiranjem odvojenih slojeva za svaki poseban zadatak. Dobrim utvrđivanjem granica odgovornosti postiže se visok stepen kohezije unutar sloja jer svaki sloj sadrži samo funkcionalnost koja je direktno povezana sa zadacima tog sloja. Komunikacija između slojeva je eksplicitna i zasnovana na interfejsima i događajima (čime se obezbeđuje slaba povezanost) i u opštem slučaju jedinice funkcionalnosti u određenom sloju mogu komunicirati samo sa jedinicama iz istog sloja ili sloja koji se nalazi direktno ispod. Sa druge strane, budući da su niži slojevi potpuno nezavisni od viših postoji mogućnost njihovog ponovnog korišćenja u drugim kontekstima. Prednost razdvajanje aplikacije na slojeve ogleda se u tome što se delovi aplikacije mogu nezavisno razvijati (čak i u različitim jezicima ukoliko je to potrebno), testirati i, što je još važnije, izmeniti (u slučaju promene u zahtevima ili raspoloživoj tehnologiji) a da to ne iziskuje nužno i izmenu celog kôda. Na primer, moguće je lako promeniti način pristupa podacima ili način prezentacije podataka (na primer web aplikacija ili desktop aplikacija) bez menjanja poslovne logike, i obrnuto.

Treba napomenuti da ovakav način podele aplikacije ne ukazuje i na to gde će se delovi aplikacije izvršavati, da li na istom računaru ili na različitim računarima. Međutim, uvođenjem komponenti ili višeslojne arhitekture otvara se prostor i za razmatranje mogućnosti da se svaka od komponenti, odnosno svaki od slojeva, izvršava na drugom računaru čime se postiže distribuiranost aplikacije[1]. Distribuirane aplikacije se sastoje od skupa komponenti koje su smeštene na različitim računarima i koje razmenjuju informacije preko mreže, pri čemu one mogu međusobno komunicirati na različite načine. Fizičko razmeštanje slojeva se postiže dodavanjem **nivoa** (*eng. tiers*). Drugim rečima, dok su slojevi vezani za logičku podelu aplikacije, nivoi se odnose na njenu fizičku podelu. Svaki nivo je u potpunosti nezavisan od svih drugih nivoa osim onih koji se nalaze direktno iznad i ispod njega dok je komunikacija između nivoa obično asinhrona kako bi se omogućila skalabilnost. Osnovne prednosti uvođenja nivoa ogledaju se u mogućnosti optimalnog korišćenja fizičkih resursa (što vodi i boljim performansama budući da izvršavanje neke komponente koja zahteva značajne resurse neće usporiti ili čak sprečiti izvršavanje drugih komponenti), većoj fleksibilnosti, lakšem upravljanju i iznad svega skalabilnosti.

Međutim, sve veća prisutnost Interneta u svakodnevnom životu sa jedne strane, i sve prisutniji trend globalizacije zajedno sa sve zahtevnijim uslovima poslovanja sa druge, doveli su i do drastičnih promena u načinu poslovanja, udruživanja poslovnih subjekata i potrebe za prelaskom na elektronsko poslovanje u svim njegovim vidovima (Business-to-Business B2B, Business-to-Consumer B2C, Business-to-Government B2G itd.). Ove promene su neminovno dovele i do potrebe za mnogo fleksibilnijim softverskim rešenjima, posebno ako se ima u vidu da je potrebno na neki način integrisati sve te različiti sisteme koji su realizovani na mnoštvu različitih platformi, jezika i tehnologija i obezbediti mogućnost njihove interakcije putem Interneta budući da oni mogu biti geografski veoma udaljeni.

[1] Pri tome je preporuka da bi aplikaciju uvek trebalo projektovati kao višeslojnu bez obzira na to kakav će se način fizičkog razmeštanja slojeva naknadno izabrati (*MacDonald, 2003*).

2.2. Servisno-orijentisani pristup razvoju aplikacija

Sposobnost brzog prilagođavanja promenama u okruženju i neprekidno usvajanje novih tehnologija postaju nezaobilazni preduslovi za održanje konkurentske prednosti u sve dinamičnijim uslovima savremenog poslovanja. Sa druge strane, konstantni napredak informacionih tehnologija i sve veća prisutnost Interneta takođe dovode do promena u načinu poslovanja preduzeća. Promene u zahtevima vezanim za poslovanje, kao i promene u raspoloživoj tehnologiji, mogu iziskivati i značajne izmene pratećih softverskih rešenja (dodavanje novih funkcionalnosti, kao i proširivanje, izmena ili izbacivanje postojećih). Promene u poslovanju koje zahtevaju izmene u razvijenim rešenjima dovode do problema održavanja (*eng. maintenance*). Prema (*Brathwaite, 1990*) pod održavanjem se podrazumeva bilo koja izmena programa nakon što je on razvijen i pušten u upotrebu i autor navodi da 80% ukupnog troška uvođenja programa odlazi na održavanje. Po njemu razlog leži u kompetetivnoj prednosti koju revitalizacija softvera obezbeđuje preduzeću (odnosno „nije dovoljno uraditi nešto, već je potrebno to uraditi brže, bolje, jeftinije i efikasnije od konkurencije“). U (*Rosen & al, 2008*) se takođe navodi da između 70 i 90 procenata IT budžeta odlazi na održavanje, 15-30% na integraciju dok samo 5-15% odlazi na razvoj novih aplikacija.

U većini prethodno razmatranih pristupa razvoju aplikacija automatizacija poslovanja se zasnivala na identifikovanju poslovnih zadataka koje je potrebno automatizovati, definisanju poslovnih zahteva i zatim razvijanju odgovarajućih rešenja. Međutim u tako razvijenim rešenjima funkcionalnost samih komponenti obično je usko povezana sa konkretnim poslovnim zahtevima ili procesima, odnosno sa ulogom te komponente u samoj aplikaciji, što umanjuje fleksibilnost celokupnog rešenja budući da je veoma teško, ako ne i nemoguće, zameniti određenu komponentu drugom. Sa druge strane, ovakav pristup takođe ne omogućava da se određena komponenta, koja je razvijena imajući u vidu jednu potrebu, ponovo iskoristi za ispunjenje neke druge potrebe čime bi se značajno smanjili troškovi razvoja.

Iako je često, u slučaju da je došlo do značajnih promena, lakše napustiti postojeće rešenja i početi razvoj novih rešenja iz početka retki su slučajevi kada je to moguće (budući da raspoloživo vreme, budžet i mogućnosti najčešće to ne dozvoljavaju) već se obično postojeća rešenja nadograđuju što dovodi do problema integracije novih rešenja sa postojećim (*eng. legacy*) rešenjima. Pored toga dodatni

problem se javlja ukoliko se za razvoj rešenja, sa ciljem smanjena troškova i vremena razvoja, kupuju i gotove komponente, koje pri tome mogu biti realizovane na različitim platformama i u različitim programskim jezicima, a koje je potrebno integrisati u postojeća rešenja. Problem integracije postojećih komponenti (bile one nasleđene ili nabavljene) neminovno otvara pitanje i mogućnosti njihovog prilagođavanja a pre svega ako se ima u vidu raznorodnost tehnologija koje su korišćene za njihov razvoj pri čemu neke od tih tehnologija mogu biti i zastarele. Dodatni izazov predstavlja potreba da se omogući integracija poslovanja i sa drugim poslovnim subjektima imajući u vidu da su IT infrastrukture različitih preduzeća najčešće veoma heterogene.

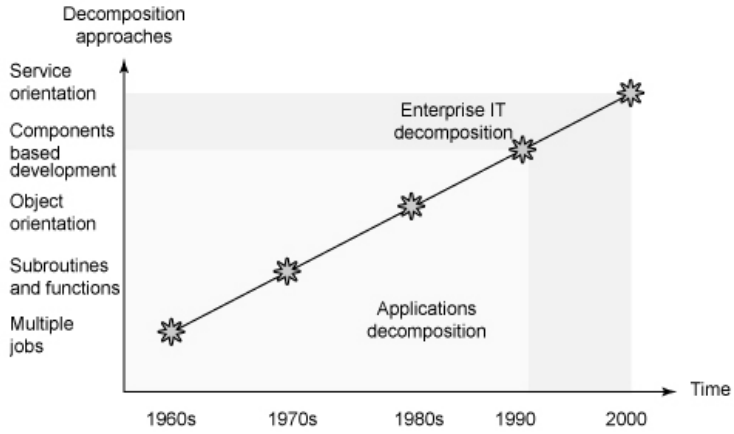
Shodno navedenim razmatranjima može se zaključiti da automatizacija poslovanja treba da se zasniva na rešenjima koja neće predstavljati prepreku, odnosno ograničavati, mogućnost uvođenja promena i koja se pri tome mogu lako integrisati sa postojećim rešenjima ali i sa poslovanjem drugih poslovnih subjekata. Drugim rečima, dobro projektovano rešenje je dovoljno fleksibilno da može da podrži promene do kojih neminovno dolazi kako i u zahtevima, tako i u raspoloživoj hardverskoj i softverskoj tehnologiji.

Servisno-orijentisani pristup razvoju aplikacija pripada novoj generaciji pristupa za razvoj fleksibilnih distribuiranih aplikacija. Cilj servisno-orijentisanog pristupa je da omogući razvoj rešenja koja se mogu kreirati, proširivati i menjati prema potrebi (*eng. on demand*). Ovo je posebno važno imajući u vidu da preduzeća čije je poslovanje bazirano na softverskim rešenjima koja su skrojena isključivo prema postojećim potrebama neće biti u mogućnosti da se brzo prilagode novim okolnostima čime gube konkurentsku prednost. Ili kao što je navedeno u (*Finkelstein, 2004*):

> „Zasnivanje budućih potreba na postojećim poslovnim procesima implicitno podrazumeva da će budućnost biti slična prošlosti. Sa druge strane jedino što je izvesno je da će se procesi koje ćemo obavljati u budućnosti veoma razlikovati od sadašnjih procesa. Stoga je neophodno projektovati rešenja koristeći aktivnosti i procese koje se mogu ponovo koristiti i koji su skrojeni prema okruženju koje se zasniva na Internetu kako bi preduzeća bila u mogućnosti da reaguju u sekundama ili minutima a ne u danima ili nedeljama. Moramo projektovati za budućnost u kojoj je jedina konstanta sama promena. “

Ključna razlika servisno-orijentisanog pristupa u odnosu na prethodno izložene pristupe se ogleda u tome što on podrazumeva da je razvoj aplikacija zapravo vođen poslovanjem odnosno poslovnim procesima organizacije. Prema (*Harmon, 2010*) do početka 1990-tih IT je u najvećoj meri bio nezavisan od samog poslovanja i više je predstavljao uslugu koja se nudi preduzeću. Konstantni napredak informacionih tehnologija je neminovno doveo do toga da su one sve više i više ne samo nezaobilazni deo savremenog poslovanja već i značajan pokretač promena u načinu samog poslovanja. Naime, kako se navodi u (*Weske, 2012*) informacione tehnologije, a pre svega informacioni sistemi danas zauzimaju značajno mesto u upravljaju poslovnim procesima budući da je svakim danom sve veći broj aktivnosti preduzeća koje se obavljaju uz podršku informacionih sistema. Pored toga informacioni sistemi omogućavaju i kompletnu automatizaciju određenih poslovnih procesa. Međutim, autor dalje navodi da u mnogim preduzećima postoji veliki raskorak između organizacionih aspekata poslovanja i primenjene informacione tehnologije. Prema (*Finkelstein, 2004*) kod tradicionalnih pristupa razvoju softverskih rešenja zahteve obično specificiraju IT eksperti u komunikaciji sa korisnicima, kako bi utvrdili njihove operativne poslovne potrebe, te tako projektovana rešenja zapravo zavise od raspoložive tehnologije. Pored toga treba istaći da ovi pristupi zapravo ne obezbeđuju podršku za upravljanje poslovnim procesima odnosno upravljanjem tokom njihovog izvršavanja (*eng. workflow*) pa samim tim ni mogućnost da se dati tok automatizuje. Sa druge strane treba imati u vidu i da se same tehnologije neprekidno menjaju.

Stoga je očigledno da je, za ostvarivanje konkurentske prednosti i opstanak na tržištu, danas neophodno istovremeno usmeriti pažnju kako na poslovne procese preduzeća (neprekidno ih unapređujući i usklađujući sa definisanim strategijama i ciljevima) tako i na softverska rešenja (stalnim usvajanjem novih tehnologija koje će obezbediti podršku izvršavanju poslovnih procesa). Kako se navodi u (*SOA Manifesto, 2009*), jedan od najvažnijih ciljeva servisno-orijentisanog pristupa je razvoj softverskih rešenja koja će omogućiti organizacijama da se brzo i lako prilagode neprekidnim promenama, kako bi se dugoročno gledano ostvarila održiva poslovna vrednost (*eng. sustainable business value*). Prepoznajući potrebu da se premosti jaz između poslovnih potreba koje se neprekidno menjaju i softverskih rešenja koja treba da obezbede odgovarajuću podršku poslovanju, fokus se premestio na poslovanje preduzeća:



Slika 1. Evolucija modela distribuiranog računarstva (*Shahid, 2008*)

Dakle, razvoj softverskih rešenja zasnovan na servisno-orijentisanom pristupu zapravo zahteva da se na osnovu postavljenih poslovnih ciljeva i strategija najpre definišu poslovni procesi. Naime, kako se navodi u (*Finkelstein, 2004*), ako poslovni zahtevi nisu na adekvatan način opisani razvijeno rešenje potencijalno neće zadovoljavati stvarne potrebe korisnika. U tu svrhu danas se najčešće koristi *Business Process Modeling – BPM* koji predstavlja metodologiju koja omogućava da se opišu različiti aspekti nekog poslovnog procesa pri čemu propisuje veći broj standarda koji omogućavaju da se dati proces grafički prikaže u vidu toka (*eng. flow*) skupa aktivnosti (*Havey, 2005*). Drugim rečima, prilikom razvoja rešenja polazi se od modela poslovnih procesa kojima se poslovni procesi identifikuju i formalno definišu. Pri tome sa poslovne tačke gledišta nije od značaja da li se dati proces obavlja ručno ili se može delimično ili čak potpuno automatizovati. Na osnovu kreiranih modela poslovnih procesa se zatim mogu definisati funkcionalnosti koje servisi treba da obezbede nezavisno od toga na koji način će se oni zapravo implementirati. Prednost ovakvog pristupa se ogleda u tome što omogućava ekspertima iz poslovnog i IT domena da komuniciraju na lakši, efikasniji i efektivniji način.

Međutim treba istaći da, iako se servisno-orijentisani pristup i BPM često prepliću i koriste zajedno, osnovna ideja servisno-orijentisanog pristupa se zapravo ogleda u razvoju čitavog niza jedinica funkcionalnosti (servisa), kojima se rešavaju određeni zadaci, ali koje su pri tome nezavisne od šireg konteksta odnosno konkretnog poslovnog procesa. Na ovaj način otvara se prostor za mogućnost kombinovanja tih funkcionalnosti na različite načine za zadovoljenje različitih potreba.

Treba napomenuti da je servisna-orijentacija zapravo apstraktna paradigma te da se ovaj opšti princip ne odnosi samo na servise koji su realizovani pomoću softverskih rešenja.

Kada je reč o razvoju softverskih rešenja, ono što razlikuje servisno-orijentisani pristup od drugih pristupa je dakle način na koji se aplikacija deli na funkcionalne celine. Servisno-orijentisana paradigma se zapravo zasniva na razvoju mnoštva nezavisnih softverskih komponenti (koje obuhvataju manje celine poslovne logike), a koje su pri tome razvijene na takav način da se mogu više puta koristiti i lako međusobno povezivati. Da bi se ovo obezbedilo paradigmom je propisan skup principa koji utvrđuju način dekompozicije funkcionalnosti aplikacije na manje logičke celine kao i pravila (odnosno preporučenih smernica) za razvoj samih komponenti (*Erl, 2008*). Primena ovih principa prilikom razvoja rezultuje jedinicama funkcionalnosti (**servisima**) koje su u potpunosti nezavisne kako od drugih jedinica funkcionalnosti tako i od okruženja (*Service Loose Coupling Principle*) i autonomne u smislu da imaju veliki stepen kontrole nad svojim izvršnim okruženjem (*Service Autonomy Principle*) te se mogu pozicionirati kao resursi preduzeća koji se mogu lako ponovo koristiti (*Service Reusability Principle*), ne zahtevajući pri tome jaku povezanost sa korisnicima, i koji se mogu lako komponovati kako bi zajedno pružili složeniju funkcionalnost (*Service Composability Principle*). Dakle, osnovni elementi (tj. gradivni blokovi) softverskih rešenja u servisno-orijentisanom pristupu su servisi.

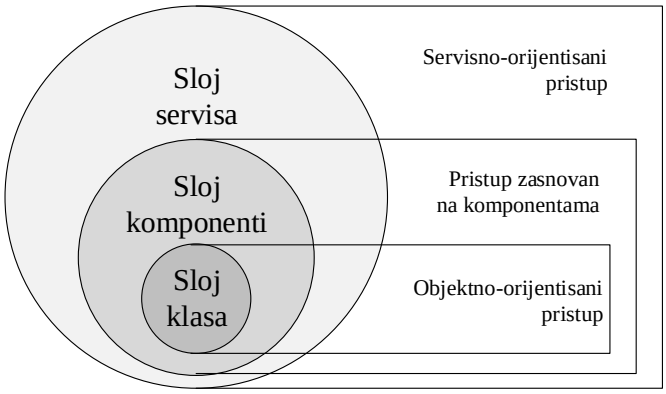
Servisno-orijentisana arhitektura (SOA) predstavlja model arhitekture za razvoj softverskih rešenja u skladu sa servisno-orijentisanim pristupom (*Chou & al, 2010*). S obzirom na to da se danas često pojam servisno-orijentisane arhitekture izjednačava sa pojmom servisno-orijentisanog pristupa grupa vodećih autora iz ove oblasti je 2009. godine formulisala formalnu deklaraciju kojom nastoje da ustanove jasno razdvajanje ova dva pojma: „*Servisno-orijentisana arhitektura je vrsta arhitekture koja je rezultat primene servisne orijentacije*" (*SOA Manifesto, 2009*). Dakle, SOA predstavlja vrstu arhitekture koja omogućava razvoj aplikacije kroz fleksibilno programsko povezivanje (kompoziciju) nezavisno razvijenih softverskih komponenti – servisa. Pri tome treba napomenuti da su servisna-orijentacija kao apstraktna paradigma i SOA kao tehnološka arhitektura zapravo nezavisni od načina implementacije samih servisa.

Svakom servisu se dodeljuje jedinstveni funkcionalni kontekst i svaki se sastoji od skupa mogućnosti (*eng. capability*) koje su u vezi sa datim kontekstom. Pri tome, oni mogu obavljati funkcije koje se kreću od jednostavnih zadataka pa sve do složenih poslovnih procesa, odnosno drugim rečima, pojedinačni servis može pružati i skup mogućnosti koje su grupisane zbog toga što se odnose na isti cilj odnosno funkcionalni kontekst uspostavljen datim servisom. To znači da servis u osnovi igra ulogu omotača za skup funkcionalnosti koje se odnose na isti funkcionalni kontekst (*Erl, 2008*).

Ono što razlikuje servis od drugih oblika modularnosti (kao što su objekti, komponente, moduli) je najpre to što on zapravo predstavlja neku poslovnu funkciju, odnosno omogućava grupisanje funkcionalnosti sa stanovišta funkcionalnog konteksta a ne sa stanovišta fizičkih svojstava i ponašanja nekog konkretnog entiteta iz realnog sveta, pa samim tim omogućava da se na adekvatniji način opišu poslovni procesi. Dakle, iako servisno-orijentisani pristup razvoju aplikacija podržava već utemeljene opšte principe dobrih softverskih arhitektura (kao što su sakrivanje informacija, modularnost i razdvajanje nadležnosti) koji su prisutni i u objektno-orijentisanom pristupu, kao i u pristupu zasnovanom na komponentama između njih ipak postoji razlika. Naime, servisno-orijentisani pristup je usmerena na poslovne procese i na to kako se oni mogu mapirati u servise, za razliku od arhitektura zasnovanih na komponentama u kojima je razvoj usmeren sam kôd. Prema (*Zimmermann & al, 2004*) osnovni nedostatak objektno-orijentisanog pristupa projektovanju je u tome što je nivo granularnosti usmeren na klase koje se nalaze na suviše niskom nivou apstrakcije od onog koji je potreban za modelovanje servisa. Autori navode da jake veze između klasa (kao što je nasleđivanje) dovode do čvrstog povezivanja (*eng. tight coupling*) a samim tim i zavisnosti između strana koje su uključene, dok nasuprot tome, servisno-orijentisana paradigma nastoji da promoviše fleksibilnost i agilnost kroz slabo povezivanje (*eng. loose coupling*) odnosno razvoj servisa koji su međusobno potpuno nezavisni[1]. Slika 2 ilustruje odnos između navedenih pristupa projektovanju aplikacija sa aspekta nivoa apstrakcije na koji su usmereni. Pored toga, treba istaći i da servisno-orijentisani pristup ne nameće bilo kakva ograničenja u pogledu alata i jezika koji će se koristiti za razvoj servisa niti platforme na kojoj će se oni izvršavati, dok se sa druge strane, kako se navodi u (*Chou & al, 2010*) razvoj i izvršavanje komponenti se obično

[1] Međutim autori takođe navode da se objektno-orijentisani pristup može uspešno koristiti za projektovanje strukture samog servisa.

zasniva na platformski zavisnim tehnologijama, te integracija postojećih komponenti u novo rešenje zahteva da date komponente budu kompatibilne pri čemu se ona najčešće vrši fizičkom ugradnjom odnosno pisanjem dodatnog kôda kojim se data komponenta integriše u rešenje.



Slika 2. Odnos između različitih pristupa projektovanju aplikacija (prema ***Zimmermann & al, 2004*****)**

U različitim pristupima razvoju softvera (strukturnim, objektno-orijentisanim, servisno-orijentisanim) princip minimalne povezanosti i maksimalne kohezije pojedinih elemenata sistema (modul, klasa, komponenta,...) se definiše i postiže na različite načine. U metodama strukturnog projektovanja kohezivnost i povezanost modula definišu se u hijerarhijskom dijagramu modula. Maksimalna kohezivnost i minimalna povezanost u objektno-orjentisanim pristupima postiže se korišćenjem velikog broja uzora (šablona). Koncept nasleđivanja (odnosno generalizacije) u objektno-orjentisanom pristupu zapravo povećava povezanost između komponenti nekog softverskog sistema. U servisno-orijentisanom pristupu ne koristi se nasleđivanje već se složeni servisi, kao komponente velike funkcionalne granularnosti, grade isključivo kompozicijom jednostavnijih servisa.

Konačno, identifikovanje i definisanje servisa treba pre svega da se zasniva na osnovnom cilju servisno-orijentisanog pristupa – mogućnosti njihovog ponovnog korišćenja, odnosno ukoliko se neka jedinica funkcionalnosti ne može ponovo iskoristiti ona zapravo ne predstavlja servis (*Zimmermann & al, 2004*).

Servisi se na konceptualnom nivou projektuju kao komponente koje se sastoje od servisnog ugovora (*eng. service contract*), kojim se izražava koje su funkcionalnosti dostupne za javno pozivanje, i od logike koja omogućava izvršavanje datog skupa funkcionalnosti. Servisni ugovor se sastoji od jednog ili više dokumenata kojima se

izlažu meta informacije o servisu i on sadrži samo one informacije o servisu koje su neophodne korisnicima (*Service Abstraction Principle*). Pored toga, dati ugovor mora biti u skladu sa određenim standardima za projektovanje servisnih ugovora (*Standardized Service Contract Principle*). Najvažniji deo servisnog ugovora čine dokumenta kojima se izlaže njegov tehnički interfejs (*eng. technical service contract*) koji sadrži osnovne informacije neophodne za pozivanje funkcionalnosti servisa tj. koji predstavlja programski interfejs (API) datog servisa. Dakle, svaki servis ima jasno definisan tehnički interfejs koji sakriva detalje vezane za njegovu implementaciju, dok funkcionalnosti servisa, izložene u interfejsu, mogu biti na različite načine implementirane pomoću jednog ili više elemenata unutar tog servisa. Dodatno servisni ugovor može sadržati i druga dokumenta (koja takođe mogu biti u računarski čitljivom formatu) kao što je garantovani nivo kvaliteta servisa (*eng. Service Level Agreement – SLA*) kojim se daju dodatne informacije o nefunkcionalnim karakteristikama servisa (*eng. Quality of Service – QoS*). Pri tome treba napomenuti da se kvalitet servisa koji se garantuje određenim SLA može razlikovati od korisnika do korisnika budući da se on može definisati u dogovoru sa pružaocem servisa.

Sami servisi se implementiraju kao fizički nezavisne softverske komponente, pri čemu treba ponoviti da su servisno-orijentisani pristup pa i sama SOA nezavisni od bilo koje konkretne tehnološke platforme i načina implementacije samih servisa, te se prilikom realizacije pojedinačnih servisa zapravo može koristiti bilo koja tehnologija. Prema (*Chou & al, 2010*), bilo koja tehnologija pomoću koje je moguće razviti distribuirani sistem može se koristiti i za razvoj servisno-orijentisanih rešenja. Oni navode da su danas tri najčešća načina realizacije servisa: komponente, web servisi i REST servisi (koji se projektuju u skladu sa REpresentational State Transfer arhitektonskim pristupom u kome su resursi ključni elementi apstrakcije, pri čemu je akcenat na jednostavnosti, skalabilnosti i mogućnosti korišćenja). Imajući u vidu navedeno u ovom radu će se često koristiti pojam servis da bi se istaklo da pristupi koji se predlažu nisu ograničeni samo na pojam web servisa.

Servise nude pružaoci servisa – organizacije ili pojedinci koje obezbeđuju implementaciju servisa, njihov opis i pružaju neophodnu poslovnu i tehničku podršku, definišući pri tome i uslove pod kojima se servis može koristiti. Korisnici servisa mogu

biti drugi poslovni subjekti ili fizička lica, ali i i druge aplikacije kako unutar preduzeća tako i izvan njega.

Budući da je cilj omogućavanje deljenja funkcionalnosti između organizacija i/ili pojedinaca, mora postojati neki mehanizam koji će omogućiti da svaka zainteresovana strana lako pronađe servise koje drugi entiteti nude. Stoga je potrebno uspostaviti repozitorijum servisa (*eng. Repository*)[1] u kome će se održavati skup raspoloživih servisa pri čemu treba obezbediti i da se u njemu na neki način klasifikuju servisi kako bi korisnik iz mnoštva servisa pronašao onaj koji mu je potreban. Uvođenjem repozitorijuma zapravo se ukida potreba za jakom povezanošću (*eng. high coupling*) pružaoca servisa i korisnika. Pored toga neophodno je obezbediti i dovoljno detaljne informacije o samim servisima kako bi bilo koji korisnik mogao efektivno da ih pronađe, interpretira i koristi te svaki servis treba da sadrži i dodatne meta podatke (*Standardized Service Contract Principle*).

Konačno, servisno-orijentisani pristup takođe promoviše korišćenje standarda kako bi se omogućila integracija tj. obezbedila interoperabilnost različitih heterogenih sistema. Naime, imajući u vidu da se servisi ne moraju izvršavati u istim okruženjima potrebno je obezbediti mehanizme koji će omogućiti njihovu interakciju a koji neće zahtevati tesno povezivanje između pružaoca i korisnika servisa, pri čemu se sama interakcija ostvaruje putem mreže koja može biti lokalna ili globalna (Internet).

Iz navedenog se dakle može zaključiti:

Servisi predstavljaju jedinice funkcionalnosti na visokom nivou apstrakcije koje su nezavisne od konteksta same aplikacije, njene tehnološke platforme i infrastrukture kao i od drugih servisa i koje se kao takve mogu više puta koristiti ne samo na nivou pojedinačnog programa ili cele aplikacije već i na nivou celog preduzeća ili čak i u drugim preduzećima i to na načine i u kontekstima koji ne moraju biti poznati u vreme njihove implementacije. Pri tome se oni mogu lako međusobno povezivati i komponovati, bez obzira na to na kojoj su platformi realizovani i u kom programskom jeziku kao i na kojoj se platformi izvršavaju, kako bi zajedno obezbedili složeniju funkcionalnost.

[1] U (*Josuttis, 2007*) se navodi da treba praviti razliku između repozitorijuma servisa (*eng. repositories*), koji treba da omoguće upravljanje servisima sa poslovne tačke gledišta i registara servisa (*eng. registries*) koji se odnose na upravljanje servisima sa tehničkog aspekta.

Primenom servisno-orijentisanog pristupa može se lako realizovati distribuirana aplikacija koja se sastoji od skupa servisa koji međusobno interaguju razmenom poruka (u nekom standardnom formatu) putem mreže. SOA kao arhitektura zapravo podrazumeva da se razvoj aplikacija ostvaruje se kompozicijom servisa o čemu će biti više reči u sledećem potpoglavlju. Činjenica da dati servisi mogu biti veoma tehnološki heterogeni kao i da se mogu izvršavati na različitim platformama i biti veoma geografski udaljeni prestaje da bude relevantna budući da se sami servisi izvršavaju lokalno (odnosno na strani pružaoca) i da se komunikacija između njih obavlja putem predefinisanih poruka. Pri tome poslovni procesi nisu više limitirani raspoloživom IT infrastrukturom dok sa druge strane raspoložive tehnologije mogu pružiti kompetetivnu prednost preduzeću.

Osnovna prednost servisno-orijentisanog pristupa razvoju aplikacija u odnosu na klasične pristupe, se pre svega, ogleda u tome što razvijeni servisi nisu vezani ni za jedan konkretan poslovni domen već se njihovom kompozicijom zapravo kreira domensko-specifična aplikacija.

Sa poslovne tačke gledišta, osnovna prednost primene servisno-orijentisanog pristupa je dramatična ušteda u vremenu potrebnom za razvoj softverskih rešenja kao i u troškovima njihovog razvoja i održavanja, kao i činjenica da se sama rešenja mogu brzo i lako menjati kako bi se prilagodila promenama u zahtevima i uslovima poslovanja. Naime, razvoj novih softverskih rešenja je značajno brži, jednostavniji i jeftiniji u odnosu na klasične pristupe jer za realizaciju novih rešenja nije više neophodno razviti sve komponente datog rešenja već se mogu iskoristiti postojeći servisi, koji pri tome mogu biti i eksterno razvijeni, pri čemu način na koji su oni realizovani više ne predstavlja prepreku njihovom korišćenju. Sa druge strane, činjenica da je servisno-orijentisani pristup orijentisan na usklađivanje poslovnih zahteva i softverskih rešenja obezbeđuje mnogo veću fleksibilnost i agilnost poslovanja (pa samim tim i konkurentsku prednost) budući da se servisi mogu brzo i lako menjati, isključivati i uključivati u postojeće rešenje u skladu sa novim poslovnim potrebama. Pri tom se novi poslovni procesi mogu veoma brzo automatizovati čime se smanjuje vreme do tržišta. Pored toga, mogućnost lakog povezivanja nezavisno razvijenih servisa koji su interoperabilni olakšava poslovanje sa poslovnim partnerima ali takođe i otvara prostor za sve veću integraciju poslovanja sa drugim subjektima što je jedan od osnovnih

zahteva savremenog poslovanja na globalnom tržištu. Konačno, uspostavljanje repozitorijuma servisa zapravo omogućava preduzeću da se promoviše budući da će razvijeni servisi biti raspoloživi mnogo širem krugu potencijalnih korisnika i poslovnih partnera što u krajnjoj liniji znači i da će se investicija u razvoj novih servisa koji se mogu više puta koristiti, na različite načine i od strane različitih korisnika, višestruko isplatiti.

Sa tehničke tačke gledišta i same aplikacije su veoma fleksibilne budući da su servisi slabo povezani te se oni mogu lako menjati i unapređivati (pod uslovom da interfejs datog servisa ostane kompatibilan), ili čak zameniti drugim servisima koji obezbeđuju istu funkcionalnost (moguće implementiranim pomoću drugih tehnologija) a da to ne utiče na druge servise, celokupno rešenje niti krajnje korisnike. Pored toga ovaj pristup ne zahteva da se servisi fizički integrišu u aplikaciju već se oni pozivaju preko mreže i izvršavaju udaljeno, pri čemu su sami servisi autonomni (tj. potpuno nezavisni jedni od drugih) što zapravo znači da se oni mogu nezavisno razvijati i održavati. Pored toga primena principa servisno-orijentisane paradigme znači da neće doći do ponavljanja iste funkcionalnosti u različitim komponentama, što je čest slučaj kada je reč o klasičnim pristupima razvoju aplikacija. Servisno-orijentisani pristup omogućava da se na jednostavan način prevaziđu problemi integracije i komunikacije različitih aplikacija koje rade na različitim platformama i koje su implementirane pomoću različitih tehnologija odnosno heterogenost tehnologija više ne predstavljaju prepreku. Šta više ovaj pristup omogućava i laku integraciju nasleđenih komponenti koje se mogu upakovati u vidu servisa i na taj način izložiti. Ovo je posebno značajno imajući u vidu da se danas veoma često dešava i da jedno preduzeće kupi drugo ili se dva preduzeća spoje što nameće potrebu za integracijom njihovih rešenja. Dodatna prednost se ogleda u tome što se na taj način omogućava korišćenje tih postojećih komponenti i u drugačijim kontekstima od onih za koje su prvobitno bili razvijeni.

Konačno, imajući u vidu navedeno, očigledno je da se servisno-orijentisani pristup razvoju aplikacija sve više primenjuje.

2.3. Kompozicija servisa

Kompozicija servisa zapravo predstavlja agregaciju servisa koji su komponovani na takav način da zajedno automatizuju određeni zadatak ili poslovni proces. Drugim rečima, kompozicijom više servisa (koji pri tome ne moraju biti obezbeđeni od strane istog pružaoca) obezbeđuju se složenije funkcionalnosti. Stoga se u servisno-orijentisanom pristupu razvoj kompleksnih aplikacija zapravo realizuje jednostavnom kompozicijom servisa. Pri tome jedna kompozicija može uključivati kako proste servise tako i kompozitne servise (odnosno druge kompozicije) pri čemu sama kompozicija može biti i inter-organizaciona. Sa druge strane treba istaći da će, u skladu sa osnovnom idejom servisno-orijentisanog pristupa, isti servis (ili kompozicija) zapravo često biti uključen u nekoliko različitih kompozicija odnosno aplikacija i to ne nužno istih poslovnih subjekata. Očigledno je da je osnovni preduslov za mogućnost komponovanja servisa, kao što je navedeno u prethodnom poglavlju, to da oni moraju biti tehnološki nezavisni, slabo povezani i interoperabilni.

Prednost ovakvog pristupa ogleda se u tome što se za automatizaciju novih poslovnih procesa mogu koristiti postojeći servisi (sopstveni ili tuđi), bilo proširenjem postojećih kompozicija, bilo kreiranjem novih, što značajno smanjuje vreme potrebno za razvoj rešenja i troškove. Sa druge strane, i sami servisi se mogu lako unapređivati (kako bi se usvajale nove tehnologije) a da to pri tome ne utiče na definisane kompozicije. Ili kako se navodi u (*Linthicum, 2007*):„servisi se prave da traju dok se kompozicije prave da budu promenljive“.

Budući da je u servisno-orijentisanom pristupu razvoj aplikacija vođen poslovnim procesima, očigledno je da je najpre neophodno opisati način realizacije određenog poslovnog procesa. Danas postoje brojni alati (tekstualni i grafički) koji omogućavaju modelovanje poslovnih procesa. Među njima se posebno izdvaja BPMN (*Business Process Model and Notation*)[1] koji obezbeđuje grafičku notaciju za kreiranje modela poslovnih procesa, pri čemu treba istaći da je prilikom formiranja modela akcenat isključivo na strukturi samih procesa i načinu njihove interakcije, dok se tehnički aspekt (tj. način njihove implementacije) ne razmatra. BPMN omogućava formalno definisanje poslovnih procesa nekog poslovnog entiteta putem dijagrama poslovnih procesa.

[1] Business ProcessManagement Initiative (BPMI) BPMN: http://www.omg.org/bpmn/

Prema (*White, 2008*) BPMN omogućava kreiranje tri vrste podmodela:

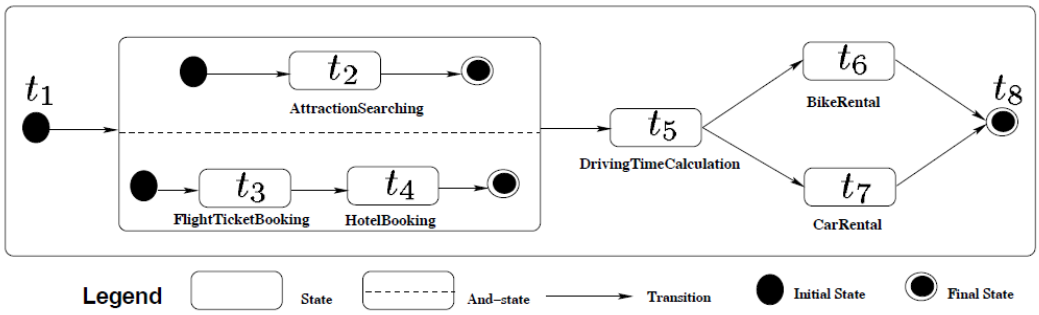
- Orkestracije (*eng. Orchestration*) – definišu redosled izvršavanja aktivnosti (tj. redosled po kome će se pozivati servisi) sa ciljem izvršavanja određene funkcionalnosti odnosno poslovnog procesa. Pri tome, kako se navodi u (*White, 2008*), ovi modeli zapravo opisuju ponašanje samo sa stanovišta pojedinačnog poslovnog entiteta, te zapravo podrazumevaju da svi definisani elementi postoje unutar jedinstvenog konteksta kao i da postoji neki centralni mehanizam koji ih koordinira odnosno upravlja njihovim izvršavanjem. Drugim rečima, pomoću orkestracija se servisi komponuju kako bi se obezbedio neki složeniji poslovni proces te će definisani poslovni proces zapravo upravljati tokom izvršavanja (*eng. control flow*) i tokom razmene podataka (*eng. data flow*) između servisa. Grubo gledano, može se reći da se orkestracijama definišu unutrašnji mehanizmi nekog kompozitnog servisa odnosno struktura same kompozicije, pri čemu se rezultujuća kompozicija takođe može izložiti kao servis i kao takva uključivati u druge kompozicije.
- Koreografije (*eng. choreography*) – kojima se opisuje koordinacija interakcija između različitih poslovnih entiteta definisanjem redosleda po kome će se poruke razmenjivati, pri čemu, kako se navodi u (*Josuttis, 2007*) fokus nije na orkestraciji aktivnosti pojedinačnih poslovnih subjekata, već na razmeni informacija (poruka) između njih. Prema (*W3C, 2004*) koreografija predstavlja model sekvence operacija, stanja i uslova koji upravljaju interakcijama između nezavisnih servisa, pri čemu interakcija koja je propisana koreografijom rezultuje izvršavanjem određene funkcionalnosti. Međutim, kako se navodi u (*White, 2008*) za razliku od orkestracija, koreografije nisu definisane u nekom dobro-oformljenom kontekstu niti postoji centralni mehanizam upravljanja. Dakle, grubo gledano, može se reći da se koreografijom zapravo definiše razmena poruka između ravnopravnih servisa pri čemu svaki servis zna kako i pod kojim uslovima treba da izvrši određene operacije ili da pozove neki drugi servis te svaki poslovni entitet zapravo upravlja samo svojim servisima.
- Kolaboracije (*eng. Collaborations*) – kojima se samo opisuju entiteti i interakcije između njih (u pogledu poruka koje se razmenjuju) ali koji za razliku od koreografija ne prikazuju i redosled po kome će se one razmenjivati.

Sledeći korak je selekcija ili razvoj samih servisa koji će se koristiti za realizovanje definisanih procesa. Naime, prema (*Zimmermann & al, 2004*) SOA kao arhitektura se zasniva na skupu softverskih servisa koji odgovaraju generičkim poslovnim servisima od kojih se određeni poslovni proces sastoji. Pretragom odgovarajućih repozitorijuma može se utvrditi koji se od neophodnih servisa nalaze na raspolaganju a koje je neophodno razviti ili nabaviti.

Treba napomenuti i da postoji veliki broj alata zasnovanih na BPM pristupu koji obezbeđuju podršku ne samo za modelovanje samih procesa već i za njihovo izvršavanje. Naime, definisanim poslovnim servisima će se pridružiti konkretni softverski servisi, i ukoliko je model procesa definisan pomoću nekog izvršnog jezika (kao što je BPEL) ili jezika koji se može prevesti u izvršni jezik (na primer BPMN koji se može mapirati u BPEL) takav model se može izvršiti pomoću odgovarajućeg „engine-a“ koji će pozivati servise u skladu sa definisanim tokom izvršavanja (*eng. control flow*), što znači da ne postoji potreba za programskim povezivanjem samih servisa. Konačno, očigledno je da je za funkcionisanje ovakve arhitekture neophodno postojanje određenog mehanizma unutar same platforme koji će upravljati lociranjem komponenti i njihovom interakcijom.

Dakle, generalno gledano, prilikom definisanja kompozicije za realizaciju nekog složenog funkcionalnog zahteva prvo je neophodno odrediti koji su to delovi funkcionalnosti koje svaka od komponenti treba da pruži kao i redosled njihovog izvršavanja (odnosno strukturu same kompozicije). U ovom koraku razmatraju se samo funkcionalne karakteristike servisa. Pri tome treba istaći da tok izvršavanja kompozicije (*eng. control flow*) nije nužno samo sekvencijalan već obuhvata i druge upravljačke strukture (uslovne i ciklične) pri čemu se neki servisi u kompoziciji mogu izvršavati i paralelno.

Jedan od primera kompozicije, koji se u literaturi najčešće navodi, odnosi se na planiranje putovanja za koje je potrebno rezervisati avionsku kartu i hotel, izabrati znamenitosti koje se žele posetiti i iznajmiti kola ili bicikl (Slika 3.). Data kompozicija, modelovana pomoću dijagrama promene stanja (*eng. statechart diagram*) podrazumeva da se neke od aktivnosti izvršavaju paralelno.



Slika 3. Primer kompozicije za planiranje putovanja (*Zeng & al, 2003*)

Kada se uspostavi struktura kompozicije sledeći korak je izbor konkretnih servisa (iz repozitorijuma servisa) koji će se uključiti u kompoziciju i obaviti definisane zadatke. Drugim rečima prvi korak je planiranje čiji je rezultat apstraktna kompozicija servisa a drugi korak je selekcija servisa u okviru koje se kompoziciji dodeljuju konkretni servisi. Ovaj drugi korak će biti predmet ovog rada i sam problem će biti detaljno izložen u Poglavlju 3.1. Nakon što se formira konkretna kompozicija ona se može izvršiti od strane odgovarajućeg engine-a koji će upravljati izvršavanjem kompozicije odnosno koji će sledeći definisani tok izvršavanja pozivati i koordinirati izabrane servise koristeći informacije dobijene iz repozitorijuma o njihovim lokacijama i načinu korišćenja (koje su opisane u interfejsima tih servisa).

Imajući u vidu navedeno, očigledno je da je potrebno obezbediti i da se sama kompozicija može lako promeniti, ili po potrebi čak i dinamički kreirati ukoliko se žele ostvariti osnovni ciljevi servisno-orijentisanog pristupa. Pored toga danas sve više postoji potreba za automatizaciju ovog postupka. Neki od pristupa koji se predlažu u literaturi omogućavaju da se delimično ili čak u potpunosti automatizuje ovaj postupak zasnivajući se pri tome na semantičkim tehnologijama i veštačkoj inteligenciji. Opsežan pregled se može naći u (*Sheng & al, 2014*). Ovde će se samo navesti neki od pristupa koji su relevantni sa stanovišta toga da obezbeđuju komponente i metode za selekciju servisa na osnovu nefunkcionalnih karakteristika te su razmatrani u narednim poglavljima. Najveći broj takvih radova podrazumeva postojanje centralizovane komponente – brokera koji će upravljati dinamičkom selekcijom servisa (*Zeng & al,2003; Aggarwal & al, 2004; Canfora & al, 2004; Zeng & al, 2004; Ardagna & Pernici, 2005; Canfora & al, Jun 2005; Canfora & al, Jul 2005; Gronmo & Jaeger, Jun*

2005; Gronmo & Jaeger, Decembar 2005; Yu & Lin, Jul, 2005; Yu & Lin, Decembar, 2005; Jaeger, 2006; Ardagna & Pernici, 2007; Yu & al, 2007; Canfora & al, 2008) dok se u (*Alrifai & Risse, 2009*) predlaže distribuirana selekcija dekomponovanjem originalnog problema na manje probleme koje je lakše rešavati. Posebno se ističe pristup predložen u (*Gronmo & Jaeger, Jun 2005; Gronmo & Jaeger, Decembar 2005; Jaeger, 2006*) koji zasnovan, na danas vodećem pristupu projektovanju rešenja – razvoj vođen modelima (*eng. Model-Driven Development – MDD*) koji podrazumeva da se razvoj može automatizovati transformacijama modela, počevši od konceptualnih modela sve do izvršnog kôda.

2.4. Web servisi

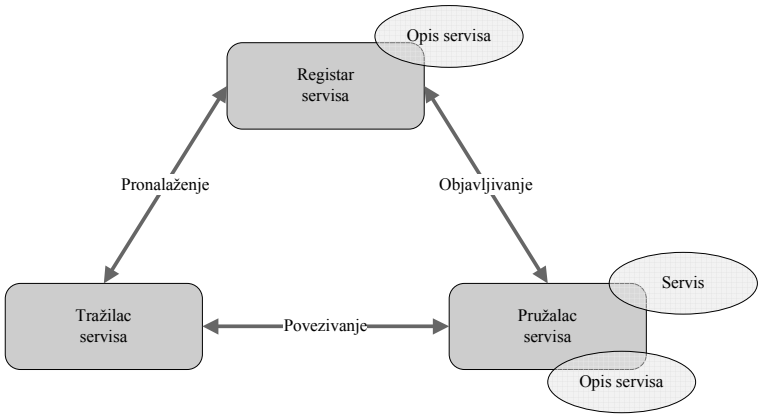
Web servisi predstavljaju jedinice funkcionalnosti implementirane kao fizički nezavisne softverske komponente koje su dostupne na Internetu pri čemu mogu obavljati funkcije koje se kreću od jednostavnih zahteva pa sve do složenih poslovnih procesa. Prema (*W3C, 2004*) web servis predstavlja softverski sistem koji je projektovan tako da podrži interoperabilnu interakciju između računara u mreži, odnosno web servis obezbeđuje standardno sredstvo za interoperabilnost između različitih softverskih aplikacija koje se izvršavaju na mnoštvu različitih platformi i/ili okruženja. U (*Andresen, 2006*) se navodi da web servisi predstavljaju modularne, samo-opisujuće aplikacije koje se mogu objaviti, locirati i pozivati sa bilo koje tačke Web-a ili lokalne mreže.

Prema (*W3C, 2004*) web servis zapravo predstavlja apstraktni resurs neke organizacije ili pojedinca (pružaoca servisa) koji obezbeđuje izvršavanje grupe zadataka koja čini koherentnu funkcionalnost. Da bi se mogao koristiti servis mora biti realizovan od strane nekog konkretnog agenta tog pružaoca. Agent predstavalja konkretan softver ili hardver koji prima i šalje poruke i kojim se realizuje data funkcionalnost, pri čemu različiti agenti mogu implementirati isti servis na različite način ili u različitim jezicima dokle god ta implementacija obezbeđuje istu funkcionalnost. Način implementacije web servisa može varirati (bilo da je reč o objektima, komponentama, ili običnom proceduralnom kôdu), pri čemu sam web servis ne mora biti povezan sa bilo kojom web

tehnologijom (*Leymann, 2003*). Skup funkcionalnosti (operacija) koje Web servis pruža izražava se putem servisnog ugovora[1].

Servise nude pružaoci servisa – organizacije ili pojedinci koje obezbeđuju implementaciju i opis servisa i pružaju neophodnu podršku. Kako se web servisi razvijaju za srednji sloj aplikacije oni ne obezbeđuju nikakav korisnički interfejs, te se ne mogu koristiti direktno od strane krajnjih korisnika, već se mogu pozivati isključivo od strane drugih aplikacija. Stoga klijenti (tražioci) servisa mogu biti samo druga rešenja, aplikacije ili procesi bilo unutar samog preduzeća bilo izvan njega kao i drugi web servisi. Pri tome treba napomenuti da pružalac i tražilac servisa zapravo ne moraju koristiti iste tehnologije.

Budući da je cilj omogućavanje deljenja funkcionalnosti između organizacija i/ili pojedinaca, mora postojati neki mehanizam koji će omogućiti da svaka zainteresovana strana lako pronađe servise koje drugi entiteti nude. Pružalac objavljuje svoje servise na Internetu u registru servisa. Registar servisa predstavlja nezavisnu kolekciju servisa koja sadrži njihove opise, fizičke lokacije, verzije, rokove važenja, dokumentaciju i politike korišćenja. Tražilac može pretraživati registar u potrazi za servisima koji odgovaraju njegovim funkcionalnim zahtevima (pri čemu prilikom pretrage za odgovarajućim servisom i nefunkcionalni zahtevi mogu biti uzeti u obzir). Ukoliko postoje odgovarajući servisi registar će mu za svaki takav servis vratiti opis interfejsa i njegovu lokaciju.



Slika 4. Arhitektura web servisa

[1] Treba napomenuti da kada se servisi implementiraju kao komponente funkcionalnost se obično nazivaju metodama dok se u slučaju implementacije pomoću web servisa nazivaju operacijama.

S obzirom na to da se servisi mogu nuditi od strane raznih pružalaca, i biti implementirani u različitim jezicima i na heterogenim platformama, potreban je skup standarda koji će obezbediti zahtevanu interoperabilnost pre svega imajući u vidu da se komunikacija obavlja putem Interneta. Stoga se web servisi zasnivaju na skupu otvorenih standarda:

1. Najpre je potrebno uspostaviti mehanizme koji će se koristiti za komunikaciju između servisa u mreži. Iako je koncept web servisa zamišljen da bude nezavisan od korišćenog protokola ipak je HTTP (*Hyper Text Transfer Protocol*) komunikacioni protokol aplikacionog sloja koji se najčešće koristi za slanje zahteva web servisu i primanje odgovora od njega putem Interneta. Pored toga za komunikaciju se mogu koristiti i drugi protokoli kao što su: FTP (*File Transfer Protocol*), SMTP (*Simple Mail Transfer Protocol*) ili čak TCP (*Transmission Control Protocol*).
2. Zatim je potrebno utvrditi format za predstavljanje podataka koji se razmenjuju a koji će biti razumljiv svim učesnicima u komunikaciji. U tu svrhu se koristi XML (*eXtensible Mark-up Language*) koji je danas de-facto standard za opis sadržaja i logičke strukture dokumenata i njihovu razmenu na Webu. Prednost korišćenja XML-a ogleda se u tome što je on zasnovan na tekstu, fleksibilan i proširiv (nema predefinisan fiksan format), samo-opisujući (informacije o podacima sastavni su deo dokumenta sa podacima čime se zapravo obezbeđuje zajednička sintaksa i prevazilazi problem usklađivanja formata podataka različitih aplikacija na različitim platformama) i dovoljno formalan za mašinsko procesiranje a istovremeno dovoljno razumljiv za korisnike. Kako XML omogućava da se podaci serijalizuju tj. prevedu u sekvencu bitova koja se lako može dekodirati (parsirati i interpretirati) na bilo kojoj platformi on je pogodan za razmenu struktuiranih podataka između web servisa bez obzira na načina implementacije samih servisa i korišćenih platformi. Za definisanje tipova podataka u porukama koje web servisi razmenjuju koristi se XML šema (XSD[1]). XSD je XML dokument kojim se, između ostalog, opisuju tipova podataka koji će se koristiti u konkretnom XML dokumentu zasnovanom na toj šemi.

[1] W3C (World Wide Web Consortium) XDS: http://www.w3.org/XML/Schema

3. Sledeći korak je definisanje samog formata tj. strukture poruka koje će se razmenjivati. SOAP (*Simple Object Access Protocol*)[1] je komunikacioni protokol za razmenu informacija, u vidu struktuiranih poruka, između aplikacija bez obzira na njihovu platformu, programsko okruženje, model podataka i način implementacije. On obezbeđuje bogat, među-platformski način predstavljanja osnovnih (opštih) tipova podataka (bilo prostih kao što su celi brojevi, decimalni brojevi, stringovi ili složenih kao što su nizovi ili strukture). Dakle dok se XML koristi za kodiranje i dekodiranje podataka SOAP se koristi za njihov prenos preko odgovarajućih Internet protokola (kao što su HTTP, SMTP i TCP). SOAP poruka je XML dokument kojim se daje se sastoji od elementa *Envelope* kojim se označava početak i kraj poruke i daje opis sadržaja i strukture poruke (odnosno informacije o tome šta se nalazi u poruci, kome je ona namenjena i na koji način se obrađuje) i elementa *Body* koji obuhvata sadržaj same poruke odnosno podatke koji se razmenjuju. Pored toga, SOAP poruka opciono može sadržati *Header* element koji sadrži dodatne informacije vezane za aplikaciju (npr. digitalni potpis, način plaćanja, broj računa u na koji se vrši uplata nadoknade za korišćenje servisa itd.) i *Fault* element koji sadrži informacije od greškama koje su nastale prilikom obrade poruke.
4. WSDL (Web Services Description Language)[2] je format zasnovan na XML-u koji opisuje sve što je klijentu potrebno za interakciju sa web servisom. To uključuje operacije servisa, tipove podataka za sve parametre i povratnu vrednost, i sve informacije koje su neophodni za komunikaciju sa servisom (lokaciju servisa, format poruke i transportne protokole). Drugim rečima WSDL zapravo obezbeđuje gramatiku za opis svih ovih informacija. Stoga, bilo koji tražilac koji može da parsira XML poruku može da koristi dati web servis čak iako je pisan u drugom jeziku ili se izvršava na računaru sa drugom platformom. Treba napomenuti da je WSDL zamišljen i razvijen da bude pre svega proširiv a zatim i nezavisan od komunikacionog protokola (npr. HTTP) i načina kodiranja (npr. SOAP) kako bi se mogao koristiti i sa budućim tehnologijama.

[1] W3C SOAP: http://www.w3.org/TR/soap/
[2] W3C WSDL: http://www.w3.org/TR/wsdl

5. Konačno UDDI (Universal Description, Discovery and Integration)[1] obezbeđuje mehanizam pomoću koga će se objavljivati i pronalaziti web servisi. UDDI je u osnovi specifikacija koja opisuje kako registar web servisa treba da bude organizovan i kako pružaoci web servisa mogu kod njega programski registrovati svoje servise. Preciznije rečeno, UDDI je zapravo poslovni repozitorijum interfejsa web servisa (opisanih WSDL dokumentima) razvijen sa namerom da omogući pružaocima da objave svoje servise na Internetu a tražiocima da pronađu potrebne servise na osnovu vrste servisa, poslovnog naziva itd.

Navedeni osnovni standardi zajedno sa celim skupom komplementarnih standarda koji propisuju različite aspekte web servisa obrazuju minimalnu infrastrukturu koju zahteva paradigma web servisa[2]. Među njima se izdvaja WS-* skup standarda (na primer WS-Security, WS-Interoperability, WS-Trust itd.) koji omogućavaju dodatnu specifikaciju servisa, pri čemu WS-Policy omogućava da se opišu i politike između ostalog i u pogled nefunkcionalnih karakteristika (QoS) web servisa koje se nude od strane pružaoca servisa odnosno zahtevaju od strane tražioca servisa. Konačno treba napomenuti da se pružalac i tražilac servisa mogu dogovoriti oko garantovanog nivoa kvaliteta putem formalnog ugovora (*eng. Service Level Agreement – SLA*) koji može biti iskazan i u računarski čitljivom formatu što je od posebnog značaja ukoliko se selekcija servisa vrši dinamički.

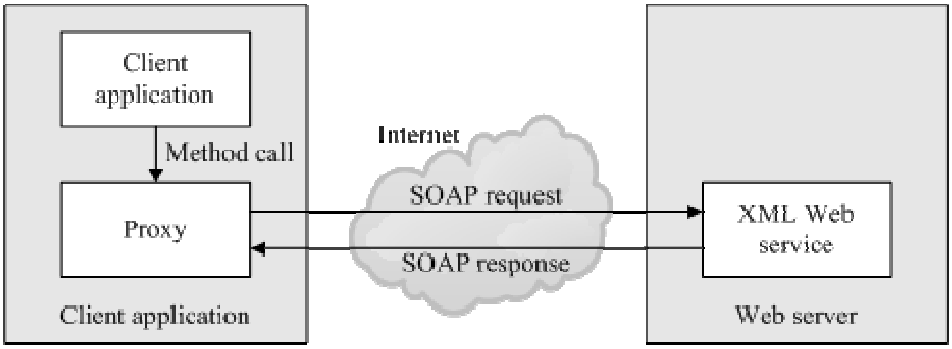
Dakle, web servisi omogućavaju preduzećima da svoje osnovne kompetencije izlože na Internetu ili intra-netu korišćenjem standardnih jezika (zasnovanih na XML-u) i protokola, pri čemu se oni mogu implementirati putem samo-opisujućih interfejsa zasnovanih na otvorenim standardima (*Papazoglou, 2003*).

Generalno gledano interakcija sa nekim web servisom se odvija na sledeći način. Tražilac pronalazi servis u registru i zahteva od njega WSDL dokument kojim je servis opisan. Na osnovu dobijenog WSDL dokumenta tražilac zatim generiše odgovarajuću „proxy“ klasu kojom se apstrahuje komunikacija sa datim web servisom odnosno omogućava rad sa web servisom kao da je on lokalna komponenta. Naime, ovom

[1] OASIS (Organization for the Advancement of Structured Information Standards) UDDI: https://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm

[2] Web Services Interoperability Organization (WS-I), koja je sada deo OASIS grupe, zadužena za promovisanje interoperabilnosti web servisa je formulisala WS-I Basic Profile kao smernice za razvoj web servisa: http://ws-i.org/Profiles/BasicProfile-1.2-2010-11-09.html

komponentom realizuje se poziv određene metode web servisa tako što se formira odgovarajuća SOAP poruka i prosleđuje web servisu koji se kao distribuirana aplikacija izvršava na nekom udaljenom web serveru. Isto tako ova klasa obrađuje i SOAP poruku koja je dobijena kao rezultat izvršavanja date operacije web servisa.



Slika 5. Interakcija XML web servisa (*MacDonald, 2003*)

Treba imati u vidu da se web servis instancira u trenutku poziva metode (*eng. just-in-time*) odnosno na početku klijentskog zahteva i uništavaju nakon što se vrati odgovor kao i da ne čuvaju stanje između dva poziva (*eng. stateless*) (*MacDonald, 2003*).

Kompozicija web servisa, kao što je objašnjeno u prethodnom poglavlju, predstavlja skup pojedinačnih web servisa kojima se obezbeđuje nova, složenija funkcionalnost[1]. Pri tome se, imajući u vidu izložene karakteristike, nezavisno razvijeni web servisi mogu veoma jednostavno automatski pronaći i dinamički integrisati u aplikacije budući da su zasnovani na skupu standardnih protokola. Takođe treba istaći da se kompozitni web servisi takođe mogu nuditi kao gotove komponente.

Kada je reč o kompozitnim web servisima, odnosno web servisima koji predstavlja agregacije više drugih web servisa, treba napomenuti da postoji ceo niz standarda koji omogućava njihovu specifikaciju među kojima se izdvajaju: BPMI-ov BPML (*Business Process Modeling Language*), IBM-ov WSFL (*Web Services Flow Language*), BPEL4WS (*Business Process Execution Language for Web Services*) ili noviji WS-BPEL (*Web Service Business Process Execution Language*)[2] promovisani od strane OASIS. Prednost nekih od ovih jezika se ogleda u tome što su oni izvršni što znači da se definisana kompozicija može interpretirati i izvršiti od strane odgovarajućeg

[1] Treba napomenuti i da se generalno gledano kompozicija web servisa može ostvariti i kompozicijom samo nekih njihovih operacija.
[2] http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

engine-a. Naime, danas vodeći standard WS-BPEL omogućava da se opiše (pomoću gramatike zasnovane na XMLu) ponašanje određenog poslovnog procesa odnosno način na koji se koordiniraju web servisi (u pogledu razmene poruka kao i redosleda izvršavanja) kako bi se ostvario dati poslovni zahtev, pri čemu se definisani model zapravo može interpretirati i izvršiti od strane BPEL engine-a. Treba napomenuti da WS-BPEL zapravo omogućava da se specificiraju dve vrste modela poslovnih procesa: apstraktni i izvršni (*eng. executable*). Apstraktni modeli se kao takvi ne mogu izvršiti međutim, nakon što se formira apstraktni model, svakom apstraktnom servisu u kompoziciji (odnosno komponenti kompozicije) se može dodeliti konkretan web servis, nakon čega engine može izvršiti datu kompoziciju u skladu se definisanim modelom. Pri tome treba istaći i da BPEL engine obezbeđuje mehanizme za obradu grešaka ali i da omogućava da se formirana kompozicija izloži kao novi kompozitni web servis.

Konačno može se zaključiti, kako se navodi u (*Papazoglou, 2003*) da budući da web servisi predstavljaju uniformni i sveprisutan način distribucije informacija za širok spektar uređaja (kao što su računari, mobilni telefoni itd.) i softverskih platformi (Unix, Windows itd.) oni predstavljaju sledeći korak u distribuiranom računarstvu . Pored toga, s obzirom na to da se web servisi razvijaju imajući u vidu komunikaciju putem Interneta i mogućnost njihovog ponovnog korišćenja oni obezbeđuju distribuiranu računarsku infrastrukturu za integraciju i kolaboraciju aplikacija unutar samih preduzeća ali i između različitih preduzeća. Sa sve većom prisutnošću Interneta implementacija servisno-orijentisanih rešenja pomoću web servisa sve više dobija na značaju. Naime, postojanje mnoštva nezavisno razvijenih softverskih komponenti dostupnih preko Interneta, koje se mogu koristiti na bilo kojoj platformi, omogućava veoma brz razvoj novih aplikacija uz veoma male troškove.

Budući da se web servisi nezavisno razvijaju i implementiraju svakim danom sve je veći broj raspoloživih web servisa realizovanih od strane različitih ponuđača koji pri tome mogu biti ponuđeni sa različitim nefunkcionalnim karakteristikama. Problem izbora odgovarajućeg servisa za određeni funkcionalni zahtev, prost ili složen, biće detaljnije razmatran u Poglavlju 3.1.

2.5. Nefunkcionalne karakteristike web servisa

U literaturi je prisutno mnoštvo različitih QoS (*eng. Quality of Service*) mera kojima se opisuju različiti kvantitativni i kvalitativni nefunkcionalni aspekti servisa kao što su performanse, dostupnost, pouzdanost, sigurnost, interoperabilnost itd. Proliferacija web servisa uslovila je pojačanu konkurenciju među pružaocima usluga, a time i povećan interes za QoS web servisa koji bi mogao da im obezbedi kompetetivnu prednost. Otuda je postalo neophodno da se posebna pažnja posveti identifikaciji mogućih QoS parametara.

2.5.1. Vrste nefunkcionalnih karakteristika i njihove osobine

Među različitim klasifikacijama QoS web servisa izdvajaju se dve osnovne:

- Specifikacija definisana od strane radne grupe W3C (*W3C, 2003*) i
- ISO 9126 standard za vrednovanje softvera (*ISO 9126, 2001*).

2.5.1.1. W3C SPECIFIKACIJA

W3C (World Wide Web Consortium) je formulisao dokument (*W3C, 2003*) kojim se propisuju zahtevi u pogledu nefunkcionalnih karakteristika web servisa. QoS zahtevi koji se specificiraju ovim dokumentom odnose se poglavito na aspekt kvaliteta web servisa sa tačke gledišta korisnika, dok se u znatno manjoj meri specificiraju QoS zahtevi koji se odnose na prenosni medijum. Predložena je sledeća kategorizacija:

- Performansa (*eng. Performance*) se izražava preko brzine kojom servis može da obradi postavljeni zahtev. Ova karakteristika može se meriti na više načina (od kojih su neki međusobno povezani):
    - Propusna moć (*eng. Throughput*) – broj zahteva koje web servis može da obradi u određenom intervalu vremena
    - Kašnjenje (*eng. Latency*) – ukupno vreme trajanja propagacije zahteva koje korisnik postavlja servisu i odgovora koje servis vraća korisniku.
    - Vreme izvršavanja (*eng. Execution Time*) – vreme za koje web servis obavi niz aktivnosti neophodnih da se izvrši postavljeni zahtev
    - Vreme odziva (*eng. Response Time*) – vreme potrebno da se zahtev izvrši. Ova vreme predstavlja zbir kašnjenja i vremena izvršavanja.

- Vreme trajanja transakcije (*eng. Transaction Time*) – vreme potrebno da web servis obavi jednu kompletnu transakciju (skup svih zahteva upućenih web servisu)

U principu dobra performansa web servisa podrazumeva veću propusnu moć, manje kašnjenje i kraća vremena izvršavanja, odziva i trajanja transakcije.

- Pouzdanost (*eng. Reliability*).

  Sposobnost web servisa da obavlja svoju funkciju pod datim uslovima u okviru određenog intervala vremena predstavlja meru njegove pouzdanosti. Ova sveobuhvatna mera sposobnosti web servisa da održi kvalitet servisa povezana je sa brojem otkaza u nekom vremenskom periodu (danu, nedelji, mesecu ili godini). Pored toga pouzdanost je takođe povezana i sa uređenom isporukom poruka koje se razmenjuju između korisnika servisa i pružaoca servisnih usluga. Web servis treba da ima visoku pouzdanost.

- Skalabilnost (*eng. Scalability*)

  Skalabilnost web servisa je sposobnost prilagođavanja servisa broju zahteva u datom vremenskom intervalu. Ovim se zapravo podrazumeva sposobnost web servisa da, po potrebi, povećava broj operacija ili transakcija koje je u stanju da podrži. Time se obezbeđuje da nezavisno od broja zahteva web servis zadrži isti nivo usluga. Na taj način je skalabilnost zapravo povezana sa performansom sistema. Web servis treba da ima veliku skalabilnost.

- Kapacitet (*eng. Capacity*)

  Kapacitet web servisa određen je maksimalnim brojem simultanih usluga koje on može da pruži uz zagarantovani nivo performanse. Web servis treba da ima specificirani kapacitet.

- Robusnost (*eng. Robustness*)

  Robusnost web servisa meri se sposobnošću ispravnog funkcionisanja u slučaju neispravnih, nekompletnih ili konfliktnih podataka. Web servis treba da ima veliku robusnost.

- Tretiranje izuzetaka (*eng. Exception handling*)

  Polazeći od činjenice da nijedan projekat web servisa ne može da specificira sve moguće ishode i alternative, posebno u slučaju kada uključuje specijalne

slučajeve i nepredviđene mogućnosti, web servis treba da ima mogućnost adekvatnog tretiranja izuzetaka. Web servis treba da poseduje ovu sposobnost.

- Tačnost (*eng. Accuracy*)

  Tačnost web servisa izražava se procentom grešaka koje se registruju u datom vremenskom intervalu.Web servis treba da im veliku tačnost, što znači da procenat grešaka treba da se minimizira.

- Integritet (*eng. Integrity*) podrazumeva sposobnost sistema da spreči neautorizovani pristup servisu, ili promenu računarskih programa ili podataka. Uočavaju se dva tipa integriteta: integritet baze podataka i integritet transakcija
  - Integritet baze podataka je obezbeđen ukoliko se podaci ne mogu menjati u toku prenošenja.
  - Integritet transakcije se odnosi na skup procedura kojima se obezbeđuje integritet baze podataka u toku transakcije.
- Mogućnost pristupa (*eng. Accessibility*)

  Mogućnost pristupa izražava sposobnost sistema da opslužuje korisničke zahteve. Veća mogućnost pristupa, koju treba da ima web servis, može da se ostvari projektovanjem sistema sa velikom skalabilnošću.
- Raspoloživost (*eng. Availability*)

  Raspoloživost servisa podrazumeva spremnost servisa da odmah odgovori na postavljeni zahtev. U tom smislu raspoloživost je određena verovatnoćom da je sistem podignut i vezana je za pouzdanost sistema. Trenutna raspoloživost servisa je povezana sa vremenom potrebnim za popravku web servisa (*TTR time-to-repair*). Web servis treba da ima visoku raspoloživost, odnosno on treba da se odazove neposredno pošto je pozvan.
- Interoperabilnost (*eng. Interoperability*)

  Interoperabilnost je vezana za mogućnost implementacije web servisa u različitim razvojnim okruženjima, čime se developeri koji koriste ove servise oslobađaju obaveze da vode računa o programskim jezicima ili operativnom sistemu na kome se servis nalazi.
- Bezbednost (*eng. Security*). Porast web usluga koje se prenose preko Interneta prirodno dovodi do zaoštravanja pitanja sigurnosti. U zavisnosti od zahteva

korisnika servisa, pružaoci web servisa mogu koristiti različite pristupe u obezbeđivanju zaštite, kao i različite nivoe zaštita.

- Autentifikacija (*eng. Authentication*) – mogućnost provere autentičnosti korisnika (ili nekog servisa) koji traži pristup datom servisu.
- Autorizacija (*eng. Authorization*) – provera prava korisnika (ili nekog servisa) da pristupi datom servisu.
- Poverljivost (*eng. Confidentiality*) – obezbeđenje podataka na taj način da samo autorizovane osobe mogu da im pristupe ili da ih menjaju.
- Odgovornost (*eng. Accountability*) – pružaoc usluge mora biti odgovoran za usluge koje web servis pruža.
- Mogućnost praćenja i provere postupka (*eng. Traceability and Auditability*) – web servis mora da obezbedi mogućnost praćenja istorije svih obavljenih transakcija.
- Kriptovanje podataka (*eng. Data Encryption*) – podaci treba da budu zaštićeni nekom od tehnika kriptovanja.
- Sprečavanje poricanja (*eng. Non-Repudiation*) – posle traženja usluge sa nekog servisa korisnik ne može da porekne tu činjenicu.

- QoS zahtevi vezani za transportnu mrežu

  Postupci koji se primenjuju u cilju ostvarivanja željenih karakteristika web servisa moraju da uzmu u obzir i karakteristike transportne mreže koje su u najvećoj meri nezavisne od same aplikacije. Osnovni QoS parametri mreže su kašnjenje mreže pri prenosu paketa, promenljivost kašnjenja koje otežava formiranje korektne sekvence paketa, kao i gubitak pojedinih paketa.

2.5.1.2. ISO 9126 SPECIFIKACIJA

ISO 9126 (*ISO 9126, 2001*) standard formiran je tako da reflektuje tri tipa kvaliteta softvera

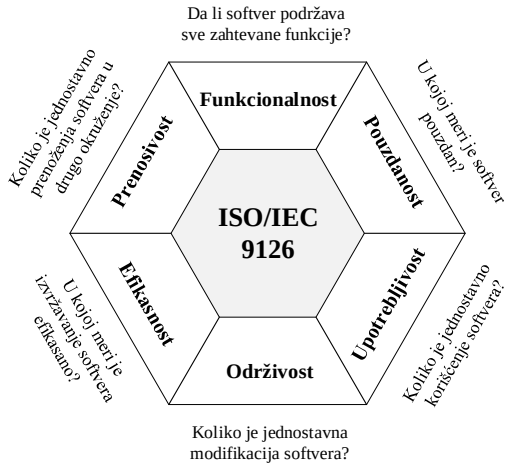
- Korisnički atributi – vezani za korisničku percepciju kvaliteta.
- Interni atributi – nezavisni od konteksta u kome se servis koristi.
- Eksterni atributi – vezani za kontekst u kome se servis koristi.

Kombinovanjem ovih aspekata standard ISO 9126 sastavljen je od četiri dela

- ISO/IEC 9126-1: 2001 Deo 1: Model kvaliteta
- ISO/IEC 9126-2: 2003 Deo 2: Eksterna metrika
- ISO/IEC 9126-3: 2003 Deo 3: Interna metrika
- ISO/IEC 9126-4: 2004 Deo 4: Metrika kvaliteta pri korišćenju

Prvi deo standarda definiše atribute kvaliteta softvera koji se koriste u procesu kontrole kvaliteta (SQC), osiguranja kvaliteta (SQA) i poboljšanju karakteristika softvera (SPI). U tom smislu preostala tri dela standarda koja se bave metrikom i merenjima atributa su od ključnog značaja za tri navedena procesa. U skladu sa time ovde će se izložiti samo model kvaliteta.

Model kvaliteta (*ISO 9126, 2001*) obuhvata 6 osnovnih karakteristika kvaliteta.



Slika 6. Šest karakteristika kvaliteta softvera ISO/IEC 9126 (prema ***Moody, 2005*****)**

Ove karakteristike se dalje dele na odgovarajuće potkarakteristike. Na najnižem nivou hijerarhije definišu se atributi zajedno sa metrikom koja se koristi za njihovo izražavanje. Prema (*Botella & al, 2004*) karakteristike su nemerljivi faktori kvaliteta koje se koriste u cilju klasifikacije potkarakteristika. Potkarakteristike su faktori kvaliteta koje se mogu, ako je to potrebno, subjektivno meriti i koje se mogu dekomponovati u druge potkarakteristike ili u atribute koji pomažu pri merenju kvaliteta. Za razliku od potkarakteristika kojim je pridružena subjektivna metrika osnovni atributi se mogu objektivno meriti nezavisno od percepcije onih koji vrše merenje. Osnovni atributi se ne mogu dalje dekomponovati. Izvedeni atributi su

kombinacija osnovnih atributa za koje je nekad moguće uvesti objektivnu metriku a nekad samo subjektivnu.

ISO 9126 standard propisuje sledeće osnovne karakteristike i potkarakteristike kvaliteta softvera (*ISO 9126, 2001*):

1. Funkcionalnost (*eng. Functionality*) je esencijalna svrha svakog proizvoda. U slučajevima kada posmatrani servis obavlja samo jednu funkciju njegova funkcionalnost se veoma lako definiše. Međutim, ako servis ima mogućnost pružanja većeg broja usluga, kao na primer, bankomat, onda se funkcionalnost mora specificirati preko liste svih funkcija koje servis pruža.

   Bitna odlika funkcionalnosti, za razliku od drugih karakteristika, je da se prisustvo ili odsustvo svake pojedinačne funkcije izražava bulovskom logikom. Pored toga, potrebno je da se istakne da se standard odnosi samo na funkcionalnost softvera koji pruža uslugu, a ne i na funkcionalnost celog procesa (u okviru koga neke funkcije ne moraju biti podržane od strane softvera).

   Potkarakteristike

   - Pogodnost (*eng. Suitability*) – mera u kojoj web servis odgovara specificiranim funkcijama.
   - Tačnost (*eng. Accurateness*) – tačnost obavljanja specificiranih funkcija.
   - Interoperabilnost (*eng. Interoperability*) – sposobnost određene komponente softvera da radi u interakciji sa drugim komponentama sistema.
   - Usaglašenost (*eng. Compliance*) – usaglašenost softvera sa odgovarajućim relevantnim zakonskim propisima.
   - Sigurnost (*eng. Security*) – mogućnost sprečavanja neautorizovanog pristupa softverskim funkcijama.
2. Pouzdanost (*eng. Reliability*) označava sposobnost sistema da funkcioniše na definisani način u okviru datog perioda vremena.

   Potkarakteristike

   - Postojanost (*eng. Maturity*) – učestanost otkaza softverskih funkcija.
   - Tolerancija greške (*eng. Fault tolerance*) – sposobnost softvera da se izdrži privremeni otkaz neke komponente i da nastavi normalan rad.

- Mogućnost oporavka (*eng. Recoverability*) – sposobnost vraćanja sistema u puni rad uključujući očuvanje konzistentnosti podataka i uspostavljene mrežne konekcije.

3. Upotrebljivost (*eng. Usability*) je tesno vezana sa funkcionalnosti sistema i odnosi se na jednostavnost njegovog korišćenja za obavljanje određene funkcije.

   Potkarakteristike

   - Razumljivost (*eng. Understandability*) – je određena jednostavnošću razumevanja funkcija sistema.
   - Lakoća učenja (*eng. Learnability*) – izražava napor koji je potrebno uložiti da bi se ovladalo korišćenjem sistema (posebno se izražava za početnike, iskusne eksperte, povremene korisnike itd.).
   - Operativnost (*eng. Operability*) – mogućnost jednostavnog korišćenja od strane određenog korisnika u datim uslovima.

4. Efikasnost (*eng. Efficiency*) je vezana za resurse koji se koriste kada sistem obezbeđuje zahtevanu funkcionalnost. Tako su, na primer, indikatori efikasnosti zauzeće prostora na disku ili memoriji, kao i korišćenje mrežnih resursa.

   Potkarakteristike

   - Vremenska efikasnost (*eng. Time behaviour*) – je određena vremenom odziva za dati protok transakcija.
   - Potrošnja resursa (*eng. Resource behaviour*) – izražava utrošene resurse pri obavljanju transakcija.

5. Održivost (*eng. Maintainability*) se odnosi na sposobnost identifikacije i otklanjanja greške u okviru softverske komponente. Ova karakteristika zavisi od čitljivosti softverskog koda, modularnosti i kompleksnosti.

   Potkarakteristike

   - Mogućnost analize (*eng. Analyzability*) – izražava sposobnost identifikacije uzroka greške u softveru.
   - Promenljivost (*eng. Changeability*) – izražava napor koji je potrebno uložiti da bi se u sistem unele izvesne promene.

- Stabilnost (*eng. Stability*) – karakteriše osetljivost sistema na promene, odnosno negativni uticaj (*eng. impact*) koji promene mogu da imaju na rad sistema.
- Mogućnost testiranja (*eng. Testability*) – mera napora koji je potrebno uložiti da bi se proverila ispravnost sistema posle promena.

6. Prenosivost (*eng. Portability*) je karakteristika kojom se izražava adaptabilnost softvera na promenu okruženja ili zahteva u pogledu funkcionalnosti.

   Potkarakteristike

   - Adaptivnost (*eng. Adaptability*) – sposobnost sistema da prihvati nove specifikacije ili izmenjeno radno okruženje.
   - Jednostavnost instalacije (*eng. Installability*) – izražava napor koji je potrebno uložiti da bi se softver instalirao.
   - Usaglašenost (*eng. Conformance*) – mogućnost korišćenja (odnosno prenosa) na različite platforme.
   - Zamenljivost (*eng. Replaceability*) – specificira „plug and play“ aspekt, odnosno jednostavnost kojom se određena softverska komponenta može zameniti u specificiranom okruženju.

Pored navedenih opštih, standardima propisanih, QoS mera u literaturi postoji i mnoštvo drugih pristupa kako u pogledu QoS mera koje se predlažu tako i u pogledu načina njihove klasifikacije. Jedan od pristupa koji se u relevantnoj literaturi najčešće navodi je (*Menascé, 2002*).

Konačno treba napomenuti da pružaoci servisa zapravo mogu različitim korisnicima nuditi iste servise sa različitim nefunkcionalnim karakteristikama. Naime, sve je češća pojava da pružaoci usluga nude nekoliko različitih QoS nivoa jednog servisa kako bi se prilagodili zahtevima korisnika.

2.5.2. Domensko-specifične nefunkcionalne karakteristike

Potrebno je napomenuti da postoje i druge, domensko-specifične, nefunkcionalne karakteristike koje često takođe imaju veliku ulogu prilikom selekcije servisa kao što su cena, geografska lokacija, mogućnost plaćanja određenom kreditnom karticom itd. Na primer, kao što se navodi u (*Canfora & al, 2006*), ukoliko je u pitanju izbor servisa za rezervaciju avionske karte i hotela, cilj bi bio postići kompromis između cene i kategorije hotela, favorizujući pri tome hotele i avionske kompanije koje pružaju najveći broj popusta. Autori takođe navode primer kompozicije za obradu fotografija kod koje je potrebno maksimizirati rezoluciju slike ili broja boja koje se pojavljuju na slici a da se pri tome minimizira vreme odziva i cena. Pristupi predloženi u (*Chen & al, 2003; Aggarwal & al, 2004;Agarwal & Lamparter, 2005; Lin & al, 2005; Canfora & al, 2006; Sohrabi & al, 2006; Zhang & al, 2006; Canfora & al, 2008; Herssens & al, Januar 2008; Sora & al, 2009; Tran & al, 2009; Garg & al, 2013*) omogućavaju da se prilikom selekcije servisa uzmu u obzir i domensko-specifične nefunkcionalne karakteristike servisa.

Jedan od sveobuhvatnijih pristupa je predložen u (*Canfora & al, 2006*) u kome se dozvoljava mogućnost specificiranja proizvoljnih QoS atributa, definisanjem tipa vrednosti (matematičke, skupovne, logičke ili tekstualne), skale, kao i načina na koji će dati atributi agregirati ukoliko je reč o kompoziciji servisa. U (*Aggarwal & al, 2004; Agarwal & Lamparter, 2005; Gronmo & Jaeger, Jun 2005; Gronmo & Jaeger, Decembar 2005; Canfora & al, 2006; Jaeger, 2006; Canfora & al, 2008; Herssens & al, Januar 2008; Agarwal & al, 2009; Sora & al, 2009;. Tran & al, 2009*) se predlažu ontologije i jezici pomoću kojih se mogu specificirati različiti aspekti, odnosno specifične osobine, nefunkcionalnih karakteristika što obuhvata i odnose koji mogu postojati među njima.

2.5.3. Nefunkcionalne karakteristike kompozicije web servisa – Agregacija

Određivanje QoS za kompoziciju postavlja sledeći izazov: potrebno je na neki način agregirati QoS vrednosti pojedinačnih servisa u kompoziciji, pri čemu način agregacije (sumiranje, množenje, minimum itd.) može zavisiti od prirode relevantnih QoS atributa kao i od strukture odnosno toka izvršavanja same kompozicije. Treba istaći da postoje i pristupi u kojima se agregacija, pa posledično i sama selekcija, vrše ne uzimajući u

obzir strukturu kompozicije (*Berbner & al, 2006; Alrifai & Risse, 2009; Claro & al, 2005; Yu & Lin, Decembar, 2005; Gao & al, 2009; Liu & al, 2009; Qi & al, 2010; Luo & al, 2011*).

Jedan od pristupa koji se najčešće navodi predložen je u (*Cardoso, 2002*). U svojoj doktorskoj disertaciji (*Cardoso, 2002*) autor najpre predlaže teorijski QoS model koji omogućava formalnu specifikaciju i predstavljanje QoS mera za različite zadatke u kompoziciji. Autor zatim predstavlja i algoritam (nazvan *Stochastic Workflow Reduction - SWR*) za automatsko izračunavanje QoS celokupne kompozicije na osnovu QoS mera zadataka od kojih se ona sastoji. Algoritam je zasnovan na matematičkom modelu kojim se formalno opisuje skup formula za izračunavanje QoS mera. Predloženi algoritam predviđa primenu skupa redukcionih pravila pomoću kojih se kompozicija svodi na jedan atomski zadatak kojim je predstavljen QoS cele kompozicije. Svaki put kada se primeni određeno redukciono pravilo struktura kompozicije se menja. Nakon nekoliko iteracija preostaće samo jedan zadatak koji će sadržati QoS mere koje odgovaraju celokupnoj kompoziciji. Autor navodi da ovaj skup pravila korespondira skupu inverznih operacija koje se mogu koristiti za konstrukciju kompozicije i u radu se ograničava na predefinisan skup operacija kako bi se onemogućilo definisanje kompozicija koje nisu validne. U skladu sa tim predviđeno je šest različitih redukcionih pravila[1]: sekvencijalno (*eng. sequential*), paralelno (*eng. parallel*), uslovno (*eng. conditional*), otporno na greške (*eng. fault-tolerant*), ciklično (*eng. loop*) i mrežno[2] (*eng. network*). Predloženi model u slučaju uslovnih i cikličnih struktura podrazumeva i da se za svaku od grana definiše verovatnoća izvršavanja (odnosno ulaska/izlaska iz ciklusa) koja može biti bilo zadata od strane korisnika bilo određena na osnovu podataka dobijenih praćenjem dotadašnjeg izvršavanja kompozicije.

U (*Zeng & al, 2003*) a zatim i u (*Zeng & al, 2004*) se daju formule za agregaciju koje se, u zavisnosti od samog QoS atributa, zasnivaju na sumi, proseku, proizvodu i kritičnom putu. Pored toga, autori za ciklične strukture predlažu primenu tehnike za odmotavanje (*eng. unfolding*) kako bi se obezbedio acikličan dijagram promene stanja koji ima konačan broj putanja izvršavanja. Predložena tehnika, podrazumeva da se na

[1] Autor navodi da su navedena pravila izabrana imajući u vidu da je njihova implementacija podržana u većini workflow sistema i šablona (*eng. pattern*), mada navodi da je moguće razviti i dodatna pravila.
[2] Svrha mrežnog šablona (*eng. network pattern*) je da obezbedi strukturnu i hijerarhijsku podelu date kompozicije u slojeve, kako bi se lakše razumeo način grupisanja povezanih zadataka u jedinice funkcionalnosti.

osnovu podataka dobijenih praćenjem dotadašnjeg izvršavanja odredi maksimalan mogući broj iteracija datog ciklusa. Stanja unutar ciklusa će se zatim klonirati, shodno ideju predloženoj u (*Gillmann & al, 2002*) onoliko puta koliko će se dati ciklus izvršiti. Ovakav pristup predlaže se i u (*Ardagna & Pernici, 2005*) dok isti autori u sledećem radu (*Ardagna & Pernici, 2007*) predlažu nešto drugačiji pristup (*eng. loop peeling*) u kome se iteracije datog ciklusa predstavljaju pomoću sekvence grana. Svaka grana obuhvata i uslov kojim se proverava da li je potrebno nastaviti iteriranje ili prekinuti ciklus. Definisanje odgovarajućih uslova za svaku od grana podrazumeva najpre utvrđivanje maksimalnog broja iteracija a zatim određivanje raspodele verovatnoća za svaki mogući broj ponavljanja ciklusa (*0, 1, ...n*, gde *n* predstavlja maksimalni mogući broj iteracija). Autori navode se verovatnoće izvršavanja grana u uslovnim strukturama kao i raspodela verovatnoća vezanih za broj iteracija u cikličnim strukturama mogu odrediti na način koji je predložen u (*Cardoso, 2002*). Agregacija samih atributa se u (*Ardagna & Pernici, 2005* i *Ardagna & Pernici, 2007*) vrši pomoću težinske sume, proizvoda, minimuma ili maksimuma. Treba napomenuti da obe grupe autora (*Zeng & al, 2003; Zeng & al, 2004; Ardagna & Pernici, 2005; Ardagna & Pernici, 2007*) navode da predloženi pristupi nisu vezani samo za izdvojeni skup QoS atributa. Pri tome u (*Ardagna & Pernici, 2007*) autori posebno ističu da smatraju da je skup QoS atributa koji su izdvojili reprezentativan, budući da pokriva sve moguće tipove agregacije. Međutim, ovi pristupi ipak podrazumevaju da su kvalitativni atributi zapravo numerički izraženi a pri tome ostaje i nejasno na koji način bi se tretirali domensko-specifični atributi koji se agregiraju pomoću korisnički definisanih funkcija, mada je u (*Ardagna & Pernici, 2005*) ipak napomenuto da se podrazumeva da se mogu koristiti isključivo oni QoS atributi koji se mogu agregirati pomoću sume, proizvoda, proseka, minimuma ili maksimuma.

Pristup agregaciji predložen u (*Canfora & al, 2004*) se zasniva na (*Cardoso, 2002*) ali se ciklične strukture tretiraju na način sličan onome predloženom u (*Zeng & al, 2004*) s tim da se predlaže da se izračunavanje QoS vrednosti ciklične strukture zasniva na procenjenom broju iteracija bez potrebe za odmotavanjem samog ciklusa. U radu je dat tabelarni prikaz (Tabela 1.) načina agregacije pojedinih QoS atributa (uključujući i domensko-specifične) uzimajući u obzir njihove karakteristike kao i korišćene strukture (sekvencijalna, uslovna, paralelna i ciklična) koji se poklapa sa

pristupom predloženim u (*Cardoso, 2002*) izuzev u slučaju cikličnih struktura. Autori napominju da se navedo samo primeri agregacije atributa koji su relevantni za izložene primere, te da data tabela nije kompletna. Izračunavanje QoS za kompoziciju se zatim ostvaruje obilaženjem čvorova u grafu i rekurzivnom primenom odgovarajućih funkcija.

Tabela 1. Načini agregacije QoS atributa u zavisnosti od korišćenih struktura (*Canfora & al, 2004*)

QoS Attr.	Sequence	Switch	Flow	Loop
Time (T)	$\sum_{i=1}^{m} T(t_i)$	$\sum_{i=1}^{n} p_{ai} * T(t_i)$	$Max\{T(t_i)_{i\in\{1\dots p\}}\}$	$k * T(t)$
Cost (C)	$\sum_{i=1}^{m} C(t_i)$	$\sum_{i=1}^{n} p_{ai} * C(t_i)$	$\sum_{i=1}^{p} C(t_i)$	$k * C(t)$
Availability (A)	$\prod_{i=1}^{m} A(t_i)$	$\sum_{i=1}^{n} p_{ai} * A(t_i)$	$\prod_{i=1}^{p} A(t_i)$	$A(t)^k$
Reliability (R)	$\prod_{i=1}^{m} R(t_i)$	$\sum_{i=1}^{n} p_{ai} * R(t_i)$	$\prod_{i=1}^{p} R(t_i)$	$R(t)^k$
Custom Attr. (F)	$f_S(F(t_i))$ $i \in \{1\dots m\}$	$f_B((p_{ai}, F(t_i)))$ $i \in \{1\dots n\}$	$f_F(F(t_i))$ $i \in \{1\dots p\}$	$f_L(k, F(t))$

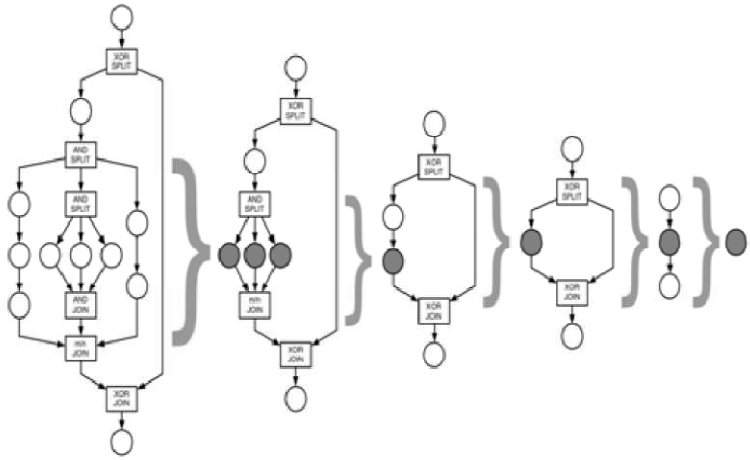
Navedeni pristup autori dalje proširuju u (*Canfora & al, 2006*) definišući jezik i alat za specifikaciju i agregaciju domensko-specifičnih atributa koji mogu biti i tekstualni, skupovni itd. (pod pretpostavkom da se njihove vrednosti na neki način mogu urediti odnosno međusobno porediti). Za svaki novi QoS atribut alat obezbeđuje mogućnost navođenja načina agregacije kroz definisanje formula (uz obavezno navođenje tipa povratne vrednosti) zasnovanih na matematičkim, logičkim, skupovnim i tekstualnim operatorima za svaku od struktura kompozicije. Pri tome autori napominju da definisani način agregacije određenog domensko-specifičnog atributa može biti bilo opšti (te se može ponovo koristiti i za druge kompozicije iz istog ili srodnog domena) ili definisan samo za potrebe razmatrane kompozicije, kao i da se isti domensko-specifični QoS atribut može na različite načine agregirati u zavisnosti od potreba korisnika.

Sveobuhvatniji pristup predložen je u (*Jaeger & al, 2004*) i dalje razrađivan u (*Gronmo & Jaeger, Jun 2005; Jaeger & al, 2005; Jaeger, 2006; Jaeger & Ladner, 2006; Jaeger & Muhl, 2007*) u kome autori predlažu mehanizam za agregaciju QoS atributa, zasnovan na pristupu koji su predložili za modelovanje kompozicije pomoću uzora za kompoziciju, u kome se agregacija postupno vrši na nivou pojedinačnih gradivnih blokova odnosno uzora.[1]. U svojoj doktorskoj disertaciji (*Jaeger, 2006*) autor daje detaljan opis predloženog pristupa koji najpre podrazumeva da se za svaki od identifikovanih uzora za kompoziciju definišu načini agregacije pojedinih QoS atributa

[1] Autori napominju da se predloženi pristup agregaciji zasniva na pristupu koji je predložen u (*Puschner & Schedl, 1997*) za računanje vremena izvršavanja strukturnih uzora u softverskim arhitekturama.

koji pri tome uzimaju u obzir i specifičnosti samih atributa. Agregirane vrednosti za svaki od uzora u datoj kompoziciji se zatim mogu dalje agregirati, primenom hijerarhijskog pristupa, kako bi se dobile agregirane QoS vrednosti celokupne kompozicije. Za izračunavanje agregiranih QoS vrednosti kompozicije modelovane pomoću grafa primenjuje se rekurzivni pristup kojim se dati graf redukuje, korak po korak, naizmeničnim agregiranjem pod-uzora počevši od pod-uzora koji ne sadrže nijedan drugi pod-uzor sve dok se graf ne svede na samo jedan čvor. U svakom od koraka primenjuje se odgovarajuće pravilo agregacije za dati uzor i dati QoS atribut i dobijene vrednosti se koriste u sledećim koracima prilikom agregacije nad-uzora (Slika 7). Primenom navedenog postupka u poslednjem koraku će se dobiti po jedan izraz za svaki od razmatranih QoS atributa.

Prilikom grananja i spajanja se pretpostavlja da svaki od servisa ima podjednaku verovatnoću da bude izabran pri čemu autori posebno napominju da u slučaju OR grananja sve moguće kombinacije moraju biti uzete u obzir. Kada su u pitanju petlje, autori navode da se očekivani broj iteracija procenjuje na osnovu raspoloživih informacija ističući da to zapravo ne mora biti adekvatno budući da taj broj najčešće zavisi od određenih logičkih uslova koji se mogu evaluirati tek prilikom samog izvršavanja kompozicije.



Slika 7. Agregacija korak po korak (*Jaeger, 2006*)

Autori dalje navode da se predloženi pristup agregaciji može primeniti kako prilikom projektovanja kompozicije tako i prilikom njenog izvršavanja, pri čemu se u tom slučaju agregiraju QoS vrednosti dobijene praćenjem (*eng. monitoring*) izvršavanja same

kompozicije. U cilju ilustracije razlike autori navode da se za one QoS atribute koji se agregiraju pomoću aritmetičke sredine, agregacijom prilikom projektovanja kompozicije zapravo dobija samo estimacija, budući da je neophodno unapred proceniti raspodelu izvršavanja pojedinačnih servisa kod uslovnih grananja, dok se praćenjem ostvarenih QoS vrednosti prilikom izvršavanja kompozicije ove raspodele mogu direktno uočiti. Na osnovu podataka dobijenih praćenjem QoS vrednosti prilikom izvršavanja kompozicije svakom pojedinačnom servisu se može dodeliti težina u skladu sa datom raspodelom (tj. brojem poziva datog servisa), te će agregacija u paralelnim strukturama rezultovati ekstrapolacijom vrednosti. Pored toga, u (*Jaeger & Muhl, 2007*) autori navode da agregacija prilikom projektovanja kompozicije može rezultovati samo gornjim i/ili donjim granicama QoS vrednosti te stoga predstavlja samo aproksimaciju vrednosti koje će se ostvariti prilikom samog izvršavanja. U (*Jaeger & al, 2004*) autori posebno daju formule za agregiranje vrednosti izdvojenih QoS atributa (za svaki od sedam identifikovanih uzora za kompoziciju). U svim ovim radovima autori posebno ističu da predloženi pristupi nisu limitirani na navedene skupove atributa već da su pojedini QoS atributi izdvojeni isključivo u cilju ilustrovanja samih pristupa. Međutim, iako je u navedenim radovima predložen sveobuhvatan model u kome se za agregiranje vrednosti primenjuju različite funkcije (kao što su: suma, proizvod, minimum, maksimum, stepenovanje itd.)[1] autori se ograničavaju isključivo na agregaciju kvantitativnih QoS atributa dok se kvalitativni atributi rangiranjem svode na numeričke vrednosti. Takođe treba napomenuti i da autori ističu da se navedeni model zasniva na pretpostavci o nezavisnosti servisa odnosno na pretpostavci da izvršavanje određenog servisa ne utiče na izvršavanje drugih servisa niti na njihove QoS nivoe. Na kraju autori daju i osvrt na model kompozicije predložen u (*Cardoso, 2002*) i navode da iako je dati pristup sličan njihovim on ne identifikuje opšte strukture već analizira određene specifične scenarije primene, te su predložene strukture ograničene na primenjene scenarije. Kao drugi nedostatak tog pristupa navode da dati model ne obuhvata slučajeve paralelnog izvršavanja u kojima je prisutno OR grananje ili spajanje zasnovano na diskriminatoru. U (*Jaeger, 2006*) autor razmatra i pristup predložen u (*Zeng & al, 2004*) navodeći da je iako je agregacija QoS atributa zasnovana na kritičnom putu u opštem slučaju opravdana (pod pretpostavkom da zadaci koji se ne

[1] U navedenim radovima date su formule za agregaciju svih razmatranih atributa i to za svaki od identifikovanih uzora. Imajući u vidu veliki broj formula one su u ovom radu izostavljene.

nalaze na kritičnom putu neće uticati na ukupni QoS kompozicije prilikom njenog izvršavanja) predloženi pristup agregaciji zasnovan na uzorima za kompoziciju ipak daje preciznije izraze za agregaciju koji na adekavatniji način odražavaju QoS koji će se ostvariti prilikom izvršavanja kompozicije i prilaže primer kojim ilustruje izneto tvrđenje. U skladu sa navedenim autor predlaže i proširenje pristupa iz (*Zeng & al, 2004*) u kome bi bi se algoritam zasnovan na kritičnom putu primenio za agregiranje svih relevantnih QoS atributa (a ne samo vremena odziva) te bi se množenjem ili sabiranjem pojedinačnih vrednosti na kritičnim putevima identifikovanim za svaki od QoS atributa zapravo dobili slični rezultati kao i primenom pristupa zasnovanog na uzorima za kompoziciju, mada dalje navodi da je definisanje pravila za agregaciju na osnovu uzora za kompoziciju značajno jednostavniji zadatak od razvoj algoritma kojim bi se identifikovali kritični putevi za svaku od relevantnih QoS karakteristika.

Većina drugih autora koristi izložene pristupe, mada treba istaći da se u velikom broju slučajeva razmatra samo sekvencijalno izvršavanje servisa kao i da se za agregaciju QoS atributa koristi samo mali broj funkcija tj. suma (*Claro & al, 2005; Liu & al, 2009; Gao & al, 2009*), suma ili proizvod (*Yu & Lin, Jul, 2005; Yu & Lin, Decembar, 2005; Yu & al, 2007; Qi & al, 2010*) i u nekim slučajevima minimum i maksimum (*Zeng & al,2003; Zeng & al, 2004; Ardagna & Pernici, 2005; Berbner & al, 2006; Ardagna & Pernici, 2007; Wada & al, 2008; Alrifai & Risse, 2009; Wang & al, 2010; Wada & al, 2012*). Pri tome se autori u takvim slučavima vezuju isključivo za kvantitativne QoS atribute (opšte i ponekad domensko-specifične).

Konačno, sveobuhvatniji pristupi su predloženi u (*Aggarwal & al, 2004; Agarwal & Lamparter, 2005; Canfora & al, 2006; Luo & al, 2011; Sun & Zhao , 2012*) u kojima se omogućava agregacija QoS atributa pomoću matematičkih, skupovnih, logičkih i lingvističkih operatora i to za sve moguću strukture kompozicije.

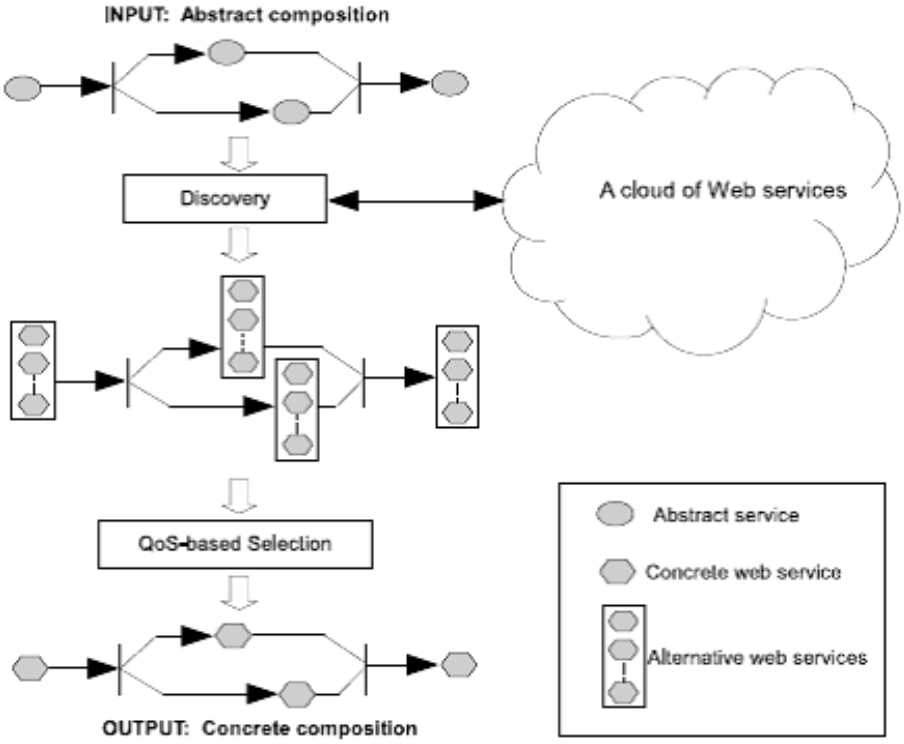
Sa gledišta ovog rada način agregacije QoS atributa zapravo nije relevantan budući da predloženi pristupi mogu podržati bilo koji način agregacije. Pri tome se pošlo od tvrdnje, izložene u (*Wada & al, 2008; Alrifai & Risse, 2009; Wada & al, 2012*), da se paralelni, uslovni i ciklični modeli mogu transformisati u sekvencijalni model redukcijom predloženom u (*Cardoso, 2002*).

3. MODELOVANJE PROBLEMA SELEKCIJE WEB SERVISA NA OSNOVU NEFUNKCIONALNIH KARAKTERISTIKA

3.1. Postavka problema

Budući da se softverske komponente nezavisno razvijaju i objavljuju na Internetu, svakim danom sve je veći broj dostupnih web servisa koji pružaju istu funkcionalnost. Očigledno je da se tada izbor konkretnog servisa iz mnoštva raspoloživih servisa mora zasnivati i na njihovim nefunkcionalnim karakteristikama (QoS). Međutim, broj relevantnih nefunkcionalnih karakteristika može biti proizvoljan, pri čemu se skup QoS atributa koji su relevantni za selekciju može razlikovati od korisnika do korisnika. Imajući u vidu navedeno, **problem selekcije pojedinačnih web servisa** bi se mogao definisati na sledeći način: *potrebno je za dati funkcionalni zahtev izabrati konkretan servis, iz skupa raspoloživih servisa koji obezbeđuju datu funkcionalnost, koji je optimalan u pogledu relevantnih nefunkcionalnih karakteristika*.

Sa druge strane, kada je za realizaciju određenog funkcionalnog zahteva neophodna kompozicija servisa ovaj zadatak postaje još kompleksniji. Prilikom definisanja kompozicije za realizaciju nekog složenog funkcionalnog zahteva prvo je neophodno odrediti koji su to delovi funkcionalnosti koje svaka od komponenti treba da pruži kao i redosled njihovog izvršavanja. U ovom koraku razmatraju se samo funkcionalne karakteristike servisa. Kada se kompozicija definiše sledeći korak je izbor konkretnih servisa koji će se uključiti u kompoziciju i obaviti definisane zadatke. Na osnovu definicije kompozicije, najpre se, pretragom registra servisa (npr. UDDI) identifikuju pogodni servisi za svaku od komponenti kompozicije uparivanjem zahtevanih funkcionalnosti sa opisima servisa dostupnih u registru. Kao rezultat dobija se skup raspoloživih servisa (tj. njihovih opisa i lokacija) za svaku od komponenti kompozicije. Nakon toga se vrši selekcija kako bi se iz raspoloživog skupa izabrali optimalni servisi za svaki od zadataka. Dakle prvi korak je planiranje čiji je rezultat apstraktna kompozicija servisa a drugi korak je selekcija čiji je rezultat konkretna kompozicija u kojoj je svakom apstraktnom servisu dodeljen konkretan servis iz skupa raspoloživih servisa (Slika 8).



Slika 8. Konceptualni pogled na problem selekcije web servisa za kompoziciju (*Alrifai &Risse, 2009*)

Pri tome se selekcija servisa vrši na takav način da rezultujuća kompozicija ne samo zadovolji postavljene funkcionalne zahteve već bude i optimalna u pogledu relevantnih QoS karakteristika. Pored toga, treba imati u vidu i da se često prilikom selekcije nameću i određena ograničenja u pogledu minimalnih nefunkcionalnih karakteristika koje kompozicija mora obezbediti, pri čemu, u opštem slučaju, skup QoS atributa na osnovu kojih se vrši selekcija ne mora nužno biti isti kao i skup QoS atributa nad kojima se nameću ograničenja. Drugim rečima, **problem selekcije web servisa za kompoziciju** bi se mogao definisati na sledeći način: *Potrebno je za svaku od komponenti kompozicije izabrati konkretan servis, iz skupa raspoloživih servisa koji obezbeđuju zahtevanu funkcionalnost, kako bi se dobila kompozicija koja je optimalna u pogledu relevantnih nefunkcionalnih karakteristika a koja istovremeno zadovoljava sva postavljena QoS ograničenja*. Potrebno je napomenuti i da se selekcija servisa za kompoziciju može vršiti i dinamički izborom optimalnog skupa servisa neposredno pre izvršavanja kompozicije ili čak u toku njenog izvršavanja. Za razliku od toga statički pristup podrazumeva da se unapred izaberu servisi i permanentno dodele kompoziciji. Međutim, skup raspoloživih servisa ne mora u svakom trenutku biti isti, odnosno u

međuvremenu se mogu pojaviti novi servisi sa boljim QoS karakteristikama ili se može desiti da u vreme izvršavanja kompozicije predviđeni servis bude nedostupan ili da se njegovi QoS nivoi značajno promene. Sa druge strane i zahtevi poslovanja se često menjaju što može iziskivati i promenu zahteva vezanih za QoS kompozicije.

Budući da se selekcija kako pojedinačnih servisa tako i servisa za kompoziciju najčešće vrši na osnovu većeg broja nefunkcionalnih karakteristika jedan od ključnih problema prilikom selekcije je heterogenost samih QoS mera koje, kao što je navedeno u prethodnom poglavlju, mogu biti opšte prihvaćene ili domensko-specifične, kvalitativne ili kvantitativne i izražene pomoću različitih, često neuporedivih, jedinica mere, pri čemu je za neke QoS atribute poželjno ostvariti maksimalnu moguću vrednost a za druge minimalnu. Ako se tome doda i činjenica da zahtevi u pogledu relevantnih nefunkcionalnih karakteristika mogu biti, a često i jesu, delimično ili u potpunosti kontradiktorni (usled prirode odabranih QoS mera), očigledno je da se selekcija ne može izvršiti bez aktivnog učešća donosioca odluke odnosno korisnika. Svaki korisnik najpre mora odabrati skup nefunkcionalnih karakteristika koji je za njega relevantan. Suočen sa činjenicom da je gotovo izvesno da se sve odabrane karakteristike ne mogu istovremeno optimizovati korisnik mora i da iskaže preference u pogledu postavljenih zahteva kao i kompromise koje je spreman da prihvati.

Stoga se može zaključiti da definisani problemi selekcije pojedinačnih web servisa i selekcije web servisa za kompoziciju zapravo predstavljaju probleme višekriterijumskog odlučivanja (*eng. Multiple Criteria Decision Making* – MCDM)[1] u kojima je potrebno iz mnoštva alternativa koje stoje na raspolaganju donosiocu odluke izabrati onu koja na najbolji način zadovoljava ceo skup postavljenih zahteva. Kod većine realnih MCDM problema prisutan je veći broj faktora koji utiču na samu odluku. Svaki od ovih faktora predstavlja neki kriterijum prema kome će se evaluirati dostizanje sveukupnog cilja problema te se svaka alternativa može okarakterisati time u kojoj meri ispunjava postavljeni kriterijum. Sa druge strane, svaka alternativa je najčešće opisana pomoću nekog skupa atributa, te kriterijumi zapravo predstavljaju određeni podskup

[1] Prema (*Hwang & Yoon, 1981*) pojam MCDM se koristi da označi sve pristupe koji se odnose bilo na višeatributivno odlučivanje (*eng. Multiple Attribute Decision Making* – MADM) bilo na višeciljno odlučivanje (*eng. Multiple Objective Decision Making* – MODM). Ukoliko bi bilo potrebno preciznije definisati moglo bi se reći da selekcija pojedinačnih web servisa zapravo predstavlja MADM problem dok problem selekcije web servisa za kompoziciju zapravo predstavlja MODM problem odnosno problem višekriterijumskog programiranja (*eng. Multiple Objective Programming* – MOP).

atributa koje je donosilac odluke izabrao za predstavljanje cilja[1]. Konkretno kada je u pitanju problem selekcije servisa alternative predstavljaju raspoložive servise (ili konkretne kompozicije servisa) opisane pomoću svojih nefunkcionalnih karakteristika. Cilj se može definisati kao izbor onog servisa (ili kompozicije servisa) koji će obezbediti najbolju moguću ukupnu QoS vrednost, pri čemu se za kriterijume uzimaju relevantni QoS atributi.

S obzirom na sve veću aktuelnost izloženog problema, o čemu svedoči i veliki broj radova koji daju pregled u ovoj oblasti (na primer *Rao & Su, 2005; Sathya &al, 2010; Strunk, 2010; Yulu & Xi, 2011; Pejman & al, 2012; Sheng & al, 2014*), u literaturi su predloženi brojni pristupi za njegovo modelovanje i rešavanje koji zavise pre svega od prirode problema odnosno cilja koji se želi postići selekcijom. Broj različitih pristupa i njihova raznovrsnost nedvosmisleno dokazuju da se metode višekriterijumskog odlučivanja i optimizacije mogu efektivno i efikasno koristiti za rešavanje problema selekcije, kako pojedinačnih servisa tako i servisa za kompoziciju, na osnovu nefunkcionalnih karakteristika. Imajući u vidu uspešnost primene ovih metoda namera ovog rada je bila da se ispita mogućnost korišćenja nekih novijih pristupa mekog računarstva i optimizacije, koji nisu do sada primenjivani u ovoj oblasti, kako bi se unapredilo rešavanje definisanog problema.

Sa jedne strane ideja je da se razviju postupci formiranja modela koji će na adekvatan način uzeti u obzir složene verbalne zahteve korisnika, različite odnose koji postoje između relevantnih nefunkcionalnih karakteristika kao i specifičnosti samih mera. Sa druge strane budući da mnogi od postojećih pristupa podrazumevaju pojednostavljenja datog problema ili nameću određena ograničenja u pogledu prirode QoS atributa koji se mogu koristiti za selekciju, čime se zapravo otvara pitanje primenljivosti takvih pristupa u realnim situacijama, predložiće se pristupi za rešavanje problema bez nametanja ovakvih ograničenja i to kako za postojeće modele problema tako i za predložene nove modele.

Shodno postavljenim ciljevima u nastavku ovog poglavlja pažnja će biti usmerena na modelovanje problema dok će se metode koje se predlažu za rešavanje

[1] Autori u (*Hwang & Yoon, 1981*) navode, pozivajući se na (*Keeney & Raiffa, 1976*), da se u literaturi pojmovi: atribut (*eng. attribute*), kriterijum (*eng. criteria*), cilj (*eng. objective* i *eng. goal*) i dimenzija (*eng. dimension*) često međusobno zamenjuju budući da ne postoje univerzalne, opšte prihvaćene, definicije tih pojmova.

postavljenih problema razmatrati u narednim poglavljima. Pri tome će se umesto uobičajenog pregleda literature u nastavku rada, kao motivacija za pristupe koji se u ovom radu predlažu, dati osvrt na neke od postojećih pristupa modelovanju i rešavanju navedenih problema koji se izdvajaju zbog svoje specifičnosti ili značaja.

Budući da selekcija pojedinačnih servisa zapravo predstavlja samo specijalan slučaj problema selekcije servisa za kompoziciju (tj. može se posmatrati i kao kompozicija koja se sastoji samo od jedne komponente) u nastavku će najpre biti data opšta matematička formulacija problema selekcije servisa.

Međutim, ovakav višekriterijumski problem nije jednostavno rešavati jer kako se navodi u (*Hwang & Yoon, 1981*) ono što karakteriše sve MCDM probleme (pa samim tim i problem selekcije servisa) je postojanje većeg broja kvantitativnih i/ili kvalitativnih kriterijuma koji pri tome mogu biti međusobno konfliktni i izraženi pomoću različitih često nesamerljivih, odnosno neuporedivih, jedinica mera. Stoga različiti autori, u cilju prevazilaženja navedenih teškoća, predlažu svođenje problema selekcija servisa na različite jednostavnije modele koji su najčešće u tesnoj vezi sa izabranim načinom rešavanja samog problema. U sledećem potpoglavlju će se, na osnovu dostupne literature, dati osvrt na neke od postojećih pristupa (koji se izdvajaju zbog svoje specifičnosti ili značaja) i objasniti motivacija za uvođenje novih pristupa.

Na kraju će biti predložena dva nova pristupa za modelovanje problema selekcije servisa koji će omogućiti da se uzme u obzir priroda samih QoS mera kao i moguća suprotstavljenost kriterijuma ali i da se na adekvatniji način iskažu preference korisnika u pogledu relevantnih nefunkcionalnih karakteristika.

Konačno treba istaći da se u ovom radu podrazumeva da je skup raspoloživih servisa unapred zadat tj. da je prethodno izvršeno sparivanje funkcionalnih zahteva sa interfejsima servisa koji se nude u odgovarajućim registrima. Pored toga se uzima da vrednost određenog QoS atributa zapravo predstavlja vrednost za koju se očekuje da će biti ostvarena na osnovu raspoloživih podataka. Pri tome se u ovom radu ne razmatra da li objavljeni QoS nivo odgovara stvarnom nivou tj. koliko je pružalac servisa pouzdan.

3.2. Opšta matematička formulacija problema

Kako bi se definisani problem selekcije servisa formalno iskazao uvode se sledeće pretpostavke. Neka je data apstraktna kompozicija *ACS* koja se sastoji od *m* komponenti $C=\{c_i\}$, $i=1,...,m$ i neka za svaku od datih komponenti c_i postoji skup servisa kandidata $S_i=\{s_{ij}\}$, $j=1,...,p_i$, pri čemu broj raspoloživih kandidata ne mora biti isti za svaku od komponenti. Jedna konkretna kompozicija *CS* zapravo predstavlja jednu od mogućih kombinacija servisa tj. skup servisa ($CS=\{s_{ij}\}, i=1,...,m,\ j\in\{1,...,p_i\}$) pri čemu za svaku od komponenti apstraktne kompozicije c_i mora biti izabran tačno jedan servis s_{ij}.

Dalje, neka je $\{q^k\}$, $k=1,..,n$, skup nefunkcionalnih karakteristika odnosno QoS atributa koji predstavljaju kriterijume selekcije od kojih je za neke potrebno ostvariti maksimalnu, a za druge minimalnu moguću vrednost. Budući da skup QoS atributa koji predstavljaju kriterijume selekcije ne mora nužno biti isti kao i skup QoS atributa nad kojima se nameću ograničenja u pogledu minimalnih nefunkcionalnih karakteristika koje kompozicija mora da zadovolji[1], neka *l* označava QoS atribute za koje su definisana ograničenja, pri čemu su granice date pomoću skupa $Q=\{Q^1,...,Q^r\}$. Konačno, neka q_{ij}^k i odnosno q_{ij}^l predstavljaju QoS vrednosti koje ostvaruje pojedinačni servis s_{ij}, dok $q_{CS}^k(q_{ij}^k)$ i $q_{CS}^l(q_{ij}^l)$ predstavljaju agregiranu QoS vrednost celokupne kompozicije za *k*-ti QoS atribut na osnovu koga se vrši selekcija odnosno za *l*-ti QoS atribut vezan za ograničenja, respektivno.

Uzimajući u obzir prirodu, odnosno smer optimizacije, relevantnih QoS atributa matematički model problema bi se sada sažeto mogao formulisati na sledeći način:

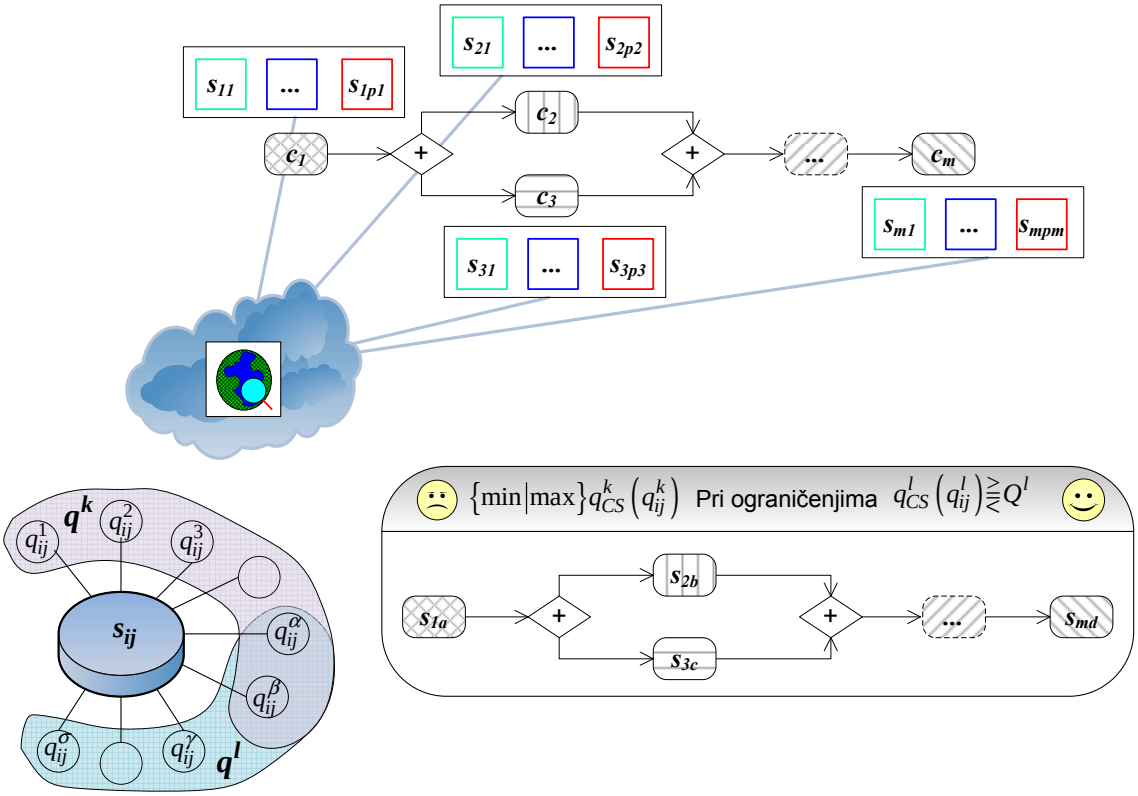
$$\{\min|\max\}\ q_{CS}^k\left(q_{ij}^k\right),\quad k=1,...,n,$$

pri ograničenjima:

$$q_{CS}^l\left(q_{ij}^l\right)\begin{matrix}\geq\\=\\\leq\end{matrix}Q^l,\quad l=1,...,r.$$

Shodno uvedenim oznakama problem selekcije servisa za kompoziciju na osnovu nefunkcionalnih karakteristika ilustrovan je na Slici 9.

[1] U opštem slučaju ova dva skupa mogu imati presek koji nije prazan skup.



Slika 9. Ilustracija problema selekcije servisa za kompoziciju

Jedno rešenje (odnosno jedna konkretna kompozicija *CS*) u prostoru rešenja može se predstaviti pomoću skupa binarnih vrednosti koje ukazuju na to da li je određeni servis uključen u kompoziciji ili ne, te je broj promenljivih u modelu zapravo jednak ukupnom broju raspoloživih servisa. Drugim rečima rešenje se može definisati kao skup vektora promenljivih $x = \{x_i\}$, $x_i = [x_{i1}, ..., x_{ip_i}]$ takvih da je $x_{ij} = 1$ ukoliko je servis s_{ij} uključen u kompoziciju *CS* i $x_{ij} = 0$ u suprotnom, kao što je ilustrovano na Slici 10.

c_1		c_2			c_n	
s_{11}	0	s_{21}	0		s_{n1}	0
s_{12}	1	s_{22}	0	...	s_{n2}	1
s_{13}	0	s_{23}	0		s_{n3}	0
⋮	⋮	s_{24}	0		s_{n4}	0
s_{1p_1}	0	s_{25}	1		⋮	⋮
		⋮	⋮		s_{np_n}	0
		s_{2p_2}	0			

Slika 10. Grafička reprezentacija rešenja

S obzirom na to da se postavljeni kriterijumi i ograničenja zapravo odnose na nefunkcionalne karakteristike celokupne kompozicije, a ne pojedinačnog servisa, pretpostaviće se da $q^k(x)$ i $q^l(x)$ predstavljaju agregiranu QoS vrednost kompozicije za k-ti QoS atribut na osnovu koga se vrši selekcija odnosno l-ti atribut vezan za ograničenja, respektivno. Imajući u vidu prirodu odnosno smer optimizacije relevantnih QoS atributa, u cilju preciznijeg formulisanja modela, pretpostaviće se da je za n_1 Qos kriterijuma poželjno ostvariti maksimalnu moguću vrednost a za preostalih $(n - n_1)$ što manju vrednost. Na isti način pretpostavlja se da je za r_1 atributa ograničenja data minimalna a za preostalih $(r - r_1)$ maksimalna QoS vrednost koju kompozicija treba da obezbedi. Sada se model može formulisati na sledeći način:

$$\max q^k(x),\ k = 1,\ldots, n_1,$$

$$\min q^k(x),\ k = n_1+1,\ldots, n,$$

pri ograničenjima:

$$q^l(x) \geq Q^l, \quad l = 1,\ldots, r_1,$$

$$q^l(x) \leq Q^l, \quad l = r_1+1,\ldots, r,$$

$$\sum_{j=1}^{p_i} x_{ij} = 1, \quad i = 1,\ldots, m,$$

$$x_{ij} \in \{0,1\}, \quad j = 1,\ldots, p_i;\ i = 1,\ldots, m.$$

Gledano potpuno opšte ovako definisan problem selekcije servisa predstavlja:

- *problem globalne optimizacije* sa ograničenjima,
- *višekriterijumski problem* budući da se selekcija servisa najčešće vrši na osnovu većeg broja nefunkcionalnih karakteristika,
- *kombinatorni problem* budući da je dopustivi skup rešenja diskretan, pri čemu se u ovom radu posmatra kao *problem celobrojnog*, odnosno preciznije rečeno, *0 – 1 programiranja* jer je dopustivi skup podskup skupa $B = \{0,1\}^{m \times p_i}$
- *problem nelinearanog programiranja* budući da, u opštem slučaju, kriterijumske funkcije i/ili funkcije ograničenja kojima se opisuju zahtevi vezani za QoS celokupne kompozicije, mogu biti nelinearne funkcije (ukoliko se selekcija vrši na osnovu QoS atributa koji se agregiraju pomoću nelinearnih funkcija).

Imajući u vidu prirodu QoS mera kao i moguću suprotstavljenost kriterijuma očigledno je da ovakav problem nije jednostavno rešavati, o čemu će biti više reči u poglavlju 5.

Međutim, treba napomenuti da kada se razmatra selekcija pojedinačnih servisa problem postaje značajno jednostavniji budući da se definisani kriterijumi zapravo odnose na nefunkcionalne karakteristike samih servisa a ne celokupne kompozicije (tj. $q^k(x) = q_{ij}^k$), te kao što je prethodno istaknuto problem zapravo postaje problem višekriterijumskog odlučivanja. Pored toga iako je u opštem slučaju, kada je reč o selekciji pojedinačnih servisa, moguće definisati i ograničenja takva ograničenja bi zapravo služila da se unapred odbace oni servisi koji ih ne zadovoljavaju.

Na kraju treba istaći da je, u cilju ilustracije pristupa koji se u ovom radu predlažu, bilo neophodno da se odabere konkretan skup QoS atributa što ne znači da su predloženi pristupi ograničeni samo na njih. Polazeći od atributa koji se navode u najvećem broju radova izabran je reprezentativni skup (predstavljen u Tabeli 2):

Tabela 2. Izabrani QoS atributi

QoS atribut	*Oznaka*	*Opis*	*Jedinica mere*
Vreme odziva (*eng. Response time*)	RT	Vreme potrebno da se pošalje zahtev i primi odgovor.	milisekund
Kašnjenje (*eng. Latency*)	L	Ukupno vreme trajanja propagacije zahteva između korisnika i servisa (u oba smera).	milisekund
Raspoloživost (*eng. Availability*)	A	Verovatnoća da je servis u funkciji.	procenat
Pouzdanost (*eng. Reliability*)	R	Odnos broja poruka o greškama i ukupnog broja poruka.	procenat
Bezbednost (*eng. Security*)	S	Nivo i vrsta zaštite koju servis pruža.	Opisna (Odlična, Vrlo dobra, Dobra, Dovoljna, Nedovoljna)
Enkripcija (*eng. Encryption*)	E	Ukazuje na to da li servis obezbeđuje enkripciju podataka	Opisna (Da/Ne)

Pored toga, radi jednostavnosti, a ne umanjujući pri tome opštost predloženih pristupa, u nastavku rada će se smatrati da je skup QoS atributa nad kojima su zadata ograničenja isti kao i skup atributa na osnovu kojih se vrši selekcija.

3.3. Postojeći pristupi modelovanju

U ovom poglavlju najpre će se dati osvrt na neke od postojećih pristupa koji se posebno izdvajaju zbog svoje relevantnosti, specifičnosti ili zato što se najčešće referenciraju u literaturi. Pri tome treba napomenuti da je usled brojnosti pristupa i neraspoloživosti određene literature nemoguće dati sveobuhvatan pregled.

Postojeći pristupi za modelovanje problema selekcije servisa na osnovu nefunkcionalnih karakteristika mogu se razmatrati sa nekoliko aspekata:

- Vrsta problema (pojedinačna selekcija ili selekcija za kompoziciju)
- Podržani QoS atributi
- Način izražavanja preferenci korisnika

i ukoliko je reč o problemu selekcije servisa za kompoziciju:

- Način agregacije atributa
- Struktura kompozicije
- Izabrani model
- Podržana QoS ograničenja

Vrsta problema

Problem selekcije pojedinačnih servisa za dati funkcionalni zahtev razmatra se u:

Chen & al, 2003; Zeng & al,2003; Seo & al, 2005; Tong & Zhang, 2006;Wang & al, 2006; Zhang & al, 2006; Xiong & Fan, 2007; Godse & al, 2008; Herssens & al, Januar, 2008; Herssens & al, Jun, 2008; Agarwal & al, 2009; Sora & al, 2009; Tran & al, 2009; Almulla & al, 2011; Karim & al, 2011; Gao & Ma, 2013; Garg & al, 2013.

Problem selekcije servisa za kompoziciju razmatra se u:

Cardoso, 2002; Zeng & al,2003; Aggarwal & al, 2004; Canfora & al, 2004; Zeng & al, 2004; Agarwal & Lamparter, 2005; Ardagna & Pernici, 2005; Canfora & al, Jun 2005; Canfora & al, Jul 2005; Claro & al, 2005; Gronmo & Jaeger, Jun 2005; Jaeger & al, 2005; Lin & al, 2005; Yu & Lin, Jul, 2005; Yu & Lin, Decembar, 2005; Berbner & al, 2006; Canfora & al, 2006; Gao & al, 2006; Jaeger, 2006; Jaeger & Ladner, 2006; Sohrabi & al, 2006; Ardagna & Pernici, 2007; Cao & al, 2007; Jaeger & Muhl, 2007; Yu & al,

2007; Ai & Tang, 2008; Canfora & al, 2008; Ming & Zhen-wu, 2007; Parejo & al 2008; Wada & al, 2008; Ye & Mounla, 2008; Alrifai & Risse, 2009; Fan & al, 2009; Gao & al, 2009; Huang & al, 2009; Liu & al, 2009; Qi & al, 2010; Tang & Ai, 2010; Wang & al, 2010; Ai, 2011; Kattepur & al, 2011; Klein &, 2011; Luo & al, 2011; Pop & al, 2011; Wada & al, 2012; Sun & Zhao , 2012.

Treba napomenuti da pristupi za selekciju servisa za kompoziciju koji se zasnivaju na lokalnoj selekciji, odnosno izboru najboljeg kandidata za svaku od komponenti kompozicije pojedinačno, nisu razmatrani budući da takav pristup generalno gledano ne mora nužno rezultovati optimalnim rešenjem sa globalnog stanovišta tj. sa stanovišta QoS celokupne kompozicije, a pre svega ako je neophodno da rezultujuća kompozicija zadovolji i postavljena globalna ograničenja.

Podržani QoS atributi

Pristupi koji se zasnivaju samo na određenom manjem podskupu numeričkih opšte prihvaćenih mera:

Cardoso, 2002; Claro & al, 2005; Yu & Lin, Jul, 2005; Yu & Lin, Decembar, 2005; Berbner & al, 2006; Tong & Zhang, 2006; Yu & al, 2007; Godse & al, 2008; Wada & al, 2008; Gao & al, 2009; Liu & al, 2009; Chen & al, 2010; Qi & al, 2010; Tang & Ai, 2010; Wang & al, 2010; Ai, 2011; Kattepur & al, 2011; Wada & al, 2012.

Pristupi koji dozvoljava mogućnost selekcije na osnovu proizvoljnog skupa kvantitativnih mera:

Zeng & al,2003; Canfora & al, 2004; Zeng & al, 2004; Ardagna & Pernici, 2005; Canfora & al, Jun 2005; Canfora & al, Jul 2005; Gronmo & Jaeger, Jun 2005; Jaeger & al, 2005; Gao & al, 2006; Jaeger, 2006; Jaeger & Ladner, 2006; Ardagna & Pernici, 2007; Cao & al, 2007; Jaeger & Muhl, 2007; Ai & Tang, 2008; Parejo & al 2008; Alrifai & Risse, 2009; Fan & al, 2009; Huang & al, 2009; Tang & Ai, 2010; Ai, 2011; Klein &, 2011; Luo & al, 2011; Sun & Zhao , 2012; Gao & Ma, 2013.

Međutim treba napomenuti da veliki broj ovih radova zapravo podrazumeva određena ograničenja u pogledu samih atributa. Najčešći zahtev koji se postavlja je da vrednosti QoS atributa moraju biti numeričke. Pristupi u kojima se vrednosti nekih, po svojoj prirodi, kvalitativnih atributa zapravo predstavljaju ili mapiraju (na primer rangiranjem) u odgovarajuće numeričke vrednosti:

Cardoso, 2002; Zeng & al, 2003; Canfora & al, 2004; Zeng & al, 2004; Ardagna & Pernici, 2005; Canfora & al, Jun 2005; Canfora & al, Jul 2005; Gronmo & Jaeger, Jun 2005; Jaeger & al, 2005; Yu & Lin, Decembar, 2005; Jaeger, 2006; Jaeger & Ladner, 2006; Tong & Zhang, 2006; Ardagna & Pernici, 2007; Jaeger & Muhl, 2007, Yu & al, 2007; Ai & Tang, 2008; Parejo & al 2008; Alrifai & Risse, 2009; Liu & al, 2009; Tang & Ai, 2010; Qi & al, 2010; Ai, 2011; Luo & al, 2011; Sun & Zhao , 2012; Gao & Ma, 2013.

Sa druge strane postoje pristupi koji omogućavaju da se vrednosti QoS atributa predstave pomoću fazi skupova:

Chen & al, 2003; Agarwal & Lamparter, 2005; Claro & al, 2005; Lin & al, 2005; Sora & al, 2009; Xiong & Fan, 2007; Almulla & al, 2011.

Konačno može se zaključiti da samo mali broj pristupa omogućava selekciju na osnovu bilo kog skupa QoS mera, kako kvantitativnih tako i kvalitativnih, uključujući i domensko-specifične QoS mere, pri čemu treba istaći da većina ovih pristupa razmatra problem selekcije pojedinačnih servisa:

Chen & al, 2003; Aggarwal & al, 2004; Agarwal & Lamparter, 2005; Lin & al, 2005; Seo & al, 2005; Canfora & al, 2006; Sohrabi & al, 2006; Zhang & al, 2006; Canfora & al, 2008; Herssens & al, Januar, 2008; Sora & al, 2009; Tran & al, 2009; Karim & al, 2011; Garg & al, 2013.

Način izražavanja preferenci korisnika

Kao što je istaknuto u prethodnom poglavlju, jedan od ključnih problema prilikom selekcije servisa na osnovu nefunkcionalnih karakteristika je heterogenost QoS mera. Imajući u vidu prirodu samih QoS mera kao i moguću suprotstavljenost kriterijuma očigledno je da značaj relevantnih nefunkcionalnih karakteristika može varirati u zavisnosti od potreba korisnika. Naime, budući da je u opštem

slučaju gotovo nemoguće istovremeno optimizovati sve odabrane karakteristike, potrebno je omogućiti donosiocu odluke da na neki način iskaže preference u pogledu postavljenih zahteva kao i kompromise koje je spreman da prihvati.

Kod najvećeg broja pristupa autori polaze od toga da su preference korisnika eksplicitno uključene u model formiranjem težinske sume kriterijuma, te se može zaključiti da se kod takvih pristupa kompromis zapravo postiže adekvatnim izborom težina, čime se ukazuje na relativni značaj atributa. Izražavanje preferenci korisnika putem težina predlaže se u:

Zeng & al,2003; Aggarwal & al, 2004; Canfora & al, 2004; Zeng & al,2004; Ardagna & Pernici, 2005; Canfora & al, Jun 2005; Canfora & al, Jul 2005; Gronmo & Jaeger, Jun 2005; Jaeger & al, 2005; Yu & Lin, Jul, 2005; Yu & Lin, Decembar, 2005; Berbner & al, 2006; Canfora & al, 2006; Gao & al, 2006; Jaeger, 2006; Jaeger & Ladner, 2006; Ardagna & Pernici, 2007; Cao & al, 2007; Yu & al, 2007; Ai & Tang, 2008; Canfora & al, 2008; Godse & al, 2008; Parejo & al 2008; Ye & Mounla, 2008; Alrifai & Risse, 2009; Gao & al, 2009; Huang & al, 2009; Liu & al, 2009; Tran & al, 2009; Qi & al, 2010; Tang & Ai, 2010; Wang & al, 2010; Ai, 2011; Kattepur & al, 2011; Klein &, 2011; Luo & al, 2011; Sun & Zhao , 2012; Gao & Ma, 2013; Garg & al, 2013.

Treba napomenuti i da je u nekim od pristupa omogućeno predstavljanje težina pomoću fazi brojeva:

Chen & al, 2003; Zhang & al, 2006; Wang & al, 2006; Xiong & Fan, 2007; Herssens & al, 2008.

Međutim, postoje i nešto drugačiji pristupi. Tako na primer u (*Jaeger & al, 2005; Jaeger, 2006* - „*Bottom-Up Approximation*“, *Jaeger & Muhl, 2007*) autori predviđaju mogućnost eksplicitnog definisanja kompromisa i navode primer u kome se kompromis između *vremena izvršavanja* i *utroška resursa* definiše na taj način što se procenat koji se dodaje optimalnom vremenu izvršavanja koristi za izračunavanje procenta koji se dodaje optimalnom utrošku resursa (pri čemu zbir ova dva procenta mora biti jednak 100) te što je kraće vreme izvršavanja veći će biti utrošak resursa, i obrnuto. Ipak treba napomenuti da autori navode da

mogućnost definisanja kompromisa nije od značaja prilikom formulisanja modela problema, već se dati kompromisi uzimaju u obzir prilikom rešavanja problema.

Model problema koji autori predlažu u (*Jaeger & Muhl, 2007*) podrazumeva formiranje funkcije pogodnosti koja predstavlja količnik čiji su brojilac i imenilac proizvodi QoS atributa koji se maksimiziraju odnosno minimiziraju, respektivno, pri čemu se samim QoS atributima ne dodeljuju težine već se smatra da su one neutralne. Međutim prilikom rešavanja se za svako od mogućih rešenja primenjuje i SAW tehnika kako bi se izračunala normalizovana QoS vrednost celokupne kompozicije agregacijom vrednosti izabranih QoS atributa. U (*Canfora & al, 2004; Canfora & al, Jun 2005; Canfora & al, Jul 2005*) autori takođe formiraju funkciju pogodnosti kao količnik težinskih suma atributa koji se maksimiziraju i minimiziraju. O ovim pristupima će biti više reči u Poglavlju 3.6.1.

Sa druge strane postoje i pristupi koji razmatraju primenu fazi logike za definisanje preferenci korisnika i kompromisa na koje je on spreman da pristane. Preference korisnika se modeluju pomoću fazi IF-THEN pravila kojima se izražava stepen prihvatljivosti za različite kombinacije atributa u (*Agarwal & Lamparter, 2005*). U pristupu predloženom u (*Lin & al, 2005*) kompromisi se predstavljaju pomoću fazi izraza, pri čemu se za svaku agregiranu QoS vrednost definiše nekoliko fazi skupova koji ukazuju na stepen prihvatljivosti datog atributa. Kombinacija rezonovanja zasnovanog na logici prvog reda i hijerarhijske fazi logičke evaluacije predlaže se u (*Zhang & al, 2006*). Definiše se stepen zadovoljenja (*eng. Degree of Sevice Satisfaction – DSS*) sa tačke gledišta korisnika i zatim se formira hijerarhijski DSS model u kome se automatski (na osnovu opisa servisa) generišu fazi skupovi atributa (više skupova). Preferenca se određuje defazifikacijom centra gravitacije za ocenu korisnika koja je nezavisna promenljiva. U (*Herssens & al, Januar 2008*) fazi pravila se automatski generišu na osnovu preferenci korisnika i na osnovu njih se, u procesu fazi zaključivanja, rangiraju servisi. Premisama fazi pravila specificiraju se moguće kombinacije vrednosti atributa, dok odgovarajući zaključak određuje stepen prihvatljivosti datog servisa. Pristup predložen (*Herssens & al, Jun, 2008*) omogućava da se definišu pravila u pogledu prioriteta kriterijuma. U (*Sora & al, 2009*) se na

osnovu predložene domenske ontologije korisnik definiše svoje zahteva na osnovu kojih se formiraju odgovarajući fazi skupovi. Fazi pravila se automatski generišu na osnovu zahteva korisnika, pri čemu se premisama specificiraju moguće kombinacije vrednosti atributa dok se posledicom definiše stepen prihvatljivosti datog servisa. Rezultujući fazi skupovi se defazifikuju i zatim se vrši rangiranje. Pristup predložen u (*Almulla & al, 2011*) podrazumeva da se QoS vrednosti najpre predstave pomoću fazi skupova i formira se odgovarajuća matrice na osnovu koje se dobija matrica fazifikovanih vrednosti (odnosno odgovarajuće funkcije pripadnosti). Na osnovu date matrice se zatim generišu statističke raspodele i računaju relativni odnosi između atributa kao fazi korelacione distance (*eng. Fuzzy Correlation Weighting*) kako bi se za svaki atribut odredila odgovarajuća težina.

Takođe postoje i pristupi koji predlažu nešto drugačije načine za specifikaciju složenih zahteva korisnika u pogledu nefunkcionalnih karakteristika. U (*Agarwal & al, 2009*) se predlaže formalni jezika za definisanje politika (*eng. policy*) koji omogućava da se definišu kompleksnije preference u pogledu kako funkcionalnih tako i nefunkcionalnih karakteristika web servisa, mada se u ovom radu preference u pogledu nefunkcionalnih karakteristika zapravo odnose na složena ograničenja koja određeni servis treba da zadovolji. Za specifikaciju preferenci korisnika je u (*Sohrabi & al, 2006*) definisan veoma bogat jezik zasnovan na Golog programskom jeziku i logici prvog reda (*eng. first-order logic – FOL*) koji pri tome obezbeđuje i potpuni poredak preferenci. U (*Tran & al, 2009*) se predlaže QoS ontologija koja omogućava da se specificiraju različiti mogući odnosi između QoS mera kao što su korelacija, kompozicija, kontradikcija, stepen uticaja itd.

Iako u svojoj doktorskoj disertaciji (*Ai, 2011*) autor navodi da relevantni kriterijumi mogu biti konfliktni on dalje podrazumeva da postoji definisana strategija za razrešavanje ovih konflikata mada je ne ilustruje.

Konačno treba napomenuti i da postoje pristupi koji ne predviđaju prevođenje originalnog višekriterijumskog problema u jednokriterijumski problem:

Claro & al, 2005; Wada & al, 2008; Fan & al, 2009; Wada & al, 2012.

Normalizacija

Pristupi koje podrazumevaju formiranje jedinstvene funkcije cilja zahtevaju i da se QoS vrednosti prethodno normalizuju, budući da mogu biti izražene pomoću različitih neuporedivih jedinica mere ili čak i istih ali sa različitim rasponima vrednosti. Pri tome je u literaturi prisutno nekoliko pristupa za normalizaciju QoS vrednosti.

Pri tome, način na koji će se vršiti normalizacija zavisi pre svega od toga da li je reč i globalnoj ili lokalnoj selekciji (tj. selekcije servisa za pojedinačnu komponentu kompozicije). Naime, u slučaju lokalne selekcije normalizacija vrednosti *k*-tog QoS atributa za pojedinačni servis s_{ij} vrši se u odnosu na minimalnu i maksimalnu moguću vrednost tog atributa koja se može obezbediti za datu komponentu c_i prema sledećim formulama (u zavisnosti od toga da li se za odgovarajući QoS atribut preferira minimalna ili maksimalna vrednost):

$$q^k_{norm_{ij}} = \frac{Q^k_{\max_i} - q^k_{ij}}{Q^k_{\max_i} - Q^k_{\min_i}}$$

odnosno

$$q^k_{norm_{ij}} = \frac{q^k_{ij} - Q^k_{\min_i}}{Q^k_{\max_i} - Q^k_{\min_i}}$$

pri čemu je:

$$Q^k_{\min_i} = \min_{\forall s_{ij} \in C_i} q^k_{ij}$$

$$Q^k_{\max_i} = \max_{\forall s_{ij} \in C_i} q^k_{ij}$$

Ovakav pristup predlaže se u (*Zeng & al, 2004 za lokalnu selekciju; ; Cao & al, 2007; Herssens & al, 2008; Alrifai & Risse, 2009 za lokalnu selekciju; Qi & al, 2010; Luo & al, 2011; Sun & Zhao , 2012*).

Međutim, ukoliko je reč o selekciji servisa za kompoziciju, kao što se navodi u (*Alrifai & Risse, 2009*), ovakav postupak ne mora biti adekvatan. Naime, kako autori navode, budući da se prethodni postupak zasniva na poređenju rastojanja između QoS vrednosti određenog servisa i lokalne maksimalne/minimalne QoS vrednosti (za datu komponentu kompozicije) sa rastojanjem između lokalne maksimalne i minimalne vrednosti za datu komponentu, ovakav pristup normalizaciji može voditi ka lokalnim umesto globalnim optimumima pod uticajem lokalnih svojstava samih komponenti. Stoga

oni predlažu pristup u kome se agregirane QoS vrednosti (q^k_{agg}) porede sa minimalnim i maksimalnim mogućim agregiranim vrednostima kako bi se osiguralo da evaluacija kandidata bude globalno validna. Minimalne i maksimalne agregirane vrednosti se dobijaju agregiranjem pojedinačnih minimalnih odnosno maksimalnih QoS vrednosti za svaku od komponenti u kompoziciji pomoću odgovarajuće formule. Na primer, u slučaju aditivne agregacije, normalizacija bi se vršila prema sledećoj formuli:

$$q^k_{norm_{agg}} = \frac{Q^k_{\max} - q^k_{agg}}{Q^k_{\max} - Q^k_{\min}}$$

odnosno

$$q^k_{norm_{agg}} = \frac{q^k_{agg} - Q^k_{\min}}{Q^k_{\max} - Q^k_{\min}}$$

pri čemu je:

$$Q^k_{\min} = \sum_{i=1}^{m} Q^k_{\min_i}$$

$$Q^k_{\max} = \sum_{i=1}^{m} Q^k_{\max_i}$$

Međutim treba napomenuti da autori ne navodi na koji način bi se vršila normalizacija QoS atributa koji se agregiraju pomoću funkcija koje nisu aditivne. Isti postupak predlaže se u (*Zeng & al, 2003; Zeng & al, 2004; Ardagna & Pernici, 2005; Canfora & al, 2006; Ardagna & Pernici, 2007; Ai & Tang, 2008; Canfora & al, 2008; Ye & Mounla, 2008; Gao & al, 2009; Liu & al, 2009; Qi & al, 2010; Tang & Ai, 2010; Wang & al, 2010; Ai, 2011; Kattepur & al, 2011; Klein &, 2011; Sun & Zhao , 2012*).

U (*Jaeger & al, 2005; Gronmo & Jaeger, Jun 2005*) autori za normalizaciju takođe primenjuju ovu formulu s tim da se prilikom određivanje minimalnih i maksimalnih vrednosti razmatra samo tekući skup kandidata (odnosno skup kandidata koji je relevantan u datom koraku predloženog algoritma za selekciju čiji su detalji dati u sledećem poglavlju), dok se u (*Jaeger, 2006; Jaeger & Ladner, 2006;*) navode iste formule ali nije precizirano koji se kandidati razmatraju prilikom evaluacije minimalne i maksimalne vrednosti.

U (*Yu & Lin, Jul, 2005; Yu & Lin, Decembar, 2005; Yu & al, 2007*) se za normalizaciju primenjuju standardni skor (z-skor):

$$q^k_{norm_{ij}} = \frac{q^k_{ij} - \mu^k}{\sigma^k}$$

odnosno

$$q^k_{norm_{ij}} = 1 - \frac{q^k_{ij} - \mu^k}{\sigma^k}$$

Pored navedenih pristupa u (*Tong & Zhang, 2006; Herssens & al, 2008*) se navode i drugi pristupi za normalizaciju QoS vrednosti koji zavise od same prirode (tj. skale) datog atributa ili metode koja se primenjuje za rešavanje problema.

Konačno, treba napomenuti i da je uobičajeno je da se težine takođe normalizuju kako bi njihov zbir bio jednak jedinici čime se zapravo ukazuje na relativni značaj pojedinačnih atributa.

Način agregacije QoS atributa

Kada se razmatra selekcija servisa za određenu kompoziciju izbor bi trebalo zasnivati na nefunkcionalnim karakteristikama celokupne kompozicije. Određivanje QoS za kompoziciju postavlja sledeći izazov: potrebno je na neki način agregirati QoS vrednosti pojedinačnih servisa u kompoziciji, pri čemu način agregacije (detaljno opisan u Poglavlju 2.5.3.) zavisi od prirode samih QoS atributa. Generalno gledano određivanje QoS za kompoziciju u opštem slučaju može rezultovati nelinearnim funkcijama, što zavisi od načina na koji se pojedinačni QoS atributi agregiraju. Međutim, zbog specifičnosti metoda koje se koriste za rešavanje postavljenog problema mnogi pristupi zahtevaju linearnu funkciju cilja i/ili ograničenja.

Zeng & al,2003; Zeng & al,2004; Ardagna & Pernici, 2005; Claro & al, 2005; Jaeger & al, 2005; Yu & Lin, Jul, 2005; Yu & Lin, Decembar, 2005; Berbner & al, 2006; Gao & al, 2006; Ardagna & Pernici, 2007; Cao & al, 2007; Yu & al, 2007; Alrifai & Risse, 2009; Liu & al, 2009; Sun & Zhao , 2012.

Stoga se, u navedenim pristupima autori zapravo vezuju za određeni skup QoS mera koje se agregiraju pomoću sume ili proizvoda dok se ostali načini agregacije

uglavnom ne razmatraju. Pri tome većina ovih autora predviđa da se multiplikativne funkcije za agregiranje svedu logaritmovanjem na linearne funkcije nije ilustrovan način na koji bi se tretirali QoS atributi koji se agregiraju pomoću nelinearnih funkcija kao što su min, max funkcija ili bilo koja druga korisnički definisana funkcija.

Dakle može se zaključiti da samo mali broj pristupa omogućava selekciju na osnovu bilo kog skupa QoS mera uključujući i domensko-specifične QoS mere (koje se agregiraju pomoću korisnički definisanih funkcija).

Struktura kompozicije

Samo sekvencijalno izvršavanje:

Berbner & al, 2006; Alrifai & Risse, 2009; Claro & al, 2005; Yu & Lin, Decembar, 2005; Gao & al, 2009; Liu & al, 2009; Qi & al, 2010; Luo & al, 2011.

Sekvenca i paralelno izvršavanje:

Wada & al, 2008; Wada & al, 2012.

Složenije strukture (Sekvenca, iteracija, ciklično i paralelno izvršavanje):

Canfora & al, 2004; Agarwal & Lamparter, 2005; Canfora & al, Jun 2005; Canfora & al, Jul 2005; Yu & Lin, Jul, 2005; Canfora & al, 2006; Yu & al, 2007; Canfora & al, 2008; Parejo & al 2008; Ai, 2011; Klein &, 2011; Sun & Zhao , 2012.

Grafovi (DAG):

Zeng & al, 2003; Zeng & al, 2004; Ardagna & Pernici, 2005; Lin & al, 2005; Gao & al, 2006; Ardagna & Pernici, 2007; Ardagna & Pernici, 2007.

Uzori za kompoziciju:

Gronmo & Jaeger, Jun 2005; Jaeger & al, 2005; Jaeger, 2006; Jaeger & Ladner, 2006; Tang & Ai, 2010.

Izabrani model

Kao što je prethodno istaknuto većina autora predlaže svođenje problema selekcija servisa za kompoziciju, koji po svojoj prirodi predstavlja višekriterijumski optimizacioni problem, na različite jednostavnije modele koji su najčešće u tesnoj vezi sa izabranim načinom rešavanja samog problema. Tako se na primer problem selekcije servisa za kompoziciju modeluje kao:

MMKP (Multi-dimensional Multiple-choice 0-1 Knapsack Problem):

Ardagna & Pernici, 2005; Jaeger & al, 2005; Yu & Lin, Jul, 2005; Yu & Lin, Decembar, 2005; Ardagna & Pernici, 2007; Cao & al, 2007; Liu & al, 2009; Luo & al, 2011.

MCOP (Multi-Constrained Optimal Path problem):

Yu & Lin, Decembar, 2005; Yu & al, 2007.

RCPSP (Resource Constrained Project Scheduling Problem):

Jaeger & al, 2005; Yu & Lin, Jul, 2005; Jaeger, 2006.

Drugi modeli zasnovani na grafovima:

Gao & al, 2006; Yu & Lin, Jul, 2005.

ILP (Integer Linear Programming Problem)

Zeng & al,2003; Zeng & al, 2004; Berbner & al, 2006; Klein &, 2011.

Model zasnovan na formiranju funkcije pogodnosti (*eng. fitness*):

Canfora & al, 2004; Canfora & al, Jun 2005; Canfora & al, Jul 2005; Canfora & al, 2006; Jaeger & Muhl, 2007; Canfora & al, 2008; Parejo & al 2008; Tang & Ai, 2010; Wang & al, 2010; Ai, 2011.

Specifični modeli:

Gronmo & Jaeger, Jun 2005; Lin & al, 2005; Jaeger, 2006; Alrifai & Risse, 2009; Qi & al, 2010; Sun & Zhao , 2012.

Treba napomenuti da samo mali broj pristupa ne predviđa prevođenje originalnog višekriterijumskog problema u jednokriterijumski problem:

Claro & al, 2005; Wada & al, 2008; Fan & al, 2009; Wada & al, 2012.

Ograničenja

Na kraju, prilikom selekcije servisa za kompoziciju često se nameću i određena ograničenja u pogledu minimalnih QoS karakteristika koje kompozicija mora da zadovolji. Međutim treba napomenuti i da neki od pristupa podrazumevaju postojanje samo jednog QoS ograničenja ili čak potpuno odsustvo ograničenja što je zapravo suviše restriktivno za većinu realnih problema:

Pristupi koji predviđaju postojanje samo jednog ograničenja:

Claro & al, 2005; Jaeger & al, 2005 („Bottom-Up Approximation“, „Constraint Optimized Selection“); Yu & Lin, Jul, 2005; Jaeger, 2006 („Bottom-Up Approximation“, „Constraint Optimized Selection“); Jaeger & Muhl, 2007 (na osnovu izloženog primera); Canfora & al, 2006.

Pristupi koji podrazumevaju odsustvo QoS ograničenja, te se problem zapravo svodi na problem lokalne selekcije odnosno za svaku od komponenti kompoziciji se rešava problem selekcije pojedinačnih servisa:

Agarwal & Lamparter, 2005; Jaeger & al, 2005 („Greedy Selection“, „Pattern-wise Selection“); Gao & al, 2006; Jaeger, 2006 („Greedy Selection“, „Pattern-wise Selection“); Huang & al, 2009; Tang & Ai, 2010.

Pristupi u kojima se navodi da mogu postojati ograničenja ali zapravo nije ilustrovan način na koji bi se ona tretirala:

Gronmo & Jaeger, Jun 2005; Gronmo & Jaeger, Decembar 2005;

Takođe treba napomenuti i da postoje pristupi u kojima se ograničenja zapravo integrišu u funkciju cilja uvođenjem određenog kaznenog faktora:

Canfora & al, Jun 2005; Canfora & al, 2006; Canfora & al, 2008; Jaeger & Muhl, 2007; Parejo & al 2008; Ai, 2011.

U (*Alrifai & Risse, 2009*) se podrazumeva da će se postavljenja globalna ograničenja dekomponovati kako bi se formirala lokalna ograničenja, budući da autori predlažu pristup u kome će se vršiti distribuirana lokalna selekcija koja će prilikom evaluacije uzeti u obzir formirana lokalna ograničenja.

Iz izloženog se može zaključiti da mnogi od pristupa često podrazumevaju pojednostavljenja datog problema najpre u pogledu prirode samih QoS atributa koji se mogu koristiti za selekciju a zatim i u smislu odsustva ograničenja, omogućavanja samo serijske veze komponenti u kompoziciji, zahteva za linearnost funkcija za agregaciju itd.

Konačno treba istaći da kako se u velikom broju radova (*Ardagna & Pernici, 2005; Jaeger & al, 2005; Yu & Lin, Jul, 2005; Yu & Lin, Decembar, 2005; Jaeger, 2006; Ardagna & Pernici, 2007; Cao & al, 2007; Yu & al, 2007; Alrifai & Risse, 2009; Liu & al, 2009; Strunk, 2010; ; Luo & al, 2011; Shi & Chen, 2011; Sun & Zhao , 2012*) navodi da se problem selekcije servisa za kompoziciju može posmatrati kao specijalna vrsta problema ranca, i to MMKP (*Multi-dimensional Multiple-choice 0-1 Knapsack Problem*), ovde će ukratko biti izložen i takav pristup i pokazaće su Poglavlju 5.2 da se neki od pristupa koji se predlažu za rešavanje navedenog problema mogu uspešno primeniti i na ovakav model.

3.3.1. Multi-dimensional Multiple-choice 0-1 Knapsack (MMKP) Model

Problem ranca (*eng. knapsack problem*) predstavlja problem kod koga je potrebno iz datog skupa artikala izabrati takav podskup koji obezbeđuje da „vrednost“ artikala u rancu bude najveća moguća, a da pri tome budu zadovoljena ograničenja u pogledu raspoloživog kapaciteta ranca. Problem selekcije servisa predstavlja MMKP problem budući da je skup artikala (u ovom slučaju servisa) podeljen u nekoliko klasa koje odgovaraju komponentama kompozicije (*multiple-choice),* a iz kojih je moguće izabrati samo po jedan servis (*0-1*) kao i da se prilikom selekcije može istovremeno razmatrati nekoliko ograničenja (*multi-dimensional).*

Da bi se problem selekcije servisa formulisao kao MMKP, najpre je potrebno da donosilac odluke svakom od relevantnih kriterijuma dodeliti težinu w_k. ($w_k \in \mathbb{R}, w_k > 0, k = 1,\ldots,n.$) kojom zapravo izražava koliko je dati kriterijum za njega značajan. Pri tome je uobičajeno i da se težine normalizuju (tj. da je $\sum_{k=1}^{n} w_k = 1$), čime se zapravo ukazuje na relativni značaj datih kriterijuma.

Uzimajući u obzir smer optimizacije relevantnih kriterijuma, opšti MMKP model problema selekcije servisa se može formulisati na sledeći način:

$$\max \quad \sum_{k=1}^{n_1} w_k q^k(x) \; - \sum_{k=n_1+1}^{n} w_k q^k(x),$$

pri ograničenjima:

$$\sum_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^l x_{ij} \geq Q^l, \quad l = 1,..., r,$$

$$\sum_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^l x_{ij} \leq Q^l, \quad l = r_1,..., r,$$

$$\sum_{j=1}^{p_i} x_{ij} = 1, \qquad i = 1,..., m,$$

$$x_{ij} \in \{0,1\}, \qquad j = 1,..., p_i;\ i = 1,..., m.$$

Međutim treba imati u vidu da formiranje ovakvog modela zapravo ipak podrazumeva i usvajanje niza pretpostavki u pogledu prirode QoS atributa na osnovu kojih se može vršiti selekcija. Naime, da bi se mogla formirati težinska suma, vrednosti samih QoS

atributa moraju biti pre svega numeričke. U skladu sa tim su, za formiranje MMKP modela selekcije servisa za kompoziciju koji će se u ovom radu rešavati, iz izabranog reprezentativnog skupa (Tabela 2.) izdvojeni samo numerički QoS atributi: *Raspoloživost*, *Vreme Odziva*, *Pouzdanost* i *Kašnjenje*. Način na koji se ovi QoS atributi agregiraju kao i smer njihove optimizacije dat je u sledećoj tabeli Tabela 3):

Tabela 3. Način agregacije i smer optimizacije izabranih QoS atributa

QoS atribut	*Oznaka*	*Način agregacije*	*Smer optimizacije*
Raspoloživost	A	$q^1(x) = \prod_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^1 x_{ij}$	max
Vreme odziva	RT	$q^2(x) = \sum_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^2 x_{ij}$	min
Pouzdanost	R	$q^3(x) = \prod_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^3 x_{ij}$	max
Kašnjenje	L	$q^4(x) = \sum_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^4 x_{ij}$	min

Pored toga, imajući u vidu da se u opštem slučaju neki QoS atributi zapravo agregiraju pomoću nelinearnih funkcija, kao što je i slučaj kada je reč o atributima *Raspoloživost* i *Pouzdanost*, potrebno je takve QoS kriterijume prevesti u linearne (npr. logaritmovanjem u slučaju proizvoda pri čemu treba istaći da se drugi oblici agregacije u ovakvim pristupima najčešće ne razmatraju):

$$\log\left(\prod_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^k x_{ij}\right) = \sum_{i=1}^{m} \sum_{j=1}^{p_i} \left(\log\left(q_{ij}^k\right)\right) x_{ij} \quad k = 1,3,$$

Navedena relacija je tačna budući da, za bilo koje dopustivo rešenje, tačno jedno x_{ij} ima vrednost *1* dok sve ostale promenljive imaju vrednost *0*, za svako $i=1,\ldots,m$. Međutim treba napomenuti da, kada je reč o atributima čije su vrednosti izražene u procentima (te se kreću između 0 i 1), primena logaritamske funkcije rezultuje negativnim vrednostima te će se dobijene vrednosti množiti sa -1 čime se zapravo menja i smer optimizacije datih kriterijuma.

Sledeći uslov koji se nameće prilikom formiranja težinske sume je uporedivost veličina koje u njoj učestvuju. Međutim, sami QoS atributi su najčešće izraženi pomoću različitih jedinica mere na primer, *Raspoloživost* i *Pouzdanost* se mere u procentima dok se *Vreme Odziva* i *Kašnjenje* mere u milisekundama, pri čemu je u slučaju poslednja dva atributa, iako se radi o istoj jedinici mere (milisekund), zapravo u pitanju različit raspon samih vrednosti. Uporedivost atributa postiže se normalizacijom kako bi se vrednosti atributa svele na opseg [0,1]. U ovom radu će se normalizacija vrednosti *k*-tog QoS atributa za pojedinačni servis s_{ij} vršiti u odnosu na minimalnu i maksimalnu moguću vrednost tog atributa prema sledećim formulama (u zavisnosti od toga da li se za odgovarajući QoS atribut preferira što veća ili što manja vrednost):

$$q^k_{norm_{ij}} = \frac{q^k_{ij} - Q^k_{\min}}{Q^k_{\max} - Q^k_{\min}} \quad \text{odnosno} \quad q^k_{norm_{ij}} = \frac{Q^k_{\max} - q^k_{ij}}{Q^k_{\max} - Q^k_{\min}}$$

pri čemu je, da bi se osiguralo da evaluacija servisa bude globalno validna, uzeto da je:

$$Q^k_{\min} = \min_{\forall s_{ij}} q^k_{ij} \quad \text{i} \quad Q^k_{\max} = \max_{\forall s_{ij}} q^k_{ij}.$$

Treba napomenuti da se u navedenim formulama, u slučaju kada je reč o QoS atributima *Raspoloživost* i *Pouzdanost* umesto vrednosti q^k_{ij} zapravo uvrštava vrednost $\left(-\log(q^k_{ij})\right)$ koju je, zbog promene smera optimizacije, potrebno minimizirati.

Uzimajući u obzir navedene pretpostavke moguće je odrediti korisnost (*eng. utility*) za svaki od servisa:

$$f_{ij} = \sum_{k=1}^{n} w_k q^k_{norm_{ij}}$$

Problem se time svodi na maksimizaciju ukupne korisnosti kompozicije:

$$max \quad \sum_{i=1}^{m} \sum_{j=1}^{p_i} f_{ij} x_{ij}$$

pri definisani ograničenjima.

Konačno, treba napomenuti da je prilikom implementacije metoda, koje se u Poglavlju 5.2. predlažu za rešavanje navedenog modela, problem zapravo preveden u ekvivalentni problem minimizacije negativne vrednosti date funkcije korisnosti. Pri tome predložene metode ne zahtevaju da ograničenja budu linearna niti zahtevaju da vrednosti atributa u ograničenjima budu normalizovane.

3.4. Motivacija za uvođenje novih pristupa

Budući da se selekcija može zasnivati na QoS merama koje mogu biti kvantitativne i kvalitativne, subjektivne i objektivne i izražene pomoću različitih neuporedivih jedinica mere, kao i da je za neke QoS atribute maksimalna vrednost poželjna dok je za druge poželjna minimalna vrednost, očigledno je da su kod skoro svih realnih problema relevantni kriterijumi delimično ili u potpunosti kontradiktorni. Pored toga u opštem slučaju ne moraju svi kriterijumi biti jednako značajni prilikom selekcije, pri čemu i značaj samih kriterijuma može varirati od problema do problema. Stoga je potrebno omogućiti donosiocu odluke da na neki način iskaže preference u pogledu postavljenih zahteva kao i kompromise koje je spreman da prihvati. Kao što se može videti u prethodnom potpoglavlju većina postojećih pristupa selekciji web servisa (kako pojedinačnih servisa i tako i servisa za kompoziciju) predlaže formiranje težinske sume za agregaciju relevantnih kriterijuma (odnosno QoS atributa) u jedinstvenu funkciju cilja bilo prilikom modelovanja problema, bilo prilikom njegovog rešavanja.

Metoda težinske sume (*MacCrimmon, 1968*) koja je u (*Hwang & Yoon, 1981*) nazvana metoda jednostavnih aditivnih težina (*eng. Simple Additive Weighting* – SAW) je jedna od najpoznatijih i najšire korišćenih metoda višekriterijumskog odlučivanja (*Yoon & Hwang, 1995*). SAW metoda podrazumeva da donosilac odluke svakom od kriterijuma najpre dodeli numeričku vrednost – težinu kojom izražava koliko je dati kriterijum za njega značajan. Zatim se skor svake od alternativa izračunava sabiranjem pojedinačnih doprinosa svakog od relevantnih kriterijuma (dobijenih množenjem vrednosti kriterijuma odgovarajućim težinama). Naime, ako je za n_1 Qos kriterijuma poželjno ostvariti maksimalnu dok je za preostalih ($n - n_1$) poželjno ostvariti što manju moguću vrednost tada se skor neke alternative x može odrediti na sledeći način:

$$\sum_{k=1}^{n_1} w_k q^k(x) \; - \sum_{k=n_1+1}^{n} w_k q^k(x),$$

pri čemu je $w \in \mathbb{R}^n, w_k > 0, k = 1, \ldots, n.$ vektor odabranih težina.

Međutim, da bi se mogla formirati težinska suma, vrednosti samih kriterijuma moraju biti pre svega numeričke a zatim i uporedive. Uporedivost kriterijuma čije vrednosti mogu biti izražene pomoću različitih heterogenih jedinica mera (ili čak i istih ali sa različitim rasponima vrednosti) postiže se normalizacijom. Osnovni cilj

normalizacije je uspostavljanje bilo uporedivih skala vrednosti koje su izražene pomoću iste jedinice mere bilo neutralnih skala vrednosti koje nisu vezane ni za jednu jedinicu mere odnosno koje su izražene pomoću mera koje nemaju svoju dimenziju (*eng. dimensionless units*), te što je veći rang to je veća preferenca. Pri tome se kada je reč o selekciji servisa, kao što je izloženo u prethodnom poglavlju, normalizacije može vršiti na različite načine. Jedan od postupaka, koji se u literaturi najčešće predlaže, se zasniva na poređenju rastojanja između QoS vrednosti određenog servisa (ili kompozicije) i maksimalne/minimalne moguće QoS vrednosti za datu komponentu (odnosno kompoziciju) sa rastojanjem između maksimalne i minimalne moguće vrednosti za datu komponentu (odnosno kompoziciju). Međutim, ovakav pristup može dovesti do toga da se onemogućava odnosno potencira selekcija onih servisa (odnosno kompozicija) čija QoS vrednost odgovara maksimalnoj ili minimalnoj mogućoj vrednosti. Pored toga, poznato je da problem normalizacije postaje još kompleksniji kada je reč o QoS atributima koji se agregiraju putem nelinearnih funkcija. Konačno, treba napomenuti i da je uobičajeno da se težine takođe normalizuju kako bi njihov zbir bio jednak jedinici čime se zapravo ukazuje na relativni značaj pojedinačnih atributa.

Sa druge strane, određivanje samih težina nije jednostavan zadatak, kao što se navodi u (*MacCrimmon, 1968*), naročito kada je potrebno verbalne iskaze vezane za značaj kriterijuma pretvoriti u odgovarajuće numeričke vrednosti. On dalje navodi da čak i kada su vrednosti dva kriterijuma svedene na istu skalu može se desiti da se množenjem dodeljenim težinama dobije da one imaju podjednak doprinos i u situacijama kad to zapravo nije slučaj. Drugim rečima, ako su dva kriterijuma podjednako značajna dodeljivanje jednakih težina svakom od njih neće uvek obezbediti da oni ravnopravno učestvuju u selekciji. Na primer, neka postoje dve alternative od kojih je prva mnogo dominantnija od druge po jednom od kriterijuma a značajno lošija po drugom, dok je druga alternativa prosečna po oba kriterijuma. U slučaju prve alternative, usled dominantnosti prvog kriterijuma, formiranjem težinske sume se potire uticaj drugog kriterijuma što može voditi izboru pogrešne alternative. Budući da je zahtev bio da oba kriterijuma ravnopravno učestvuju u selekciji imalo bi više smisla izabrati alternativu koja je prosečna po oba kriterijuma ali čija je težinska suma manja. Još jedan problem vezan za određivanje težina pojedinačnih QoS atributa istaknut je u (*Jaeger, 2006*) gde autor navodi da ono postaje još kompleksnije kada su u pitanju

nelinearne funkcije koje se prevode u linearne budući da tada težine moraju biti takve da kompenzuju nelinearnu operaciju koja se primenjuje na QoS vrednosti.

Sa druge strane, sabiranje otežanih vrednosti kriterijuma za neku alternativu pretpostavlja da zapravo ne postoji zavisnost između samih kriterijuma, što znači da se oni mogu zasebno razmatrati. U protivnom SAW pristup može navoditi na pogrešne zaključke. Kako se navodi u (*Tzeng & Huang, 2011*) složenost MCDM problema proističe iz ne samo iz problema vezanih za određivanje težina kriterijuma već i problema vezanih za moguću uslovljenost odnosno zavisnosti između samih preferenci kao i moguću suprotstavljenosti kriterijuma. Kako autori dalje navode, pozivajući se na (*Keeney & Raiffa, 1976*), SAW metoda, pa i fazi SAW metoda, se zasnivaju na pretpostavci da ne postoji međusobna uslovljenost preferenci tj. da odnos preferenci (odnosno prihvatljiv kompromis) u pogledu bilo koja dva kriterijuma ne zavisi od vrednosti preostalih kriterijuma, ili drugačije rečeno, kako se navodi u (*Yoon & Hwang, 1995*), doprinos pojedinačnog kriterijuma sveukupnom cilju ne zavisi od doprinosa preostalih kriterijuma. Naime, kako se navodi u (*Figueira, &al 2005, Poglavlje 14*) aditivna forma podrazumeva da između kriterijuma ne postoji bilo kakva interakcija te da ispunjenje svakog od kriterijuma doprinosi ispunjenju sveukupnog cilja. Dalje se navodi da u realnosti između kriterijuma ipak često postoji određeni stepen zavisnosti odnosno interakcije te da ispunjenje jednog od kriterijuma zapravo može uticati na korisnost (tj. upotrebnu vrednost) nekog drugog kriterijuma, bilo da zahteva da i drugi kriterijum mora biti ispunjen (tzv. komplementarni kriterijumi), bilo da ispunjenje drugog kriterijuma postaje irelevantno (tzv. međusobno zamenljivi kriterijumi), što se ne može izraziti sumom.

Dakle, dodatni izazov predstavlja činjenica da su u realnosti zahtevi korisnika u pogledu relevantnih nefunkcionalnih karakteristika često dosta složeniji. Na primer korisnik može imati potrebu da definiše zahteve u kojima se lošija vrednost po nekom od kriterijuma može nadomestiti nekim drugom kriterijumom, odnosno u kojima značaj kriterijuma zavisi od ostvarenih vrednosti drugih kriterijuma. Generalno gledano, ovakvi zahtevi korisnika se mogu izraziti samo pomoću verbalnih iskaza.

Još jedan aspekt koji treba uzeti u obzir je mogućnost postojanja korelacije između različitih QoS mera. Na primer prema *Vreme Odziva* (*eng. Response Time*) servisa predstavlja vreme koje je potrebno da servis izvrši određeni zahtev, pri čemu se

ono zapravo predstavlja zbir *Kašnjenj*a (*eng. Latency*) tj. trajanja propagacije zahteva koje korisnik postavlja servisu i odgovora koje servis vraća korisniku i *Vremena Izvršavanja* (*eng. Execution Time*) tj. vremena za koje servis obavi niz aktivnosti neophodnih da se izvrši postavljeni zahtev. Pored toga, *Raspoloživost* (*eng. Availability*) koja predstavlja verovatnoću da je server na kome se nalazi servis operativan je najčešće vezana za *Pouzdanost* (*eng. Reliability*) koja predstavlja sposobnost servisa da obavlja svoju funkciju pod datim uslovima u okviru određenog intervala vremena, itd.

Uzimajući u obzir prethodna razmatranja može se zaključiti da pristupi kod kojih se mogućnost definisanja kompromisa svodi isključivo na neku strategiju za dodeljivanje težina kriterijumima zapravo ne uzimaju u obzir činjenicu da QoS kriterijumi na osnovu kojih se vrši selekcija mogu uticati jedni na druge, biti u korelaciji i/ili kontradikciji čime se dovodi u pitanje koliko je njihova primena adekvatna. Pored toga određivanje samih težina je u opštem slučaju arbitrarno i u krajnjoj liniji ne mora adekvatno odražavati korisnikove preference niti kompromise koje je spreman da prihvati. Konačno treba napomenuti i da većina postojećih pristupa zahteva linearnu funkciju cilja a da određivanje QoS za kompoziciju u opštem slučaju može rezultovati nelinearnim funkcijama (u zavisnosti od toga kako se pojedinačni QoS atributi agregiraju) te primena ovih pristupa podrazumeva svođenje nelinearnih funkcija na linearne (na primer logaritmovanjem). Sa druge strane primena SAW tehnike takođe podrazumeva i da su vrednosti samih kriterijuma numeričke i uporedive. Stoga se većina autora zapravo vezuje za određeni predefinisani skup QoS mera koje se mogu numerički izraziti i koje se agregiraju pomoću sume ili proizvoda dok se ostali načini agregacije uglavnom ne razmatraju.

Imajući u vidu navedeno jedan od ciljeva ovog rada je definisanje novih pristupa za modelovanje problema selekcije servisa na osnovu nefunkcionalnih karakteristika koji ne zahtevaju dodeljivanje težina niti normalizaciju vrednosti QoS atributa, a koji će omogućiti da se na adekvatniji način uzmu u obzir specifičnosti samih mera ali i različiti odnosi koji postoje između relevantnih nefunkcionalnih karakteristika kao i zahtevi korisnika (koji mogu biti izraženi i pomoću složenih verbalnih iskaza). U skladu sa ovim ciljem u nastavku rada će biti predložena dva nova pristupa za modelovanje problema selekcije servisa, dok će u Poglavljima 4 i 5 biti i pokazano da se oni mogu uspešno koristiti za rešavanje navedenog problema.

3.5. Nov model zasnovan na primeni konzistentne fazi logike

Iskustvo pokazuje da čovek zaključuje i donosi odluke na osnovu prethodnog iskustva, prikupljenih činjenica, okvirnih pravila, maglovitih koncepata, nedefinisanih intuicija, sumnji, verovanja, procena, pogađanja,.... U nemogućnosti da se opisani proces odlučivanja do kraja sagleda, a samim tim i pretoči u matematički model bilo koje složenosti, teorija odlučivanja se opredelila za svođenje problema odlučivanja na definisanje cilja (ili skupa ciljeva) koji se žele postići kao i kriterijuma kojima se evaluira „cena“ dostizanja cilja. Šta više, u nastojanju da se ostvari kompromis između adekvatnosti modelovanja i jednostavnosti izračunavanja najveći broj metoda opredelio se za specifikaciju cilja preko sume svih relevantnih kriterijuma, pri čemu se značaj pojedinih kriterijuma iskazuje težinom sa kojom on učestvuje u sumi. Ovim se implicitno podrazumeva, kao što je i izloženo u prethodnom potpoglavlju, da su svi faktori linerano povezani, da nisu uzajamno korelisani i da pri donošenju odluke zapravo nema kriterijuma koji su uzajamno isključivi. Nije, međutim, teško uočiti da u praksi postoji veoma mali broj problema koji ispunjavaju navedene uslove. Nesumnjivo je da u svim takvim slučajevima težinska suma ili neka njena varijacija predstavlja izuzetno grubu aproksimaciju, koja ne može dati adekvatne rezultate.

U skladu sa prethodno iznetom analizom može se zaključiti da bi za predstavljanje preferenci korisnika vezanih za nefunkcionalne karakteristike servisa bila adekvatnija primena logičkih funkcija od uobičajenog arbitrarnog dodeljivanja težina, budući da one omogućavaju uzimanje u obzir različitih odnosa koji mogu postojati između relevantnih QoS atributa kao i specifičnosti samih mera. Sa druge strane, u većini realnih slučajeva, zahtevi korisnika se jedino i mogu izraziti pomoću složenih verbalnih iskaza koje najčešće nije ni moguće realizovati dodeljivanjem težina. U tom smislu osnovni zadatak je da se obezbedi odgovarajuća funkcionalna zavisnost koja će omogućiti da se na adekvatan način iskažu ovakvi logički zahtevi.

Fazi logika se nameće kao prirodno rešenje za specifikaciju ovakvih zahteva korisnika budući da je usklađena sa kvalitativnim opisima koje ljudi koriste u svakodnevnoj komunikaciji. Naime, u želji da se, bar u nekoj meri, prevaziđe dihotomija između procesa donošenja odluka u realnom životu i optimizacije kriterijumskih funkcija Zadeh je predložio da se problemu modelovanja i odlučivanja priđe kroz fundamentalno drugačiji pristup – fazi modeliranje i odlučivanje (*Zadeh*,

1973). Polazeći od stanovišta da u osnovi procesa razmišljanja i odlučivanja ne leže brojevi povezani težinskom sumom, već niz pojmova koji se izražavaju jezički i koji su većoj ili manjoj meri međusobno povezani i relevantni za donošenje odluke, fazi modelovanje napušta koncept preslikavanja skupova brojnih vrednosti, te predlaže da se modelovanju pristupi kroz uspostavljanje logičkih veza između pojmova koji opisuju parametre koji utiču na donošenje odluke.

U literaturi postoje pristupi koji razmatraju primenu fazi logike za definisanje preferenci korisnika i kompromisa na koje je on spreman da pristane prilikom selekcije na osnovu nefunkcionalnih karakteristika mada neki od njih ne uzimaju u obzir korelaciju koja može postojati između QoS atributa kao ni situaciju u kojoj korisnik može želeti da lošiju vrednost nekog QoS atributa nadomesti drugim (tj. da definiše jedan zahtev koji sadrži alternativne opcije i/ili negaciju nekog od uslova umesto formiranja velikog broja različitih pravila) (*Chen & al, 2003; Agarwal & Lamparter, 2005; Lin & al, 2005; Tong & Zhang, 2006; Wang & al, 2006; Xiong & Fan, 2007; Sora & al, 2009; Almulla & al, 2011*). Pored toga najveći broj takvih pristupa se odnosi isključivo na problem selekcije pojedinačnih servisa za dati funkcionalni zahtev (*Tong & Zhang, 2006; Wang & al, 2006; Zhang & al, 2006; Xiong & Fan, 2007; Herssens & al, 2008; Sora & al, 2009; Almulla & al, 2011*) dok se problem selekcije servisa za kompoziciju razmatra samo u malom broju takvih radova (*Agarwal & Lamparter, 2005; Lin & al, 2005*). Treba napomenuti i da je u nekim pristupima omogućeno predstavljanje pomoću fazi brojeva težina (*Chen & al, 2003; Zhang & al, 2006; Wang & al, 2006; Xiong & Fan, 2007; Herssens & al, 2008*) ili samih QoS vrednosti (*Chen & al, 2003; Agarwal & Lamparter, 2005; Claro & al, 2005; Lin & al, 2005; Sora & al, 2009; Xiong & Fan, 2007; Almulla & al, 2011*) samih vrednosti QoS atributa.

Međutim, polazna pretpostavka ovog rada je da bi logičke funkcije kojima se predstavljaju zahtevi korisnika trebalo da zadovoljavaju sve zakone klasične logike, odnosno da budu definisane u Bulovom okviru. Budući da nijedna konvencionalna teorija fazi skupova nije u Bulovom okviru (*Radojević, 2008a*) u radu se predlaže primena konzistentne fazi logike, izložene u (*Radojević, 2008b*). Naime, uočavajući da klasična fazi logika, nasuprot izuzetno dobrim rezultatima u procesima zaključivanja koji se sreću, na primer, u teoriji upravljanja, ima izvesne nedostatke u teoriji odlučivanja Radojević (*Radojević, 2000*) je predložio dalju razradu fazi koncepta kroz

usaglašavanje sa Bulovom logikom. Namera rada je da se pokaže da se ovakav pristup može primeniti za definisanje složenih korisničkih zahteva vezanih za relevantne QoS atribute prilikom modelovanja problema selekcije web servisa uzimajući u obzir i korelaciju koja može postojati među nefunkcionalnim karakteristika kao i prirodu samih QoS mera. Na ovaj način bi se omogućilo da se selekcija vrši na osnovu bilo kog skupa QoS mera uključujući i domensko-specifične mere koje se agregiraju pomoću korisnički definisanih funkcija.

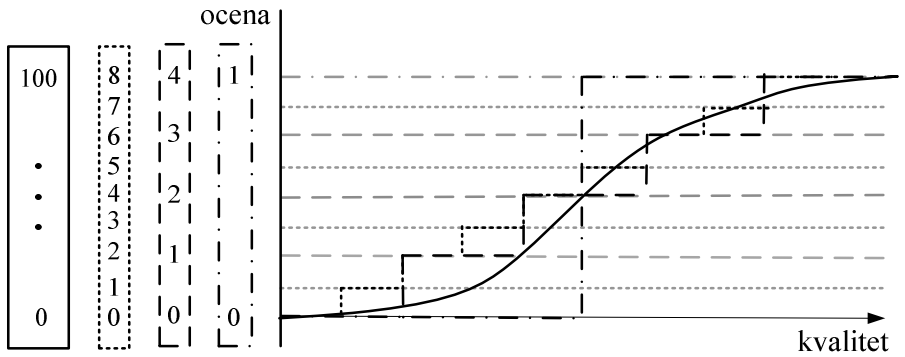
Konačno, u Poglavlju 4. će se ilustrovati da primena konzistentne fazi logike ne mora rezultovati istim izborom servisa kao primena konvencionalne fazi logike, zbog čega se prednost daje konzistentnom pristupu nad konvencionalnim, dok će se u narednom potpoglavlju objasniti i razlog ovog odstupanja.

3.5.1. Predstavljanje atributa – Fazi skup

Nema nikakve sumnje da niz QoS atributa izabranih za kriterijume, kao što su na primer bezbednost, postojanje prateće dokumentacije i slično, ne može da bude objektivno izmeren, pa je prirodno da se opisuje opisno. Međutim čak i u onim slučajevima kada je merenje moguće (vreme odziva, raspoloživost, itd.) fazi pristup predlaže drugačiju interpretaciju izmerenih vrednosti. Naime, u svakodnevnom životu suočen sa pitanjem ocene percepcije neke pojave čovek će koristiti opisne prideve tipa „brzo“ ili „sporo“, „malo“ ili „mnogo“, „prijatno“ ili „neprijatno“ uz sve nijanse koje postoje između tih suprotstavljenih pojmova. Tako će se na primer za kvalitet dokumentacije reći da je „izvrstan“, „dobar“, „prilično dobar“, „osrednji“, „loš“ ili čak „nedopustivo loš“. Međutim, statistička obrada odgovora svakog istraživanja, ili konvencionalan način formiranja kriterijumske funkcije, zahtevaju da se ove ocene izraze brojčanim vrednostima što znači da će donosioci odluke biti prinuđeni da svoj utisak iskažu brojčano na skali ponuđenih ocena. Ova skala može biti gruba i na taj način čitavu nijansu opisnih ocena osiromašiti svođenjem na nekoliko brojeva. Skala naravno može biti i daleko finija, odnosno sadržati veliki broj ponuđenih ocena čime se samo na izgled ostvaruje bolje slaganje između opisne procene i brojčane vrednosti.

Ako se pretpostavi da je pojam „dobro“ označen ocenom 15, a pojam „veoma dobro“ ocenom 20, tada je donosilac odluke prinuđen da na neki način odmeri da li je njegov osećaj da je kvalitet dokumentacije datog servisa prilično zadovoljavajući

(odnosno da se nalazi između „dobar“ i „veoma dobar“) bliži recimo oceni 17 ili 18. Nezavisno od toga koju od te dve ocene izabere on gotovo izvesno neće biti u stanju da ponudi obrazloženje svog izbora. Drugim rečima, čak i kada broj podeoka na skali ocena raste, ona i dalje ostaje izdeljena po segmentima koji se ne preklapaju, tako da se svakom izboru dodeljuje uvek samo jedan broj, kao što je to ilustrovano na Slici 11:



Slika 11. Polivalentni skupovi ocenjivanja kvaliteta

Brojčanu ocene percepcije nekog pojma moguće je zameniti *fazi skupom* (*Zadeh, 1965*) koji predstavlja generalizaciju klasičnog binarnog skupa i to tako što se pripadnost (*1*), odnosno nepripadnost (*0*) elementa skupu proširuje pojmom stepena pripadnosti koji pripada intervalu [0,1]. U tom smislu se definiše funkcija pripadnosti (*eng. membership function*) fazi skupa koja preslikava svaki elemenat univerzalnog skupa na interval realnih brojeva:

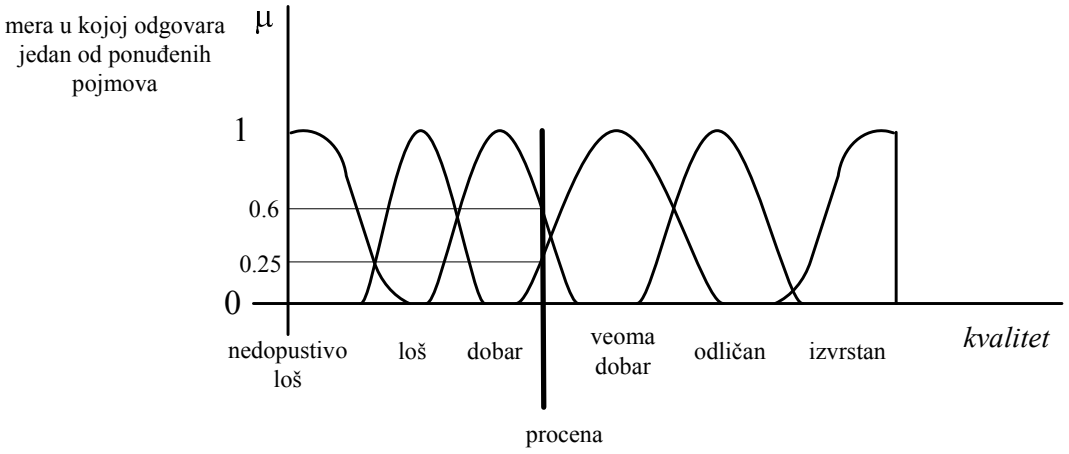
$$\mu_A(x): X \rightarrow [0,1]$$

Fazi skup *A* nad univerzalnim skupom *U* definiše se kao skup uređenih parova:

$$A = \{(x, \mu_A(x): x \in U\}$$

Kao što je već rečeno, funkcija pripadnosti predstavlja meru slaganja promenljive *x* sa pojmom *A*, odnosno svojstvima pojma *A*. Pri tome $\mu_A(x) = 1$ označava potpuno slaganje promenljive *x* sa pojmom *A*, $\mu_A(x) = 0$ iskazuje da se promenljiva *x* ni na koji način ne slaže sa pojmom *A*, dok sve ostale vrednosti funkcije pripadnosti izražavaju stepen do koga promenljiva *x* odgovara pojmu *A*.

Ukoliko bi se sada istom donosiocu odluke pokazala Slika 12 na kojoj su prikazani fazi skupovi pojma „kvalitet“ on bi sasvim sigurno umeo da pokaže da je po njegovoj oceni kvalitet u izvesnoj meri (recimo 60%) dobar i u izvesnoj meri (25%) veoma dobar.



Slika 12. Fazi skupovi koji čine okvir spoznaje pojma „kvalitet“

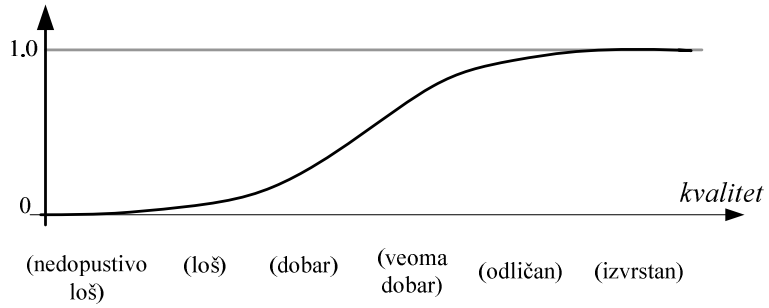
Izloženi primer se zapravo nalazi u osnovi teorije fazi skupova koja se može razumeti kao nastojanje da se „veštački i uređen svet matematike približi stvarnom i neuređenom svetu u kome živimo“ (*Kosko, 1993*). Drugim rečima, umesto da se neka pojava kvalifikuje nizom brojčanih vrednosti na utvrđenoj skali ocena, ona se opisuje nizom svojstava (lingvističkih promenljivih) koje obuhvataju sve moguće nijanse ljudske percepcije - *okvir spoznaje* (*eng. frame of cognition*). Ukoliko su fazi skupovi odabrani tako da obezbeđuju potpuno pokrivanje svih mogućih ocena tada će svaka pojedinačna percepcija naći svoje mesto unutar definisanih skupova.

Skup svih svojstava kojima se opisuje jedan pojam čini *okvir spoznaje* tog pojma. Formalno, fazi skup $A=\{A_1, A_2, ..., A_n\}$ je *okvir spoznaje* pojma X ako su ispunjeni sledeći uslovi:

- A pokriva univerzum X - svaki element pojma X pridružen je bar jednom fazi skupu sa ne-nultim stepenom pripadnosti,
- $\forall x \in X, \quad \exists \mu_{Ai}(x) > \varepsilon, \quad \varepsilon > 0,$ određuje nivo pokrivanja,
- Elementi skupa A su unimodalni skupovi (postoji oblast x koja ima jasno semantičko značenje i u kojoj samo jedan od fazi skupova A_i ima izrazito veliku funkciju pripadnosti),
- Fazi skupovi se delimično prekrivaju – (veliko prekrivanje zamagljuje sliku, malo prekrivanje vodi ka diskretnoj logici).

Okvir spoznaje zavisi od broja svojstava koja se žele koristiti za specificiranje posmatrane veličine[1].

Neki pojmovi mogu se na zadovoljavajući način opisati i pomoću okvira spoznaje koji sadrži samo jedan fazi skup. U posmatranom primeru pojam „kvalitet“ bi se mogao prikazati jednim fazi skupom (prikazanim na Slici 13) na kome su svojstva implicitno uključena preko vrednosti funkcije pripadnosti. Iako takav prikaz u izvesnoj meri smanjuje preciznost opisa, činjenica je da je samim uvođenjem fazi skupa otvorena mogućnost za formiranje fazi relacije i korišćenje fazi konjunkcije i disjunkcije.



Slika 13. Okvir spoznaje predstavljen jednim fazi skupom

3.5.2. Fazi Relacije

Uvođenje pojma fazi skupa je prirodno dovelo i do definisanja fazi relacija nad tim skupovima. U opštem slučaju fazi relacija *S* između dva univerzalna skupa *V* i *W* je fazi skup definisan u prostoru proizvoda:

$$S = V \odot W = \left\{ (v,w), \mu_S(v,w) : v \in V, w \in W \right\}$$

Ovaj skup okarakterisan je funkcijom pripadnosti:

$$\mu_S : V \odot W \to [0,1]$$

Različiti izbori operatora $\odot$ omogućavaju formiranje različitih relacija između fazi skupova.

Negacija, fazi komplement skupa

Polazeći od činjenice da u nekom fazi skupu *A* funkcija $\mu_A(v)$ izražava stepen saglasnosti uočene karakteristike sa datim pojmom, onda je izvesno da funkcija

[1] Imajući u vidu da psiholozi smatraju da je ljudsko biće u stanju da koristi 7±2 koncepata jednog pojma (*Pedrycz, 1993*), izvesno je da sa brojem svojstava ne treba preterivati.

$1-\mu_A(v)$ izražava stepen nesaglasnosti. To znači da se svakom fazi skupu *A* može pridružiti fazi skup $\neg A$ koji predstavlja njegov komplement:

$$\neg A = \{(v, \mu_{\neg A}(v) = 1 - \mu_A(v) : v \in V\}$$

Izložena definicija se može uopštiti na sledeći način. Neka $\neg a$ označava komplement funkcije pripadnosti elementa *a* jednog fazi skupa ($a \in [0,1]$) tada komplement fazi skupa mora zadovoljavati sledeće uslove:

1^0 $\neg 0 = 1, \quad \neg 1 = 0$

2^0 $\forall a,b \in [0,1]$ ako je $a \le b$ tada je $\neg a \ge \neg b$

3^0 kontinualnost

4^0 involutivnost negacije

Presek fazi skupova

Relacija preseka dva fazi skupa, koja ima značenje konjunkcije, definiše se pomoću *t-norme*, odnosno preslikavanjem:

$$t : [0,1] \times [0,1] \to [0,1]$$

koje za $a,b \in [0,1]$ zadovoljava sledeće aksiome:

1^0 $a t b = b t a$ – komutativnost

2^0 $a t (b t c) = (a t b) t c$ – asocijativnost

3^0 $a \le b \Rightarrow a t c \le b t c$ – monotonost[1]

4^0 $1 t a = a$ – postojanje elementa identiteta (1)

5^0 $a t a \ge a$ – subidempotentnost (samo u konvencionalnoj fazi logici)

Iz navedenih uslova se vidi da izložena definicija *t-norme* predstavlja svojevrsnu ekstenziju logičke „i“ operacije (preseka) binarnih skupova. Naime prva dva aksioma su direktno preuzeta iz bivalentne logike. Osobina monotonosti uključena je da bi se obezbedilo da konjunkcija ne opada ako istinitost pojma (odnosno stepen pripadnosti) raste. Konačno zahtev da broj *1* bude jedinični elemenat odgovara interpretaciji funkcije pripadnosti, odnosno činjenici da je iskaz apsolutno tačan ako funkcija pripadnosti ima vrednost *1*, odnosno

[1] U nekim logikama se uslov monotonosti funkcije zamenjuje strožijim uslovom striktne monotonosti i kontinualnosti.

apsolutno netačan ako je vrednost *0*. Naime, iz uslova monotonosti i egzistencije jediničnog elementa direktno sledi da je $0 t a = 0$.

Najčešće korišćeni oblici *t-norme* su:

1^0 $a t b = min(a,b)$ (standardni presek - Gödel-Dummett)

2^0 $a t b = a \cdot b$ (algebarski proizvod - logičko "i")

3^0 $a t b = max(0, a+b-1)$ (ograničena razlika - Lukašijevič)

Za ove norme važi:

$$max(0,a+b-1) \le a \cdot b \le min(a,b)$$

Izbor određene *t-norme* zavisi od konkretnog iskaza koji se želi modelovati. Tako, na primer, iskaz „mala cena i dobre performanse“ kojim se, sa gledišta uslova poslovanja, izražavaju suprotni zahtevi podrazumeva korišćenje proizvoda, odnosno logičke „i“ operacije. Međutim, budući da u iskazu „dobre performanse i dobra dokumentacija“ nema korelacije za njegovo prikazivanje pogodnije je koristiti operator „min“.

Sledeći definiciju *t-norme*, presek fazi skupova *A* i *B* definisanih nad univerzalnim skupovima *V* i *W* specificira se na sledeći način:

$$A \cap B = \{(v,w), \mu_{A \cap B}(v,w) = \mu_A(v) t \mu_B(w) : v \in V, w \in W\}$$

Unija fazi skupova

Relacija unije dva fazi skupa koja ima značenje disjunkcije definiše se pomoću *s-norme* (*t-konorme*), odnosno preslikavanjem:

$$s : [0,1] \times [0,1] \to [0,1]$$

koje za $a,b \in [0,1]$ zadovoljava sledeće aksiome:

[1] U nekim logikama se uslov monotonosti funkcije zamenjuje strožijim uslovom striktne monotonosti i kontinualnosti.

$1^0\ a \text{ s } b = b \text{ s } a$ – komutativnost

$2^0\ a \text{ s } (b \text{ s } c) = (a \text{ s } b) \text{ s } c$ – asocijativnost

$3^0\ a \le b \Rightarrow a \text{ s } c \le b \text{ s } c$ – monotonost[1]

$4^0\ 0 \text{ s } a = a$ – postojanje elementa identiteta (0)

$5^0\ a \text{ s } a \ge a$ – subidempotentnost (samo u konvencionalnoj fazi logici)

Iz definicije obe norme vidi se da *s-norma* predstavlja *t-konormu*, odnosno da je:

$$a \text{ s } b = 1 - (1 - a) \text{ t } (1 - b)$$

Najčešće korišćeni oblici *s-norme* su:

$1^0\ a \text{ s } b = max(a, b)$ (standardna unija - Gödel-Dummett)

$2^0\ a \text{ s } b = a + b - ab$ (algebarska suma - logičko "ili")

$3^0\ a \text{ s } b = min(1, a + b)$ (ograničena suma - Lukašijevič)

Za ove norme važi:

$$max(a, b) \le a \text{ s } b \le min(1, a + b)$$

Konačno granice *t-* i *s-norme* određene su sledećim relacijama:

$$x \text{ t } y \le \min(x, y) \le \max(x, y) \le x \text{ s } y$$

Sledeći definiciju *s-norme* unija fazi skupova *A* i *B* definisanih nad univerzalnim skupovima *V* i *W* specificira se na sledeći način:

$$A \cup B = \{(v, w), \mu_{A \cup B}(v, w) = \mu_A(v) \text{ s } \mu_B(w) : v \in V, w \in W\}$$

3.5.3. Formiranje fazi skupova nefunkcionalnih karakteristika

Da bi se formirali fazi skupovi za nefunkcionalne karakteristike koje su odabrane za kriterijume na osnovu kojih će se vršiti selekcija potrebno je za svaki od tih QoS atributa specificirati odgovarajući okvir spoznaje, što znači da treba odabrati interval vrednosti na kome se posmatra QoS mera, broj fazi skupova na tom intervalu, kao i oblik samih fazi skupova.

Interval vrednosti QoS mera

Interval vrednosti određen je najmanjom i najvećom vrednošću koju QoS mera može da ima. Drugim rečima vrednosti atributa, za koje je stepen pripadnosti *0* odnosno *1*, određene su samom prirodom atributa, što znači da je donja granica (Q_{min}^{k}) po pravilu određena vrednošću ispod koje se atribut smatra nedopustivo lošim (ako je u pitanju rastuća funkcija) odnosno izuzetno dobrim (ukoliko je u pitanju opadajuća funkcija) dok za gornju granicu (Q_{max}^{k}) prihvatljivosti važi obrnuto tvrđenje. Na primer, ako se posmatra *Raspoloživost* servisa može se reći da je ona neprihvatljiva ukoliko je recimo ispod 50% dok je gornja granica prihvatljivosti recimo 99.5%.

Za kvalitativne QoS atribute čije se vrednosti izražavaju opisno uvodi se skala koja će biti proporcionalna broju opisnih ocena, tako da je interval vrednosti određen opsegom date skale.

Sa druge strane, ukoliko je reč o selekciji servisa za kompoziciju teoretski je moguće da se fazifikacija vrši na svakom pojedinačnom atributu ili na agregiranim vrednostima atributa. Međutim, predstavljanje pojedinačnih atributa pomoću fazi skupova zahtevalo bi i da se agregacija vrši formiranjem fazi relacije između tih fazi skupova. Imajući u vidu raznolikost prirode samih atributa pre svega u pogledu načina na koji se oni agregiraju (npr. ukupno vreme odziva kao zbir vremena odziva pojedinačnih servisa u kompoziciji, ukupna pouzdanost kao proizvod pojedinačnih pouzdanosti, ukupna enkripcija kao prisustvo u svim servisima ili odsustvo makar u samo jednom, opisna ocena ukupne bezbednosti kao minimum bezbednosti pojedinačnih servisa, itd) postavlja se pitanje u kojoj meri bi formiranje fazi relacije moglo da očuva smisao „ukupne vrednosti“ određenog QoS atributa. Otuda je odabrano da se fazifikacija u slučaju izbora servisa za kompoziciju vrši nad agregiranim vrednostima atributa ($q^k(x)$) a ne nad pojedinačnim vrednosti (q_{ij}^{k}) jer je na taj način očuvan pravi smisao svake QoS mere. To znači da se stepen pripadnosti fazi skupa zapravo definiše kao $\mu_{A^k}\left(q^k(x)\right)$. Ovako predstavljeni atributi imaće funkcije pripadnosti između *0* i *1* tako da normalizacija nije potrebna.

Na osnovu svega izloženog i granične vrednosti prihvatljivosti (Q^k_{min} i Q^k_{max}) definišu se za agregirane minimalne i maksimalne QoS vrednosti tj. u opštem slučaju:

$$Q^k_{min} \le q^k\left(Q^k_{min_i}\right), \quad i = 1,...,m.$$
$$Q^k_{max} \ge q^k\left(Q^k_{max_i}\right), \quad i = 1,...,m.$$

Treba napomenuti da znak jednakosti ukazuje na teoretske granice, čije bi usvajanje zapravo vodilo ka tome da se automatski isključe iz razmatranja, odnosno potenciraju, one kompozicije čije su QoS vrednosti jednake datim granicama. Drugim rečima, moglo bi se recimo desiti da se za neku dopustivu kompoziciju čija je QoS vrednost sa stanovišta datog atributa na granici neprihvatljivosti dobije stepen pripadnosti *0* čime se zapravo onemogućava njena selekcija. Stoga, sa praktične tačke gledišta, donja granica treba da bude nešto manja a gornja nešto veća od teoretskih vrednosti npr. za 1%.

Primera radi ukoliko je reč o QoS atributima koji se agregiraju pomoću sume granice bi se mogle odrediti na sledeći način:

$$Q^k_{min} = \left(\sum_{i=1}^{m} Q^k_{min_i}\right) \cdot 0.99$$
$$Q^k_{max} = \left(\sum_{i=1}^{m} Q^k_{max_i}\right) \cdot 1.01.$$

dok bi se u slučaju QoS atributa koji se agregiraju pomoću proizvoda granice mogle odrediti kao:

$$Q^k_{min} = \left(\prod_{i=1}^{m} Q^k_{min_i}\right) \cdot 0.99$$
$$Q^k_{max} = \left(\prod_{i=1}^{m} Q^k_{max_i}\right) \cdot 1.01.$$

Očigledno je da će se za QoS atribute, koji se agregiraju na neki drugačiji način, donja i gornja granica prihvatljivosti dobiti na analogan način tj. korigovanjem vrednosti koje se dobijaju primenom odgovarajućeg operatora agregacije na minimalne odnosno maksimalne moguće vrednosti datog QoS atributa za svaku od komponenti kompozicije, respektivno.

Konačno treba napomenuti da je prilikom određivanja ovih granica moguće uzeti u obzir i postavljena ograničenja u pogledu minimalnih nefunkcionalnih karakteristika koje kompozicija mora da zadovolji ($Q=\{Q^1,...,Q^r\}$) ukoliko su u pogledu datog QoS kriterijuma selekcije istovremeno zadata i ograničenja.

Broj fazi skupova

Budući da je osnovna ideja korišćenja fazi skupova predstavljanje QoS atributa pomoću funkcija pripadnosti dovoljno je da se za svaki od atributa definiše po jedan fazi skup koji pokriva celokupni interval vrednosti datog atributa. To znači da se svaki atribut q^k zapravo predstavlja samo sa po jednim fazi skupom A^k.

Oblik fazi skupa

Nesumnjivo je da ukoliko je poželjno da vrednost datog atributa bude što je moguće veća stepen pripadnosti datom fazi skupu će biti rastuća funkcija, dok će u slučaju da je poželjno ostvariti najmanju moguću vrednost stepen pripadnosti biti opadaću funkcija. Polazeći od smisla koji se želi postići datom funkcijom pripadnosti logično je da se fazi skup izrazi linearnom ili *S* funkcijom. Naime, ukoliko se procenjuje da se u sve mere ravnopravne u pogledu odnosa onda je pogodno odabrati linearnu funkciju. U tom slučaju fazi skupovi imaju sledeći oblik:

Rastući linearni fazi skup:

$$\mu_{A^k}(q_{ij}^k)=\begin{cases}0, & q_{ij}^k < Q_{min}^k \\ \dfrac{q_{ij}^k - Q_{min}^k}{Q_{max}^k - Q_{min}^k}, & q_{ij}^k \in \left(Q_{min}^k, Q_{max}^k\right) \\ 1, & q_{ij}^k > Q_{max}^k\end{cases}$$

Opadajući linearni fazi skup:

$$\mu_{A^k}(q_{ij}^k)=\begin{cases}0, & q_{ij}^k < Q_{min}^k \\ \dfrac{q_{ij}^k - Q_{min}^k}{Q_{max}^k - Q_{min}^k}, & q_{ij}^k \in \left(Q_{min}^k, Q_{max}^k\right) \\ 1, & q_{ij}^k > Q_{max}^k\end{cases}$$

Međutim ukoliko se želi svojevrsna favorizacija vrednosti bliskih najpoželjnijim vrednostima, odnosno kažnjavanje nepovoljnih vrednosti onda bi trebalo odabrati *S*-funkciju.

Rastuća S- funkcija:

$$\mu_{A^k}(q_{ij}^k)=\begin{cases}0 & q_{ij}^k < Q_{min}^k \\ 2\left(\dfrac{q_{ij}^k - Q_{min}^k}{Q_{max}^k - Q_{min}^k}\right)^2 & Q_{min}^k \le x \le \dfrac{Q_{min}^k + Q_{max}^k}{2} \\ 1-2\left(\dfrac{q_{ij}^k - Q_{max}^k}{Q_{max}^k - Q_{min}^k}\right)^2 & \dfrac{Q_{min}^k + Q_{max}^k}{2} \le q_{ij}^k \le Q_{max}^k \\ 1 & q_{ij}^k > Q_{max}^k \end{cases}$$

Opadajuća S-funkcija:

$$\mu_{A^k}(q_{ij}^k)=\begin{cases}1 & q_{ij}^k < Q_{min}^k \\ 1-2\left(\dfrac{q_{ij}^k - Q_{min}^k}{Q_{max}^k - Q_{min}^k}\right)^2 & Q_{min}^k \le q_{ij}^k \le \dfrac{Q_{min}^k + Q_{max}^k}{2} \\ 2\left(\dfrac{q_{ij}^k - Q_{max}^k}{Q_{max}^k - Q_{min}^k}\right)^2 & \dfrac{Q_{min}^k + Q_{max}^k}{2} \le q_{ij}^k \le Q_{max}^k \\ 0 & q_{ij}^k > Q_{max}^k \end{cases}$$

U slučaju kada je reč o kompoziciji servisa u navedenim formulama se kao nezavisna promenljiva koristi agregirana QoS vrednost ($q^k(x)$).

3.5.4. Konvencionalna fazi logika u modelovanju procesa odlučivanja

Polazeći od koncepta predstavljanja QoS atributa pomoću fazi skupova prirodno je da se ispita mogućnost korišćenja fazi skupova u procesu selekcije. Budući da proces donošenja odluke najčešće podrazumeva da se uspostavi funkcionalna zavisnost između kriterijuma koja iskazuje valjanost pojedinačne alternative tako da se najbolja alternativa dobija minimizacijom ili maksimizacijom date funkcije, pokazaće se da fazi logika predstavlja jednu od mogućnosti za formiranje ovakve funkcije.

U osnovi fazi odlučivanja je modelovanje iskaza tipa implikacija koje povezuju uslov (*A*) i posledicu (*B*), a koji se mogu izraziti kao:

Ako je *A*, **tada** je *B*

U skladu sa tim neophodno je da se pre svega razmotri način na koji se modeluju uslov i implikacija, kao i da se ustanovi na koji način se za neki konkretni uslov određuje odgovarajuća posledica. Konačno neophodno je i da se precizira način na koji će se izvršiti optimizacija dobijenog iskaza.

Modelovanje uslova

Uslov (***ako)*** izražava logičku povezanost veličina koje utiču na posledicu. To znači da se on formira konjunkcijom i/ili disjunkcijom pojedinačnih uslova predstavljenih fazi skupovima, koji čine okvir spoznaje svake od relevantnih veličina koji utiču na posledicu, odnosno presekom ili unijom fazi skupova. Ako se sa A^i ($i = 1,2,\ldots,n$) označe sve veličine koje mogu uticati na posledicu, a sa A^i_j ($j = 1,2 \ldots, m$) fazi skupovi koji predstavljaju okvir spoznaje veličine A^i tada se u opštem slučaju uslovi izražavaju preko relacija oblika:

$$A=\bigcap_{i,j} A_j^i; \quad i \in \{1,...,n\},\ j \in \{1,...,m\}$$

$$A=\bigcup_{i,j} A_j^i; \quad i \in \{1,...,n\},\ j \in \{1,...,m\}$$

pri čemu se u svakom iskazu odabiraju fazi skupovi (A^i_j) onih veličina koje utiču na datu posledicu.

Potrebno je istaći da usvojeni način modelovanja uslova ne omogućava da se u jednom iskazu kombinuju konjunkcija i disjunkcija uslova. Navedeni problem se prevazilazi kombinacijom ovih elementarnih iskaza o čemu će kasnije biti više reči.

Modelovanje implikacije

Implikacija (***tada***), iskazuje se fazi relacijom između fazi skupa uslova i fazi skupa posledice koja izražava logiku samog iskaza koji se modeluje. To znači da rezultujući fazi skup treba da bude opisan funkcijom pripadnosti koja izražava meru istinitosti posmatranog iskaza u svakoj tački ravni (*uslov,posledica*). Pri tome, relacija treba da bude takva da funkcija pripadnosti fazi skupa koji se dobija njenom primenom ima vrednost blisku jedinici za one tačke u ravni (*uslov,posledica*) u kojima su funkcije pripadnosti uslova i posledice bliske jedinici, što znači da treba da predstavlja neku vrstu preseka dva skupa. Polazeći od ovog zahteva očigledno je da je za predstavljanje implikacije moguće koristiti bilo koju *t-normu*.

Najčešće se kao *t-norma* koriste:

- Korelacioni minimum:

$$S_{min} = A \circ B = \left\{(v, w), \mu_{S_{min}}(v, w) = \min(\mu_A(v), \mu_B(w)) : v \in V, w \in W\right\}$$

- Korelacioni proizvod:

$$S_{prod} = A \bullet B = \left\{(v, w), \mu_{Sprod}(u, w) = \mu_A(u) \cdot \mu_B(w) : v \in V, w \in W\right\}$$

Određivanje fazi skupa posledice

Uspostavljanje fazi relacije *S* između uslova i posledice omogućava da se za svaku konkretnu vrednost uslova (*A*) odredi odgovarajuća implikacija (*B*). Ova relacija se dobija primenom *max-t-kompozicije* (*Nguyen & Walker, 2005*) definisane na sledeći način:

$$B = Rel_{max\,t}(S, A) = \left\{w, \mu_B(w) = \max_{v \in V}\{\mu_A(v) t \mu_S(v, w)\} : v \in V, w \in W\right\}$$

pri tome se kao *t-norma* najčešće koristi minimum ili proizvod pa se govori o *max-min* i *max-proizvod* kompozicijama.

Korišćenje *max-t kompozicije* za određivanje fazi skupa neke konkretne implikacije zahteva da se, kao prvo, konkretna vrednost uslova predstavi odgovarajućim fazi skupom. Budući da se radi o samo jednoj vrednosti, recimo v_0, ona može da se predstavi singlton fazi skupom A_c:

$$A_c = \{v, \mu_{Ac}(v) : v \in V; \mu_{Ac}(v_0) = 1, \mu_{Ac}(v) = 0, v \neq v_0\}$$

U tom slučaju *max-t-kompozicija* daje fazi skup konkretne implikacije B_c. Relacija određivanja ovog skupa zavisi od toga da li je za modelovanje implikacije korišćen korelacioni minimum ili proizvod.

$$B_c = Rel_{max\,t}(S_{min}, A_c) = \left\{(w), \mu_{Bc}(w) = \mu_{S_{min}}(v_0, w) = \min(\mu_A(v_0), \mu_B(w) : w \in W\right\};$$

$$B_c = Rel_{max\,t}(S_{prod}, A_c) = \left\{(w), \mu_{Bc}(w) = \mu_{S_{prod}}(v_0, w) = \mu_A(v_0) \cdot \mu_B(w) : w \in W\right\}$$

Agregacija (predstavljanje složenih iskaza)

Kao što je već istaknuto jedan uslov može predstavljati ili samo konjunkciju ili samo disjunkciju što znači da se ne može modelovati iskaz tipa:

Ako je (A1 *i* A2 *i*…) ***ili*** **ako** je (A3 *ili* A4 *ili*…) ***ili*** **ako** je…, **tada** je B

Ovaj problem se prevazilazi tako što se posledica specificira nizom jednostavnijih (elementarnih) iskaza oblika:

Ako je (A1 *i* A2 *i*…) **tada** je B
Ako je (A3 *ili* A4 *ili*…) **tada** je B
...

tako da se rezultujući fazi skup posledice dobija agregacijom fazi skupova (B_{ci}) koji se dobijaju na osnovu pojedinačnih iskaza. Budući da su iskazi povezani disjunkcijom agregacija predstavlja uniju ovih skupova, što znači da se realizuje bilo kojom *s-normom*.

$$B_c = \bigcup_i B_{ci}$$

Površina odlučivanja

U opštem slučaju, rezultujuće fazi skupove moguće je optimizovati bilo uvođenjem neke fazi mere preference (*Figueira & al, 2005, Poglavlje 14*) bilo tako što će se najpre oni defazifikovati (tj. odrediti brojna vrednosti koja reprezentuje fazi skup), pa zatim optimizovati skup tako dobijenih vrednosti.

Ako se primenom *max-t-kompozicije* odrede fazi skupovi posledice za sve dozvoljene vrednosti uslova tada je moguće da se svaki od ovih skupova defazifikuje, odnosno da se odrede brojne vrednosti posledice u funkciji ulaznih promenljivih. Ova funkcija određuje *površinu odlučivanja* na kojoj treba odrediti minimum ili maksimum.

3.5.4.1. ILUSTRATIVNI PRIMER

U cilju ilustracije opisanog fazi pristupa posmatraće se problem selekcije pojedinačnih servisa, na osnovu Vremena Odziva (*eng. Response Time*) i Raspoloživosti (eng. *Availability*), pri čemu se ocena samog servisa može okarakterisati sledećim iskazom:

Ako je *Vreme odziva* ***I Ako*** *je Raspoloživost* **tada** je *Ocena*

Potrebno je zapaziti da se ovim iskazom izražava čitav niz implikacija. Sa gledišta donosioca odluke *Vreme odziva* može, naime, da se kreće od veoma malog, znači jako dobrog, do veoma velikog, odnosno nedopustivo lošeg. Na isti način i *Raspoloživost* može biti mala, odnosno slaba, umereno dobra ili izuzetno velika. U svim tim situacijama *Ocena* je rezultat kombinacije ove dve veličine. Drugim rečima, posmatrani iskaz u sebi sažima niz sledećih iskaza:

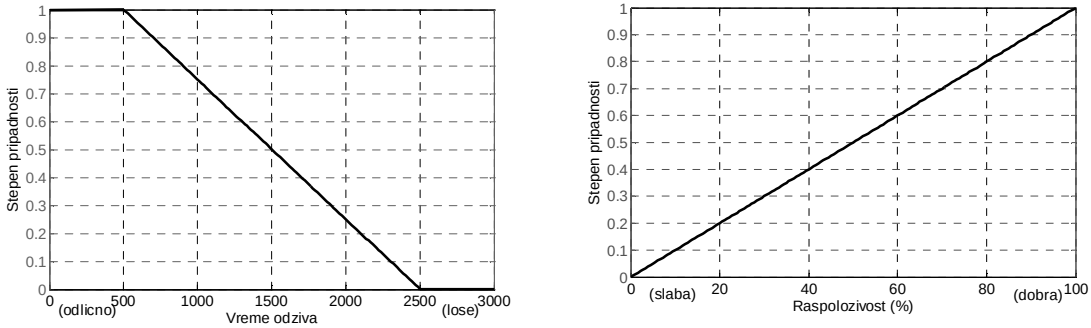
Ako je *Vreme odziva* malo ***I Ako*** *je Raspoloživost* velika **tada** je *Ocena* odlična

Ako je *Vreme odziva* malo ***I Ako*** *je Raspoloživost* srednja **tada** je *Ocena vrlo* dobra

Ako je *Vreme odziva* srednje ***I Ako*** *je Raspoloživost* mala **tada** je *Ocena* dovoljna

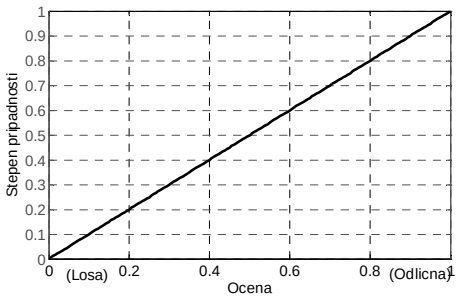
itd..

Da bi se formirao model kojim će se predstaviti svi ovi iskazi, neophodno je da se kao prvo, usvoje fazi skupovi promenljivih *Vreme odziva* i *Raspoloživost*. Ako se pretpostavi, shodno razmatranjima iz Potpoglavlja 3.5.3, da su okviri spoznaje oba pojma predstavljeni sa po jednim fazi skupom, onda ti skupovi mogu imati izgled prikazan na Slici 14. Pri tome je potrebno istaći da je oblik skupova određen činjenicom da je sa gledišta selekcije servisa poželjno da *Vreme odziva* bude što manje, a *Raspoloživost* što veća. Pored toga, usvojeno je da servis mora da ima neko konačno *Vreme odziva*, što znači da donja granica ne može biti jednaka nuli. Isto tako usvojeno je i da postoji neka gornja granica iznad koje je *Vreme odziva* apsolutno neprihvatljivo. Treba napomenuti da je ista logika mogla da se primeni i za fazi skup *Raspoloživost* za koju je najčešće neprihvatljivo da bude ispod 50%. U tom slučaju dobila bi se bolja rezolucija stepena pripadnosti u opsegu koji je od interesa. Međutim, ovde je u cilju postizanja veće preglednosti površina odlučivanja usvojeno da *Raspoloživost* bude u nekoj meri prihvatljiva od 0 do 100%.



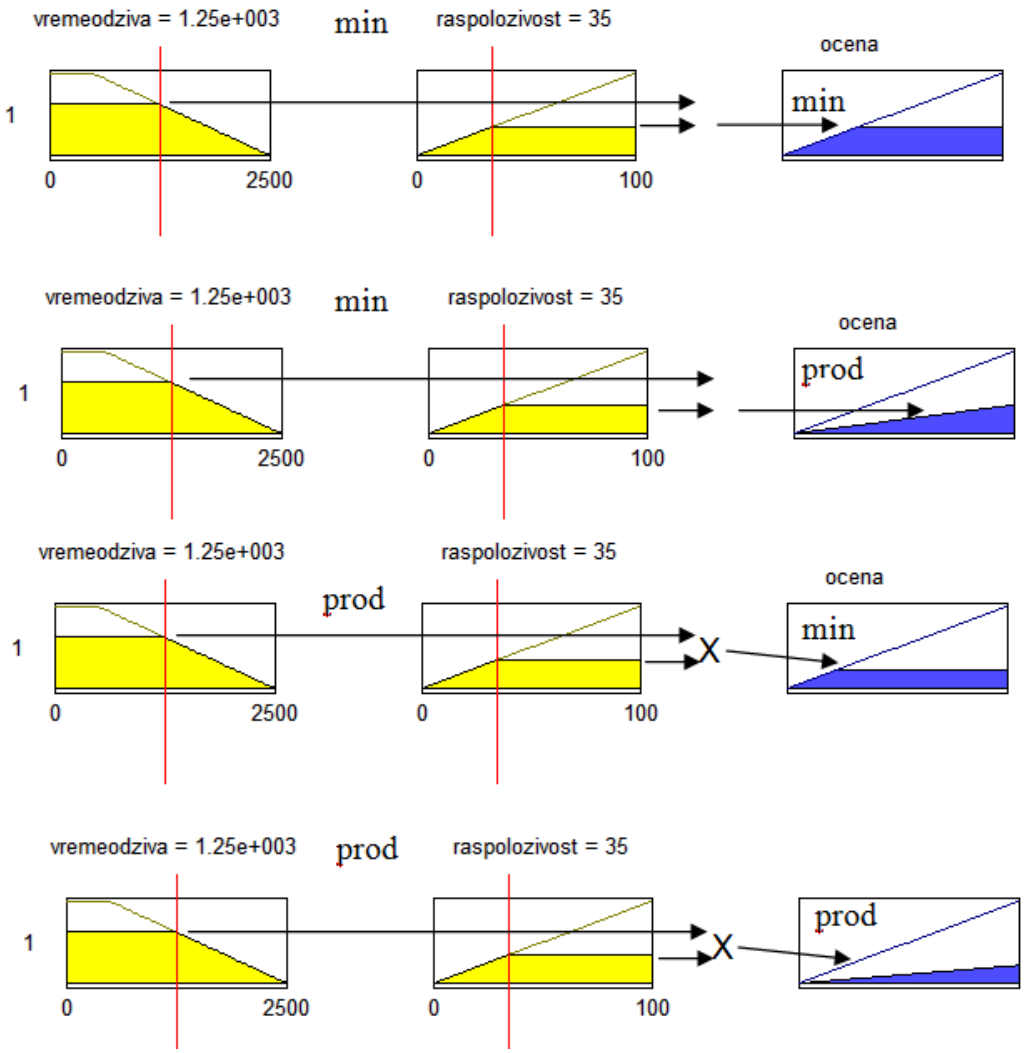
Slika 14. Fazi skupovi uslova: *Vreme odziva* i *Raspoloživost*

Imajući u vidu da se ocena kreće od nedopustivo loše do odlične, pojmu *Ocena* može se pridružiti fazi skup prikazan na Slici 15.



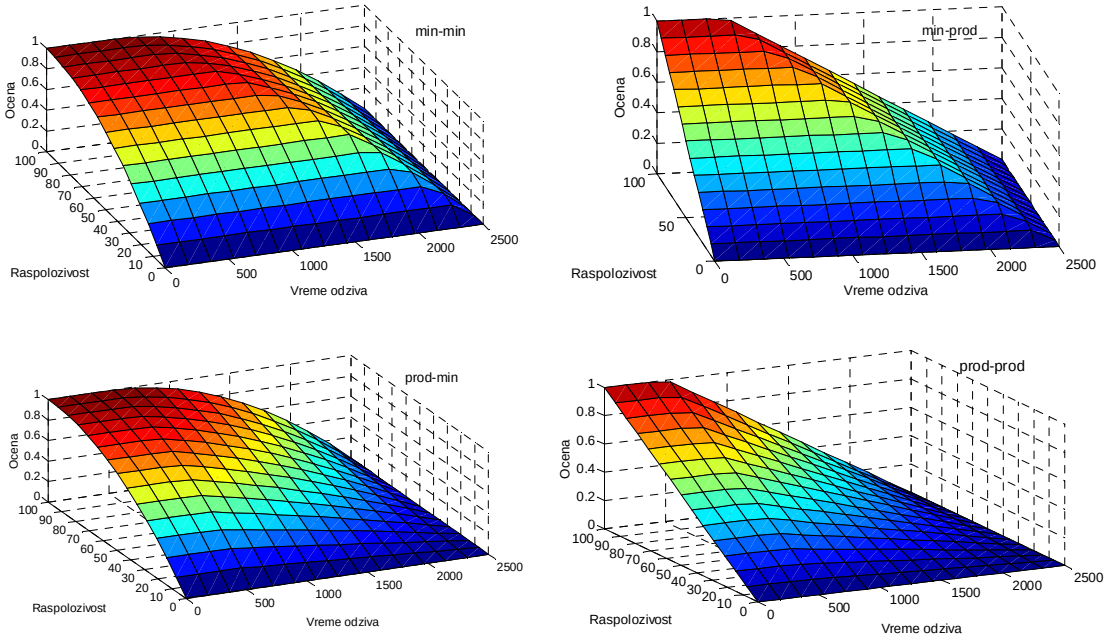
Slika 15. Fazi skup posledice: *Ocena*

Fazi skup *Ocena* koji će se dobiti za neke konkretne vrednosti uslova zavisiće od izbora *t-norme* za formiranje preseka uslova, kao i od načina na koji se modelira implikacija. Na Slici 16 data je ilustracija formiranja fazi skupa izlaza za sve četiri moguće kombinacije izbora *t-normi:*



Slika 16. Ilustracija načina formiranje fazi skupa posledice

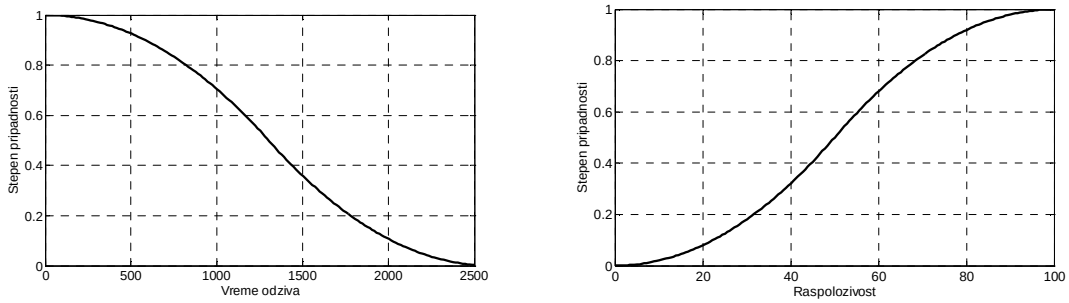
Konačno, ako se optimizacija vrši na osnovu brojnih vrednosti mogućih ocena, neophodno je još da se izvrši defazifikacija dobijenog skupa. Imajući u vidu značenje same ocene čini se da je najpogodnije da se defazifikacija izvrši tako što će se za svaki fazi skup posledice odrediti njegova površina. Dodatno, budući da je maksimalna površina rezultujućeg fazi skupa jednaka 0.5, uzimanjem dvostruke površine dobiće se vrednosti ocena u opsegu od 0 do 1[1]. Površine odlučivanja u posmatranom primeru za sve četiri moguće kombinacije izbora *t-normi* date su na Slici 17. Kao što se vidi izbor samih *t-normi* ne utiče bitno na oblik površine odlučivanja. Suština je, zapravo, da svaka kombinacija omogućava da se za date vrednosti *Vremena Odziva* i *Raspoloživosti* odredi brojna vrednost ocene koja se zatim može optimizovati.



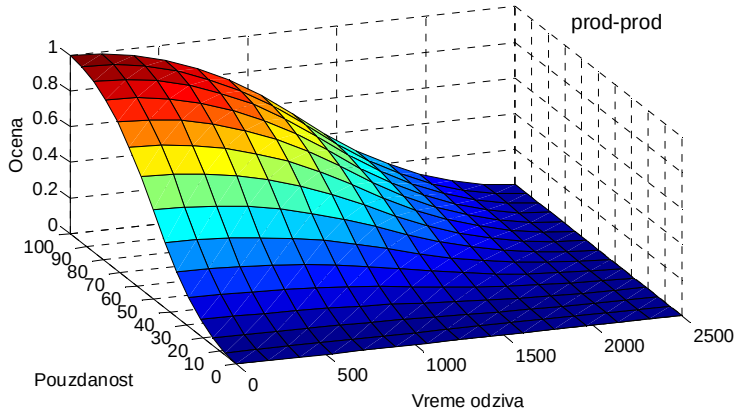
Slika 17. Površine odlučivanja za različite izbore *t-normi*

Potrebno je istaći da je u posmatranom primeru moguće dobiti i drugačije površine odlučivanja. Ako se, na primer, umesto linearnih fazi skupova uslova koriste *S* funkcije (Slika 18), kojima se zapravo potencira neprihvatljivost lošijih vrednosti i prihvatljivost dobrih, tada se dobija i drugačija površina odlučivanja kao što je pokazano na Slici 19.

[1] Potrebno je napomenuti da kada se defazifikacija vrši preko površine isti rezultat se može dobiti i kada se kao fazi skup izlaza usvoji skup čije su sve funkcije pripadnosti jednake 1, te je njegova površina takođe jednaka 1. U tom slučaju implikacijom se samo skalira visina pravougaonika.



Slika 18. Fazi skupovi veličina uslova modelovanih pomoću *S* funkcija



Slika 19. Površina odlučivanja za S fazi skupove uslova

Konačno, važno je uočiti i činjenicu da fazi pristup omogućava da se selekcija servisa zasniva i na nefunkcionalnim karakteristikama koje nisu merljive već se opisuju lingvističkim promenljivima.

3.5.5. Konzistentna fazi logika

Čitav niz primena izloženog pristupa fazi modelu odlučivanja odnosi se na situacije u kojima je svaki iskaz nezavisan. Odsustvo korelacije između iskaza nužno implicira i nepostojanje alternativnih mogućnosti, što znači da se ni u jednom iskazu ne može sresti negacija nekog od uslova (tj. komplement odgovarajućeg fazi skupa). Iskazi se naime vezuju isključivo za skupove koji čine okvir spoznaje svakog od pojmova, pa samim tim nema ni mesta za njihove komplemente.

Međutim, problem donošenja odluke na osnovu većeg broj kriterijuma je, po pravilu, vezan za pravljenje svojevrsnog kompromisa između različitih korelisanih, a često i suprotstavljenih kriterijuma. Drugim rečima iskazi koji se modeluju gotovo uvek uključuju i skup i njegov komplement.

U posmatranom primeru kriterijumska funkcija uključivala je *Vreme Odziva* i *Raspoloživost*. Međutim, često se pri oceni servisa pravi kompromis između *Raspoloživosti* i *Pouzdanosti (eng. Reliability).* Naime, ukoliko je *Raspoloživost* servisa dovoljno velika tada može biti od većeg značaja uzeti u razmatranje *Vreme Odziva* dok će u suprotnom slučaju (ukoliko je *Raspoloživost* mala) od većeg značaja biti *Pouzdanost*, što se zapravo može izraziti sledećim logičkim iskazom:

(Ako je „*Vreme odziva*“ **I** „*Raspoloživost*“)
ILI
(Ako „*Nije raspoloživost*“ **I** „*Pouzdanost*“)
tada je „*Ocena*“

Izloženi primer zapravo ilustruje problem sa kojim se sreće primena konvencionalne fazi logike u teoriji odlučivanja. Naime, postojanje iskaza kojima se zamenjuje odsustvo nekog drugog uslova podrazumeva da fazi relacija konzistentno tretira problem kontradikcije i isključenja trećeg.

Princip kontradikcije podrazumeva da ocena nekog pojma ne može istovremeno pripadati nekom skupu i njegovom komplementu što implicira da sve funkcije pripadnosti fazi skupa iskaza $A\ i\ neA$ treba da budu jednake nuli.

$$A \cap \neg A = 0$$

Na isti način ako se ocena nekog pojma ne nalazi u skupu *A* ona mora pripadati komplementarnoj oceni – *princip isključenja trećeg* – što znači da bi sve funkcije pripadnosti fazi skupa formiranog na osnovu iskaza $A\ ili\ neA$ trebalo bi da budu jednake jedinici.

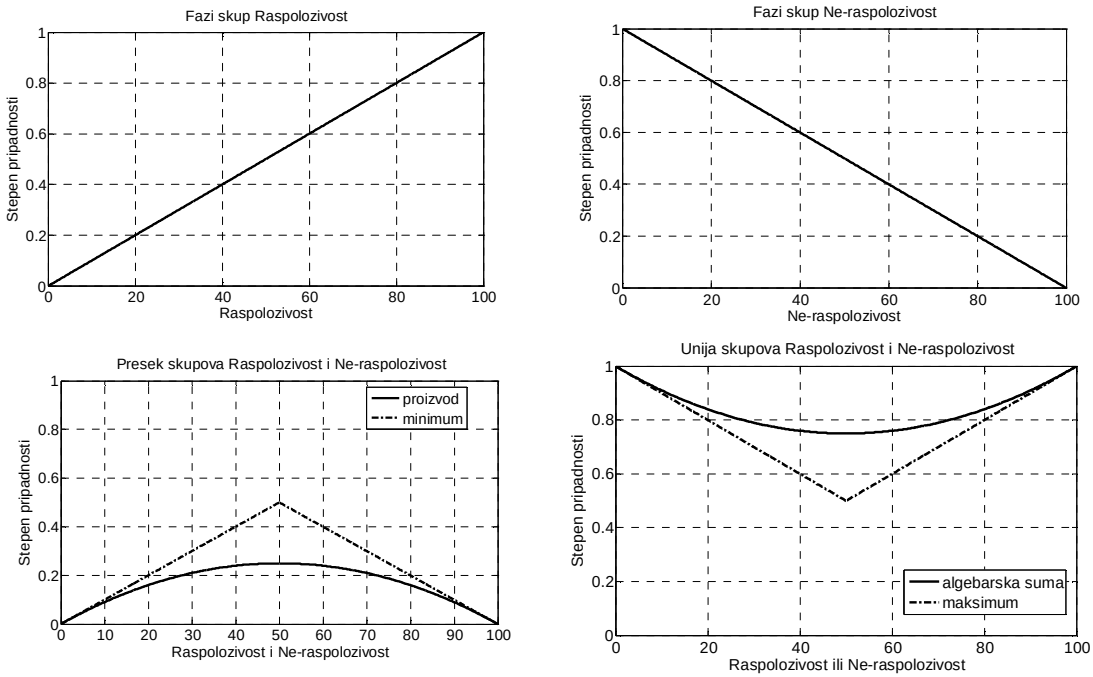
$$A \cup \neg A = 1$$

Imajući u vidu način na koje se određuju presek i unija skupova vidi se da princip kontradikcije i isključena trećeg zahteva da, pored navedenih uslova, *t*- i *s-norma* ispunjavaju i sledeće uslove (*Radojević, 2008c*):

$$\forall a \in U, \quad at\neg a = 0 \quad i \quad as\neg a = 1, \text{ gde } \neg a \text{ označava komplement } a.$$

Lako je pokazati da definicija *t-norme* u opštem slučaju ne podrazumeva ispunjenje navedenih uslova. Ako se, na primer, posmatraju fazi skup „*Raspoloživost*“ i njegov komplement „*Ne-raspoloživost*“ prikazani na Slici 20 tada se primenom algebarskog proizvoda i sume za formiranje preseka i unije ta dva skupa dobijaju fazi skupovi iz

kojih sledi da postoji područje kod kojih je servis u nekoj meri istovremeno i raspoloživ i ne-raspoloživ, čime je narušen princip kontradikcije. Isto tako, vidi se da postoji područje u kome servis u nekoj meri nije ni raspoloživ ni ne-raspoloživ, što znači da je narušen princip isključenja trećeg.



Slika 20. Presek i unija fazi skupa i njegovog komplementa

Opisani problem je rezultat činjenice, kao što je istaknuto u (*Radojević, 2008b*) da se pri definisanju *t-* i *s- norme* sledila ideja proširenja Bulove logike u smislu istinitnosne funkcionalnosti, odnosno principa izračunavanja složenog izraza na osnovu istinitnosnih vrednosti njegovih komponenti. Pri tome se nije vodilo računa o ostalim aksiomima Bulove logike.

Činjenica da jedan objekat može istovremeno da ne pripada ni skupu ni njegovom komplementu, kao i da bude i prisutan i odsutan, nužno dovodi do nekonzistenosti pri odlučivanju u svim slučajevima u kojima su kriterijumi korelisani ili alternativni (odnosno kompromisni). U cilju otklanjanja ovog nedostatka neophodno je da se umesto konvencionalne fazi logike koristi **konzistentna fazi logika** u kojoj se definiše generalizovani proizvod kao posebna potklasa *t-norme* pomoću koga se koncept fazi logike smešta u okvir Bulove algebre (*Radojević, 2000*).

U osnovi konzistentne fazi logike leži interpolativna Bulova algebra (IBA) koja se koristi kao okvir za tretiranje funkcija pripadnosti kao mere istinitosti neke tvrdnje. Razlog za ovo opredeljenje leži u činjenici da IBA, nasuprot klasičnoj fazi logici, ima

osobinu da na vrednosnom nivou održava sve zakone na kojima počiva Bulova algebra. Ovo se ostvaruje razdvajanjem same algebre na dva nivoa: simbolički i vrednosni. Na simboličkom nivou definišu se objekti koji zapravo čine zakone Bulove algebre nad atomima, odnosno elementarnim članovima skupa koji se ne mogu izraziti ni jednom Bulovom funkcijom drugih elemenata skupa. Na vrednosnom nivou, izračunavanje se obavlja tek nakon što se data relacija pomoću transformacija, koje se izvode na simboličkom nivou, svede na oblik koji sadrži samo atome. Na taj način se prirodno obezbeđuje očuvanje svih zakona Bulove algebre.

U cilju ilustracije ove ideje posmatraće se ponovo izloženi problem preseka i unije fazi skupa i negacije tog skupa.

Neka je za neku vrednost promenljive x_0 funkcija pripadnosti fazi skupa *A:*

$$\mu_A(x_0) = 0.4$$

tada je funkcija pripadnosti komplementa fazi skupa $\neg A$ za istu vrednost promenljive:

$$\mu_{\neg A}(x_0) = 1 - 0.4 = 0.6$$

Primenom klasične fazi logike dobija se:

$$\mu_A(x_0) \mathbin{t} \mu_{\neg A}(x_0) = 0.4 \cdot 0.6 = 0.24 \neq 0$$
$$\mu_A(x_0) \mathbin{s} \mu_{\neg A}(x_0) = 0.4 + 0.6 - 0.4 \cdot 0.6 = 0.76 \neq 1$$

Međutim, pošto se funkcija pripadnosti komplementa dobija preko funkcije pripadnosti originalnog skupa ona ne predstavlja atomski (elementarni) element fazi skupa. Sledeći izloženu ideju transformacije, operacije preseka i unije (*t*- i *s*- *norma*) se u posmatranom primeru mogu odrediti i na sledeći način:

$$\mu_A(x_0) \mathbin{t} \mu_{\neg A}(x_0) = \mu_A(x_0) \mathbin{t} (1 - \mu_A(x_0)) = \mu_A(x_0) \mathbin{t} 1 - \mu_A(x_0) \mathbin{t} \mu_A(x_0) = \mu_A(x_0) - \mu_A(x_0) \mathbin{t} \mu_A(x_0)$$

Ako bi se *t-norma* ograničila na one oblike, koji pored navedenih uslova imaju i osobinu idempotentnosti, tada bi prethodni izraz bio jednak *0*, što znači da princip kontradikcije važi za sve vrednosti funkcija pripadnosti.

$$\forall \mu_A \in V, \mu_A \mathbin{t} \mu_A = \mu_A \Rightarrow \mu_A(x_0) \mathbin{t} \mu_{\neg A}(x_0) = 0$$

Šta više, zadržavanjem iste osobine *t-norme*, korišćenjem *t-konorme* pokazuje se da važi i princip isključenja trećeg:

$$\begin{aligned}
\mu_A(x_0) \,\mathrm{s}\, \mu_{\neg A}(x_0) &= 1-\big(1-\mu_A(x_0)\big)\,\mathrm{t}\,\big(1-\mu_{\neg A}(x_0)\big) = \\
&= 1-\big[1\,\mathrm{t}\,1-\mu_A(x_0)\,\mathrm{t}\,1-1\,\mathrm{t}\,\mu_{\neg A}(x_0)+\mu_A(x_0)\,\mathrm{t}\,\mu_{\neg A}(x_0)\big] = \\
&= 1-\big[1-\mu_A(x_0)-1\,\mathrm{t}\,\big(1-\mu_A(x_0)\big)+\mu_A(x_0)\,\mathrm{t}\,\mu_{\neg A}(x_0)\big] = \\
&= 1-\big[1-\mu_A(x_0)-1\,\mathrm{t}\,1+1\,\mathrm{t}\,\mu_A(x_0)+\mu_A(x_0)\,\mathrm{t}\,\big(1-\mu_A(x_0)\big)\big] = \\
&= 1-\big[1-\mu_A(x_0)-1+\mu_A(x_0)+\mu_A(x_0)\,\mathrm{t}\,1-\mu_A(x_0)\,\mathrm{t}\,\mu_A(x_0)\big] = \\
&= 1-\big[\mu_A(x_0)-\mu_A(x_0)\big] = 1
\end{aligned}$$

Izloženi primer ne treba razumeti kao formalno proširenje uslova koje treba da ispuni *t-norma*, već samo kao ilustraciju činjenice da se pomoću dodatnih uslova može prevazići problem kontradikcije i isključenja trećeg.

3.5.6. Predstavljanje fazi skupova pomoću interpolativne Bulove algebre

Detaljan prikaz interpolativne Bulove algebre (IBA) kao i konzistentne fazi logike koja na njoj počiva može se naći u (*Radojević, 2008b*). U cilju razumevanja njenog korišćenja pri formiranju fazi modelovanju problema selekcije servisa ovde će se navesti samo osnovni rezultati.

Realizacija fazi skupova u okviru Bulove algebre polazi od ideje da atributi (svojstva) koji se posmatraju čine domen Bulove algebre. Pri tome se posmatra samo skup primarnih atributa, odnosno onih koje se ne mogu predstaviti Bulovom funkcijom nekih drugih primarnih atributa. Skup primarnih atributa $\Omega=\{a_1,\cdots a_n\}$ generiše konačnu Bulovu algebru posmatranih atributa $BA_p(\Omega)$. Bulova algebarska struktura posmatranih atributa je:

$$\left(BA_p(\Omega),\cap,\cup,C\right)$$

Svaki element Bulove algebre, odnosno u posmatranom slučaju, fazi relacije $\varphi\in BA_p(\Omega)$ može se jednoznačno predstaviti pomoću sledeće disjunktivne kanonične forme:

$$\varphi = \bigcup_{S\in P(\Omega)\,|\,\sigma_\varphi(S)=1} \alpha(S)$$

gde su:

- $\alpha(S)=\bigcap_{a_i\in S} a_i \bigcap_{a_j\in\Omega\setminus S} Ca_j$ - atomi (atomski element $BA_p(\Omega)$), najjednostavniji elementi koji ne sadrže nijedan element Bulove algebre u sebi osim trivijalne nulte konstante, što implicira da je konjunkcija dva atomska elementa jednaka nuli.
- $\sigma_\varphi(S),\left(S\in P(\Omega)\right)$ struktura posmatrane fazi relacije, odnosno skup atomskih elemenata sadržanih u relaciji, definisana pomoću karakteristične funkcije strukture (strukturalne funkcije):

$$\sigma_\varphi(S)=\begin{cases}1, & \alpha(S)\subseteq\varphi\\ 0, & \alpha(S)\not\subset\varphi\end{cases}\quad\left(\varphi\in BA(\Omega);S\in P(\Omega)\right)$$

U skladu sa ovom definicijom vidi se da je struktura primarnih atributa data sledećim skupom funkcija:

$$\sigma_{a_i}(S)=\begin{cases}1, & a_i\in S\\ 0, & a_i\notin S\end{cases}\quad\left(S\in P(\Omega),a_i\in\Omega\right)$$

Uvođenje disjunktivne kanonične forme ima za cilj da se svaka složena fazi relacija razloži na atomske elemente i to izvođenjem operacija na simboličkom nivou. U tom smislu struktura zapravo određuje koji atomski elementi učestvuju u jednoj relaciji. Pri tome, struktura neke relacije se određuje direktno na osnovu struktura elemenata koji učestvuju u relaciji i sledećih pravila:

$$\begin{aligned}&\sigma_{\varphi\cup\psi}(S)=\sigma_\varphi(S)\vee\sigma_\psi(S)\\ &\sigma_{\varphi\cap\psi}(S)=\sigma_\varphi(S)\wedge\sigma_\psi(S)\quad;\left(S\in P(\Omega)\right)\\ &\sigma_{C\varphi}(S)=\neg\sigma_\varphi(S)\end{aligned}$$

gde simboli $\neg,\wedge$ i $\vee$ predstavljaju klasične binarne operatore negacije, konjunkcije i disjunkcije.

Tek kada je fazi relacija raščlanjena na simboličkom nivou moguće je izračunavanje njene vrednosti za neke date vrednosti argumenata, odnosno funkcija pripadnosti. Ovo nadalje znači da u određivanju vrednosti neke relacije učestvuju isključivo vrednosti atomskih elemenata. Ideja da se relacija posmatra sa gledišta strukturne funkcionalnosti obezbeđuje očuvanje svih postulata Bulove logike, koji važe

za sve atomske elemente shodno njihovoj definiciji. U tom smislu princip istinitnosne funkcionalnosti koji leži u osnovi klasične fazi logike ovde predstavlja samo posledicu strukturne funkcionalnosti.

Na osnovu izloženog koncepta vidi se da je u cilju izračunavanja vrednosti neke relacije potrebno da se utvrde vrednosti primarnih atributa i atomskih elemenata. Budući da su primarni atributi predstavljeni fazi skupovima njihove vrednosti određene su samim funkcijama pripadnosti ($\mu_{a_i}, (i=1,2,\ldots,n)$).

Za izračunavanje vrednosti atomskih elementa, a preko njih i svih drugih relacija, koristi se generalizovani Bulov polinom (GBP) koji uključuje operacije sabiranja (+), oduzimanja (-) i generalizovanog proizvoda ($\otimes$).

Generalizovani proizvod je bilo koja funkcija $\otimes : [0,1]\times[0,1]\to[0,1]$ koja zadovoljava sva četiri aksioma *t-norme* uz još jedan dodatni peti aksiom:

1^0 Komutativnost:

$$a(x)\otimes b(x)=b(x)\otimes a(x),\quad (a,b\in\Omega; a(x),b(x)\in[0,1]; x\in X)$$

2^0 Asocijativnost:

$$a(x)\otimes(b(x)\otimes c(x))=(a(x)\otimes b(x))\otimes c(x),\quad (a,b,c\in\Omega; a(x),b(x),c(x)\in[0,1]; x\in X)$$

3^0 Monotonost

$$a(x)\le b(x)\Rightarrow a(x)\otimes c(x)\le b(x)\otimes c(x),\quad (a,b,c\in\Omega; a(x),b(x),c(x)\in[0,1]; x\in X)$$

4^0 Ograničenost

$$a(x)\otimes 1=a(x),\quad (a\in\Omega; a(x)\in[0,1]; x\in X)$$

5^0 Uslov ne-negativnosti

$$\sum_{K\in P(\Omega\setminus S)}(-1)^{|K|}\bigotimes_{a_i\in K\cup S}a_i(x)\ge 0,\quad (\Omega=\{a_1,\ldots,a_n\}, S\in P(\Omega), a_i(x)\in[0,1]; x\in X)$$

Neka je $\alpha^{\otimes}(S)$ vrednosna realizacija, odnosno funkcija pripadnosti, atomskog elementa $\alpha(S), (S\in P(\Omega))$, $\alpha^{\otimes}(S): X\to[0,1]$. Ona se izražava pomoću generalizovanog atomskog Bulovog polinoma:

$$\alpha^{\otimes}(S)(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n})=\sum_{K\in P(\Omega\setminus S)}(-1)^{|K|}\bigotimes_{a_i\in K\cup S}\mu_{a_i},\quad (a_i\in\Omega,\mu_{a_i}\in[0,1], i=1,2,\ldots,n)$$

Funkcije pripadnosti svih drugih relacija izračunavaju se preko odgovarajućeg generalizovanog Bulovog polinoma:

$$\varphi^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)=\sum_{S\in P(\Omega)\mid\sigma_{\varphi}(S)=1}\alpha^{\otimes}(S)\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)=$$

$$\sum_{S\in P(\Omega)}\sigma_{\varphi}(S)\alpha^{\otimes}(S)\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right),$$

$$\left(a_i\in\Omega,\mu_{a_i}\in[0,1],i=1,2,\ldots,n\right).$$

Ovaj polinom se može predstaviti i kao skalarni proizvod dva vektora:

$$\varphi^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)=\vec{\sigma}_{\varphi}(S)\vec{\alpha}^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)$$

gde su:

$\vec{\sigma}_{\varphi}=\left[\sigma_{\varphi}(S)\middle|S\in P(\Omega)\right]_{1\times N}\quad\left(N=2^{|\Omega|}\right)$ – vektor strukture

$\vec{\alpha}^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)=\left[\alpha^{\otimes}(S)\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)\middle|S\in P(\Omega)\right]_{1\times N}^{T}$ – vektor atomskih polinoma

$\left(a_i\in\Omega,\mu_{a_i}\in[0,1],i=1,2,\ldots,n,N=2^{|\Omega|}\right)$

U skladu sa tim svaki fazi skup $\varphi^{\otimes}$ čije su funkcije pripadnosti $\varphi^{\otimes}\left(x\right)$ može se prikazati kao unija relevantnih atomskih skupova:

$$\varphi^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)=\bigcup_{S\in P(\Omega)\mid\sigma_{\varphi}(S)=1}\alpha^{\otimes}(S)\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)$$

Iz prethodne definicije se vidi da strukturni vektor, budući da je vrednosno neutralan, ima algebarsku prirodu što znači da su svi aksiomi i teoreme Bulove algebre očuvani (*Radojevic, 2013*), kao što je to prikazano u sledećoj tabeli.

Tabela 4. Aksiomi i teoreme Bulove algebre u konzistentnoj fazi logici

Aksiome		*Teoreme*	
$\vec{\sigma}_{\varphi\cup(\psi\cup\vartheta)}=\vec{\sigma}_{(\varphi\cup\psi)\cup\vartheta}$	$\vec{\sigma}_{\varphi\cap(\psi\cap\vartheta)}=\vec{\sigma}_{(\varphi\cap\psi)\cap\vartheta}$	$\vec{\sigma}_{\varphi\cup\varphi}=\vec{\sigma}_{\varphi}$	$\vec{\sigma}_{\varphi\cap\varphi}=\vec{\sigma}_{\varphi}$
$\vec{\sigma}_{\varphi\cup\psi}=\vec{\sigma}_{\psi\cup\varphi}$	$\vec{\sigma}_{\varphi\cap\psi}=\vec{\sigma}_{\psi\cap\varphi}$	$\vec{\sigma}_{\varphi\cup0}=\vec{\sigma}_{\varphi}$	$\vec{\sigma}_{\varphi\cap1}=\vec{\sigma}_{\varphi}$
$\vec{\sigma}_{\varphi\cup(\varphi\cap\psi)}=\vec{\sigma}_{\varphi}$	$\vec{\sigma}_{\varphi\cap(\varphi\cup\psi)}=\vec{\sigma}_{\varphi}$	$\vec{\sigma}_{\varphi\cup1}=\vec{1}$	$\vec{\sigma}_{\varphi\cap0}=\vec{0}$
$\vec{\sigma}_{\varphi\cup(\psi\cap\vartheta)}=\vec{\sigma}_{(\varphi\cup\psi)\cap(\varphi\cup\vartheta}$	$\vec{\sigma}_{\varphi\cap(\psi\cup\vartheta)}=\vec{\sigma}_{(\varphi\cap\psi)\cup(\varphi\cap\vartheta}$	$\vec{\sigma}_{C0}=\vec{1}$	$\vec{\sigma}_{C1}=\vec{0}$
$\vec{\sigma}_{\varphi\cup C_{\varphi}}=1$	$\vec{\sigma}_{\varphi\cap C_{\varphi}}=0$	$\vec{\sigma}_{C(\varphi\cup\psi)}=\vec{\sigma}_{C\varphi\cup C\psi}$	$\vec{\sigma}_{C(\varphi\cap\psi)}=\vec{\sigma}_{C\varphi\cap C\psi}$
		$\vec{\sigma}_{CC\varphi}=\vec{\sigma}_{\varphi}$	

Opšti postupak za transformisanje proizvoljne fazi relacije u odgovarajući generalizovani Bulov polinom definisan je na sledeći način:

$$\varphi\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right),\psi\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)\in BA(\Omega)$$

$$(\varphi\wedge\psi)^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)=_{def}\varphi^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)\otimes\psi^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)$$

$$(\varphi\vee\psi)^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)=_{def}\varphi^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)+\psi^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)-$$
$$-\varphi^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)\otimes\psi^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)$$

$$(\neg\varphi)^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)=_{def}1-\varphi^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)$$

gde za primarne atribute važi:

$$\left(a_i\wedge a_j\right)^{\otimes}=_{def}\begin{cases}\mu_{a_i}\otimes\mu_{a_j} & i\neq j\\ \mu_{a_i} & i=j\end{cases}$$

$$\left(a_i\vee a_j\right)^{\otimes}=_{def}\mu_{a_i}+\mu_{a_j}-\left(a_i\wedge a_j\right)^{\otimes}$$

$$\left(\neg a_i\right)^{\otimes}=_{def}1-\mu_{a_i}$$

$$\left(a_i,a_j\in\Omega,\mu_{a_i},\mu_{a_j}\in[0,1]\right)$$

Izračunavanje funkcije pripadnosti neke fazi relacije zahteva da se prethodno odabere funkcija koja će igrati ulogu generalizovanog proizvoda. Pošto iz definicije ovog proizvoda sledi da on ispunjava sledeći uslov:

$$max\left(0,\mu_a(x)+\mu_b(x)-1\right)\leq\mu_a(x)\otimes\mu_b(x)\leq min\left(\mu_a(x),\mu_b(x)\right)$$

vidi se da se mogu koristiti sve *t-norme* čije su vrednosti između Lukašijevičeve norme i minimuma. Otuda je prirodno da se za generalizovani proizvod uzmu tri najčešće korišćene *t-norme* u fazi logici:

$$\mu_a(x)\otimes\mu_b(x)=min\left(\mu_a(x),\mu_b(x)\right)$$
$$\mu_a(x)\otimes\mu_b(x)=\mu_a(x)\cdot\mu_b(x)$$
$$\mu_a(x)\otimes\mu_b(x)=max\left(0,\mu_a(x)+\mu_b(x)-1\right)$$

Konkretan izbor neke od ovih formi zavisi od prirode promenljivih koje učestvuju u relaciji.

Jedan od kriterijuma može biti i korelisanost. Ukoliko promenljive nisu korelisane onda je logično da se koristi algebarski proizvod (sa značenjem „i jedna i

druga“). Međutim, ukoliko su promenljive potpuno korelisane onda je prirodno da se generalizovani proizvod formira kao minimum (samo jedna i to manja od dve korelisane promenljive). Sa druge strane kada su promenljive negativno korelisane onda se ta korelisanost najbolje izražava Lukašijevičevom formom (zbir promenljivih koji odražava njihovu zajedničku promenu).

Sa gledišta smisla relacije izvesno je da kada se vrši konjunkcija jedne promenljive sa samom sobom ili ako se poređenje vrši po istom atributu onda, budući da se radi o korelisanim promenljivima, takođe treba koristiti minimum.

Ukoliko se u jednoj relaciji nalaze i promenljive iste prirode i promenljive različite prirode onda se kao generalizovani proizvod mogu koristiti i minimum i proizvod i Lukašijevičeva forma, već u zavisnosti od uzajamnog odnosa promenljivih koje učestvuju u svakoj pojedinačnoj operaciji.

3.5.6.1. ILUSTRATIVNI PRIMER

Da bi se uočila razlika između korišćenja konvencionalne i konzistentne fazi logike posmatraće se površine odlučivanja koje se dobijaju modelovanjem iskaza:

(Ako je „*Vreme odziva*“ **I** „*Raspoloživost*“)
ILI
(Ako „*Nije raspoloživost*“ **I** „*Pouzdanost*“)
tada je „*Ocena*“

Ako se *Vreme Odziva* označi kao RT, *Raspoloživost* kao A, i *Pouzdanost* kao R, tada se posmatrani iskaz se može izraziti logičkom relacijom

$$(RT \wedge A) \vee (\neg A \wedge R)$$

Neka su dalje *Vreme Odziva* i *Raspoloživost* predstavljene fazi skupovima prikazanim na Slici 14. Pored toga, budući da se *Pouzdanost* isto kao i *Raspoloživost* izražava u procentima za njeno predstavljanje može se usvojiti fazi skup istog oblika kao i za *Raspoloživost*. Uz navedene pretpostavke konvencionalna fazi logika se može direktno primeniti na navedeni izraz pri čemu se mora odabrati *t-norma* za predstavljanje operacije "i" i *s-norma* za predstavljane operacije "ili".

Primena konzistentne fazi logike, zahteva da se dati izraz pre svega transformiše u generalizovani atomski Bulov polinom. U skladu sa izloženim opštim pristupom ova transformacija se izvodi na sledeći način:

$$
\begin{aligned}
\left((RT \wedge A) \vee (\neg A \wedge R)\right)^{\otimes} &= (RT \wedge A)^{\otimes} + (\neg A \wedge R)^{\otimes} - \left((RT \wedge A) \wedge (\neg A \wedge R)\right)^{\otimes} = \\
&= RT \otimes A + \left((1-A) \wedge R\right)^{\otimes} - (RT \wedge A)^{\otimes} \otimes (\neg A \wedge R)^{\otimes} = \\
&= RT \otimes A + (1-A) \oplus R - RT \otimes A \otimes \left((1-A) \otimes R\right) = \\
&= RT \otimes A + R - A \otimes R - RT \otimes A \otimes R + RT \otimes A \otimes A \otimes R = \\
&= RT \otimes A + R - A \otimes R
\end{aligned}
$$

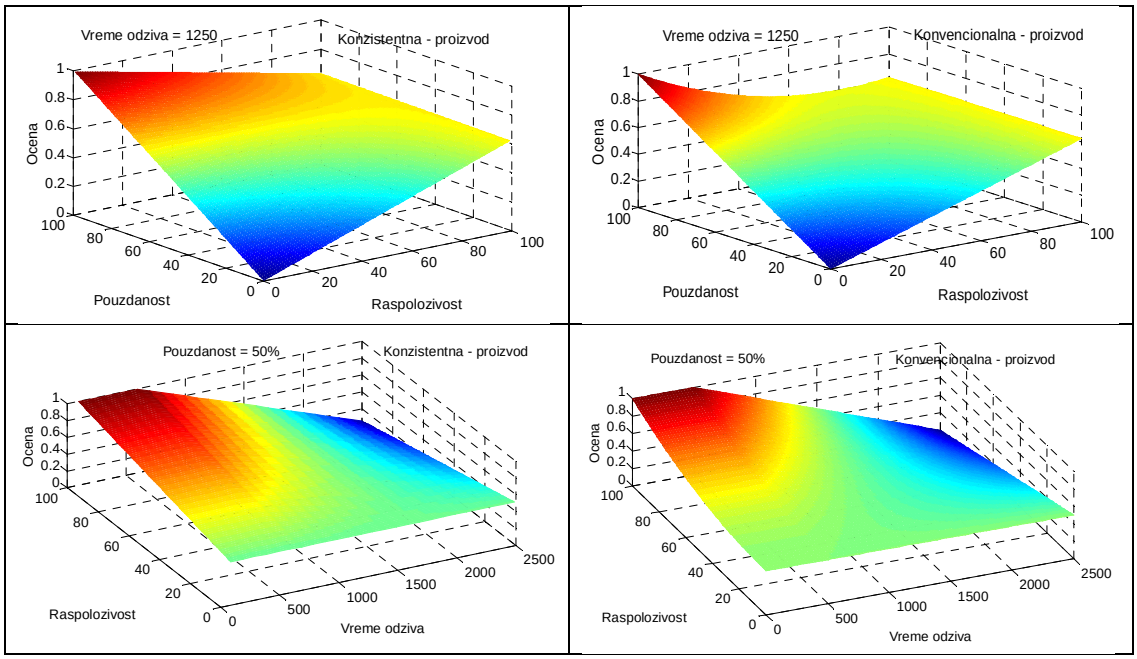
Budući da primena konzistentne fazi logike na fazi skupove izražene funkcijama pripadnosti uvek daje brojne vrednosti izraza, prirodno je i da se u cilju poređenja sa konvencionalnom fazi logikom usvoji defazifikacija preko površine izlaznog fazi skupa. Na slikama 21 i 22 su date površine odlučivanja dobijene za sledeći izbor *t*- i *s-normi* kod konvencionalne fazi logike odnosno operatora ⊗ kod konzistentne fazi logike:

- Konvencionalna fazi logika:

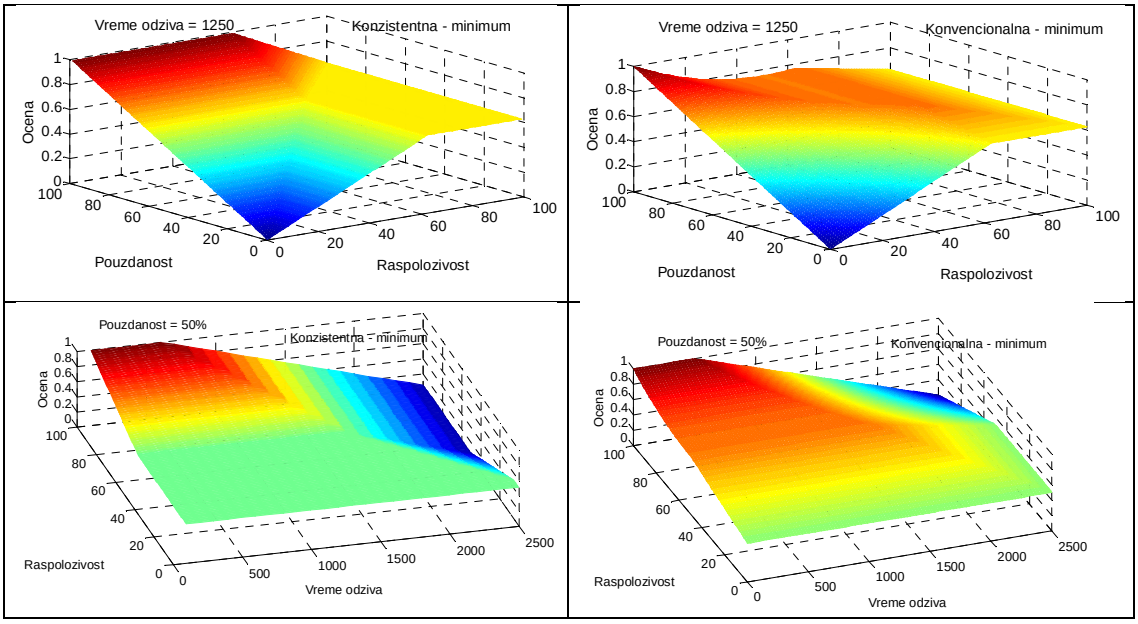
  t-norma proizvod ili minimum, *s-norma* algebarska suma

- Konzistentna fazi logika:

  generalizovani proizvod (⊗): proizvod ili minimum



Slika 21. Poređenje površina odlučivanja – proizvod



Slika 22. Poređenje površina odlučivanja - minimum

Kao što se vidi razlika između površina je posebno izražena u onim delovima u kojima, zbog oblika fazi skupa *Raspoloživost*, dolazi do izražaja nepoštovanje principa kontradikcije i isključenja trećeg.

Konačno potrebno je istaći da korišćenje konzistentne fazi logike omogućava, kao što je pokazano u (*Radojević, 2008c*), definiciju logičkog operatora agregacije kao pseudo logičke funkcije, odnosno linearne konveksne kombinacije generalizovanih Bulovih polinoma:

$$Agg_{\eta}^{\otimes}\left(\mu_{a_1},\mu_{a_2},\ldots,\mu_{a_n}\right)=\sum_{S\in P(\Omega)}\eta(S)\sum_{C\in P(\Omega\setminus S)}(-1)^{|C|}\bigotimes_{a_i\in S\cup C}\mu_{a_i}$$

U skladu sa definicijom generalizovanog Bulovog polinoma iz gornjeg izraza se vidi da mera agregacije $\eta(S)$ predstavlja strukturu logičkog operatora agregacije. To znači da je mera agregacije funkcija skupa $\eta: P(\Omega)\to[0,1]$ definisana pomoću sledećeg izraza:

$$\eta(S)=\sum_{i=1}^{m}w_i\sigma_{\varphi_i}(S),\quad\left(S\in P(\Omega),\sum_{i=1}^{m}w_i=1,w_i\geq 0,\varphi_i\in BA(\Omega),i=1,2,\ldots m\right)$$

Na osnovu datih relacija sledi da logički operator agregacije zavisi od dva faktora:

- izbor mere agregacije
- operator generalizovanog proizvoda.

Odgovarajućim izborom ovih faktora mogu se dobiti svi poznati operatori agregacije kao što je to prikazano u sledećoj tabeli, gde je za generalizovani proizvod izabran *min*:

Tabela 5. Logički operatori agregacije

Naziv	*Mera agregacije i (težina)*	*Logički operator agregacije*
Težinska suma	$\eta_{add}(S)=\sum_{i=1}^{n} w_i \mu_{\sigma_{ai}}(S), S\in P(\Omega)$ $\left(\sum_{i=1}^{n} w_i = 1, w_i \geq 0\right)$	$Agg^{min}_{\eta_{add}}\left(\mu_{a_1},\ldots,\mu_{a_n}\right)=\sum_{a_i\in\Omega} w_i\mu_{a_i}$
Aritmetička sredina	$\eta_{sred}(S)=\frac{\lvert S\rvert}{\lvert\Omega\rvert}$ $\left(w_i=\frac{1}{n}\right)$	$Agg^{min}_{\eta_{sred}}\left(\mu_{a_1},\ldots,\mu_{a_n}\right)=\frac{1}{n}\sum_{a_i\in\Omega}\mu_{a_i}$
Samo k-ti atribut	$\eta_k(S)=\begin{cases}1 & a_k\in S\\ 0 & a_k\notin S\end{cases}$ $\left(w_i=\begin{cases}1 & i=k\\ 0 & i\neq k\end{cases}\right)$	$Agg^{min}_{\eta_k}\left(\mu_{a_1},\ldots,\mu_{a_n}\right)=\mu_{a_k}$
Min vrednost atributa	$\eta_{AND}(S)=\begin{cases}1 & S=\Omega\\ 0 & S\neq\Omega\end{cases}$	$Agg^{min}_{\eta_{AND}}\left(\mu_{a_1},\ldots,\mu_{a_n}\right)=min\left\{\mu_{a_1},\ldots,\mu_{a_n}\right\}$
Max vrednost atributa	$\eta_{OR}(S)=\begin{cases}1 & S\neq 0\\ 0 & S=0\end{cases}$	$Agg^{min}_{\eta_{OR}}\left(\mu_{a_1},\ldots,\mu_{a_n}\right)=max\left\{\mu_{a_1},\ldots,\mu_{a_n}\right\}$
Uređena težinska agregacija	$\eta_{OWA}(S)=\begin{cases}0 & S=0\\ \sum_{i=1}^{m} w_i & \lvert S\rvert=m\end{cases}$ $\left(\sum_{i=1}^{n} w_i=1, w_i\geq 0\right)$	$Agg^{min}_{\eta_{OWA}}\left(\mu_{a_1},\ldots,\mu_{a_n}\right)=\sum_{i=1}^{n} w_i\mu_{a_{(i)}}$ $\mu_{a_{(1)}}\leq\mu_{a_{(2)}}\leq\ldots\leq\mu_{a_{(n)}}$
Statistika k-tog reda	$\eta_{k-ti}(S)=\begin{cases}0 & \lvert S\rvert<k\\ 0 & \lvert S\rvert\geq k\end{cases}$	$Agg^{min}_{\eta_{k-ti}}\left(\mu_{a_1},\ldots,\mu_{a_n}\right)=\mu_{a_{(k)}}$ $\mu_{a_{(1)}}\leq\mu_{a_{(2)}}\leq\ldots\leq\mu_{a_{(n)}}$
Diskretni Šokeov integral	η_{mon} bilo koja monotona mera	$Agg^{min}_{\eta_{mon}}\left(\mu_{a_1},\ldots,\mu_{a_n}\right)=C_{\eta_{mon}}\left(\mu_{a_1},\ldots,\mu_{a_n}\right)=$ $=\sum_{k=1}^{n}\left(\mu_{a_{(k)}}-\mu_{a_{(k-1)}}\right)\eta_{mon}\left(A_{(k)}\right)$ $\mu_{a_{(1)}}\leq\mu_{a_{(2)}}\leq\ldots\leq\mu_{a_{(n)}};$ $A_{(k)}=\left\{\mu_{a_{(k)}},\ldots,\mu_{a_{(n)}}\right\}$
Generalizovani diskretni Šokeov integral	η_{mon} bilo koja ne monotona mera	$Agg^{min}_{\eta}\left(\mu_{a_1},\ldots,\mu_{a_n}\right)=GC_{\eta}\left(\mu_{a_1},\ldots,\mu_{a_n}\right)=$ $=\sum_{k=1}^{n+1}\left(\mu_{a_{(k)}}-\mu_{a_{(k-1)}}\right)\eta\left(A_{(k)}\right)$ $0=\mu_{a_{(0)}}\leq\mu_{a_{(1)}}\leq\mu_{a_{(2)}}\leq\ldots\leq\mu_{a_{(n+1)}}=1;$ $A_{(k)}=\left\{\mu_{a_{(k)}},\ldots,\mu_{a_{(n)}}\right\}, A_{(n+1)}=0,$ $\left(\eta(0)\neq 0 \text{ u opštem slučaju}\right)$

Na osnovu svega izloženog može se zaključiti da fazi logika otvara mogućnost formiranja kriterijumskih funkcija preko logičkih izraza. Šta više predstavljanje QoS atributa pomoću fazi skupova omogućava da se objedine različite vrste nefunkcionalnih karakteristika nezavisno od toga da li su iskazane brojnim vrednostima ili lingvističkim promenljivima. Dodatno izborom oblika fazi skupova i njihovih granica moguće je svaki aspekt precizirati na najadekvatniji način. Međutim, kao što je izloženo, konvencionalna fazi logika pokazuje izvesne nedostatke pri defazifikaciji iskaza koji uključuju negaciju nekog od uslova. To zapravo znači da je, u slučaju kada postoji negacija, u postupku određivanje brojne vrednosti izlaznog fazi skupa (ocene) usled nepoštovanja principa kontradikcije i isključenja trećeg moguće dobiti neadekvatne vrednosti. Sa druge strane primena konzistentne fazi logike zahteva da se najpre na strukturnom nivou izvrši transformacija formulisanog logičkog izraza u generalizovani atomski Bulov polinom (GBP) da bi se tek onda u njega uvele vrednosti, što zapravo znači da u određivanju vrednosti neke fazi relacije učestvuju isključivo vrednosti atomskih elemenata čime je obezbeđeno očuvanje svih Bulovih postulata. Šta više, kako se navodi u (*Radojević, 2010; Radojević, 2013*), GBP na vrednosnom nivou preslikava određeni element Bulove algebre u njemu odgovarajuću vrednost iz realnog intervala [0,1] čime se obezbeđuje očuvanje parcijalnog poretka na vrednosnom nivou, što nije slučaj kada je reč o drugim više-vrednosnim i/ili fazi pristupima. Dakle konzistentna fazi logika omogućava da se, odgovarajućim izborom generalizovanog proizvoda, realizuju sve logičke i pseudo-logičke funkcije koje se sreću pri formulisanju kriterijumskih funkcija budući da se prema (*Radojević, 2008b*) svaka logička funkcija može jednoznačno prevesti u odgovarajući GBP primenom IBA.

Konačno, potrebno je napomenuti da logička agregacija za rezultat ima novu strukturu komponenti što nije slučaj kada je reč o SAW pristupu (*Radojević, 2008c; Radojević, 2010*). Iako bi se prema (*Figueira, &al 2005; Tzeng & Huang, 2011*) u cilju prevazilaženja problema neaditivnosti, koji je posledica interakcije koja postoji između samih kriterijuma, umesto SAW pristupa mogao koristiti Šoke-ov integral, treba istaći da je on zapravo samo specijalan slučaj IBA (*Radojević, 1999*). Dakle, budući da IBA uključuje sve logičke funkcije i sve interpolativne operatore (generalizovani proizvod) umesto samo jednog aritmetičkog interpolativnog operatora (min funkcije) može se zaključiti da ovakav pristup pruža daleko šire mogućnosti primene.

3.6. Nov model zasnovan na višekriterijumskom razlomljenom programiranju

Drugi pristup, koji nije do sada predlagan za modelovanje problema, mogao bi se primeniti u slučajevima kada ne postoje logički zahtevi vezani za QoS kriterijume ali su QoS kriterijumi heterogeni i izraženi pomoću nekoliko različitih jedinica mere. U navedenom slučaju problem bi se mogao modelovati kao diskretan problem višekriterijumskog razlomljenog programiranja (*eng. Multi-Objective Fractional Programming Problem* – MOFPP).

3.6.1. Osnovna teorijska polazišta

Generalno gledano problemi razlomljenog programiranja predstavljaju klasu optimizacionih problema kod kojih je funkcija cilja zapravo količnik (*eng. ratio*) dve funkcije:

$$\underset{x\in X}{opt}\left\{z(x)=\frac{N(x)}{D(x)}\right\} \text{ pod uslovom da je } D(x)>0, \forall x\in X.$$

pri čemu se pretpostavlja da je $X \subseteq \mathbb{R}^n$, $N : X \to \mathbb{R}$, $D : X \to \mathbb{R}$.

Ukoliko je potrebno istovremeno optimizovati (maksimizirati ili minimizirati) nekoliko različitih količnika funkcija onda navedeni problem predstavlja problem višekriterijumskog razlomljenog programiranja – MOFPP koji se može formalno može iskazati na sledeći način:

$$\underset{x\in X}{opt}\left\{z(x)=\left(\frac{N_1(x)}{D_1(x)},\frac{N_2(x)}{D_2(x)},\ldots,\frac{N_v(x)}{D_v(x)}\right)\right\}, v\geq 2$$

pod uslovom da je $D_g(x)>0, \forall x\in X, \forall g=1,\ldots,v.$

Problemi razlomljenog programiranja se u literaturi često sreću. Kako se navodi u (*Frenk & Schaible, 2005*) ovakve vrste modela se najviše primenjuju u oblasti ekonomije kada je potrebno maksimizirati efikasnost sistema iskazanu odnosom određenih tehničkih i/ili ekonomskih pokazatelja (na primer optimizacija različitih finansijskih racija, maksimizacija produktivnosti, minimizacija odnosa troškova i vremena, maksimizacija odnosa izlaza i ulaza sistema itd.) ali i u drugim oblastima kao što su na primer tehnika, numerička analiza, statistika, fizika itd. Iscrpan pregled problema koji su modelovani kao problemi razlomljenog programiranja se može naći i na primer u (*Craven, 1988; Schaible, 1995; Stancu-Minasian, 1997; Schaible, 2002;*

Stancu-Minasian, 2006; Frenk & Schaible, 2009). Pored toga razlomljeno programiranje se uspešno primenjuje i kod diskretnih problema pa čak i problema *0-1* programiranja, pregled se na primer može naći u (*Barros, 1998; Radzik, 1998; Prokopyev, 2009*). Pregledom navedene literature se takođe vidi da se ovakav pristup uspešno primenjuje i u oblasti informacionih sistema kako za probleme kod kojih se razmatraju funkcionalne karakteristike (npr. selekcija svojstava (*eng. feature*) i prepoznavanje šablona u kontekstu data mining-a i text mining-a, formiranje optimalnog logičkog izraza upita nad bazom podataka, prepoznavanje oblika, optimalna alokacije resursa za testiranje softvera itd.) tako i za optimizaciju nefunkcionalnih karakteristika (npr. određivanje kapaciteta i propusne moći komunikacionog kanala, optimizacija performansi bežičnog prenosa, izbora procesora i raspodelu zadataka po procesorima itd.)

Međutim od posebnog je značaja činjenica da se MOFP može primeniti i u slučajevima kada originalni problem zapravo ne predstavlja problem razlomljenog programiranja. Naime, u (*Frenk & Schaible, 2005*) autori navode da postoje brojni problemi u oblasti menadžmenta i donošenja odluka kod kojih se indirektno dolazi do razlomljenih modela iako originalna funkcija cilja zapravo ne uključuje količnike i daju nekoliko primera takvih pristupa.

Konkretno kada je u pitanju selekcija servisa za kompoziciju, iako autori to ne naglašavaju, indirektna primena razlomljenog programiranja se može naći na primer u (*Canfora & al, 2004; Canfora & al, Jul 2005*) gde se predlaže pristup modelovanju problema pomoću funkcije pogodnosti (*eng. fitness*) koja je definisana kao količnik u čijem se brojiocu nalazi težinska suma QoS atributa koje je potrebno maksimizirati dok se u imeniocu nalazi težinska suma QoS atributa koji se minimiziraju. Pri tome u (*Canfora & al, Jun 2005*) autori proširuju navedeni model integrisanjem ograničenja u funkciju pogodnosti (u kojoj su zamenili brojilac i imenilac) tj. dodavanjem otežane kaznene funkcije zasnovane na udaljenosti od zadatih vrednosti ograničenja, u skladu sa pristupom predloženim u (*Fang, 1994*). Međutim treba napomenuti da autori u sledećim radovima (*Canfora & al, 2006; Canfora & al, 2008*) napuštaju ovakav pristup i normalizacijom svode sve QoS atribute (i one koji se maksimiziraju i one koji se minimiziraju) na isti raspon vrednosti te funkciju pogodnosti umesto količnika definišu kao težinsku sumu agregiranih QoS vrednosti sa dodatom težinskom kaznenom

funkcijom pomoću koje se u funkciju pogodnosti uključuju ograničenja (*Canfora & al, 2006*) dok se u (*Canfora & al, 2008*) funkcija pogodnosti dobija množenjem težinske sume agregiranih QoS vrednosti sa dopunom do jedinice težinske kaznene funkcije.

Model problema koji autori predlažu u (*Jaeger & Muhl, 2007*) takođe podrazumeva formiranje funkcije pogodnosti pri čemu autori navode da se data funkcija pogodnosti zasniva na pristupu predloženom u (*Canfora & al, Jun 2005*). Međutim, formula koja je u radu data se zapravo razlikuje od njihove u nekoliko pogleda: (1) brojiocem se predstavljaju atributi koji se maksimiziraju a imeniocem se predstavljaju atributi koji se minimiziraju, (2) brojilac i imenilac količnika predstavljaju proizvode a ne težinske sume QoS atributa koji se maksimiziraju odnosno minimiziraju, respektivno, (3) QoS atributima se ne dodeljuju težine već se smatra da su one neutralne i (4) količnik se množi kaznenim faktorom. U daljem tekstu autori razmatraju i druge oblike funkcije pogodnosti dobijene skaliranjem njenih vrednosti na opseg [0,1] ili množenjem vrednosti funkcije pogodnosti određenim faktorom kako bi se raspon njenih vrednosti povećao. Za svako od rešenja primenjuje se SAW tehnika da bi se izračunala normalizovana QoS vrednost celokupne kompozicije agregacijom vrednosti izabranih QoS atributa. Treba napomenuti da u izloženom primeru autori predviđaju postojanje samo jednog ograničenja.

3.6.2. Predloženi pristup modelovanju

Potrebno je istaći da se u prethodno navedenim pristupima formira samo jedan količnik što zahteva da se svakom od QoS atributa dodeli težina kojom će se izraziti preference korisnika. Sa druge strane, osnovna ideja ovog pristupa se sastoji u sparivanju QoS kriterijuma sa ciljem da se optimizuju njihovi količnici čime se početni problem zapravo prevodi u MOFPP. Dakle, iako početni problem zapravo ne predstavlja problem razlomljenog programiranja, navedeni pristup omogućava donosiocu odluke da uparivanjem kriterijuma iskaže kompromise koji su za njega prihvatljivi.

QoS kriterijumi bi se sparivali uzimajući u obzir njihovu prirodu (jedinicu mere i vrstu optimizacije – minimizacija ili maksimizacija) čime bi se dobile razlomljene funkcije cilja koje imaju iste jedinice mere i koje se optimizuju u istom smeru. Treba napomenuti da je tako formirani MOFPP po svojoj prirodi diskretan budući da je dopustivi skup X konačan.

Prednosti predloženog pristupa nad SAW pristupom proizilaze iz činjenice da MOFPP ne podrazumeva formiranje jedinstvene funkcije cilja. Kao što je prethodno izloženo da bi se u SAW pristupu omogućilo agregiranje vrednosti različitih QoS atributa u jedinstvenu funkciju cilja neophodno ih je prethodno normalizovati budući da mogu biti izraženi pomoću različitih jedinica mera ili čak i istih ali sa različitim rasponima vrednosti, a zatim svakom od njih dodeliti i odgovarajuće težine kojima se izražavaju preference korisnika u pogledu datih atributa. Za razliku od toga predloženi MOFP pristup ne zahteva normalizaciju podataka niti dodeljivanje arbitrarnih težina QoS kriterijumima. Otuda predloženi pristup ima prednost u onim slučajevima kada se normalizacijom remeti originalni odnos između vrednosti QoS atributa.

Konačno važno je uočiti da formirani model u opštem slučaju može biti nelinearan (ako se u brojiocima ili imeniocima količnika nađu QoS atributi koji se agregiraju pomoću nelinearnih funkcija), što znači da predloženi pristup ne nameće ograničenje u pogledu linearnosti funkcija za agregiranje QoS atributa. U tom kontekstu je od posebnog značaja činjenica da se ne zahteva normalizacija.

Međutim ukoliko se, u cilju lakšeg rešavanja, želi primeniti neka metoda koja je razvijena za probleme linearnog razlomljenog programiranja nelinearni agregirani QoS kriterijumi se mogu primenom odgovarajućih transformacija (na primer, logaritmovanjem) prevesti u linearne kriterijume čime se dobija MOLFPP (*eng. Multi-Objective Linear Fractional Programming Problem*).

Problem zapravo predstavlja MOLFPP ako su i brojioci i imenioci afine funkcije (tj. zbir linearne funkcije i neke konstante). MOLFPP se može formalno iskazati na sledeći način:

$$\underset{x\in X}{opt} \left\{ z(x) = \left(\frac{N_1(x)}{D_1(x)}, \frac{N_2(x)}{D_2(x)}, \ldots, \frac{N_v(x)}{D_v(x)} \right) \right\}, v \geq 2$$

pri čemu je: $N_g(x) = c_g x + \alpha_g, \quad c_g \in \mathbb{R}^n, \alpha_g \in \mathbb{R}, \forall g = 1, \ldots, v,$

$$D_g(x) = d_g x + \beta_g, \quad d_g \in \mathbb{R}^n, \beta_g \in \mathbb{R}, \forall g = 1, \ldots, v,$$

$$d_g x + \beta_g > 0, \qquad \forall g = 1, \ldots, v, \forall x \in X,$$

Treba napomenuti da MOLFPP zahteva i da sama ograničenja, koja određuju dopustivi skup *X*, budu linearna (odnosno afine funkcije).

3.6.2.1.ILUSTRATIVNI PRIMER

Ukoliko bi se razmatrali QoS atributi Raspoloživost (eng. *Availability*), Pouzdanost (eng. *Reliability*), Vreme Odziva (*eng. Response Time*) i Kašnjenje (*eng. Latency*), mogla bi se formirati dva količnika. Agregirana *Raspoloživost* (q_1) i agregirano *Vreme Odziva* (q_2) bi sačinjavali prvi količnik dok bi odnos agregirane *Pouzdanosti* (q_3) i agregiranog *Kašnjenja* (q_4) bio predstavljen drugim količnikom. Budući da je *Vreme Odziva* i *Kašnjenje* potrebno minimizirati dok je *Raspoloživost* i *Pouzdanost* potrebno maksimizirati prva dva kriterijuma su izabrana za imenioce, dok su druga dva kriterijuma izabrana za brojioce datih razlomaka. Oba dobijena razlomljena kriterijuma će u tom slučaju imati istu jedinicu mere (procenat/milisekund) i oba će predstavljati kriterijume koje je potrebno maksimizirati. Sada se odgovarajući matematički MOFP model može iskazati na sledeći način:

$$\max_{x \in X} \left\{ z(x) = \left(\frac{q^1(x)}{q^2(x)}, \frac{q^3(x)}{q^4(x)} \right) \right\}$$

pri ograničenjima: $q^k(x) \le Q^k, \quad k = 1,..., 4,$

$$\sum_{j=1}^{p_i} x_{ij} = 1, \qquad i = 1,..., m,$$

$$x_{ij} \in \{0,1\}, \qquad j = 1,..., p_i;\ i = 1,..., m.$$

Uključivanje drugih atributa u razmatranje rezultovalo bi samo povećanjem broja razlomaka u novoj funkciji cilja.

Pošto se *Raspoloživost* i *Pouzdanost* agregiraju pomoću proizvoda navedeni model predstavlja primer nelinearnog diskretnog MOFPP.

Ukoliko se date funkcije logaritmovanjem prevedu u linearne funkcije[1]:

$$\log\left(\prod_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^k x_{ij} \right) = \sum_{i=1}^{m} \sum_{j=1}^{p_i} \left(\log\left(q_{ij}^k \right) \right) x_{ij}, \quad k = 1,3,$$

onda se odgovarajući diskretan MOLFPP može formulisati na sledeći način:

[1] Navedena relacija je tačna budući da, za bilo koje dopustivo rešenje, tačno jedno x_{ij} ima vrednost *1* dok sve ostale promenljive imaju vrednost *0*, za svako $i=1,\ldots,m$.

$$\max_{x \in X} \left\{ z(x) = \left(\frac{\sum_{i=1}^{m} \sum_{j=1}^{p_i} \left(\log\left(q_{ij}^{1} \right) \right) x_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^{2} x_{ij}}, \frac{\sum_{i=1}^{m} \sum_{j=1}^{p_i} \left(\log\left(q_{ij}^{3} \right) \right) x_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^{4} x_{ij}} \right) \right\}$$

pri ograničenjima: $\sum_{i=1}^{m} \sum_{j=1}^{p_i} \left(\log\left(q_{ij}^{k} \right) \right) x_{ij} \geq \log\left(Q^{k} \right), \quad k = 1,3,$

$$\sum_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^{k} x_{ij} \leq Q^{k}, \qquad k = 2,4,$$

$$\sum_{j=1}^{p_i} x_{ij} = 1, \qquad i = 1,\ldots, m,$$

$$x_{ij} \in \{0,1\}, \qquad j = 1,\ldots, p_i;\ i = 1,\ldots, m.$$

U izloženom primeru, s obzirom na to da su funkcije q_2 i q_4 linearne i pozitivne, imenioce nije bilo potrebno dodatno transformisati. Ipak, u opštem slučaju, i u imeniocima se mogu naći funkcije koje bi se logaritmovanjem transformisale u linearne funkcije. Međutim, imajući u vidu zahtev da imenioci razlomaka moraju biti pozitivne funkcije, a kako logaritamske funkcije ne ispunjavaju ovaj uslov, potrebno je tako dobijene funkcije skalirati nekom vrednošću (β) da bi se obezbedila njihova pozitivnost[1]:

$$D_g = \sum_{i=1}^{m} \sum_{j=1}^{p_i} \left(\log\left(q_{ij}^{k} \right) \right) x_{ij} + \beta_g > 0, \quad \forall x \in X.$$

Vrednost β_g bi se pri tome mogla odrediti na sledeći način:

$$\beta_g = \omega + \sum_{i=1}^{m} \left| \min_{j=1,\ldots,p_i} \log\left(q_{ij}^{k} \right) \right|$$

gde je ω parametar koji je potrebno podesiti kako bi se osiguralo da najmanja moguća suma uvek bude striktno veća od najmanje moguće vrednosti sume logaritama date kriterijumske (agregirajuće) funkcije (npr. $\omega = 1$).

[1] Poznato je da dodavanje konstante funkciji cilja ne može promeniti optimalno rešenje početne funkcije.

4. Nova metoda za selekciju pojedinačnih servisa

4.1. Metoda Analitičkog Hijerarhijskog Procesa (AHP)

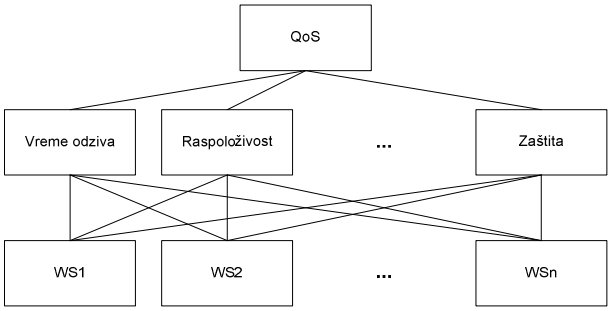
Kada je u pitanju selekcija pojedinačnih servisa za dati funkcionalni zahtev problem se, kao što je izloženo u Poglavlju 3.1., može posmatrati kao problem višekriterijumskog odlučivanja (MCDM) koji podrazumeva izbor jedne alternative (iz mnoštva raspoloživih alternativa) na osnovu većeg broja, često konfliktnih kriterijuma, pri čemu je potrebno izabrati onu alternativu koja na najbolji način ispunjava ceo skup postavljenih zahteva, odnosno onu koja je najbolja sa stanovišta svih relevantnih kriterijuma. Imajući u vidu mane ovog pristupa koji, kao što je izloženo u Poglavlju 3.4, ne omogućava da se uzme u obzir interakcija koja može postojati između samih kriterijuma u literaturi su predložene nove metode, kao što je na primer metoda hijerarhijskog analitičkog procesa (*eng. Analytic Hierarchy Process – AHP*), i novi alati kao što su fazi logika i teorija operatora agregacije (*eng. theory of aggregation operators*) kako bi se MCDM pristupi unapredili (*Torfi & al, 2010*). Sa druge strane, ukazano je i na problem vezan za određivanje samih težina ukoliko je potrebno doneti odluku na osnovu velikog broja kriterijuma, budući da je u tom slučaju veoma teško odrediti koliko je neki kriterijum značajniji od drugih te se najčešće se koristi aproksimativni pristup. Kako se navodi u (*Tzeng & Huang, 2011*) AHP je jedna od metoda koja se može primeniti za određivanje težina relevantnih kriterijuma, umesto da se one arbitrarno dodeljuju.

Stoga se u ovom radu za rešavanje navedenog problema predlaže proširenje AHP metode kao jedne od najčešće korišćenih metoda višekriterijumskog odlučivanja (*Saaty, 1988*). Pri tome treba napomenuti da, iako je u opštem slučaju moguće definisati ograničenja i kada je reč o selekciji pojedinačnih servisa takva ograničenja bi zapravo služila da se unapred odbace oni servisi koji ih ne zadovoljavaju.

AHP metoda predložena u (*Saaty, 1980*) polazi od toga su problemi odlučivanja u realnosti često veoma složeni te da je u cilju savladavanja te složenosti najpre neophodno razložiti takav problem na skup hijerarhijski organizovanih potproblema koji su lakši za rešavanje. Drugim rečima, u AHP metodi dekompozicija problema zapravo podrazumeva da je najpre potrebno identifikovati sve faktore koji su relevantni za donošenje odluke a zatim ih struktuirati u vidu hijerarhije u kojoj su elementi nekog

nivoa zavisni od (odnosno pod uticajem) elemenata na prethodnom nivou (tj. nivou koji je neposredno iznad datog) dok su elementi koji se nalaze na istom nivou međusobno nezavisni. U opštem slučaju to znači da je najpre potrebno definisati ciljeve koji se žele postići, zatim uspostaviti kriterijume za evaluaciju dostizanja sveukupnog cilja problema, koji će omogućiti da se izabere ona alternativa koja na najbolji način ispunjava ceo skup postavljenih ciljeva (odnosno onu koja je najbolja sa stanovišta svih relevantnih kriterijuma), i na kraju utvrditi koje su alternative na raspolaganju donosiocu odluke. Na ovaj način se zapravo uspostavlja hijerarhijski model (kao osnova za donošenje odluka) koji se sastoji od nekoliko uzastopnih nivoa: cilja (na vrhu), kriterijuma i potkriterijuma (u sredini i to krećući se postepeno od opštijih prema konkretnijim) i raspoloživih alternativa (na dnu).

Konkretno, za problem selekcije pojedinačnih web servisa, kao cilj se uzima izbor onog servisa koji će obezbediti najbolju moguću ukupnu QoS vrednost, za kriterijume se mogu uzeti relevantni QoS atributi dok će alternative zapravo predstavljati raspoložive web servise (Slika 23). Treba napomenuti da, u opštem slučaju, navedeni model može sadržati složenije kriterijume kao i potkriterijume o čemu će biti više reči nešto kasnije.



Slika 23. Jednostavni AHP model za problem selekcije pojedinačnih web servisa

Osnovna ideja AHP metode se zatim ogleda u vrednovanju odnosno ocenjivanju samih elemenata, nivo po nivo, što zapravo podrazumeva da se svakom od elemenata u hijerarhiji dodeli odgovarajući težinski koeficijent kojim se izražava relativni značaj (*eng. relative priority* ili *local priority*) datog elementa u odnosu na element koji se nalazi neposredno iznad njega u hijerhiji. Na kraju se sve dobijene vrednosti agregiraju kako bi se odredio globalni značaj (*eng. global priority*) svake od alternativa u odnosu na sveukupni cilj na osnovu koga se konačno mogu rangirati raspoložive alternative.

Međutim, kao što je diskutovano u poglavlju 3.4, određivanje težina nije jednostavan zadatak, imajući pre svega u vidu to da atributi (pomoću kojih su predstavljeni elementi datog problema odlučivanja) mogu, u opštem slučaju, biti veoma heterogeni i izraženi pomoću različitih, često nesamerljivih, jedinica mera ili čak i istih ali sa različitim rasponima vrednosti. Prepoznajući činjenicu da se različite skale ne mogu kombinovati Sati je u (*Saaty, 1980*) predložio koncept prioriteta kao apstraktnu jedinicu mere koja će biti validna po svim skalama. Prema (*Saaty, 1990*) prioriteti se definišu u skladu sa skalom preferenci koje ukazuju na to u kojoj meri je određeni element značajniji (odnosno dominantniji) od nekog drugog elementa u odnosu na određeno svojstvo sa čijeg stanovišta se ti elementi porede. Navedena skala, koja se naziva *Satijeva skala 9 tačaka*, zapravo omogućava da se verbalno iskazane preference donosioca odluke prevedu u odgovarajuće numeričke vrednosti. Naime, ova skala pretpostavlja da se odnos između dva elementa može okarakterisati rasponom iskaza koji se kreću od toga da su dva elementa jednako značajna, ili da se jedan element neznatno, umereno, znatno ili izrazito preferira u odnosu na drugi pri čemu datim iskazima odgovaraju numeričke vrednosti (*1, 3, 5, 7* i *9*, respektivno). Pored toga, navedena skala obezbeđuje i međuvrednosti (*2, 4, 6* i *8*) ukoliko je potrebno finije definisati preference. Stoga je jedna od prednosti AHP metode to što ona omogućava donošenje odluka kako na osnovu kvantitativnih podatka (i to bez obzira na njihovu jedinicu mere ne zahtevajući pri tome normalizaciju vrednosti) tako i na osnovu kvalitativnih podataka.

Vrednovanje elemenata se vrši poređenjem u parovima (*eng. pairwise comparisons*) pri čemu će donosilac odluke, razmatrajući svaki par elemenata posebno, dati svoj sud o tome koliko je jedan element značajniji od drugog sa stanovišta elementa koji je neposredno iznad njih u hijerarhiji.

Rezultati ovih poređenja se prikazuju u vidu matrica poređenja (*eng. comparison matrices*) za svaki od nivoa hijerarhije. Pri tome za svaki element neke matrice $A=(a_{ij})$, $i, j=1,\ldots n$, važi sledeća relacija: $a_{ji}=1/a_{ij}$.

Takođe je neophodno proveriti da li je donosilac odluke bio logički konzistentan prilikom poređenja budući da, kako se navodi u (*Saaty, 1990*), nekonzistentnost prilikom vrednovanja može dovesti do nekonzistentnih zaključaka. Konzistentnost sudova donosioca odluke se može evaluirati poređenjem glavnog sopstvenog vektora

(λ_{max}) date matrice poređenja i njenog traga (n). Naime, u idealnom slučaju, ako je donosilac odluke bio apsolutno konzistentan u svojim procenama, glavni desni sopstveni vektor date matrice poređenja A će biti jednak njenom tragu ($\lambda_{max}=n$). U suprotnom, stepen konzistentnost njegovih sudova se može evaluirati pomoću koeficijenta konzistentnosti (*eng. Consistency Ratio – CR*) koji zapravo pokazuje koliko su dati sudovi konzistentni u poređenju sa velikim uzorcima potpuno nasumičnih sudova. Da bi se odredio ovaj koeficijent najpre je potrebno izračunati indeks konzistentnosti (*eng. Consistency Index – CI*) kao meru odstupanja λ_{max} od traga date matrice koja, prema (*Saaty, 1990*), zapravo predstavlja negativni prosek svih preostalih korena karakterističnog polinoma date matrice:

$$CI = \frac{\lambda_{\max} - n}{n-1}.$$

Zatim se CR dobija poređenjem ovog indeksa sa indeksom nasumične konzistentnosti (*eng. Random Consistency Index – RI*) koji zapravo predstavlja prosečnu vrednost indeksa konzistentnosti nad velikim brojem matrica (istog reda) čiji su elementi nasumično izabrani iz Satijeve skale. Tabela 6 preuzeta iz (*Saaty, 1990*) prikazuje vrednosti ovog indeksa za matrice različitih veličina:

Tabela 6. Vrednosti indeksa nasumične konzistentnosti (RI)

Veličina matrice	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
RI	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49

Sada se CR može izračunati na sledeći način:

$$CR = \frac{CI}{RI}.$$

Ukoliko je CR jednak *0* to znači da je donosilac odluke bio savršeno konzistentan prilikom vrednovanja. Prema (*Saaty, 1990*) vrednost CR ispod 10% se smatra prihvatljivom dok vrednosti CR preko 10% ukazuju na odsustvo konzistentnosti. Iako se u praksi ponekad dozvoljavaju i vrednosti veće od 10% u opštem slučaju će se smatrati da je skup sudova suviše nekonzistentan da bi bio pouzdan te da ih je neophodno revidirati. Pri tome treba napomenuti da odsustvo konzistentnosti ne mora nužno biti rezultat loše subjektivne procene donosioca odluke već može biti i rezultat

objektivnih okolnosti kao što su na primer znatna odstupanja u vrednostima samih elementima.

Relativni značaj (odnosno lokalni prioriteti) elemenata se zatim mogu dobiti iz obradom ovih matrica što se u literaturi najčešće svodi na određivanje normalizovanih glavnih desnih sopstvenih vektora datih matrica (*Saaty, 1980*).

Globalni prioriteti elemenata u odnosu sa sveukupni cilj se zatim dobijaju agregiranjem lokalnih prioriteta svakog elementa hijerarhije po svim putanjama koje vode od date alternative ka cilju. U literaturi su prisutni brojni pristupi koji se mogu primeniti za agregaciju na primer aditivne metode agregacije (*Saaty, 1980*) ili multiplikativne metode agregacije (*Lootsma, 1993; Barzilai & Lootsma, 1997*), pri čemu se detaljni pregled različitih pristupa može naći u (*Ishizaka & Labib, 2011*). Treba istaći da se u literaturi najčešće primenjuje aditivna agregacija.

Dakle, klasična AHP metoda (za pojednostavljeni problem koji podrazumeva postojanje samo tri nivoa u hijerarhiji: ciljeva, kriterijuma i alternativa) sastoji iz sledećih koraka:

1. Definisanje problema odnosno formiranje hijerarhijskog modela
2. Određivanje relativnog značaja (težina) alternativa u odnosu na kriterijume.
3. Određivanje relativnog značaja (težina) kriterijuma u odnosu na cilj.
4. Izračunavanje globalnih prioriteta čime se omogućava poređenje i rangiranje alternativa.

4.2. Proširenje AHP metode fazi logikom

U ovom radu za rešavanje problema selekcije pojedinačnih servisa predlaže proširenje AHP metode koje će omogućiti rešavanje problema modelovanog pomoću predloženog pristupa primene konzistentne fazi logike koja omogućava da se različiti zahtevi u pogledu kriterijuma, koji u opštem slučaju mogu biti logičke funkcije, integrišu u jedinstven globalni cilj.

Naime, klasična AHP metoda i njena matematička podloga, kao što se može videti iz prethodnog potpoglavlja, podrazumevaju da su elementi koji se nalaze na istom nivou u hijerarhiji međusobno nezavisni, te zapravo ne omogućava da se cilj i/ili kriterijumi definišu kao logički izrazi koja uzimaju u obzir međuzavisnost elemenata. Iako metoda Analitičkog Mrežnog Procesa (*eng. Analytical Network Process – ANP*) (*Saaty, 2004*) omogućava specificiranje kriterijuma koji su međusobno povezani ona zapravo ne omogućava da se na adekvatan način predstave složeni verbalni zahtevi donosioca odluke, posebno kada se imaju u vidu kompromisi koje je donosilac odluke spreman da prihvati. Kako se navodi u (*Chin & al, 2009*) ANP jedino obezbeđuje da se odnosi između različitih elemenata izraze putem relativnih težina koje se generišu poređenjem po parovima i konvergencijom tzv. „super-matrice“, te ne omogućava da se uticaj između samih elemenata kvantitativno niti eksplicitno iskaže. Na postojanje lingvističkih problema se takođe ukazuje i u (*Wolfslehner & al, 2005*) gde autori navode da semantika koja se koristi prilikom poređenja po parovima (tj. pitanje koliko je značajniji dati element) zapravo često ne omogućava da se prilikom poređenja uzmu u obzir složeniji odnosi koji mogu postojati između elemenata u mreži.

Fazi odlučivanje podrazumeva situacije u kojima vrednosti, težine ali i kriterijumi i ciljevi mogu biti fazi skupovi. Većina pristupa u kojima se razmatra korišćenje fazi logike u klasičnom AHP pristupu tzv. fazi AHP (*eng. Fuzzy AHP*) (*Van Laarhoven & al, 1983; Chang, 1996; Kahraman & al, 1998; Buckley & al, 2001; Torfi & al, 2010*) samo omogućava donosiocima odluka da adekvatnije izraze svoje preference prilikom poređenja po parovima budući da se iskustvo i ekspertiza donosioca odluke u opštem slučaju može bolje predstaviti verbalnim iskazima nego numeričkim rangiranjima.

Međutim, osnovni nedostatak AHP metode pa i fazi AHP metode se ogleda u tome što agregacija najčešće postiže težinskom sumom koeficijenata koja je aditivna i

često neadekvatna. Naime, kao što je izloženo u Poglavlju 3.4 nedostatak metode težinskih koeficijenata je to što ona podrazumeva nezavisnost samih elemenata te odražava samo linearne veze među njima. Međutim, kriterijumi od kojih zavisi odluka su često međusobno povezani ili kontradiktorni. Dakle postavlja se pitanje na koji način se može omogućiti donosiocu odluke da definiše cilj kao logički zahtev koji odražava kompromise koje je spreman da prihvati. Dodatni izazov predstavlja situacija u kojoj značaj samih kriterijuma varira u zavisnosti od ostvarenih vrednosti drugih kriterijuma tj. u kome se lošija vrednost po nekom od kriterijuma zapravo može nadomestiti nekim drugom kriterijumom. Ukoliko se cilj ili kriterijumi žele definisati kao logički zahtevi bilo bi nemoguće formirati odgovarajuće matrice poređenja koje bi na adekvatan način odrazile ovaj zahtev.

Stoga je predloženo proširenje AHP metode koje će omogućiti da se uzmu u obzir definisane logičke relacije. Drugim rečima, za određivanje značaja kriterijuma u odnosu na cilj, umesto matrica poređenja, predlaže korišćenje logičke funkcija. Drugim rečima, kriterijumi bi se kombinovali u jedinstveni globalni kriterijum primenom logičke funkcije kao operatora logičke agregacije. Predloženi pristup se može na različiti načine inkorporirati u AHP metodu, odnosno može se primeniti na različitim nivoima hijerarhije, što će biti ilustrovano primerima u sledećem poglavlju.

Treba istaći da u (*Shuai, 2009*) autor takođe prepoznaje potrebu da se omogući uzimanje u obzir moguće zavisnosti između elemenata i predlaže pristup zasnovan na primeni ne-aditivnih fazi mera i fazi integrala za određivanje značaja kriterijuma i samim tim određivanja ukupnog značaja alternativa. Kako je istaknuto u Potpoglavlju 3.5.6.1 IBA uključuje sve logičke funkcije i sve interpolativne operatore, te pristup koji se u ovom radu predlaže pruža daleko šire mogućnosti primene.

Generalno gledano AHP metoda bi se na ovaj način mogla proširiti primenom bilo klasične bilo konzistentne fazi logike. Međutim, kao što je izloženo u Potpoglavlju 3.5.6.1, njihova primena ne mora voditi istim rezultatima. Konačno, imajući u vidu da se Satijeva skala zapravo može posmatrati kao jedna specifična vrsta fazifikacije, pri čemu je fazi skup definisan kao diskretan sa ukupno 9 komponenti, u ovom radu, se radi pojednostavljena prikaza predložene metode, neće razmatrati fazi AHP metoda. Imajući u vidu da predloženi pristup predstavlja opšti postupak, koji zapravo ne zavisi od toga na koji način će korisnik izraziti svoje preference, očigledno je da se predloženim

pristupom može proširiti bilo klasična AHP metoda bilo fazi AHP metoda kod koje su preference i/ili same QoS vrednosti zadate pomoću fazi skupova.

Predloženi opšti pristup će, u narednim potpoglavljima, biti ilustrovan na nekoliko različitih primera problema selekcije pojedinačnih servisa koji je modelovan pomoću predloženog pristupa zasnovanog na primeni konzistentne fazi logike pri čemu su vrednosti relevantnih atributa date u sledećoj tabeli:

Tabela 7. QoS vrednost raspoloživih servisa

	Vreme odziva	*Raspoloživost*	*Pouzdanost*	*Zaštita*	*Enkripcija*
WS_1	2000 *ms*	93%	71%	Dovoljna	Da
WS_2	1000 *ms*	98%	75%	Odlična	Ne
WS_3	1700 *ms*	96%	78%	Slaba	Da

U prvom primeru se razmatra jednostavni slučaj kada je samo cilj definisan kao logički izraz. Pored toga u ovom primeru će se pokazati da primena klasične fazi logike u određenim slučajevima i može ka izboru lošije alternative.

Drugi primer podrazumeva nešto složeniju situaciju u kojoj su i kriterijumi i potkriterijumi takođe logički izrazi. Pri tome će, za ovaj primer dva različita pristupa za određivanje globalnih prioriteta biti prikazana i upoređena (SAW i logička agregacija). Pokazaće se da primena logičke agregacije za određivanje globalnih prioriteta može dovesti do drugačijeg rangiranja alternativa u odnosu na klasični SAW pristup imajući u vidu da ona za razliku od SAW pristupa ima za rezultat novu strukturu komponenti, kao što je izloženo u Potpoglavlju 3.5.6.1.

Konačno u trećem primeru se pokazuje da se predloženi pristup može proširiti u slučajevima kada donosilac odluke želi da donese odluku na osnovu više različitih zahteva vezanih za date kriterijume. Ovo proširenje se zasniva na upotrebi pseudo-logičke agregacije kako bi se omogućilo poređenje i rangiranje alternativa uzimajući u obzir različite zahteve. Kao što je izloženo u Poglavlju 3.5.6.1 pseudo-logička agregacija omogućava kreiranje težinske sume parcijalnih zahteva, koji, u opštem slučaju, mogu biti logičke funkcije. U izloženom primeru ilustrovana je situacija u kojoj bi donosilac odluke želeo da donese odluku i na osnovu rangiranja dobijenog klasičnom AHP metodom i na osnovu rangiranja dobijenog primenom predloženim proširenjem AHP metode. Međutim treba napomenuti da je predloženo rešenje zapravo nezavisno od načina na koji će se vršiti samo rangiranje, odnosno da se može primeniti i na zahteve koji podrazumevaju da je rangiranje izvršeno i na neki drugačiji način.

Na kraju treba istaći da se u literaturi predlažu brojni pristupi za selekciju pojedinačnih servisa na osnovu nefunkcionalnih karakteristika na primer Fazi MCDM (*Fuzzy Multi-Criteria Decision Analysis – Fazi MCDM*) se predlaže u (*Tong & Zhang, 2006; Xiong & Fan, 2007*; *Sora & al, 2009*) ili PROMETHEE[1] (*Seo & al, 2005*). U (*Kattepur & al, 2011*) se primenjuje AHP metoda za određivanje težina QoS kriterijuma. U (*Chen & al, 2003*) se primenuje najpre Fazi AHP za određivanje relativnih težina QoS kriterijuma a zatim Fazi TOPSIS (*Fuzzy Technique for Order Preference by Similarity to Ideal Solution – FTOPSIS*) za rangiranje samih alternativa. Pristupi predstavljeni u (*Godse & al, 2008; Garg & al, 2013*) podrazumevaju primenu klasične AHP metode te zapravo ne uzimaju u obzir korelaciju koja može postojati između atributa kao ni situaciju u kojoj korisnik može želeti da lošiju vrednost nekog QoS atributa nadomesti drugim.

Među pristupima koji omogućavaju da se uzme u obzir odnos koji može postojati između relevantnih kriterijuma mogu se izdvojiti sledeći radovi. U (*Herssens & al, Januar, 2008*) se predlaže Fazi MCDA (*Fuzzy Multi-Criteria Decision Analysis*) pristup koji podrazumeva formiranje fazi referentnih skupova na osnovu kojih se vrši rangiranje alternativa. Referentni skup se pravi na osnovu preferenci korisnika i informacija dobijenih od pružaoca servisa o međusobnoj zavisnosti QoS atributa. Pristup predložen u (*Herssens & al, Jun, 2008*) predlažu jezik koji omogućava da se definišu pravila u pogledu prioriteta QoS kriterijuma a zatim se primenjuje PROMETHEE metoda. U (*Tran & al, 2009*) autori definišu ontologiju koja, između ostalog, omogućava i da se opišu odnosi koji može postojati između QoS atributa a zatim se na osnovu date ontologije formira hijerarhijski model problem. U (*Karim & al, 2011*) se takođe prepoznaje potreba za uzimanjem u obzir zavisnosti koja može postojati između QoS kriterijuma te se predlaže pristup zasnovan na primeni ANP i PROMETHE metoda.

Međutim, nedostatak ovih pristupa se ogleda u tome što oni ne omogućava da se formulišu složeni logički zahtevi u kojima značaj određenog kriterijuma varira u zavisnosti od ostvarenih vrednosti nekih drugih kriterijuma.

[1] Preference Ranking Organization Method for Enrichment Evaluation

4.3. Ilustrativni primeri

4.3.1. Cilj definisan kao logički zahtev (Primer 1.)

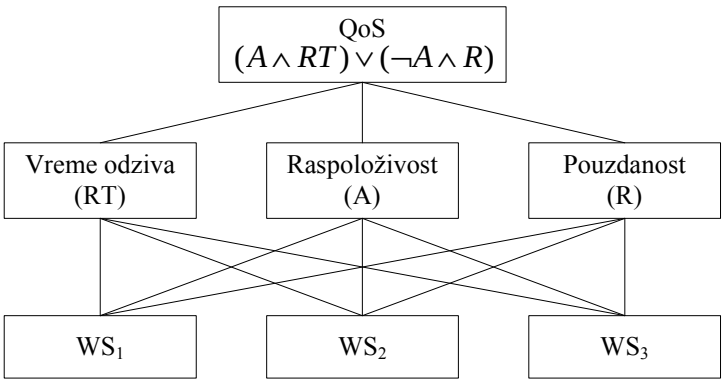
Prvi primer predstavlja jednostavni problem selekcije servisa kod koga je samo cilj definisan kao logički zahtev. U cilju ilustracije predloženog pristupa pretpostaviće se da su na raspolaganju tri različita web servisa (*WS1*, *WS2* i *WS3*) i da odluka zavisi od tri kriterijuma (za ovaj primer su kao kriterijumi od značaja izdvojeni sledeći QoS atributi: *Vreme odziva* – **RT**, *Raspoloživost* – **A** i *Pouzdanost* – **R**). Prilikom definisanja cilja pošlo se od činjenice da su *Raspoloživost* i *Pouzdanost* najčešće usko povezani te je sam cilj (izbor najbolje alternative sa stanovišta ukupnog QoS nivoa) zapravo formulisan na sledeći način: „ukoliko je *Raspoloživost* servisa dovoljno velika tada može biti od većeg značaja uzeti u razmatranje *Vreme odziva* dok će se suprotnom (ukoliko *Raspoloživost* mala) od većeg značaja biti *Pouzdanost*“. Navedeni logički zahtev:

(Ako je „*Raspoloživost*“ **I** „*Vreme odziva*“)
ILI
(Ako „*Nije raspoloživost*“ **I** „*Pouzdanost*“)
tada je „*Ocena*“

se zapravo može formalno izraziti sledećim logičkim izrazom:

$$(A \wedge RT) \vee (\neg A \wedge R).$$

Shodno izloženom AHP pristupu prvi korak je formiranje hijerarhijskog modela, koji će se u ovom slučaju sastojati samo iz tri nivoa (Slika 24).



Slika 24. Hijerarhijski model za problem selekcije pojedinačnih web servisa kada je samo cilj definisan kao logički zahtev

Zatim se za svaki od kriterijuma formira matrica poređenja kojom se predstavljaju rezultati ocenjivanja svake od alternativa prema tom kriterijumu:

Tabela 8. Matrice poređenja (Primer 1.)

	Vreme odziva			*Raspoloživost*			*Pouzdanost*		
	WS_1	WS_2	WS_3	WS_1	WS_2	WS_3	WS_1	WS_2	WS_3
WS_1	1	1/9	1/4	1	1/7	1/3	1	1/5	1/7
WS_2	9	1	2	7	1	2	5	1	1/2
WS_3	4	1/2	1	3	1/2	1	7	2	1

Za svaku od ovih matrica se zatim određuje vektor sopstvenih vrednosti na osnovu kojih je moguće rangirati alternative spram svakog od kriterijuma:

Tabela 9. Sopstveni vektori i odgovarajući rangovi (Primer 1.)

	Vreme odziva		*Raspoloživost*		*Pouzdanost*	
WS_1	0.0724	3	0.0926	3	0.0755	3
WS_2	0.6263	**1**	0.6150	**1**	0.3338	2
WS_3	0.3013	2	0.2924	2	0.5907	**1**

Konačno neophodno je ispitati konzistentnosti donosioca odluke (Tabela 10). Budući da su svi koeficijenti konzistentnosti značajno ispod 10% njegovi sudovi se mogu smatrati pouzdanim. Šta više, moglo bi se reći da je u ovom primeru donosilac odluke bio skoro savršeno konzistentan.

Tabela 10. Evaluacija konzistentnosti (Primer 1.)

	Vreme odziva	*Raspoloživost*	*Pouzdanost*
λ_{max}	3.002	3.004	3.020
CI	0.001	0.002	0.010
CR	0.20%	0.32%	1.70%

Sledeći korak u klasičnoj AHP metodi bi podrazumevao da se isti postupak zatim primeni za određivanje relativnog značaja kriterijuma u odnosu na cilj i da se na kraju izračunaju globalni prioriteti.

Međutim, budući da je cilj zapravo definisan kao logički zahtev u kojima značaj samih kriterijuma zavisi od ostvarenih vrednosti drugih kriterijuma – ukoliko je *Raspoloživost* servisa dovoljno velika tada može biti od većeg značaja uzeti u razmatranje *Vreme Odziva* dok će u suprotnom slučaju (ukoliko je *Raspoloživost* mala) od većeg značaja biti *Pouzdanost*, bilo bi nemoguće formirati odgovarajuće matrice poređenja koje bi na adekvatan način odrazile ovaj zahtev.

U opštem slučaju, definisani logički izraz bi se mogao evaluirati primenom pravila klasične fazi logike tj $(A\cdot RT)+((1-A)\cdot R)-(A\cdot RT\cdot(1-A)\cdot R)$ i rezultati ove evaluacije su dati u Tabeli 11.

Ipak, kao što je objašnjeno u Potpoglavlju 3.5.6.1, u ovom radu se preferira primena konzistentne fazi logike, (prevashodno zbog postojanja negacije) te će se ovaj logički izraz najpre transformisati u odgovarajući generalizovani atomski Bulov polinom. U ovom konkretnom slučaju primenjeni su sledeći koraci transformacije:

$$\begin{aligned}((A\wedge RT)\vee(\neg A\wedge R))^{\otimes} &= (A\wedge RT)^{\otimes}+(\neg A\wedge R)^{\otimes}-((A\wedge RT)\wedge(\neg A\wedge R))^{\otimes}=\\ &= A\otimes RT+((1-A)\wedge R)^{\otimes}-(A\wedge RT)^{\otimes}\otimes(\neg A\wedge R)^{\otimes}=\\ &= A\otimes RT+(1-A)\otimes R-A\otimes RT\otimes((1-A)\otimes R)=\\ &= A\otimes RT+R-A\otimes R-A\otimes RT\otimes R+A\otimes RT\otimes A\otimes R=\\ &= A\otimes RT+R-A\otimes R.\end{aligned}$$

Zatim se pažnja usmerava na vrednosni nivo izborom standardnog algebarskog proizvoda kao adekvatnog operatora za generalizovani proizvod i unošenjem odgovarajućih vrednosti (rezultati su prikazani u Tabeli 11).

Tabela 11. Poređenje rezultata za Primer 1.

	Konzistentna *Fazi Logika*		Klasična *Fazi Logika*	
WS_1	0.0752	3	0.0747	3
WS_2	**0.5137**	**1**	0.4642	2
WS_3	0.5061	2	**0.4693**	**1**

Kao što se i očekivalo, rezultati ova dva pristupa se razlikuju. Dakle, iako bi se generalno gledano AHP metoda mogla proširiti na opisani način i primenom klasične fazi logike očigledno je da bi, u određenim slučajevima njena primena mogla voditi ka izboru lošije alternative (tj. u izloženom primeru se njenom primenom dobija poredak $WS_3 \succ WS_2 \succ WS_1$ umesto poretka $WS_2 \succ WS_3 \succ WS_1$ koji se dobija primenom konzistentne fazi logike). Pri tome je očigledno da je WS_2 bolji izbor u pogledu definisanih zahteva, budući da je bolji od WS_3 po dva od tri kriterijuma.

Ilustracije radi, vrednosti u Tabeli 11 za WS_1 su izračunate na sledeći način:

Konzistentna fazi logika:

0.0752 *= 0.0926·0.0724+0.0755-0.0926·0.0755.*

Klasična fazi logika:

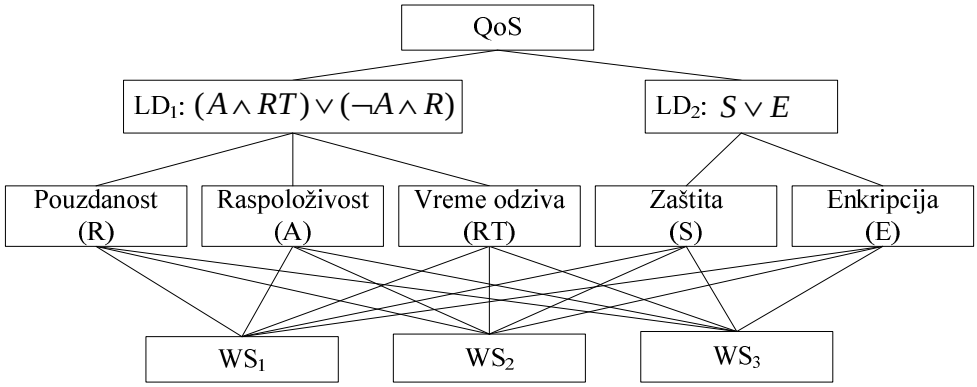
0.0747 *= 0.0926·0.0724 + (1-0.0926)·0.0755 - 0.0926·0.0724·(1-0.0926)·0.0755.*

4.3.2. Kriterijumi definisani kao logički zahtevi (Primer 2.)

U opštem slučaju, ukoliko je potrebno doneti odluku na osnovu većeg broja veoma složenih kriterijuma u hijerarhijski model problema bi se u tom slučaju mogli uključiti i dodatni nivoi. Na primer, donosilac odluke bi, pored prethodno definisanog logičkog zahteva vezanog za *Vreme odziva*, *Raspoloživost* i *Pouzdanost* servisa (u daljem tekstu **LD$_1$**), mogao želeti da uzme u razmatranje i QoS atribute koji se odnose na nivo bezbednost samog servisa (tj. *Zaštita* – **S** i *Enkripcija* – **E**). Ukoliko je donosilac odluke spreman da prihvati visok nivo bilo koja od ova atributa odgovarajući logički zahtev (u daljem tekstu **LD$_2$**) bi se tada mogao formulisati na sledeći način: „visok nivo bilo *Zaštite* bilo *Enkripcije* ukazuje na to da je određeni servis bezbedan“ što bi se moglo formalno iskazati sledećim izrazom:

$$S \vee E.$$

Hijerarhijski model ovako definisanog problema prikazan je na sledećoj slici:



Slika 25. Hijerarhijski model za problem selekcije pojedinačnih web servisa kada su kriterijumi definisani kao logički zahtevi

Matrice poređenja, u kojima se daje ocena svake od alternativa u odnosu na ova dva nova potkriterijuma, date su u sledećoj tabeli:

Tabela 12. Matrice poređenja (Primer 2.)

	Zaštita			*Enkripcija*		
	WS_1	WS_2	WS_3	WS_1	WS_2	WS_3
WS_1	1	1/5	2	1	1	1/7
WS_2	5	1	9	1	1	1/7
WS_3	1/2	1/9	1	7	7	1

Vektori sopstvenih vrednosti i rangovi alternativa spram ovih potkriterijuma su:

Tabela 13. Sopstveni vektori i odgovarajući rangovi (Primer 2.)

	Zaštita		*Enkripcija*	
WS_1	0.1577	2	0.1111	3
WS_2	0.7606	**1**	0.1111	3
WS_3	0.0817	3	0.7778	**1**

Konačno Tabela 14 potvrđuje da je donosilac odluke i ovaj put bio konzistentan.

Tabela 14. Evaluacija konzistentnosti (Primer 2.)

	Zaštita	*Enkripcija*
λ_{max}	3.002	3
CI	0.001	0
CR	0.19%	0%

Kao i u prethodnom primeru, logički izraz $(S \vee E)$ se evaluira primenom pravila konzistentne fazi logike (uzimajući standardni algebarski proizvod za operator generalizovanog proizvoda):

$$\begin{aligned}(S \vee E)^{\otimes} &= S + E - (S \wedge E)^{\otimes} = \\ &= S + E - S \otimes E.\end{aligned}$$

Sledeći korak u klasičnoj AHP metodi bi bio da se odredi relativni značaj kriterijuma (LD_1 i LD_2) u odnosu na cilj. U slučaju da donosilac odluke smatra da su oba logička zahteva jednako značajna mogla bi se primeniti dva različita pristupa za njihovo agregiranje u jedinstveni cilj.

Prvi pristup bi bio da se, u skladu sa klasičnom AHP metodom, odredi relativni značaja ovih kriterijuma u odnosu na cilj, putem matrica poređenja:

Tabela 15. Matrica poređenja, relativne težine i rang kriterijuma (Primer 2.)

	LD_1	LD_2
LD_1	1	1
LD_2	1	1

$\Rightarrow$

	QoS	
LD_1	0.5	**1**
LD_2	0.5	**1**

Tada se globalni prioriteti alternativa (odnosno servisa) u odnosu na cilj dobijaju agregiranjem lokalnih prioriteta elementa u hijerarhiji koji se nalaze na svakoj mogućoj putanji koja vode od date alternative do cilja, primenom aditivne agregacije.

Međutim, kako se navodi u (*Saaty, 1990*) u opštem slučaju izborom preferenci se može uticati na to da prilikom formiranja težinske sume neka alternativa koja je dominantna po makar jednom kriterijumu, koliko god on nevažan bio, dobije najveću konačnu vrednosti čime ona postaje najpoželjnija. Drugim rečima, ako dva kriterijuma treba da budu podjednako značajna dodeljivanje jednakih težina svakom od njih ne mora uvek adekvatno odražavati postavljeni logički zahtev ($LD_1 \wedge LD_2$). Naime, kao što je prethodno istaknuto u Poglavlju 3.4., ukoliko postoje dve alternative (od kojih je prva mnogo dominantnija od druge po jednom od kriterijuma a značajno lošija po drugom, dok je druga alternativa prosečna po oba kriterijuma) u slučaju prve alternative, usled dominantnosti prvog kriterijuma, formiranjem težinske sume se potire uticaj drugog kriterijuma što može voditi izboru pogrešne alternative. Budući da je zahtev bio da oba kriterijuma ravnopravno učestvuju u selekciji imalo bi više smisla izabrati alternativu koja je prosečna po oba kriterijuma ali čija je težinska suma manja. Što se jedino može postići primenom logičkih funkcija koje će omogućiti da se izbor podjednako zasniva na vrednostima oba zahteva.

Stoga se kao drugi pristup određivanju globalnih prioriteta, umesto formiranja matrica poređenja, predlaže primena iste tehnike zasnovane na evaluaciji značaja kriterijuma putem logičkih funkcija. Tada će se globalni prioriteti zapravo odrediti primenom navedene logičke agregacije. Pri tome budući da ovakav zahtev ne uključuje negaciju mogla bi se primeniti i klasična fazi logika.

Tabela 16. Poređenje rezultata za Primer 2.

	SAW		Conventional *Fuzzy Logic*	
WS_1	0.1632	3	0.0189	3
WS_2	0.6505	2	**0.4044**	**1**
WS_3	**0.6510**	**1**	0.4028	2

Dobijeni rezultati pokazuju da ova dva pristupa neće voditi ka istom izboru, što je u skladu sa prethodno iznetom diskusijom kao i opservacijom iznetom u Potpoglavlju 3.5.6.1 da logička agregacija ima za rezultat novu strukturu komponenti što nije slučaj kada je reč o SAW pristupu.

4.3.3. Primena pseudo logičke agregacije (Primer 3.)

Konačno treći primer se odnosi na situaciju u kojoj donosilac odluke želi da donese odluku na osnovu više različitih zahteva vezanih za date kriterijume te se ilustruje mogućnost primene pseudo-logičke agregacije za integrisanje tih različitih zahteva (koji, u opštem slučaju, mogu biti logičke funkcije) u jedinstveni globalni zahtev. Na primer, ukoliko bi donosilac odluke želeo da iskoristi prednosti oba prikazana pristupa odnosno, ukoliko bi želeo da donese odluku i na osnovu rangiranja dobijenog AHP metodom i na osnovu rangiranja dobijenog primenom predloženim proširenjem AHP metode, mogla bi se primenom pseudo-logičke agregaciju parcijalnih zahteva, kreirati težinska suma ova dva zahteva.

U ovom primeru će se razmatrati isti logički zahtev vezan za *Vreme odziva*, *Raspoloživost* i *Pouzdanost* servisa pri čemu će donosilac odluke takođe preferirati servis koji pruža viši nivo *Zaštite* (**S**) nezavisno od vrednosti ostalih atributa:

$$\left(\left(A \wedge RT\right) \vee \left(\neg A \wedge R\right)\right) \wedge S$$

Pri tome će se u ovom slučaju posmatrati problem kod koga je na raspolaganju pet servisa. Vrednosti izabranih QoS atributa za svaki od razmatranih servisa (alternativa) date su u sledećoj tabeli:

Tabela 17. Vrednosti izabranih QoS atributa (Primer 3.)

	Vreme odziva	*Raspoloživost*	*Pouzdanost*	*Zaštita*
WS_1	1000 *ms*	95%	72%	Dobra
WS_2	2000 *ms*	85%	70%	Nedovoljna
WS_3	1500 *ms*	98%	72%	Odlična
WS_4	1200 *ms*	85%	75%	Dobra
WS_5	1500 *ms*	92%	67%	Odlična

Primenom uobičajenog postupka formiranja matrica poređenja dobijaju se vektori sopstvenih vrednosti i rangovi alternativa spram svakog od kriterijuma:

Tabela 18. Sopstveni vektori i rangovi alternativa u odnosu na kriterijume (Primer 3.)

	Vreme odziva		*Raspoloživost*		*Pouzdanost*		*Zaštita*	
WS_1	0.0374	5	0.2668	2	0.1811	2	0.0991	3
WS_2	0.5623	**1**	0.0405	4	0.0851	4	0.0312	5
WS_3	0.1626	2	0.5067	**1**	0.1811	2	0.3853	**1**
WS_4	0.0750	4	0.0405	4	0.5185	**1**	0.0991	3
WS_5	0.1626	2	0.1455	3	0.0342	5	0.3853	**1**

Primena klasične AHP metode podrazumeva da se isti postupak ponovi i za ocenjivanje značaja svakog od kriterijuma prema cilju. Ako je dobijena matrica relativnih težina:

[0.4321 0.2075 0.2549 0.1055]

poslednji korak bi bio izračunavanje kompozitnog vektora koji zapravo predstavlja proizvod relativnih težinskih vektora svakog od nivoa u hijerarhiji. Na taj način bi se dobio relativni značaj svake od alternativa prema cilju (Tabela 19.) čime bi se omogućilo njihovo poređenje i rangiranje.

Primena proširene AHP metode zahteva da se najpre izvrši transformacija:

$$\begin{aligned}\left(\left((A\wedge RT)\vee(\neg A\wedge R)\right)\wedge S\right)^{\otimes} &= \left((A\wedge RT)\vee(\neg A\wedge R)\right)^{\otimes}\otimes S = \\ &= \left((A\wedge RT)^{\otimes}+(\neg A\wedge R)^{\otimes}-\left((A\wedge RT)\wedge(\neg A\wedge R)\right)^{\otimes}\right)\otimes S = \\ &= \left(A\otimes RT+\left((1-A)\wedge R\right)^{\otimes}-(A\wedge RT)^{\otimes}\otimes(\neg A\wedge R)^{\otimes}\right)\otimes S = \\ &= \left(A\otimes RT+(1-A)\otimes R-A\otimes RT\otimes\left((1-A)\otimes R\right)\right)\otimes S = \\ &= \left(A\otimes RT+R-A\otimes R-A\otimes RT\otimes R+A\otimes RT\otimes A\otimes R\right)\otimes S = \\ &= A\otimes RT\otimes S+R\otimes S-A\otimes R\otimes S\end{aligned}$$

a zatim su, primenom standardnog algebarskog proizvoda, izračunate vrednosti za svaki od servisa i rezultati su prikazani u sledećoj tabeli:

Tabela 19. Rang alternativa u odnosu na cilj (Primer 3.)

	AHP metoda		*Proširena AHP metoda*	
WS_1	0.1281	5	0.0141	4
WS_2	**0.2764**	**1**	0.0033	5
WS_3	0.2622	2	**0.0662**	**1**
WS_4	0.1834	3	0.0496	2
WS_5	0.1498	4	0.0204	3

Kako bi se dalje omogućilo poređenje i rangiranje alternativa uzimajući u obzir oba postavljena zahteva koristiće se pseudo logička agregacija. Drugim rečima formiraće se težinska suma tih pojedinačnih zahteva, pri čemu se u ovom primeru oba zahteva tretiraju ravnopravno tj. težina svakog od njih je *0.5*. Konačne vrednosti dobijene primenom ovog pristupa i redosled alternativa dati su u Tabeli 20.

Tabela 20. Rang alternativa određen pomoću pseudo-logičke agregacije (Primer 3.)

	WS_1	WS_1	WS_3	WS_4	WS_5
Globalni prioritet	0.0715	0.1398	**0.1642**	0.1165	0.0851
Rang	5	2	**1**	3	4

5. Nove metode za selekciju servisa za datu kompoziciju

Problem selekcije servisa za kompoziciju, kao što je izloženo u Poglavlju 3.2., predstavlja problem globalne optimizacije. Lokalna selekcija, odnosno izbor najboljeg kandidata za svaku od komponenti pojedinačno, ne mora nužno rezultovati optimalnim rešenjem sa globalnog stanovišta tj. sa stanovišta QoS celokupne kompozicije, a pre svega ako je neophodno da rezultujuća kompozicija zadovolji i postavljena globalna ograničenja. Sa druge strane, globalna selekcija, koja podrazumeva da se za datu kompoziciju evaluiraju sve moguće kombinacije kandidata, iako rezultuje optimalnom kombinacijom servisa (ukoliko, imajući u vidu postavljena ograničenja, takva kombinacija postoji)[1], najčešće nije primenjiva budući da broj mogućih kombinacija servisa eksponencijalno raste sa porastom broja komponenti kompozicije i/ili broja raspoloživih servisa za svaku od komponenti. Stoga je očigledno da, kod problema velikih dimenzija, potpuno pretraživanje (*eng. exhaustive search*) svih dopustivih kombinacija (kako bi se pronašlo optimalno rešenje) često nije moguće ostvariti u razumnom vremenu tj. računarska složenost[2] datog problema (u pogledu potrebnih računarskih resursa i vremena) zapravo ograničava mogućnost njegove primene u većini realnih slučajeva.

Imaju u vidu navedeno, a s obzirom na sve veću aktuelnost razmatranog problema, može se zaključiti da je potrebno obezbediti metode koje će omogućiti njegovo efikasno rešavanje naročito kada je raspoloživo vreme ograničeno, što je posebno izraženo kada je potrebno obezbediti mogućnost dinamičke selekcije servisa odnosno mogućnost da se selekcija vrši neposredno pre izvršavanja kompozicije, ili čak u toku njenog izvršavanja (umesto da se servisi unapred izaberu i permanentno dodele kompoziciji).

[1] Kao što je poznato (*Ehrgott, 2005*) za svaki kombinatorni problem je uvek moguće pronaći optimalno rešenje budući da je dopustivi skup rešenja (pa samim tim i kriterijumski skup) konačan, naravno pod pretpostavkom da dopustivi skup nije prazan, odnosno u ovom slučaju da postoji kombinacija servisa koja zadovoljava sva postavljena ograničenja.

[2] Prema (*Ehrgott, 2005*) računska složenost (*eng. computational complexity*) se meri brojem operacija (elementarnih koraka) algoritma koje je potrebno izvršiti kako bi se dobilo rešenje nekog problema odlučivanja za najveću moguću dimenziju datog problema. Kako se dalje navodi problemi optimizacije i problemi odlučivanja su usko povezani te se problemi kombinatorne optimizacije mogu lako prevesti u probleme odlučivanja i obrnuto.

Shodno tome, u literaturi su predloženi brojni pristupi, kako egzaktni tako i heuristički, koji se razlikuju pre svega po cilju koji se želi postići selekcijom a samim tim i po formiranim modelima koji često podrazumevaju određena pojednostavljenja samog problema.

Imajući u vidu broj različitih metoda kao i uspešnost njihove primene namera ovog rada je bila da se ispita mogućnost korišćenja nekih novijih pristupa optimizacije, koji nisu do sada primenjivani u ovoj oblasti, kako bi se unapredilo rešavanje definisanog problema. Naime, kada je reč o selekciji servisa za kompoziciju, ideja da se obezbede metode koje omogućavaju efikasno rešavanje navedenog problema u uslovima kada je potrebna dinamička kompozicija servisa imajući u vidu činjenicu da je ovo jedan od zahteva koji je u poslednje vreme sve aktuelniji. Kako, sa jedne strane, skup raspoloživih servisa ne mora u svakom trenutku biti isti (u međuvremenu se mogu pojaviti novi servisi sa boljim QoS karakteristikama ili se može desiti da u vreme izvršavanja kompozicije predviđeni servis bude nedostupan ili da se njegovi QoS nivoi značajno promene) dok se sa druge strane i zahtevi poslovanja često menjaju (što u opštem slučaju iziskuje i promenu zahteva vezanih za očekivani i/ili zahtevani QoS nivo) mogućnost brze i dinamičke kompozicije servisa postaje imperativ. Pri tome je namera bila da se obezbede novi pristupi i za egzaktno i za heurističko rešavanje postavljenog problema.

U skladu sa tim, u nastavku rada će najpre, za egzaktno rešavanje navedenog problema modelovanog pomoću predloženog MOLFP pristupa (Poglavlje 3.6.2), biti predložena, po prvi put, primena jedne novije tehnike izložene u (*Stanojević & Stanojević, 2013*) koja će u ovom radu biti prilagođena za diskretan MOLFPP. Pokazaće se da primena ove tehnike omogućava generisanje jako efikasnih rešenja formulisanog problema u veoma kratkom vremenskom roku čak i za probleme većih dimenzija. Pri tome je, kako se navodi u (*Stanojević & Stanojević, 2014*), modifikacijom navedenog pristupa moguće dobiti i sva efikasna rešenja postavljenog problema.

Zatim će pažnja biti usmerena na primenu metoda za približno rešavanje problema (tzv. heurističkih metoda) u slučajevima kada primena egzaktnih metoda nije adekvatna u pogledu računarskih resursa i raspoloživog vremena, imajući u vidu da broj mogućih kombinacija servisa za datu kompoziciju eksponencijalno raste sa porastom broja raspoloživih servisa. U takvim slučajevima prednost se daje heurističkim

metodama koje omogućavaju, kako rešavanje vremenski kritičnih problema većih dimenzija u okviru raspoloživog vremena, tako i rešavanje problema koji su slabo struktuirani. Shodno tome, za rešavanje problema selekcije servisa za kompoziciju, će se, po prvi put, predložiti primena poznate metaheurističke metode – metode promene okolina (*eng. Variable Neighborhood Search* – VNS). Takođe će biti predložena i jedna nova hibridna metaheuristika, zasnovana na VNS i tabu pretraživanju kojom bi se unapredilo rešavanje problema selekcije servisa za kompoziciju u uslovima kada je neophodna brza i dinamička selekcija. Pokazaće se da se navedene metode mogu efikasno primeniti, kako za rešavanje problema modelovanog kao MMKP (budući da je ovaj pristup modelovanju problema selekcije u literaturi najzastupljeniji), tako i za rešavanje problema iskazanog pomoću predloženog novog pristupa modelovanju zasnovanog na primeni konzistentne fazi logike (izloženog u Poglavlju 3.5.) Uz to u radu je predložena i jedna nova metoda koja bi se mogla koristiti za efikasno pronalaženje početnog dopustivog rešenja problema modelovanog kao MMKP.

5.1. Egzaktna metoda

5.1.1. Osnovna teorijska polazišta

Egzaktne metode omogućavaju pronalaženje optimalnog rešenja nekog optimizacionog problema (naravno ukoliko takvo rešenje uopšte postoji) obezbeđujući pri tome i dokaz optimalnosti pronađenog rešenja. Ukoliko se posmatra problem maksimizacije[1] neke funkcije *f(x)* nad dopustivim skupom $X \subset \mathbb{R}^m$, dopustivo rešenje x^{opt} se smatra *optimalnim* ako zadovoljava sledeći uslov:

$$f(x^{\text{opt}}) \geq f(x), \forall x \in X.$$

Kada je reč o višekriterijumskoj optimizaciji:

$$\max_{x \in X} \left[f_1(x), f_2(x), \ldots, f_n(x) \right], n \geq 2$$

tada je optimalno rešenje, koje se u (*Vujošević & al, 1996*) naziva i *savršeno rešenje*, ono dopustivo rešenje koje je najbolje po svim kriterijumima istovremeno:

$$f_k(x^{\text{opt}}) \geq f_k(x), \forall x \in X, k = 1, \ldots, n.$$

To znači da bi u slučaju selekcije servisa za kompoziciju, optimalno rešenje predstavljalo ono rešenje x^{opt} (tj. konkretna kompozicija servisa *CS* u kojoj je svakoj od komponenti pridružen tačno po jedan servis) koje obezbeđuje optimalnu vrednost svih relevantnih QoS atributa:

$$q^k(x^{\text{opt}}) \geq q^k(x), \forall x \in X, k = 1, \ldots, n.$$

Međutim, imajući u vidu činjenicu da kriterijumi mogu biti (i najčešće jesu) delimično ili čak potpuno konfliktni, očigledno je da kod ovakvih problema u opštem slučaju nije moguće pronaći rešenje koje bi bilo optimalno u navedenom smislu jer rešenje koje je optimalno po nekom kriterijumu neće biti optimalno i po drugim kriterijumima. Stoga se koristi nešto drugačiji koncept optimalnosti tzv. *Pareto optimalnost*.

Budući da je tada cilj pronaći ono rešenje koje obezbeđuje optimalni kompromis (*eng. trade-off*) u pogledu svih relevantnih kriterijuma, najpre se nameće pitanje na koji način će se meriti kvalitet rešenja odnosno kako će se upoređivati sama rešenja, ili preciznije rečeno njima odgovarajuće vrednosti kriterijuma, kako bi se utvrdilo koji je kompromis „bolji“.

[1] Zadatak minimizacije neke funkcije se, kao što je poznato, može prevesti u zadatak maksimizacije negativne vrednosti te funkcije.

Naime, jednom dopustivom rešenju x zapravo odgovara skup vrednosti kriterijuma, odnosno vektor $f(x) = \left(f_1(x), f_2(x), \ldots, f_n(x) \right)$, koji određuje jednu moguću tačku f u kriterijumskom prostoru $\mathbb{R}^n$. Drugim rečima, skup dopustivih rešenja X se preslikava u *kriterijumski* skup S koji u suštini predstavlja skup svih ostvarivih vrednosti vektora kriterijumskih funkcija u kriterijumskom prostoru:

$$S = f(X) = \left\{ f \in \mathbb{R}^n : f = f(x), x \in X \right\}.$$

Međutim, kako se navodi u (*Ehrgott, 2005*) pošto kriterijumski skup S, kao podskup skupa $\mathbb{R}^n$ ($n \geq 2$), nije totalno uređen potrebno je mapirati vektore kriterijumskih funkcija iz $\mathbb{R}^n$ u neki skup uređen odgovarajućim relacijama parcijalnog ili (striktno parcijalnog) poretka. Izborom relacije poretka, pomoću koje će se upoređivati vektori kriterijumskih funkcija, delimično se daje odgovor na pitanje koji je vektor „bolji“ odnosno koji se kompromis preferira.

Kod Pareto optimalnosti relacija parcijalnog poretka se zasniva na konceptu dominantnosti pri čemu se vektori kriterijumskih funkcija zapravo upoređuju po komponentama (*eng. componentwise*). Naime, za neku tačku u kriterijumskom prostoru f^1 se kaže da dominira nad nekom drugom tačkom f^2 ako su sve komponente njenog vektora f_k^1 makar jednako dobre kao njima odgovarajuće komponente vektora f_k^2 pri čemu postoji barem jedna komponenta k po kojoj je vektor f^1 bolji od f^2. Ovakva vrsta dominantnosti, koja predstavlja relaciju striktnog parcijalnog poretka (*eng. strict partial order*)[1], se često naziva i jakom (*eng. strong*). Ukoliko se traži da sve komponente vektora f^1 budu strogo bolje od njima odgovarajućih komponenti vektora f^2 onda se govori o strogoj (*eng. strict*) dominantnosti. Sa druge strane, ukoliko se izabere relacija parcijalnog poretka koja nije striktna odnosno ukoliko se dozvoli i da dva vektora budu jednaka po svim komponentama, tada je reč o slaboj (*eng. weak*) dominantnosti.

Na isti način za neko dopustivo rešenje x_1 se kaže da *dominira* nad nekim drugim dopustivim rešenjem x_2 ukoliko je bolje od njega po barem jednom kriterijumu dok po svim ostalim kriterijumima ostvaruje bolju ili makar jednaku vrednost kao x_2 (tj. ako f^1 dominira nad f^2). Sa druge strane, ukoliko je x_1 striktno bolje po svim kriterijuma u odnosu na x_2 u pitanju je *stroga* dominantnost dok se odnos između dva dopustiva

[1] Kako se navodi u (*Ehrgott, 2005*) binarna relacija je relacija *striktnog parcijalnog poretka* ako je tranzitivna i pri tome nije refleksivna dok je binarna relacija koja je refleksivna, tranzitivna i antisimetrična relacija *parcijalnog poretka*.

rešenja može okarakteristati kao *slaba* dominatnost ako x_1 dominira nad x_2 ili su njima odgovarajući vektori $f(x_1)$ i $f(x_2)$ jednaki (tj. ukoliko je $f_k(x_1) = f_k(x_2)\ \forall k = 1,\dots n$).

Konačno treba napomenuti da ukoliko rešenje x_1 ne dominira nad rešenjem x_2 to ne mora nužno implicirati da x_2 dominira nad x_1 budući da se može desiti da je prvo rešenje bolje od drugog po nekim kriterijumima dok je drugo bolje od prvog po nekim drugim kriterijumima. Imajući u vidu prethodno iznetu opservaciju da kriterijumski skup nije totalno uređen (tj. da se vektori kriterijumskih funkcija ne mogu u potpunosti urediti) očigledno je da, iako je za neka rešenja moguće nedvosmisleno utvrditi da su „bolja“ od drugih, postoji ceo skup rešenja (odnosno kompromisa) za koja nije moguće matematički utvrditi koje je bolje. Pored toga ovakav parcijalno uređen skup može imati više maksimalnih (odnosno minimalnih) elemenata koji će zapravo odgovarati Pareto optimalnim rešenjima.

Dakle, Pareto optimalno rešenje, koje se u literaturi između ostalog naziva i efikasno rešenje (*eng. efficient solution*)[1], predstavlja ono dopustivo rešenje nad kojim ne dominira nijedno drugo dopustivo rešenje. Drugačije rečeno za neko dopustivo rešenje x^* se može reći da je efikasno ako ne postoji neko drugo dopustivo rešenje koje dominira nad njim, pri čemu karakter efikasnosti rešenja zapravo zavisi od vrste dominantnosti odnosno od izabrane relacije poretka.

Naime, x^* je *jako* efikasno rešenje[2] (*eng. strongly efficient solution*) ako u dopustivom skupu ne postoji neko drugo rešenje koje je od njega bolje po svim kriterijumima. Formalno iskazano, ukoliko je u pitanju problem maksimizacije, za neko dopustivo rešenje x^* se može reći da je Pareto optimalno ako ne postoji neko drugo dopustivo rešenje x takvo da je:

$$f_k(x) \geq f_k(x^*), \quad k = 1,\dots,n \quad \text{i}$$

$$f_{k_0}(x) > f_{k_0}(x^*) \quad \text{za barem jedno } k_0 \in \{1,\dots,n\}.$$

Navedeni uslov praktično znači da, iako bi neko drugo dopustivo rešenje možda obezbedilo bolju vrednost nekih kriterijuma, postoji barem jedan kriterijum za koji bi takvo rešenje imalo goru vrednost.

[1] Kako se navodi u (*Ehrgott, 2005*) u literaturi se ovakva rešenja nazivaju još i dominantnim, nedominiranim ili neinferiornim.

[2] Treba napomenuti da se za ovakvo rešenje u literaturi često koristi samo naziv *efikasno* rešenje.

Sa druge strane, *x** predstavlja *slabo* efikasno rešenje (*eng. weakly efficient solution*) ako ne postoji nijedno drugo dopustivo rešenje koje strogo dominira nad njim, tj. ako ne postoji neko drugo dopustivo rešenje *x* takvo da je:

$$f_k(x) > f_k(x^*), \ \forall k = 1, \dots n.$$

Navedeni uslov praktično znači da, iako bi neko drugo dopustivo rešenje možda obezbedilo bolju vrednost nekih kriterijuma, ne postoji rešenje koje je bolje po svakom od kriterijuma.

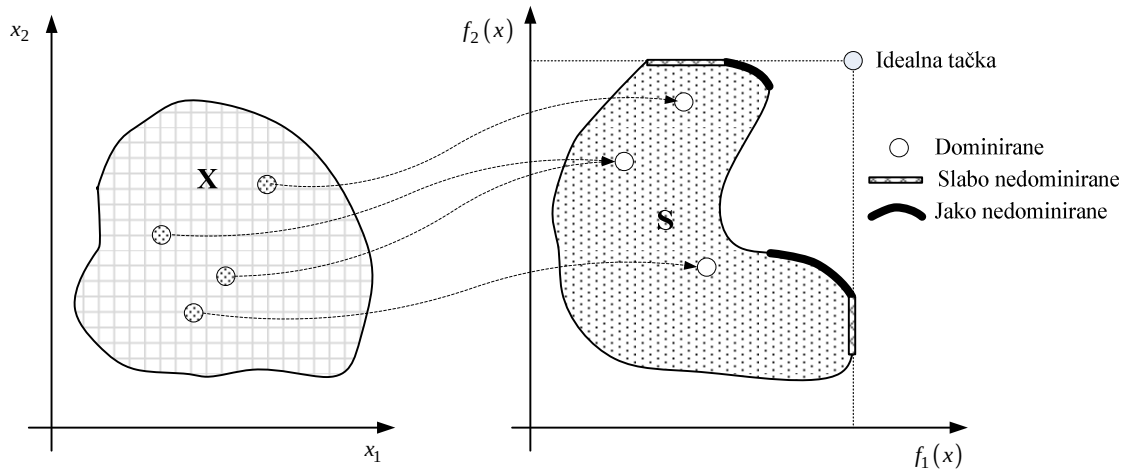
Konačno, *x** bi bilo *strogo* efikasno rešenje (*eng. strictly efficient solution*) ako ne postoji nijedno drugo dopustivo rešenje koje slabo dominira nad njim, tj. ako ne postoji neko drugo dopustivo rešenje *x* takvo da je:

$$f_k(x) \geq f_k(x^*), \ k = 1, \dots n, \quad \text{i} \ x \neq x^*.$$

Tačke u kriterijumskom prostoru koje odgovaraju ovakvim rešenjima se nazivaju *jako nedominiranim* (ili samo *nedominiranim*), *slabo nedominiranim* i *strogo nedominiranim* tačkama, respektivno, pri čemu se tačka koja bi odgovarala savršenom rešenju naziva *idealna* tačka. Kao što je prethodno istaknuto savršeno rešenje najčešće ne pripada skupu dopustivih rešenja te se u tom slučaju idealna tačka nalazi van skupa ostvarivih tačaka u kriterijumskom prostoru. Međutim kako se navodi u (*Ehrgott, 2005*) komponente idealnog vektora, koje odgovaraju optimalnim vrednostima koje bi se mogle postići po pojedinačnim kriterijumima (tj. dobiti rešavanjem *n* jednokriterijumskih problema[1]), zapravo daju gornje (odnosno, u slučaju minimizacije, donje) granice (*eng. bounds*) svakog od kriterijuma[2]. Odnos između navedenih tačaka, za slučaj maksimizacije dva kriterijuma, prikazan je na Slici 26. Treba napomenuti i da, prema (*Ehrgott, 2005*) koncept stroge nedominantnosti zapravo ne postoji u kriterijumskom prostoru budući da po definiciji stroga efikasnost zabranjuje postojanje tzv. ekvivalentnih rešenja odnosno dva rešenja sa istim vrednostima po svim kriterijumima ($f(x_1) = f(x_2)$) ili, drugačije rečeno, ne dozvoljava mogućnost postojanja više rešenja kojima bi odgovarala ista tačka u kriterijumskom prostoru.

[1] Rešenja ovih problema (x^{k*}) predstavljaju *marginalna* rešenja jednog višekriterijumskog problema.

[2] Kako autor dalje navodi, donje (odnosno, u slučaju minimizacije, gornje) granice predstavljaju komponente kriterijumskog vektora koji odgovara tački *nadira*. Komponente ovog vektora su zapravo jednake minimalnim (tj. maksimalnim) vrednostima odgovarajućih kriterijuma nad skupom nedominiranih tačaka. U opštem slučaju ni tačka nadira, kao ni idealna tačka, ne mora pripadati kriterijumskom skupu tj. ne mora odgovarati nekom dopustivom rešenju.



Slika 26. Pareto optimalnost u kriterijumskom prostoru

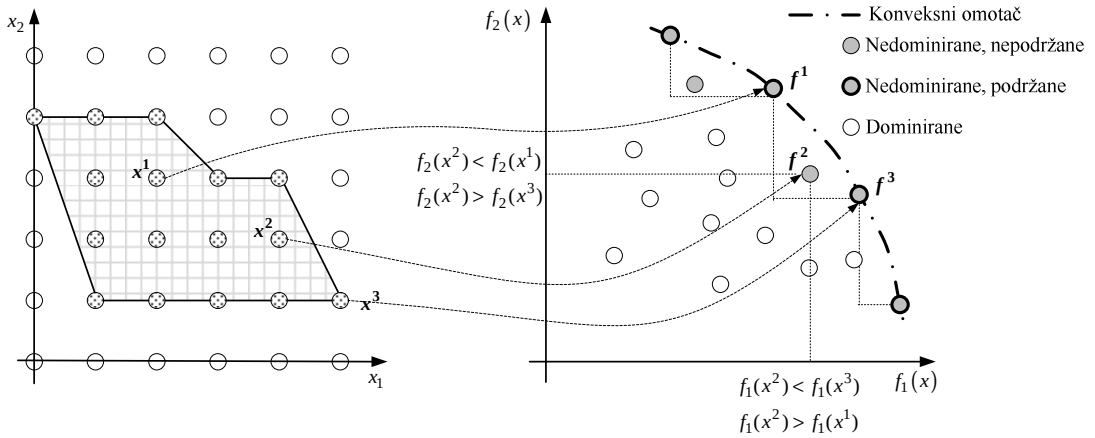
Skup tačaka u kriterijumskom prostoru koji odgovara svim Pareto optimalnim rešenjima se naziva Pareto front. Ovaj skup predstavlja kompromise koji dominiraju nad svim preostalim kompromisima koji ne pripadaju datom skupu. Shodno navedenom očigledno je da se preferiraju ona rešenja koja su, u nekom smislu, efikasna budući da je u suprotnom moguće ostvariti poboljšanje po svim kriterijumima te se preostala dominirana rešenja svakako mogu isključiti iz daljeg razmatranja. Sa druge strane, kako se navodi u (*Vujošević & al, 1996*), poboljšanje vrednosti bilo kog kriterijuma (u odnosu na efikasno rešenje) bi uvek prouzrokovalo pogoršanje vrednosti nekog drugog kriterijuma, dok u slučaju slabo efikasnog rešenja zapravo nije moguće istovremeno poboljšati sve kriterijume a da pri tome rešenje i dalje ostane dopustivo.

Međutim, budući da se sva efikasna rešenja (koja predstavljaju najbolje moguće kompromise) generalno gledano mogu smatrati jednako prihvatljivim, izbor najboljeg rešenja (kompromisa) će zapravo zavisiti od preferenci donosioca odluke. Pri tome donosilac odluke može svoje preference u pogledu kriterijuma iskazati unapred (tj. ugraditi u model ili metodu za rešavanje) te će cilj biti pronalaženje samo jednog efikasnog rešenja koje će najviše odgovarati njegovim preferencama (tzv. *a priori* pristup) ili mu se mogu ponuditi sva efikasna rešenja nakon čega će on izabrati ono koje preferira (tzv. *a posteriori* pristup), pri čemu je naravno moguća i kombinacija ova dva pristupa (tzv. *interaktivni* pristup).

Konačno treba napomenuti da u slučaju da se radi o kombinatornom problemu kod koga je dopustivi skup diskretan i konačan tj. $X \subset \mathbb{Z}^m$ (ili čak $X \subseteq \{0,1\}^m$) problem određivanja efikasnih tačaka u opštem slučaju postaje nešto složeniji. Naime, može se desiti da određeni broj efikasnih rešenja nije moguće dobiti budući da je kriterijumski skup u tom slučaju nekonveksan[1]. Shodno tome, neko efikasno rešenje *x** predstavlja *podržano* efikasno rešenje (*eng. supported efficient solution*) ako postoji vektor težina $w \in \mathbb{R}^n, w_k > 0, k = 1,\ldots,n.$ takav da je *x** optimalno rešenje sledećeg jednokriterijumskog problema:

$$\max_{x \in X} f(x) = \sum_{k=1}^{n} w_k f_k(x)$$

dok u suprotnom *x** predstavlja *nepodržano* efikasno rešenje (*eng. nonsupported efficient solution*). Njima odgovarajuće tačke u kriterijumskom prostoru se tada nazivaju *podržane* i *nepodržane* nedominirane tačke, respektivno. Drugim rečima, kao što je prikazano na Slici 27 (na slici je i kriterijumski skup diskretan tj. $S \subset \mathbb{Z}^n$) podržane nedominirane tačke su one koje se nalaze na obodu konveksnog omotača (*eng. convex hull*) kriterijumskog skupa dok je nepodržana nedominirana tačka ona za koju postoji neka konveksna kombinacija nekog para nedominiranih tačaka koja dominira nad njom (te se zapravo nalaze u unutrašnjosti konveksnog omotača).



Slika 27. Podržane i nepodržane nedominirane tačke

[1] Pored toga, postavlja se i pitanje povezanosti (*eng. connectedness*) skupa nedominiranih tačaka u kriterijumskom prostoru odnosno skupa efikasnih rešenja u dopustivom prostoru. Naime, može se dokazati (*Ehrgott, 2005*) da ukoliko je skup nedominiranih tačaka (ili skup efikasnih rešenja) povezan tada se ceo skup nedominiranih tačaka (ili efikasnih rešenja) može ispitati pri čemu i izbor konačnog kompromisnog rešenja postaje znatno lakši budući da ne postoje prekidi u efikasnom skupu.

Kako se navodi u (*Ehrgott, 2005*) većina metoda za pronalaženje efikasnih rešenja nekog višekriterijumskog kombinatornog problema, zasnovanih na tehnikama skalarizacije (pomoću kojih se na različite načine dati višekriterijumski problem svodi na jednokriterijumski problem kao što su na primer metoda težinskih koeficijenata, metoda ε-ograničenja itd.) ili ne omogućavaju pronalaženje svih efikasnih rešenja početnog višekriterijumskog problema ili rezultuju skalarizovanim problemima koji su NP-teški[1].

Budući da razmatrani problem selekcije servisa za kompoziciju upravo i predstavlja kombinatorni problem, očigledno je da u slučaju da je potrebno pronaći sva efikasna rešenja neophodno primeniti metode koje će omogućiti i pronalaženje nepodržanih efikasnih rešenja. Međutim, ukoliko se pođe od pretpostavke da je neophodna dinamička selekcija servisa onda će cilj zapravo biti pronaći bilo koje efikasno rešenje u što kraćem vremenskom roku.

[1] Prema (*Cvetković & al, 1996*) neki problem odlučivanja je NP (*eng. nondeterministic polynomial time*) težak ako ne postoji algoritam za njegovo egzaktno rešavanje koji se izvršava u broju koraka koji je ograničen polinomom po dimenziji problema.

5.1.2. Postojeći pristupi za egzaktno rešavanje problema

Među metodama koje se u literaturi predlažu za egzaktno rešavanje problema selekcije servisa za kompoziciju mogu se izdvojiti:

- Celobrojno programiranje (*eng. integer programming – IP*)

  Zeng & al,2003; Aggarwal & al, 2004; Zeng & al, 2004; Ardagna & Pernici, 2005; Ardagna & Pernici, 2007; Yu & al, 2007; Kattepur & al, 2011.
- Metode grananja i ograničavanja (*eng. branch and bound*)

  Lin & al, 2005;Yu & Lin, Decembar, 2005; Yu & al, 2007.
- Dinamičko programiranje (*eng. dynamic programming*)

  Gao & al, 2006; Huang & al, 2009; Yu & Lin, Jul, 2005.
- Metode optimizacije na grafovima

  Yu & Lin, Jul, 2005; Yu & Lin, Decembar, 2005; Yu & al, 2007.
- Specijalno razvijene metode

  Gronmo & Jaeger, Jun 2005; Gronmo & Jaeger, Decembar 2005; Jaeger, 2006; Lin & al, 2005; Yu & Lin, Decembar, 2005; Yu & al, 2007.

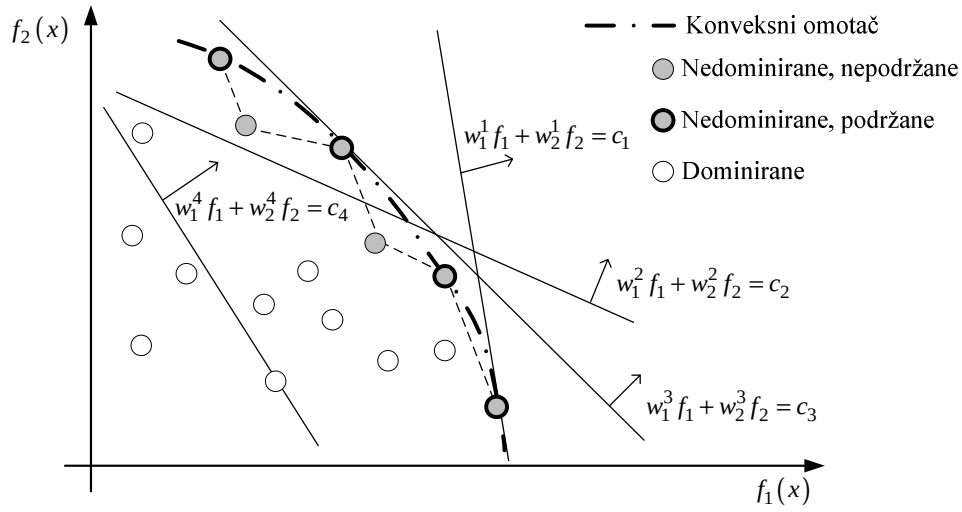
Na prvom mestu treba istaći da neki od navedenih pristupa (*Gao & al, 2006; Jaeger, 2006 – „Greedy Selection“, „Pattern-wise Selection“; Huang & al, 2009*) podrazumevaju odsustvo QoS ograničenja ili postojanje samo jednog QoS ograničenja (*Yu & Lin, Jul, 2005; Jaeger, 2006 – „Bottom-Up Approximation“, „Constraint Optimized Selection“*) što je suviše restriktivno za većinu realnih problema.

Pored toga treba napomenuti da se većina navedenih pristupa zasniva na agregiranju višestrukih kriterijuma u jedinstvenu funkciju cilja (bilo prilikom formiranja modela bilo prilikom njegovog rešavanja) primenom SAW tehnike. Drugim rečima, ukoliko je od ukupnog broja relevantnih kriterijuma n za n_1 kriterijuma poželjno ostvariti maksimalnu moguću vrednost dok je za preostalih ($n - n_1$) poželjno ostvariti što manju vrednost onda se odgovarajuća jednokriterijumska funkcija cilja može formirati na sledeći način:

$$\max_{x \in X} f(x) = \sum_{k=1}^{n_1} w_k f_k^{norm}(x) - \sum_{k=n_1+1}^{n} w_k f_k^{norm}(x), \quad w_k \geq 0, k = 1, \ldots n.$$

gde su: $w_k > 0, k = 1, \ldots n,$ težine samih kriterijuma, dok $f_k^{norm}, k = 1, \ldots n,$ predstavljaju normalizovane kriterijumske funkcije.

Kako se navodi u (*Ehrgott, 2005*) ukoliko su sve težine pozitivne (tj. $w_k > 0, k = 1,\ldots n.$), optimalno rešenje ovako formulisanog problema uvek predstavlja efikasno rešenje originalnog višekriterijumskog problema[1]. Zapravo, izborom težina formira se hiperravan koja je tangenta kriterijumskog skupa. Otuda, ukoliko je kriterijumski skup konveksan, variranjem samih težina moguće je dobiti sva efikasna rešenja originalnog problema. Međutim, ukoliko kriterijumski skup nije konveksan, što je i slučaj kada je reč o kombinatornim problemima, primena navedene tehnike ne omogućava pronalaženje svih efikasnih rešenja. Naime, nepodržane nedominirane tačke se ne mogu dostići optimizacijom bilo koje konveksne kombinacije kriterijumskih vektora kao što je i prikazano na Slici 28.



Slika 28. Geometrijska interpretacija metode težinskih koeficijenata

Primena ove metode otvara problem izbora težinskih faktora. Sa jedne strane donosilac odluke je suočen sa problemom adekvatne valorizacije pojedinih kriterijuma (koji je detaljno opisan u Poglavlju 3.4.). Sa druge strane ukoliko se rešenje traži variranjem nagiba tangentne hiperravni onda, pored izuzetne kompleksnosti u određivanju težina, same težine ni na koji način ne odražavaju značaj pojedinih kriterijuma.

Pored toga još jedan nedostatak ovog pristupa se ogleda u tome što on zahteva da se relevantni kriterijumi (koji po svojoj prirodi mogu biti veoma heterogeni) prethodno normalizuju kako bi se mogli agregirati u jedinstveni kriterijum (kao što je istaknuto u Poglavlju 3.4).

[1] U slučaju da su težine nenegativne (tj. $w_k \geq 0, k = 1,\ldots n.$) ova metoda garantuje dobijanje makar slabo efikasnog rešenja pri čemu, ukoliko je dobijeno optimalno rešenje jedinstveno, onda ono zapravo predstavlja efikasno rešenje originalnog problema.

Međutim ovde će se razmatrati rešavanje navedenog problema modelovanog pomoću MOLFP pristupa predloženog u Poglavlju 3.6.2. čija se osnovna ideja ogleda u sparivanju relevantnih QoS kriterijumi (uzimajući u obzir njihovu prirodu) sa ciljem da se optimizuju njihovi količnici. Najpre treba napomenuti da prema (*Costa, 2007*) rešavanje MOLFP problema u opštem slučaju zahteva značajne računarske resurse čak i kada je potrebno odrediti samo jedno efikasno rešenje.

Potrebno je istaći da pristupi predloženi u (*Canfora & al, 2004; Canfora & al, Jun 2005; Canfora & al, Jul 2005; Jaeger & Muhl, 2007*), koji tretiraju problem selekcije servisa za kompoziciju kao problem jednokriterijumskog razlomljenog programiranja, zapravo ne predlažu egzaktne metode za njegovo rešavanje već se zasnivaju na primeni heuristika odnosno preciznije rečeno genetskog algoritma.

Sa druge strane u literaturi su prisutni brojni pristupi za rešavanje MOFP problema koji bi se mogli primeniti i za navedeni problem. Iscrpan pregled se može naći na primer u (*Craven, 1988; Schaible, 1995; Stancu-Minasian, 1997; Schaible, 2002; Frenk & Schaible, 2005; Frenk & Schaible, 2009*) ili za kombinatorne probleme u (*Barros, 1998; Radzik, 1998; Prokopyev, 2009*). Kako autori navode u (*Frenk & Schaible, 2005*) većina predloženih pristupa polazi od modela kod kojih je potrebno minimizirati najveći od količnika („*min-max fractional program*“)[1] ili optimizovati težinsku sumu količnika („*sum-of-ratios fractional program*“). Iako bi se generalno gledano za dati problem kompromisno rešenje moglo tražiti optimizacijom težinske sume formiranih QoS količnika, pored prethodno istaknutih teškoća vezanih za određivanje efikasnih rešenja takvog problema, takođe treba istaći da takav problem prema (*Schaible & Shi, 2003*) spade među probleme razlomljenog programiranja koje je najteže rešavati. Naime, autori ističu da u opštem slučaju, imajući u vidu prirodu (konveksnost odnosno konkavnost) samih funkcija koje predstavljaju brojioce i imenioce datih razlomaka, lokalni optimum težinske sume razlomaka najčešće ne predstavlja i njen globalni optimum, pri čemu autori navode da dato tvrđenje stoji čak i kada je reč o sumi samo dva razlomka kod kojih su i brojioci i imenioci afine funkcije.

Stoga se u ovom radu predlaže primena jedne novije tehnike za generisanje jako efikasnih rešenja za MOLFP počevši od bilo kog dopustivog rešenja (*Stanojević & Stanojević, 2013*), koja ne zahteva formiranje težinske sume.

[1] U literaturi se navedeni problem često formuliše i kao problem maksimizacije najmanjeg od količnika („*max-min fractional program*“).

5.1.3. Novi pristup egzaktnom rešavanju problema

U ovom potpoglavlju biće izložena kompletna metodologija za rešavanje problema selekcije servisa za kompoziciju (modelovanog kao MOLFP) primenom novije tehnike za generisanje jako efikasnih rešenja za MOLFP počevši od bilo kog dopustivog rešenja (*Stanojević & Stanojević, 2013*). Najpre će biti izložena sama tehnika a zatim će se na ilustrativnom primeru pokazati na koji način bi se ona mogla primeniti na navedeni problem. Konačno će biti prikazani eksperimentalni rezultati koji pokazuju da primena navedene tehnike omogućava dobijanje jako efikasnog rešenja u veoma kratkom vremenskom roku čak i za probleme većih dimenzija.

U (*Stanojević & Stanojević, 2013*) autori polaze od opšte formulacije MOLFP problema za neki konveksan i ograničen dopustivi skup $X \subseteq \mathbb{R}^n$:

$$\max_{x \in X} \quad \left\{ z(x) = \left(\frac{N_1(x)}{D_1(x)}, \frac{N_2(x)}{D_2(x)}, \ldots, \frac{N_v(x)}{D_v(x)} \right) \right\}, \qquad v \geq 2$$

pri čemu je: $N_g(x) = c_g x + \alpha_g, \qquad c_g \in \mathbb{R}^n, \alpha_g \in \mathbb{R}, \forall g = 1, \ldots, v,$

$$D_g(x) = d_g x + \beta_g, \qquad d_g \in \mathbb{R}^n, \beta_g \in \mathbb{R}, \forall g = 1, \ldots, v,$$

$$d_g x + \beta_g > 0, \qquad \forall g = 1, \ldots, v, \forall x \in X,$$

uz dodatni uslov:

$$c_g x + \alpha_g > 0, \qquad \forall g = 1, \ldots, v, \forall x \in X.$$

Shodno razmatranjima iz Poglavlja 5.1.1. za neko dopustivo rešenje *x** se može reći da je *jako efikasno* rešenje navedenog problema ako i samo:

$$\nexists x \in X : \frac{N_g(x)}{D_g(x)} \geq \frac{N_g(x^*)}{D_g(x^*)}, \forall g = 1, \ldots, v,$$

pri čemu postoji bar jedno g_0 takvo da je:

$$\frac{N_{g_0}(x)}{D_{g_0}(x)} > \frac{N_{g_0}(x^*)}{D_{g_0}(x^*)}.$$

Drugim rečima efikasna rešenja MOLFP problema su ona dopustiva rešenja kod kojih nije moguće dodatno povećati bilo koji količnik a da to pri tome ne dovede do smanjenja nekog od preostalih količnika.

Autori zatim pokazuju da se pristup za testiranje efikasnosti nekog dopustivog rešenja primenom linearnog programiranja koji je predložen u (*Lotfi & al, 2010*) može zapravo iskoristiti za generisanje jako efikasnih rešenja za MOLFP počevši od bilo kog dopustivog rešenja[1].

Naime, autori dokazuju da će arbitrarno dopustivo rešenje x^* predstavljati jako efikasno rešenje navedenog MOLFP problema ako i samo ako je optimalna vrednost sledećeg problema jednaka *nuli*.

$$\max \quad \sum_{g=1}^{v} (d_g^+ + d_g^-)$$

pri ograničenjima: $c_g x + \alpha_g - d_g^+ = \left(c_g x^* + \alpha_g\right)\theta_g, \quad g = 1,..,v,$

$$d_g x + \beta_g + d_g^- = \left(d_g x^* + \beta_g\right)\theta_g, \quad g = 1,..,v,$$

$$x \in X,$$

$$d_g^+, d_g^-, \theta_g \geq 0 \qquad g = 1,..,v.$$

Stoga, oni predlažu iterativnu tehniku koja podrazumeva da se počevši od nekog arbitrarnog početnog dopustivog rešenja ($x^*=x^0$) u svakoj iteraciji h pronalazi optimalno rešenje datog problema $(x^h, \theta^h, d^{h+}, d^{h-})$ i iteriranje se nastavlja (uzimajući da je $x^*=x^h$) dokle god je $\sum_{g=1}^{v}(d_g^{h+} + d_g^{h-}) \neq 0.$ Osnovna ideja ovog pristupa ogleda se u tome da se maksimizacijom sume odstupanja, vrednosti svih kriterijuma povećavaju ili u najmanju ruku ostaju iste. Stoga se u svakoj iteraciji zapravo dobija bolja tačka, pri čemu iteriranje staje kada nije više moguće ostvariti poboljšanje. Drugim rečima, navedena tehnika će generisati sekvencu rešenja x^1, x^2,..., x^h,... koja će konvergirati ka jako efikasnom rešenju postavljenog MOLFP problema.

Konačno treba napomenuti da izložena tehnika pripada grupi *a priori* metoda budući da rezultat iterativnog postupka zavisi od izbora početnog dopustivog rešenja.

[1] Treba napomenuti da autori u (*Stanojević & Stanojević, 2013*) takođe predlažu i tehniku za generisanje slabo efikasnih rešenja navedenog problema.

Ilustrativni primer

Primena izložene tehnike će biti ilustrovana na primeru diskretnog MOLFP modela problema iz poglavlja 3.6.2. u kome se razmatraju količnici agregirane *Raspoloživosti* (q^1) i agregiranog *Vremena Odziva* (q^2), odnosno agregirane *Pouzdanost* (q^3) i agregiranog *Kašnjenje* (q^4):

$$\max_{x\in X} \left\{ z(x) = \left(\frac{\sum_{i=1}^{m}\sum_{j=1}^{p_i}\left(\log\left(q_{ij}^1\right)\right)x_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{p_i} q_{ij}^2 x_{ij}}, \frac{\sum_{i=1}^{m}\sum_{j=1}^{p_i}\left(\log\left(q_{ij}^3\right)\right)x_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{p_i} q_{ij}^4 x_{ij}} \right) \right\}$$

pri ograničenjima: $\sum_{i=1}^{m}\sum_{j=1}^{p_i}\left(\log\left(q_{ij}^k\right)\right)x_{ij} \geq \log\left(Q^k\right)$, $k = 1,3,$

$$\sum_{i=1}^{m}\sum_{j=1}^{p_i} q_{ij}^k x_{ij} \leq Q^k, \qquad k = 2,4,$$

$$\sum_{j=1}^{p_i} x_{ij} = 1, \qquad i = 1,\ldots, m,$$

$$x_{ij} \in \{0,1\}, \qquad j = 1,\ldots, p_i;\ i = 1,\ldots, m.$$

Međutim da bi se navedena tehnika mogla primeniti potrebno je najpre izvršiti određena prilagođavanja definisanog modela problema. Naime, pošto navedena tehnika zahteva da i brojioci i imenioci razlomaka moraju biti pozitivne funkcije, a budući da logaritamske funkcije ne ispunjavaju ovaj uslov, potrebno ih je skalirati nekom vrednošću (α_g) kako bi se obezbedila njihova pozitivnost čime se, kao što je poznato, neće promeniti optimalno rešenje početne funkcije. Kao što je istaknuto u poglavlju 3.6.2. vrednosti (α_g) mogu se odrediti na sledeći način:

$$\alpha_1 = \omega + \sum_{i=1}^{m}\left|\min_{j=1,\ldots,p_i}\log\left(q_{ij}^1\right)\right|$$

$$\alpha_2 = \omega + \sum_{i=1}^{m}\left|\min_{j=1,\ldots,p_i}\log\left(q_{ij}^3\right)\right|$$

pri čemu je ω parametar koji se odabira tako da se osigura da najmanja moguća suma uvek bude striktno veća od najmanje moguće vrednosti sume logaritama date kriterijumske (odnosno agregirajuće) funkcije (npr. $\omega=1$). U navedenom primeru, s obzirom na to da su funkcije q_2 i q_4 linearne i pozitivne, imenioce nije bilo potrebno dodatno transformisati. Dakle, prilagođeni diskretan MOLFP model za dati problem je formiran na sledeći način:

$$\max_{x\in X} \left\{ z(x) = \left(\frac{\sum_{i=1}^{m}\sum_{j=1}^{p_i}\left(\log\left(q_{ij}^{1}\right)\right)x_{ij} + \alpha_1}{\sum_{i=1}^{m}\sum_{j=1}^{p_i} q_{ij}^{2}x_{ij}}, \frac{\sum_{i=1}^{m}\sum_{j=1}^{p_i}\left(\log\left(q_{ij}^{3}\right)\right)x_{ij} + \alpha_2}{\sum_{i=1}^{m}\sum_{j=1}^{p_i} q_{ij}^{4}x_{ij}} \right) \right\}$$

pri ograničenjima:

(1)

$$\sum_{i=1}^{m}\sum_{j=1}^{p_i}\left(\log\left(q_{ij}^{k}\right)\right)x_{ij} \geq \log\left(Q^{k}\right), \quad k = 1,3,$$

$$\sum_{i=1}^{m}\sum_{j=1}^{p_i} q_{ij}^{k}x_{ij} \leq Q^{k}, \quad k = 2,4,$$

$$\sum_{j=1}^{p_i} x_{ij} = 1, \quad i = 1,\ldots, m,$$

$$x_{ij} \in \{0,1\}, \quad j = 1,\ldots, p_i;\ i = 1,\ldots, m.$$

Sada se može formulisati diskretan problem linearnog programiranja koji, prema (*Stanojević & Stanojević, 2013*), predstavlja osnovu za konstruisanje procedure za generisanje efikasnih rešenja MOLFP problema. Pri tome treba napomenuti da ovaj problem zapravo predstavlja prilagođenu verziju problema od koga se polazi u (*Stanojević & Stanojević, 2013*) budući da se u navedenom radu razmatra kontinualni MOLFP problem, dok je problem selekcije servisa za kompoziciju po svojoj prirodi diskretan. Uvođenjem sledećih oznaka:

$$F^{1}(x) = \sum_{i=1}^{m}\sum_{j=1}^{p_i}\left(\log\left(q_{ij}^{1}\right)\right)x_{ij}$$

$$F^{3}(x) = \sum_{i=1}^{m}\sum_{j=1}^{p_i}\left(\log\left(q_{ij}^{3}\right)\right)x_{ij}$$

odgovarajući problem se može formulisati sledeći način:

$$\max\left(d_1^+ + d_2^+ + d_1^- + d_2^-\right)$$

pri ograničenjima:

$$F^1(x) + \alpha_1 - d_1^+ = \left(F^1\left(x^*\right) + \alpha_1\right)\theta_1, \qquad (2)$$

$$F^3(x) + \alpha_2 - d_2^+ = \left(F^3\left(x^*\right) + \alpha_2\right)\theta_2,$$

$$q^2(x) + d_1^- = q^2\left(x^*\right)\theta_1,$$

$$q^4(x) + d_2^- = q^4\left(x^*\right)\theta_2,$$

$$F^k(x) \geq \log\left(Q^k\right), \qquad k = 1,3,$$

$$q^k(x) \leq Q^k, \qquad k = 2,4,$$

$$\sum_{j=1}^{p_i} x_{ij} = 1, \qquad i = 1,\ldots, m,$$

$$x_{ij} \in \{0,1\}, \qquad j = 1,\ldots, p_i;\ i = 1,\ldots, m.$$

$$d_1^+, d_2^+, d_1^-, d_2^-, \theta_1, \theta_2 \geq 0$$

Konačno sledeća procedura obezbeđuje dobijanje jako efikasnog rešenja prilagođenog diskretnog MOLFPP (*1*) za bilo koje dopustivo rešenje x^0.

Algoritam 1. Generisanje efikasnog rešenja MOLFP problema

function MOLFPP_GENERATE_EFFICIENT_SOLUTION(x^0)

$x^*=x^0$; $h=1$;

repeat

Rešiti problem (2). Neka je $\left(x^h, \theta^h, d^{h+}, d^{h-}\right)$ njegovo optimalno rešenje.

$x^*=x^h$; $h=h+1$;

until $d_1^{h+} + d_2^{h+} + d_1^{h-} + d_2^{h-} = 0$

return x^*

Dobijeno rešenje x^* je jako efikasno rešenje prilagođenog diskretnog MOLFPP (*1*).

5.1.4. Eksperimentalni rezultati

Predložena tehnika je testirana na 50 problema različitih dimenzija za nasumično generisano početno rešenje. Naime variran je broj komponenti same kompozicije (m) i broj servisa koji su na raspolaganju za date komponente (p)[1] i za svaku takvu kombinaciju generisano je po 10 test instanci.

Sam algoritam je implementiran korišćenjem *GNU MathProg* jezika a zatim su optimizacioni problemi rešavani korišćenjem *GLPK* (GNU Linear Programming Kit) softvera. Dobijeni rezultati (dati u Tabeli 21) su upoređeni sa rezultatima koji bi se dobili potpunim pretraživanjem svih mogućih kombinacija, u slučajevima kada je bilo moguće izvršiti potpuno pretraživanje u razumnom vremenu.

Tabela 21. Eksperimentalni rezultati - MOLFPP

m=3 p=10		*m=7 p=10*		*m=10 p=10*		*m=10 p=100*		*m=10 p=1000*	
Instanca	*Broj iteracija*	*Instanca*	*Broj iteracija*	*Instanca*	*Broj iteracija*	*Instanca*	*Broj iteracija*	*Instanca*	*Broj iteracija*
1	1	*1*	1	*1*	2	*1*	2	*1*	2
2	1	*2*	1	*2*	2	*2*	2	*2*	3
3	1	*3*	1	*3*	2	*3*	2	*3*	2
4	1	*4*	1	*4*	2	*4*	2	*4*	3
5	1	*5*	1	*5*	2	*5*	2	*5*	2
6	1	*6*	1	*6*	2	*6*	3	*6*	2
7	1	*7*	1	*7*	2	*7*	3	*7*	2
8	1	*8*	1	*8*	2	*8*	2	*8*	2
9	1	*9*	1	*9*	2	*9*	2	*9*	2
10	1	*10*	1	*10*	2	*10*	2	*10*	2

Prosečno vreme potrebno za dobijanje egzaktnog rešenja potpunim pretraživanjem:				
15.625 ms	46859.4 ms	26218031.25 ms	N/A	N/A

Eksperimentalni rezultati pokazuju da je predložena tehnika uvek generisala jako efikasno rešenje definisanog MOLFP problema počevši od nekog, proizvoljno izabranog, dopustivog rešenja. Pored toga za svaku od 50 instanci efikasno rešenje je dobijeno za manje od 1 sekunde, osim u slučaju problema većih dimenzija (10x1000) kod kojih su pojedinačne iteracije trajale između 40 i 400 sekundi.

[1] Prilikom testiranja pretpostavljeno je da je broj servisa koji je na raspolaganju za svaku od komponenti isti (tj. $p = p_1 = p_2 = ... = p_m$) što ni na koji način ne umanjuje mogućnost primene navedenog pristupa i u slučajevima kada se broj raspoloživih servisa za svaku od komponenti razlikuje.

Primera radi egzaktno rešenje za kompoziciju koja se sastojala od 10 komponenti, pri čemu je za svaku od komponenti na raspolaganju bilo 100 servisa, je pronađeno za manje od 2 sekunde, dok je potpuno pretraživanje svih mogućih kombinacija trajalo 7.5 sati. Čak i u slučaju većih instanci (sa preko 1000 servisa po komponenti) predložena tehnika je dala rešenje za manje od nekoliko minuta dok potpuno pretraživanje nije moglo da se obavi u razumnom vremenskom intervalu.

Kao što se, na osnovu rezultata eksperimenata može videti, primena ove tehnike omogućava dobijanje jako efikasnog rešenja problema selekcije servisa za kompoziciju u veoma kratkom vremenskom intervalu, i to u samo nekoliko iteracija, čak i za probleme većih dimenzija. To znači da bi se ova metoda mogla uspešno primeniti u slučajevima kada je neophodno obezbediti mogućnost dinamičke selekcija servise odnosno pronalaženje bilo kog efikasnog rešenja u što kraćem vremenskom intervalu.

Sa druge strane, ukoliko se želi omogućiti „a posteriori“ odlučivanje sva efikasna rešenja postavljenog problema se, kao što je pokazano u (*Stanojević & Stanojević, 2014*), mogu dobiti modifikacijom izloženog pristupa, polazeći od konveksnih kombinacija marginalnih rešenja. Prednost ovakvog pristupa se ogleda u tome što omogućava dobijanje celog skupa alternativnih rešenja te donosilac odluke njihovim međusobnim poređenjem može steći uvid u moguće kompromise i analizirati koji su od njih za njega prihvatljivi.

Konačno, treba napomenuti da u opštem slučaju skup svih efikasnih rešenja jednog višekriterijumskog problema može biti suviše velik da bi omogućio efikasan izbor najboljeg rešenja naročito ako se ima u vidu činjenica da broj efikasnih rešenje raste sa porastom broja relevantnih kriterijuma. Stoga se još jedna od prednosti predloženog MOFP pristupa ogleda u tome što se uparivanjem kriterijuma u formi razlomaka zapravo smanjuje dimenzija kriterijumskog prostora, odnosno broja efikasnih rešenja, pa se time pojednostavljuje njihovo pronalaženje, a u krajnoj liniji, i značajno olakšava problem izbora konačnog rešenja.

5.2. Heurističke metode

Generalno gledano, kada su u pitanju problemi velikih dimenzija, primena egzaktnih metoda može biti neefikasna budući da vreme potrebno za njihovo izvršavanje može biti nedopustivo dugo. Naime, egzaktno rešavanje zahteva kombinatornu pretragu čije vreme izvršavanja raste eksponencijalno sa povećanjem dimenzije problema (*Karp, 1972*). Šta više, kada se zahteva dinamička kompozicija servisa, odnosno selekcija servisa neposredno pre izvršavanja kompozicije ili čak u toku njenog izvršavanja, vreme koje će se utrošiti na rešavanje problema postaje od kritičnog značaja. Stoga se za rešavanje vremenski kritičnih problema većih dimenzija u opštem slučaju, prednost daje računski efikasnijim heurističkim metodama koje iako, za razliku od egzaktnih metoda ne garantuju nalaženje optimalnog rešenja ili konvergenciju ka tom rešenju, ipak omogućavaju dobijanje dovoljno „dobrih“[1] rešenja i to u okviru raspoloživog vremena. Pored toga, treba napomenuti da budući da se problem selekcije servisa za kompoziciju u literaturi najčešće modeluje kao MMKP, a kako je poznato da je MMKP problem pripada klasi NP teških problema, primena egzaktnih algoritama za njegovo rešavanje nije adekvatna u većini realnih slučajeva.

Sa druge strane, kao što je poznato, heurističke metode se primenjuju i za rešavanje problema koji su slabo struktuirani tj. problema za koje je teško ili čak nemoguće formirati precizni matematički model koji bi na zadovoljavajući način odrazio njihovu strukturu. Prema (*Krčevinac & al, 2004*) ono što karakteriše ovakve probleme je vrlo kompleksna struktura sa velikim brojem raznorodnih ograničenja koja se često ne mogu u potpunosti matematički formalizovati, prisustvo elementa neodređenosti, neizvesnosti i subjektivne procene, nelinearnost nekih zavisnosti, ili potreba za zadovoljenjem više od jednog cilja. Zbog toga se za njihovo rešavanje, umesto egzaktnih metoda, koje zahtevaju postojanje precizno definisanih matematičkih modela, koriste heurističke metode koje omogućavaju rešavanje modela koji vernije odražavaju strukturu problema i mnogo više odgovaraju stvarnosti.

Imajući u vidu navedena razmatranja i činjenicu da problem selekcije servisa za kompoziciju u opštem slučaju, kao što je izloženo u Poglavlju 3.2, upravo predstavlja

[1] Kao što je navedeno u (*Krčevinac & al, 2004*) budući da heuristike ne koriste klasično formalizovane matematičke postupke bazirane na teoriji, njihova primena pri rešavanju problema ne garantuje nalaženje optimalnog rešenja. Međutim, one zapravo omogućavaju formalizovanje najrazličitijih inteligentnih pravila koja, primenjena u procesu nalaženja rešenja, mogu obezbediti da rešenja u proseku budu dovoljno bliska optimumu.

takvu vrstu problema u ovom poglavlju će se razmatrati heurističke metode za njegovo rešavanje.

Najpre treba napomenuti da se različite metaheuristike[1] u literaturi već uspešno primenjuju za rešavanje problema selekcije servisa za kompoziciju kao na primer:

- Simulirano kaljenje (*eng. simulated annealing* – SA)

  Berbner & al, 2006; Gao & al, 2009.
- Tabu pretraživanje (*eng. tabu search*– TS)

  Parejo & al 2008; Pop & al, 2011.
- Genetski algoritmi (*eng. genetic algorithms*– GA)

  Canfora & al, 2004; Canfora & al, Jun 2005; Canfora & al, Jul 2005; Claro & al, 2005; Canfora & al, 2006; Jaeger & Muhl, 2007; Ai & Tang, 2008; Canfora & al, 2008;Parejo & al, 2008; Wada & al, 2008; Ai, 2011; Wada & al, 2012.
- Optimizacija rojevima čestica (*eng. partical swarm optimization* – PSO)

  Ming & Zhen-wu, 2007; Fan & al, 2009; Wang & al, 2010.
- Hibridne heuristike

  Ye & Mounla, 2008; Tang & Ai, 2010; Ai, 2011.
- Hibridne metaheuristike (zasnovane na kombinaciji postojećih metaheuristika)

  Parejo & al 2008; Gao & al, 2009.

Pored toga specijalno razvijene heuristike se predlažu u (*Jaeger & al, 2005; Jaeger, 2006; Berbner & al, 2006; Sohrabi & al, 2006; Cao & al, 2007; Alrifai & Risse, 2009; Liu & al, 2009; Qi & al, 2010; Klein &, 2011; Luo & al, 2011; Sun & Zhao , 2012*). Konačno treba istaći da se u poslednje vreme sve više ispituje i mogućnost primene drugih evolutivnih algoritama zasnovanih na prirodnim procesima na primer mravljih kolonija, imunih algoritama, rojevima pčela itd.

Imajući u vidu uspešnost primene ovih metoda namera ovog rada je bila da se ispita mogućnost korišćenja metaheuristika koji nisu do sada primenjivane u ovoj oblasti kako bi se unapredilo rešavanje definisanog problema.

[1] Za razliku od specijalnih heuristika, koje se dizajniraju za posebne vrste optimizacionih problema poštujući pri tome svojstva i specifičnosti ovih problema, tzv. opšte heuristike (odnosno metaheuristike) predstavljaju sistematizovane heurističke metodologije koje daju opšta uputstva za rešavanje problema nezavisno od specifičnosti strukture samih problema (*Krčevinac & al, 2004*).

Metaheuristike koje se u ovom radu predlažu za rešavanje problema selekcije servisa za kompoziciju se prema (*Krčevinac & al, 2004*) mogu klasifikovati u grupu metoda lokalnog pretraživanja (tzv. sekvencijalnih metoda), te će najpre biti dat opis ovog pristupa. Zatim se za rešavanje problema selekcije servisa za kompoziciju predlaže primena poznate metaheurističke metode, koja nije do sada korišćena za selekciju servisa, Variable Neighborhood Search – VNS (*Mladenović & Hansen, 1997; Hansen & Mladenović, 2001; Hansen & Mladenović, 2005*). Uz to se predlaže i nova metoda koja bi se mogla koristiti za efikasno pronalaženje početnog dopustivog rešenja. Pored toga je razvijena i nova hibridna metaheuristika zasnovana na VNS i tabu pretraživanju. Pri tome će se pokazati da se predloženi pristupi mogu efikasno primeniti, kako za rešavanje problema modelovanog kao MMKP (budući da je ovaj pristup modelovanju problema selekcije u literaturi najzastupljeniji), tako i za rešavanje problema iskazanog pomoću predloženog novog pristupa modelovanju zasnovanog na primeni konzistentne fazi logike. Treba, pri tome istaći da oba navedena modela podrazumevaju agregaciju višestrukih kriterijuma u jedinstvenu funkciju cilja.

Konačno, treba imati u vidu da većina postojećih pristupa, kao što je izloženo u Poglavlju 3.3, podrazumeva da u modelu, i ograničenja[1] i funkcije cilja, moraju biti linearne funkcije, a da određivanje QoS za kompoziciju u opštem slučaju može rezultovati nelinearnim funkcijama (što zavisi od načina na koji se pojedinačni QoS atributi agregiraju). Primena ovih pristupa dakle podrazumeva svođenje nelinearnih funkcija na linearne (na primer primenom logaritamske funkcije na proizvode kako bi se preveli u sumu logaritama). Stoga se većina autora vezuje za određeni predefinisani skup QoS mera koje se agregiraju pomoću sume ili proizvoda dok se ostali načini agregacije (kao što su min, max funkcija ili bilo koja druga korisnički definisana funkcija) uglavnom ne razmatraju. Namera ovog rada je i da se pokaže da bi se pristupi koji se ovde predlažu mogli iskoristiti za rešavanje problema bez nametanja ovakvih ograničenja. Na ovaj način se omogućava da se selekcija vrši na osnovu bilo kog skupa QoS mera uključujući i domensko-specifične QoS mere koje se agregiraju pomoću korisnički definisanih funkcija.

[1] Pored toga treba napomenuti i da neki od pristupa podrazumevaju postojanje samo jednog QoS ograničenja ili čak njihovo potpuno odsustvo.

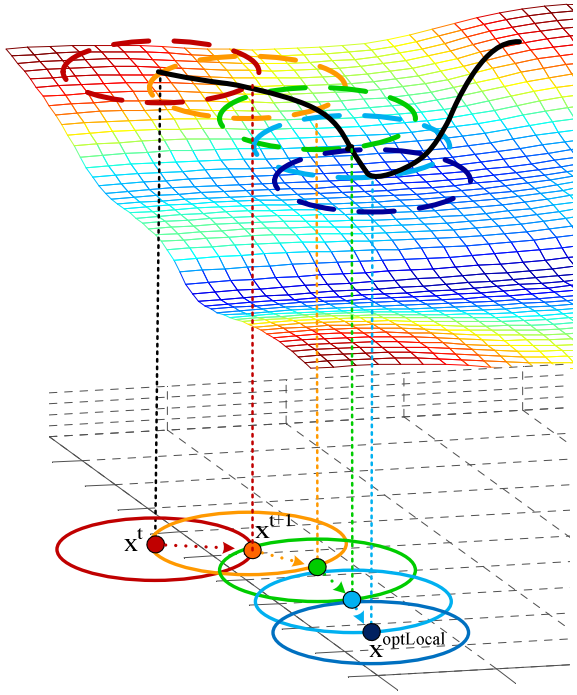
5.2.1. Osnovna teorijska polazišta

Generalno gledano kada je reč o problemima globalne optimizacije osnovno pitanje koje se postavlja je na koji način će se vršiti pretraživanje dopustive oblasti kako bi se pronašao globalni optimum. Budući da je najčešće nemoguće temeljno ispitati celu dopustivu oblast, izbor skupa probnih tačaka (rešenja) se može vršiti bilo stohastički bilo deterministički. Pored toga, kako se navodi u (*Törn & Žilinskas, 1989*), najčešće se primenjuju dve strategije za izbor tačaka u kojima će se vršiti ispitivanje: globalna pouzdanost (*eng. global reliability*) koja se zasniva na ideji da ne treba zanemariti nijedan deo dopustive oblasti i lokalno profinjavanje (*eng. local refinement*) koje polazi od činjenice da su šanse za pronalaženje novog boljeg rešenja veće u okolini nekog rešenja koja ima relativno bolju vrednost funkcije cilja nego u okolini rešenja u kome je vrednosti funkcije cilja relativno lošija, pri čemu većina metoda globalne optimizacije zapravo kombinuje globalnu i lokalnu strategiju. Sa druge strane autori navode i da se metode globalne optimizacije, između ostalog, mogu klasifikovati i po tome da li se izbor sledeće tačke u sekvenci zasniva na informacijama o vrednosti funkcije cilje u prethodnim tačkama ili ne, pa tako razlikuju lokalne metode (kod kojih sledeća tačka bira isključivo u neposrednom okruženju prethodnog rešenja) i ne-lokalne metode (kod kojih izbor sledeće tačke u sekvenci ne zavisi od vrednosti funkcije u prethodnim tačkama već se pretraga premešta u neke potpuno druge oblasti u pokušaju da se pronađe globalni optimum ili u najmanju ruku bolji lokalni optimum).

Osnovna ideja principa **lokalnog pretraživanja** se ogleda u tome da se, počevši od nekog početnog dopustivog rešenja x^0, sekvencijalno pretražuje dopustivi prostor (u potrazi za optimalnim rešenjem), pri čemu se u svakom koraku zapravo ispituje samo manji deo ovog prostora, tj. neposredna „okolina“ (*eng. neighborhood*) tekućeg rešenja iz koje se, prema nekim unapred zadatim pravilima, bira neko rešenje koje zatim postaje tekuće rešenje u sledećem koraku. Naime, umesto nasumičnog generisanja nekog skupa probnih rešenja, kod lokalnog pretraživanja se zapravo, sistematičnim kretanjem po dopustivom prostoru (odnosno premeštanjem (*eng. move*) iz rešenja u rešenje u tom prostoru), generiše konačna sekvenca probnih rešenja (x^0, x^1,…, x^t,..., x^*) u pokušaju da se dostigne optimalno rešenje (x^{opt}). Pri tome se, prilikom izbora rešenja u koje će se pretraga u narednom koraku premestiti (tj. sledećeg rešenja u sekvenci x^{t+1}), evaluiraju, naspram određenog predefinisanog kriterijuma, samo ona rešenja koja se nalaze u

neposrednoj okolini tekućeg rešenja x^t. Drugim rečima, u svakom koraku, izbor pravca u kome će se nastaviti kretanje zavisi isključivo od lokalnih informacija odnosno od lokalnih karakteristika funkcije cilja u okolini tekućeg rešenja.

Dakle, kako se navodi u (*Törn & Žilinskas, 1989*) kretanje po prostoru rešenja se zapravo postiže time što se tekuća tačka (odnosno sekvenca probnih rešenja) pomera po nekoj trajektoriji koja bi u idealnom slučaju trebalo da vodi kroz sve lokalne optimume:



Slika 29. Lokalno pretraživanje

Očigledno je da primena ove metode zahteva da se najpre definišu:

- struktura okoline,
- način generisanja početnog rešenja,
- pravilo izbora sledećeg rešenja u svakoj iteraciji i
- kriterijuma zaustavljanja.

Struktura okoline

Generalno gledano „okolina“ nekog rešenja *x* se može definisati kao skup rešenja *N(x)* koji su u nekom smislu „bliski“ datom rešenju (*Aarts & Lenstra, 2003*):

$$N(x) \subseteq X, \quad x \notin N(x)$$

tada se može reći da elementi ovog skupa predstavljaju „susede“ rešenja *x*.

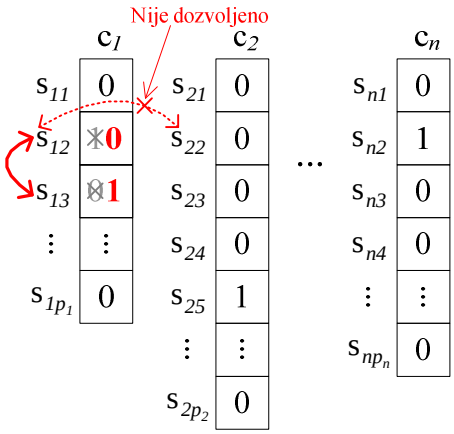
Dimenzija okoline odgovara broju rešenja koji se u njoj nalaze dok njena struktura zapravo zavisi od toga na koji način će se definisati „bliskost“ dva rešenja, odnosno od izbora same metrike.

Drugim rečima, prema (*Pirlot, 1996*), pod „susedom“ nekog rešenja se smatra svako rešenje koje se može dobiti direktno iz datog rešenja primenom neke, unapred definisane, lokalne modifikacije nazvane pomak (*eng. elementary move*) te okolina tog rešenja zapravo predstavlja sva ona rešenja koja bi se iz njega mogla dobiti primenom takve modifikacije. Stoga se, definisanje strukture okoline se zapravo svodi na definisanje pomaka, odnosno funkcije $N : X \rightarrow 2^X$, koja će iz tekućeg rešenja generisati ceo skup susednih rešenja (tj. okolinu).

U opštem slučaju pomak podrazumeva izmenu samo malog dela strukture tekućeg rešenja mada generalno gledano on može biti i veoma kompleksni te podrazumevati izmenu značajnog dela strukture tekućeg rešenja. Treba napomenuti da način na koji je definisan pomak, pa samim tim i struktura okoline, umnogome utiče na efikasnost lokalnog pretraživanja (*Pirlot, 1996*). Međutim, kako se navodi u (*Aarts & Lenstra, 2003*) ne postoje neka opšta pravila za njegovo definisanje već to pre svega zavisi od prirode samog problema i od metode koja će se primeniti za njegovo rešavanje, pri čemu je, u opštem slučaju, zapravo moguće na različite načine definisati pomak za isti problem.

Konkretno, u ovom radu, pomak će se zasnivati na principu jednostavne zamene, koji podrazumeva zamenu samo jednog servisa u kompoziciji, te će se okolina tekućeg rešenja sastojati iz rešenja koja se od datog rešenja razlikuju tačno po jednom servisu. Pri tome se vodi računa o tome da rezultujuća kompozicija mora biti validna odnosno da svakoj od komponenti mora biti pridružen tačno po jedan servis. Preciznije rečeno pomak će se ostvariti tako što će, neki servis s_{ij} koji pripada tekućem rešenju (u kome je svakoj od komponenti kompozicije pridružen tačno po jedan servis) biti zamenjen nekim drugim servisom s_{il} koji obezbeđuje istu funkcionalnost, odnosno koji je na raspolaganju za datu komponentu c_i. Kako se jedno rešenje u prostoru rešenja predstavlja pomoću skupa binarnih vrednosti (koje ukazuju na to da li je servis s_{ij} uključen u kompoziciji ili ne) za definisanje strukture okoline se kao metrika može koristiti Hemingovo rastojanje.

Stoga se pomak zapravo ostvaruje prebacivanjem (*eng. flip*) odgovarajuće promenljive x_{ij} u rešenju sa *1* na *0* ako servis s_{ij} nije više deo rešenja odnosno sa *0* na *1* ako servis s_{ij} treba da bude uključen u rešenje (Slika 30).



Slika 30. Elementarni pomak

Sada se okolina nekog rešenja x^t može formalno definisati na sledeći način:

$$N(x^t) = \left\{ x \mid x_{ij} = x_{ij}^t, i = 1,\ldots,n,\ j = 1,\ldots p_i,\ \exists_1 i \wedge \exists_2 j : x_{ij} \neq x_{ij}^t \right\}.$$

Drugim rečima, Hemingovo rastojanje[1] između bilo kog susednog rešenja x i tekućeg rešenja x^t:

$$d(x, x^t) = 2$$

što znači, da okolina $N(x^t)$ zapravo predstavlja hipersferu oko tekućeg rešenja x^t čiji je poluprečnik jednak 2.

Pri tome treba napomenuti da ovako definisan pomak, zapravo predstavlja modifikaciju principa jednostavne zamene koji se u literaturi najčešće primenjuje kada su u pitanju 0-1 problemi. Ova modifikacija je bila neophodna kako bi se obezbedilo generisanje isključivo validnih kompozicija, tj. kompozicija u kojima je svakoj od komponenti pridružen tačno po jedan servis. Sa druge strane, kako bi se održala jednostavnost pomaka, pri generisanju okoline se ne uzima u obzir dopustivost dobijenih rešenja. Drugim rečima, predloženim načinom generisanja susednih rešenja dobiće se i kompozicije koje, iako su validne, ne zadovoljavaju postavljena QoS ograničenja.

[1] Kao što je poznato iz teorije kodiranja Hemingovo rastojanje između dva vektora *x* i *y* je jednako broju koordinata po kojima se ta dva vektora razlikuju (*Hamming, 1950*).

Način generisanja početnog rešenja

Jedan od najčešće korišćenih pristupa za dobijanje početnog rešenje je njegovo nasumično generisanje. Međutim, u sledećem poglavlju će biti izložena i jedna nova heuristika koja bi se mogla primeniti za pronalaženje početnog dopustivog rešenja problema selekcije servisa za kompoziciju, modelovanog kao MMKP. Predložena heuristika, koja je specijalno razvijena sa ciljem da obezbedi „dobro" početno rešenje, pripada kategoriji konstruktivnih heuristika zasnovanih na principu proždrljivosti (*eng. greedy-add*).

Pravilo izbora sledećeg rešenja u svakoj iteraciji

Izbor rešenja, iz okoline razmatranog rešenja $N(x^t)$, koje će postati tekuće rešenje u narednoj iteraciji, takođe se može izvršiti na različite načine. U opštem slučaju se, kako se navodi u (*Gendreau, 2003*), kretanje po prostoru rešenja najčešće postiže iterativnom primenom niza lokalnih promena kojima se početno rešenje postepeno unapređuje sa ciljem da se ostvari bolja vrednost funkcije cilja. Drugim rečima, princip tzv. iterativnog poboljšavanja (*eng. iterative improvement*), kako se naziva u (*Aarts & Lenstra, 2003*), podrazumeva da se u svakoj iteraciji kao novo rešenje bira neki od suseda koji obezbeđuje poboljšanje funkcije cilja. Stoga se u okolini nekog rešenja najpre traži pravac kretanja koji će obezbediti poboljšanje vrednosti funkcije cilja a zatim se, prateći taj pravac, pretraga pomera u bolje susedno rešenje i kretanje se nastavlja na ovaj način dokle god je moguće pronaći pravac kretanja koji bi vodio ka poboljšanju. [1].

Pri tome su moguća dva različita načina realizacije ovog pristupa: tzv. prvo poboljšanje (*eng. first improvement*) kada se za novo rešenje uzima prvi pronađeni sused u kome funkcija cilja ima bolju vrednost, odnosno, tzv. najbolje poboljšanje (*eng. best improvement*) koje podrazumeva da se u svakoj iteraciji pretraži cela okolina (odnosno da se ispitaju svi susedi) a zatim se za novo rešenje uzima onaj sused koji je najbolji sa stanovišta definisane funkcije cilja.

Ukoliko se izabere strategija najboljeg poboljšanja, u svakoj iteraciji će se, kao početno rešenje za sledeću iteraciju x^{t+1}, uzeti onaj sused iz okoline tekućeg

[1] Navedeni pristup se u literaturi naziva tehnika spuštanja (*eng. descent*) kada je u pitanju problem minimizacije ondosno tehnika penjanja (*eng. hill climbing*) u slučaju maksimizacije.

rešenja x^t koji zadovoljava sledeće uslove (pod pretpostavkom da je se traži minimalna vrednost funkcije cilja):

$$f(x^{t+1}) \leq f(x), \forall x \in N(x^t),$$
$$f(x^{t+1}) \leq f(x^t),$$
$$x^{t+1} \in X$$

pri čemu je ovaj poslednji uslov bilo neophodno dodati s obzirom na to da predloženi pristup generisanju susednih rešenja može rezultovati okolinom koja sadrži i nedopustiva rešenja.

Konkretno, u ovom radu će se primeniti sledeći postupak:

Za svaku komponentu kompozicije c_i, servis s_{ij} koji je uključen u tekuće rešenje se iz njega isključuje (tj. odgovarajuća promenljiva x_{ij} se prebacuje sa *1* na *0*) a zatim se svi preostali servisi koji su na raspolaganju za datu komponentu c_i ispituju. Za svaki od servisa se najpre proverava da li bi njegovo uključivanje u kompoziciju vodilo ka narušavanju postavljenih ograničenja. Ukoliko je dobijeno rešenje dopustivo sledeći korak je izračunavanje vrednosti funkcije cilja. Konačno, metoda će rezultovati onim rešenjem u kome se ostvaruje najbolja vrednost funkcije cilja među razmatranim rešenjima.

Algoritam 2. Pronalaženje najboljeg pomaka

function BEST_ MOVE(x^t)
  for $i = 1,...,m$
    for $j = 1,...,p_i$
      if $x^t_{ij} \neq 1$ **then** *//servis s_{ij} nije deo tekućeg rešenja*
        $x' = x^t$;
        $x'_{ij} = 1$; $x'_{il} = 0$; // *zameniti servis s_{il} koji je deo tekućeg rešenja (tj. za koji je $x^t_{il} = 1$) servisom s_{ij}*
        if x' je dopustivo **then**
          if $f(x') \leq f^{t+1}$ **or** pronađen je prvi dopustivi sused **then**
            $x^{t+1} = x'$; $f^{t+1} = f(x')$;
    end
  end
  return x^{t+1}

U suprotnom, ukoliko se kao strategija odabere princip prvo poboljšanja tada će se za sledeće rešenje u sekvenci (x^{t+1}), uzeti prvi sused iz okoline tekućeg rešenja x^t koji zadovoljava sledeće uslove:

$$f(x^{t+1}) < f(x^t),\ \ x^{t+1} \in N(x^t)$$
$$x^{t+1} \in X$$

Konkretno, u ovom radu će se primeniti sličan postupak kao i u prethodnom slučaju, dakle svaka komponenta i svaki servis će se evaluirati ponaosob, ali će metoda rezultovati prvim pronađenim rešenjem u kome se ostvaruje bolja vrednost funkcije cilja.

Algoritam 3. Pronalaženje prvog boljeg pomaka

```
function FIRST_IMPROVING_ MOVE(x^t)
    for i = 1,...,m
        for j = 1,...,p_i
            if x^t_ij ≠ 1 then   //servis s_ij nije deo tekućeg rešenja
                x' = x^t;
                x'_ij = 1; x'_il = 0;   //   zameniti servis s_il koji je deo tekućeg
                                        //   rešenja (tj. za koji je x^t_il = 1) servisom s_ij
                if x' je dopustivo then
                    if f(x') < f(x^t) then
                        return x'
        end
    end
    return ne postoji pomak
```

Konačno, treba napomenuti da u slučaju da su sve agregatne funkcije linearne ili sve mogu određenim transformacijama svesti na linearne funkcije (kao što je i slučaj kada je reč o MMKP modelu problema) mogao bi se pojednostaviti postupak na taj način sto će se računati samo doprinos (odnosno promena u vrednosti funkcije cilja) koji bi se ostvario uključivanjem datih servisa u kompoziciju. To jest u navedenim algoritmima se u svakoj iteraciji umesto računanja vrednosti funkcije cilja (što u opštem slučaju može zahtevati veliki računarski napor) može samo proveriti da li je $f'_{ij} \leq f^{t+1}_{il}$ odnosno $f'_{ij} < f_{il.}$

Kriterijum zaustavljanja

U opštem slučaju pretraživanje staje kada nije više moguće postići poboljšanje funkcije cilja odnosno, kada u okolini tekućeg rešenja nije moguće pronaći nijedno rešenje koje je bolje od njega, odnosno nijedan pomak koji bi vodio kao poboljšanju vrednosti funkcije cilja.

Sa druge strane, ukoliko je potrebno, vreme izvršavanja algoritma se može ograničiti bilo vremenski bilo zadavanjem maksimalnog broja iteracija. Pored toga, ukoliko je primenjena neka druga metoda za izbor sledećeg rešenja, iterativni postupak se može zaustaviti ako u određenom, unapred zadatom, broju iteracija nije ostvareno poboljšanje funkcije cilja.

Ukoliko se izbor sledećeg rešenja u svakoj od iteracija zasniva na principu tzv. iterativnog poboljšavanja onda će, u opštem slučaju, poslednje dobijeno rešenje ujedno predstavljati i optimum. Treba napomenuti da, kako se prilikom izbora pravca kretanja ispituje samo neposredna okolina tekućeg rešenja, takav optimum zapravo predstavlja *lokalni* optimum:

$$f(x^{optLocal}) \leq f(x), \forall x \in N(x^{optLocal}).$$

Međutim ,ovako dobijeno rešenje u opštem slučaju ne mora biti i globalni optimum o čemu će biti više reči nešto niže.

Algoritam lokalnog pretraživanja

Algoritam lokalnog pretraživanja koji će se koristiti u ovom radu, kao što je naznačeno u prethodnim razmatranjima, podrazumeva da se susedna rešenja dobijaju principom jednostavne zamene.

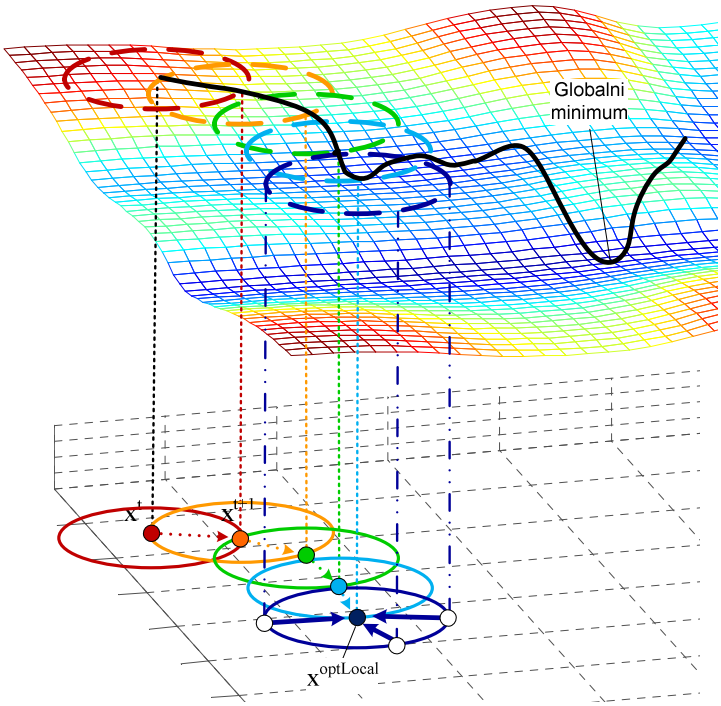
Algoritam pri tome polazi od nekog početnog dopustivog rešenja x^0 koje se može generisati bilo nasumično bilo primenom algoritma predloženog u poglavlju 5.2.2. ukoliko je reč o MMKP modelu problema. Zatim se, u svakoj iteraciji, u zavisnosti od izbora korisnika, može bilo kompletno pretražiti okolina tekućeg rešenja pre nego što se izabere sledeće rešenje u nizu, bilo primeniti princip prvog poboljšanja. Ukoliko vrednost funkcije cilja u pronađenom rešenju nije gora od vrednosti funkcije cilja u tekućem rešenju pretraga se premešta u dato novo rešenje koje postaje tekuće rešenje u

sledećoj iteraciji i pretraživanje se na isti način nastavlja počevši od novog rešenja. Ako nije moguće pronaći rešenje iteriranje staje. Konačno sam algoritam se može predstaviti na sledeći način:

Algoritam 4. Lokalno pretraživanje

```
function LOCAL_SEARCH(x^0, vrsta poboljšanja)
    x* = x^0; f* = f(x^0);
    repeat
        if vrsta poboljšanja = BEST_MOVE
            x^t = BEST_ MOVE(x*)
        else
            x^t = FIRST_IMPROVING_ MOVE(x*)
        if f(x^t) ≤ f* then
            x* = x^t; f* = f(x^t);
        else
            return x*
    return x*
```

U idealnom slučaju trajektorija bi trebalo da vodi kroz sve lokalne optimume. Međutim, može se desiti da se pretraživanje „zaglavi“ u nekom lokalnom optimumu, odnosno u dolini (*eng. valley*) u kojoj se taj lokalni optimum nalazi, budući da u svakom koraku izbor pravca u kome će se nastaviti kretanje zavisi isključivo od lokalnih karakteristika funkcije cilja u okolini tekućeg rešenja. Drugim rečima, pretraživanje može stati pre nego što se dosegne globalni optimum. Naime, budući da se u opštem slučaju u metodi lokalnog pretraživanja dozvoljavaju samo pomaci koji vode ka poboljšavanju funkcije cilja, može se desiti da se pretraživanje „zaglavi“ u okolini nekog lokalnog optimuma iz koje se ne može „izvući“ korišćenjem samo poboljšavajućih pomaka.



Slika 31. Nedostatak lokalnog pretraživanja

Očigledno je da, u neposrednoj okolini najboljeg pronađenog rešenja $N(x^{optLocal})$ ne postoji nijedno rešenje koje je bolje od njega, ali da pri tome u nekoj daljoj okolini postoji dublja dolina koja zapravo sadrži bolje rešenje (odnosno globalni optimum).

Dakle, nedostatak metoda lokalnog pretraživanja se ogleda u tome što je, shodno njihovom nazivu, pretraga usmerena samo na manji deo dopustivog prostora. Kako bi se pretraga usmerila i na druge obećavajuće oblasti, u kojima bi se mogla nalaziti i znatno bolja rešenja, potrebno je na neki način diversifikovati pretragu. Drugim rečima, ukoliko je dolina u kojoj se nalazi lokalni optimum suviše velika da bi se pretraga „izvukla“ iz takve doline potrebno je proširiti pretraživanje i na neke druge oblasti prostora dopustivih rešenja *X*.

Generalno gledano, kako se navodi u (*Gendreau, 2003*) potrebno je obezbediti mehanizme koji će omogućiti da se pretraga na neki način usmerava i po potrebi dodatno intenzivira (u oblastima za koje je prethodno utvrđeno da su obećavajuće) ali i diversifikuje (odnosno usmeri u nove oblasti). Stoga, različite metode koje su zasnovane na lokalnom pretraživanju (kao što su VNS, tabu pretraživanje, simulirano kaljenje, multi-start local search itd.) koriste različite tehnike kako bi se tokom pretrage izbeglo zaglavljivanje u lokalnim optimumima odnosno dolinama u kojima se ti lokalni optimumi nalaze.

5.2.2. Pohlepna heuristika sa vraćanjem unazad za pronalaženje početnog dopustivog rešenja

Imajući u vidu da je prilikom generisanja početnog rešenja potrebno obezbediti da ono zadovoljava sva postavljena ograničenja, moglo bi se desiti da se u nasumičnom generisanju utroši veliki napor da bi se pronašlo neko dopustivo rešenje. Šta više, kako se navodi u (*Hanafi & al, 2010*), čak i pronalaženje nekog dopustivog rešenja MMKP problema NP-težak problem.

U ovom radu se predlaže jedna nova heuristika za pronalaženje početnog dopustivog rešenja MMKP problema koja pripada grupi konstruktivnih metoda, čiji je cilj da, primenom odgovarajućih inteligentnih pravila, generišu samo jedno dopustivo rešenje problema koje treba da bude blisko optimumu (*Krčevinac & al, 2004*). Predložena heuristika je zasnovana na principu pohlepnog dodavanja (*eng. greedy add approach*) te će se u svakom koraku, iz skupa mogućih servisa, izabrati onaj servis koji je „najvredniji“ u odnosu na postavljene kriterijume i dodati u kompoziciju. Pored toga u svakom koraku se vodi računa i da izabrano rešenje pri tome obezbeđuje zadovoljenje svih postavljenih ograničenja. Pri tome, predložena metoda kombinuje princip pohlepnosti sa tehnikom vraćanja unazad (*eng. backtracking*).

Predložena metoda podrazumeva da je najpre potrebno svakoj od komponenti kompozicije dodeliti težinu (koja ukazuje na značaj te komponente u skladu sa preferencama korisnika) i komponente se zatim sortiraju, u opadajućem redosledu, prema dodeljenim težinama. Nakon toga se za svaku od komponenti, skup servisa koji je na raspolaganju za tu komponentu sortira prema rastućem redosledu korisnosti čime se zapravo ukazuje na „vrednost“ koju svaki od servisa ima.

Algoritam polazi od početnog rešenja koje je prazno tj. od kompozicije u kojoj nijednoj od komponenti nije dodeljen servis. Najpre se razmatra skup servisa koji je na raspolaganju za onu komponentu koja ima najveću težinu i iz njega se bira onaj servis koji ima najveću korisnost. Ukoliko bi uključivanje datog servisa u kompoziciju vodilo ka narušavanju postavljenih ograničenja prelazi se na sledeći servis u datom sortiranom skupu servisa za tu komponentu. Pretraživanje se nastavlja sve dok se ne pronađe dopustivi servis ili dok se zadati skup ne iscrpi. Zatim se prelazi na sledeću komponentu po težini i isti postupak se ponavlja se za tu komponentu itd.

Ukoliko, u nekom koraku, nijedan servis koji je na raspolaganju za datu komponentu nije moguće uključiti u kompoziciju a da se pri tome ne naruše postavljena ograničenja, algoritam se vraća unazad (*eng. backtrack*) na prethodno razmatranu komponentu. Servis koji je u prethodnom koraku dodeljen datoj komponenti se isključuje iz rešenja i pretraživanje sortiranog skupa raspoloživih servisa se nastavlja, počevši od isključenog servisa, u potrazi za sledećim dopustivim servisom. Kada se takav servis pronađe, pretraga se nastavlja od prvog servisa sledeće komponente.

Algoritam se na isti način nastavlja, krećući se unapred ili vraćajući se unazad, sve dok svakoj od komponenti ne bude pridružen servis. Pri tome predloženi algoritam može rezultovati nedopustivim rešenjem samo ukoliko zapravo ne postoji dopustivo rešenje problema. Naime, budući da on dozvoljava samo dopustiva rešenja i vraća se unazad ukoliko takvo rešenje nije moguće pronaći, u najgorem slučaju ovaj algoritam će rezultovati potpunom pretragom (*eng. exhaustive search*).

Algoritam 5. Generisanje dobrog početnog rešenja MMKP problema

function GREEDY_ADD_WITH_BACKTRACKING(w)

$x^0 = \{x_i\}$, $x_i = [0,...,0]$, $i = 1,...,m$;

Sortirati komponente prema opadajućem redosledu vektora težina w;

Sortirati servise za svaku komponentu prema rastućem redosledu korisnosti f_{ij};

for $i = 1...m$ // *po sortiranim komponentama*

for $j = 1...p_i$ // *po sortiranim servisima*

$x^0_{ij} = 1$;

if x^0 je dopustivo **then**

$i = i + 1$; $j = 1$; // *preći na sledeću komponentu*

else

$x^0_{ij} = 0$; $j = j + 1$;

end

if $j > p_i$ **then**

$i = i - 1$; // *vratiti se na prethodnu komponentu*

$x_{il} = 0$; // *isključiti servis* s_{il} *koji je bio izabran za datu komponentu*

$j = l+1$; // *preći na sledeći servis po redu*

end // *za svaku od komponenti je izabran neki servis*

return x^0 ako je pronađeno u suprotnom ne postoji dopustivo rešenje.

Kao što se može videti iz izloženog ovaj algoritam je zapravo projektovan tako da obezbedi „dobro“ početno rešenje, što može biti od velikog značaja ako se ima u vidu da su kod mnogih kombinatornih problema lokalni optimumi često veoma blizu jedan drugome (*Hansen & Mladenović, 2003a*). Ovako dobijeno početno rešenje se zatim može poboljšavati primenom bilo koje egzaktne ili heurističke metode.

Konačno, treba napomenuti da se princip pohlepnog dodavanja sa vraćanjem unazad se u literaturi uspešno primenjuje za MMKP problem dok se ideja sortiranja servisa (za određenu komponentu) prema korisnosti, kada je reč o selekciji servisa predlaže u (*Luo & al, 2008*) s tim da se u navedenom radu algoritam ne vraća unazad. Pored toga, metoda koja se u ovom radu predlaže podrazumeva da se i same komponente takođe sortiraju prema korisnosti.

5.2.3. Metoda promene okolina (VNS)

VNS (*eng. Variable Neighborhood Search*) predstavlja metaheuristiku koja je razvijena 90-tih godina prošlog veka (*Mladenović, & Hansen, 1997; Hansen & Mladenović, 1999*). čija se osnovna ideja ogleda u dinamičkoj promeni okolina tokom pretraživanja, čime se zapravo proširuje radijus pretrage sa ciljem da se izbegne „zaglavljivanje“ u lokalnim optimumima odnosno u dolinama u kojima se ti lokalni optimumi nalaze. Naime, ukoliko je dolina u kojoj se nalazi lokalni optimum suviše velika, da bi se pretraga „izvukla“ iz takve doline potrebno je proširiti pretraživanje i na neke druge oblasti prostora dopustivih rešenja (X) pa će se, umesto samo jedne strukture okolina, zapravo koristiti ceo skup predefinisanih struktura okolina. Dakle, VNS metoda, za razliku od većine drugih metoda zasnovanih na lokalnom pretraživanju, ne prati određenu trajektoriju već sistematično (ili u nekim varijantama i nasumično) pretražuje okoline koje su sve dalje i dalje od tekućeg rešenja kako bi se pronašao globalni optimum ili u najmanju ruku bolji lokalni optimum.

Pri tome, kako autori navode u (*Hansen & Mladenović, 2003a*), osnovna ideja VNS metode se zasniva na tri opservacije. Prvo, lokalni optimum za jednu strukturu okoline ne mora nužno biti i lokalni optimum neke druge strukture okoline. Drugo, globalni optimum zapravo predstavlja lokalni optimum u odnosu na moguće strukture okolina. Konačno, u mnogim kombinatornim problemima lokalni optimumi su često veoma blizu jedan drugome i obično se nalaze u nekoliko manjih delova dopustive oblasti, te se može zaključiti da će pronađeni lokalni optimumi zapravo obezbeđivati implicitne informacije o drugim bliskim optimalnim rešenjima koja su bolja od njih ili čak globalno optimalna.

U skladu sa tim VNS, u opštem slučaju, tokom pretraživanja dozvoljava samo pomake u bolja susedna rešenja, te ukoliko se takvi pomaci ne mogu naći, vrši se prestruktuiranje okoline trenutnog rešenja i u tako formiranoj novoj okolini se nastavlja pretraga (bilo nasumično bilo lokalnim pretraživanjem). Ukoliko u novoj okolini i dalje nije moguće pronaći bolje rešenje ponovo će se promeniti okolina i tako dalje sve dok se ne pronađe neki pravac kretanja koji bi obezbedio poboljšanje nakon čega se pravi korak u tom pravcu i pretraživanje se na isti način nastavlja počevši od prve okoline novog rešenja.

Primena VNS metode zahteva da se najpre definiše struktura okolina oko bilo koje tačke iz skupa dopustivih rešenja. U tom smislu se definiše ceo skup okolina $N_1(x), N_2(x), ..., N_{d_{\max}}(x)$ koje u opštem slučaju ne moraju biti povezane, pri čemu se broj okolina d_{max} unapred zadaje kao parametar. Pri tom, prema (*Hansen & Mladenović, 2005*), ukoliko je potrebno obezbediti da se pretraže sve doline, odnosno kako bi se pokrila cela dopustiva oblast *X*, okoline treba da budu definisane tako bude zadovoljen sledeći uslov:

$$X \subseteq N_1(x) \cup N_2(x) \cup ... \cup N_{d_{\max}}(x), \forall x \in X.$$

Generalno gledano, svaka od okolina može imati drugačiju strukturu odnosno biti definisana pomoću drugačije metrike. Alternativno, za definisanje strukture okolina može se koristiti ista metrika pri čemu će se u tom slučaju promena strukture zapravo ostvariti povećavanjem rastojanje između tekućeg rešenja i njemu susednih rešenja.

Konkretno, za razmatrani problem selekcije servisa za kompoziciju, izabran je ovaj drugi pristup, pri čemu se kao metrika i dalje uzima Hemingovo rastojanje koje će se povećavati. Stoga se okolina N_d nekog rešenja (u kome je svakoj od komponenti kompozicije pridružen tačno po jedan servis) može definisati kao skup svih onih rešenja koji se od njega razlikuju po tačno *d* servisa. Drugim rečima, pomak će se ostvariti tako što će, u ukupno *d* komponenti datog rešenja, izabrani servis biti zamenjen nekim od preostalih servisa koji su na raspolaganju za datu komponentu, što se zapravo postiže prebacivanjem (*eng. flip*) njima odgovarajućih promenljivih (x_{ij}) sa *1* na *0* kada servis s_{ij} nije više uključen u rešenje odnosno sa *0* na *1* kada servis s_{ij} treba da bude uključen u rešenje.

Uzimajući u obzir da je takođe neophodno odrediti redosled po kome će se okoline ispitivati, a imajući u vidu i kompleksnost samog pomaka kao i prethodno iznetu opservaciju da su lokalni optimumi često blizu jedan drugome i da obezbeđuju neke podatke o globalnom optimumu, u ovom radu će se usvojiti pretpostavka da su okoline ugnježdene (*eng. nested*) odnosno da svaka sledeća zapravo sadrži prethodnu:

$$N_1(x) \subset N_2(x) \subset ... \subset N_{d_{\max}}(x), \forall x \in X.$$

Samim tim, redosled ispitivanja okolina je prirodno određen rastućom dimenzijom okoline ($|N_d(x)|$). Pored toga, kako autori navode u (*Hansen & Mladenović, 2003b*) definisanje okolina na ovakav način omogućava da se neke povoljne karakteristike tekućeg rešenja (npr. činjenica da je većina promenljivih već dostigla svoju optimalnu

vrednost) zadrže i iskoriste za pronalaženje obećavajućih oblasti koje mogu sadržati rešenja bliska optimalnom. Naime, kako autori navode, budući da se ne može znati koje su promenljive u pitanju potrebno je detaljnije ispitati okoline tekućeg lokalnog optimuma u pokušaju da se se pronađe bolji lokalni optimum. Šta više, bliže okoline će zapravo biti temeljnije pretražene od daljih okolina (a ukoliko u tim bližim okolinama nije moguće ostvariti poboljšanje pretraga će zatim biti usmerena i ka daljim okolinama).

Sa druge strane, kako se navodi u (*Hansen & Mladenović, 2005*) ovakav pristup definisanju struktura okolina je jednostavnije implementirati budući da se svaka sledeća okolina dobija elementarnim pomakom iz prethodne što praktično znači da je samo potrebno definisati vrstu elementarnog pomaka a zatim se promena okolina postiže njegovom iterativnom primenom. U ovom radu, kao što je u navedeno u prethodnom poglavlju, elementarni pomak podrazumeva zamenu samo jednog servisa u rešenju (s_{ij}) nekim drugim servisom koji na raspolaganju za datu komponentu (s_{il}), te se zapravo Hemingovo rastojanje u svakoj iteraciji povećava za 2[1].

Kao što je u prethodnom poglavlju ilustrovano primenom definisanog pomaka moguće je generisati ceo skup probnih rešenja (kandidata) koji će se u datom koraku lokalnog pretraživanja ispitivati. Međutim, treba napomenuti da ukoliko se prilikom lokalnog pretraživanja primenjuje princip prvog poboljšanja onda ovako definisan pomak zapravo ne zahteva da se unapred generišu sva moguća susedna rešenja, već će se u svakom koraku zameniti samo po jedan servis u kompoziciji.

Treba napomenuti da postoji čitav niz varijanti VNS metode koje se razlikuju pre svega po tome da li su determinističke, stohastičke ili kombinovane, a zatim i po tome na koji način se poboljšava tekuće rešenje (primenom lokalnog pretraživanja ili nasumičnom pretragom) i konačno u kom trenutku, pod kojim uslovima i na koji način će se izvršiti promena okolina (pri čemu se, u zavisnosti od izabrane metode, ona može izvršiti prilikom izvlačenja iz dolina i/ili i prilikom lokalnog pretraživanja). U nastavku će ukratko biti izložene osnovne varijante.

Deterministička varijanta VNS metode, tzv. metoda promene okolina u spustu (*eng. Variable Neighborhood Descent – VND*), se prema (*Hansen & Mladenović, 2005*)

[1] Treba napomenuti da je ovako definisan pomak zadovoljava zahteve u pogledu kompleksnosti njegove primene budući da se ostvaruje jednostavnom zamenom dve promenljive u rešenju.

zasniva na prethodno iznetoj opservaciji da lokalni optimum za jednu strukturu okoline ne mora nužno biti i lokalni optimum neke druge strukture okoline. Stoga VND podrazumeva da se prilikom lokalnog pretraživanja umesto samo jedne strukture okoline koristi ceo skup predefinisanih okolina. Naime, najpre će se ispitati prva okolina tekućeg rešenja, pri čemu se za izbor sledećeg pravca kretanja najčešće primenjuje princip najboljeg poboljšanja mada se, ukoliko je reč o vremenski kritičnim problemima, može primeniti i princip prvog poboljšanja. Ukoliko nije pronađen nijedan pomak koji bi vodio ka poboljšanju funkcije cilja pretražiće se sledeća okolina itd. Sa druge strane, ukoliko je pronađeno neko rešenje kojim se ostvaruje poboljšanje funkcije cilja (bolji lokalni optimum), pretraga se premešta u dato rešenje i postupak se nastavlja počevši od prve okoline tog novog rešenja. Budući da pretraga staje kada se pretraže sve okoline, a da pri tome nije pronađen nijedan novi pomak koji bi vodio ka poboljšanju funkcije cilja, očigledno je da će dobijeno rešenje predstavljati lokalni optimum u odnosu na sve definisane okoline. Pri tome, ukoliko su okoline definisane tako da pokrivaju celu dopustivu oblast onda pronađeni lokalni optimum zapravo predstavlja i globalni optimum.

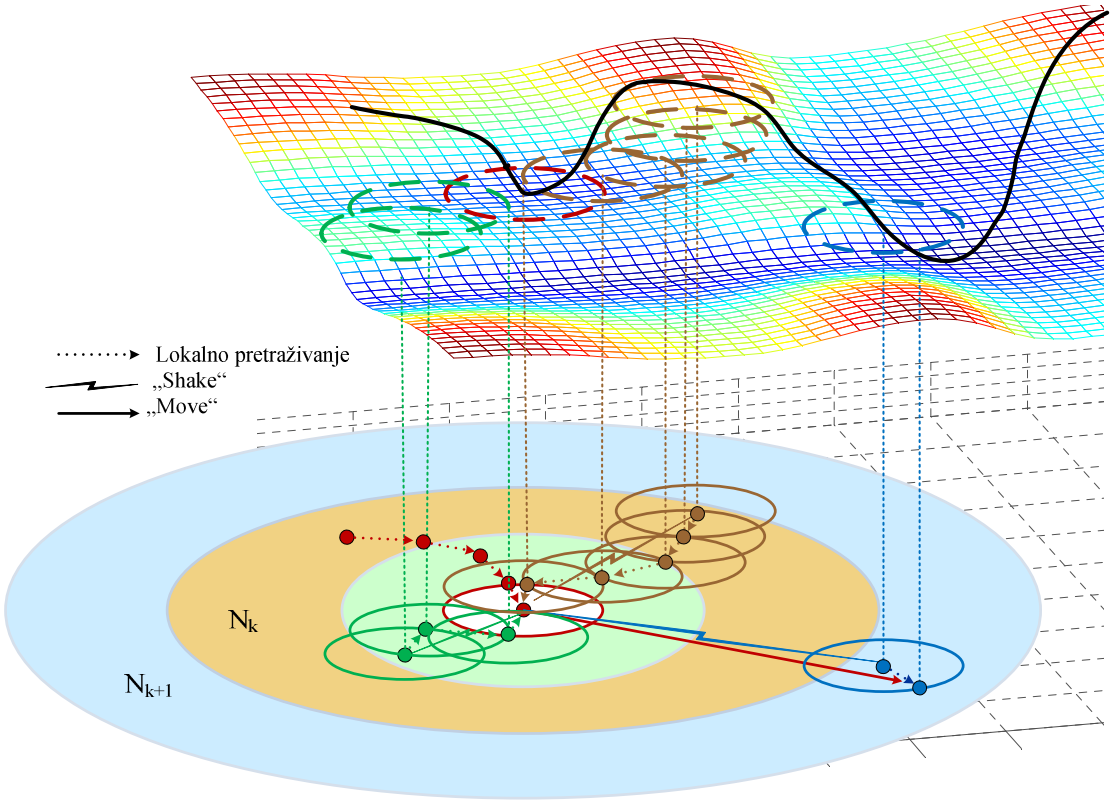
Sa druge strane, tzv. redukovana metoda promene okolina (*eng. Reduced Variable Neighborhood Search – RVNS*) predstavlja potpuno stohastičku varijantu VNS metode koja ne podrazumeva primenu lokalnog pretraživanja već nasumičan izbor probnih rešenja u pokušaju da se pronađe bolje rešenje. Međutim, za razliku od nasumičnog ispitivanja cele dopustive oblasti, osnovna ideja ove metode se prema (*Hansen & Mladenović, 2005*) zasniva na opservaciji da su lokalni optimumi često veoma blizu jedan drugome i da se obično nalaze u nekoliko manjih delova dopustive oblasti te da je poželjno najpre ispitati okolinu tekućeg rešenja a ukoliko se ne pronađe bolje rešenje postepeno će se ispitivati sve dalje i dalje okoline. Drugim rečima osnovna ideja ove metode se zapravo ogleda u sistematičnoj promeni okolina (za koje se najčešće pretpostavlja da su ugnježdene) oko tekućeg rešenja pri čemu je u svakom koraku nasumično ispitivanje ograničeno isključivo na tekuću okolinu. Stoga će, ako se pretpostavi da su okoline ugnježdene, bliže okoline zapravo biti temeljnije ispitane a ukoliko njima nije moguće ostvariti poboljšanje pretraga će zatim biti usmerena i ka daljim okolinama. Preciznije rečeno, najpre se nasumično bira neko rešenje (tzv. „*shake*“) iz prve okoline tekućeg rešenja. Ukoliko dato rešenje nije bolje od tekućeg

rešenja prelazi se na sledeću okolinu i iz nje se opet nasumično bira neko susedno rešenje itd. Kada se pronađe neko bolje rešenje pretraga se premešta u to rešenje i nasumični izbor probnih rešenja se nastavlja počevši od prve okoline novog rešenja. Kada se razmotre sve okoline postupak ponovo započinje od prve okoline tekućeg, najboljeg pronađenog, rešenja i nastavlja se na isti način dokle god se ne ispuni neki unapred zadati kriterijum zaustavljanja. Kako autori navode u (*Hansen & Mladenović, 2001*) ova metoda je veoma pogodna kada je reč o problemima velikih dimenzija, kod kojih lokalno pretraživanje može zahtevati prevelik računarski napor, ako je potrebno brzo pronaći neko dobro rešenje koje ne mora nužno biti blizu optimalnom. Sa druge strane ova metoda se, imajući u vidu navedeno, često primenjuje i za pronalaženje „dobrog" početnog rešenja.

Konačno, osnovna varijanta (*eng. Basic Variable Neighborhood Search*) za koju se u literaturi najčešće koristi samo skraćenica VNS, podrazumeva kombinovanje neke metode klasičnog lokalnog pretraživanja[1], kako bi se pronašli lokalni optimumi, i nasumičnog sistematičnog ispitivanja sve daljih okolina oko pronađenih lokalnih optimuma, kako bi se pronašle nove obećavajuće oblasti u kojima će se iznova primeniti lokalno pretraživanja, te zapravo obuhvata i determinističke i stohastičke komponente. Naime, prema (*Hansen & Mladenović, 2005*) osnovna ideja ove metode se ogleda u tome da se postepeno pretražuju sve dalje i dalje okoline oko tekućeg, odnosno do tada najboljeg pronađenog, lokalnog optimuma pri čemu se u svakom koraku neko rešenje, koje je nasumično izabrano iz tekuće okoline ovog rešenja, koristi kao polazno rešenje za lokalno pretraživanje koje će voditi ka nekom novom lokalnom minimumu. Pri tome se, kako se navodi u (*Hansen & al, 2010*), pravac pomeranja bira nasumično (tzv. „*shake*") kako bi se u potrazi za nekim drugim obećavajućim oblastima (tj. dubljim dolinama), sa jedne strane obezbedila mogućnost da se dostigne bilo koja tačka (u bilo kojoj dolini dopustive oblasti X), a sa druge stane sprečilo eventualno vraćanje u neko od prethodno ispitanih rešenja (kao što bi se moglo desiti ukoliko bi se primenilo neko determinističko pravilo).

[1] Ukoliko se za lokalno pretraživanje primeni VND dobija se opšta metoda promene okolina (*eng. General Variable Neighborhood Search*) koja takođe podrazumeva da se pre početka izvršavanja alogritma početno rešenje najpre poboljša primenom RVNS metode. Pri tome treba napomenuti da u opštem slučaju strukture okolina koja će se koristi prilikom lokalnog pretraživanja primenom VND metode ne mora biti ista kao strukture okolina koja će se koristiti prilikom pronalaženja novih obećavajućih oblasti.

Dakle, najpre se rešenje, koje je nasumično generisano u prvoj okolini tekućeg rešenja, primenom neke metode lokalnog pretraživanja poboljšava sve dok se ne dostigne lokalni optimum. Ukoliko pronađeni lokalni optimum nije bolji od tekućeg najboljeg lokalnog optimuma opisani postupak (tj. nasumični izbor rešenja i lokalno pretraživanja kako bi se iz tog susednog rešenja stiglo u lokalni optimum) se nastavlja u sledećoj okolini itd. Pri tome će se pretraga premestiti (tzv. „*move*“) iz trenutnog lokalnog optimuma u neko drugo rešenje ako i samo ako bi se time ostvarilo poboljšanje vrednosti funkcije cilja, nakon čega će pretraživanje iznova započeti ali sada od prve okoline ovog novog najboljeg lokalnog optimuma. Ukoliko se ispitaju sve okoline tekućeg najboljeg rešenja, a da pri tome nije pronađeno nijedno rešenje koje je bolje od njega, pretraga ponovo započinje od prve okoline tog rešenja i nastavlja se na isti način. Ceo postupak, u kome se smenjuju faza diversifikacije (u kojoj se pretraga postepeno proširuje u nove oblasti) i faza intenzifikacije (u kojoj se u tim oblastima traži novi lokalni optimum), se iterativno ponavlja dokle god se ne dostigne neki unapred zadati kriterijum zaustavljanja (Slika 32.).



Slika 32. Metoda promene okolina (VNS)

Konačno, treba napomenuti da se u fazi lokalnog pretraživanja mogu koristiti različite metode čak i nelinearnog programiranja (*Mladenović & al, 2008*). Kao što je pokazano

u (*Hanafi & al, 2010*) VNS se može uspešno primeniti za rešavanje MMKP problema dok se u (*Mladenović & al, 2008*) pokazuje i da se ona može uspešno primeniti i za rešavanje nelinearnih problema.

Konkretno, VNS algoritam koji je primenjen u ovom radu polazi od nekog početnog dopustivog rešenja x^0 (koje se može dobiti bilo nasumično bilo primenom algoritma opisanog u prethodnom poglavlju ukoliko je reč o MMKP modelu problema) u kome je svakoj od komponenti kompozicije pridružen tačno po jedan servis. Ovo rešenje se zatim optimizuje primenom lokalnog pretraživanja kako bi se odredio lokalni optimum koji postaje prvi najbolji lokalni optimum x^*.

Počevši od prve okoline tekućeg najboljeg lokalnog optimuma ($N_1(x^*)$), u svakoj iteraciji je najpre potrebno nasumično izabrati pravac kretanja („*shake*“) koji će omogućiti da se napusti dolina u kojoj se taj lokalni optimum nalazi. Drugim rečima, u svakoj iteraciji će se nasumično generisati neko novo rešenje x' što se zapravo postiže isključivanjem d nasumično izabranih servisa iz tekućeg rešenja (gde d odgovara dimenziji okoline koja se u tom trenutku ispituje) i uključivanjem u rešenje d drugih servisa, koji su takođe nasumično izabrani, vodeći pri tome računa da svaki isključeni servis može biti zamenjen isključivo nekim od servisa koji obezbeđuje istu funkcionalnost odnosno koji je na raspolaganju za komponentu kompozicije kojoj je bio pridružen isključeni servis. Preciznije rečeno, budući da su susedne okoline ugnježdene (odnosno da se promena okoline ostvaruje zamenom samo jednog servisa u rešenju) u svakoj iteraciji se zapravo nasumično bira jedna promenljiva x_{il} koja će biti uključena u rešenje (tj. postavljena na jedinicu) dok će promenljiva koja odgovara servisu koji je tada potrebno isključiti iz rešenja (x_{ij}) biti postavljena na nulu.

Zatim se primenjuje metoda lokalnog pretraživanja (prikazana u prethodnom poglavlju) kako bi se od generisanog rešenja x' stiglo do novog lokalnog optimuma x^t. Pri tome treba napomenuti da će se u okviru izloženog algoritma lokalne pretrage u svakom koraku, u zavisnosti od preferenci korisnika, bilo u potpunosti pretražiti neposredna okolina ovog rešenja pre nego što se izabere najbolji pomak (tj. pomak koji vodi ka rešenju u kome funkcija cilja ostvaruju najbolju vrednost u datoj okolini), bilo pronaći prvi najbolji pomak (budući da potpuno pretraživanje okoline može biti veoma vremenski zahtevno), kao i da će se prilikom evaluacije pomaka razmatrati isključivo dopustiva rešenja. Ukoliko je pronađeni lokalni optimum x^t bolji od tekućeg lokalnog

optimuma pretraga se premešta u taj novi lokalni optimum i ceo proces počinje iz početka počevši od prve okoline (N_1) novog rešenja. U suprotnom, bilo da je pretraga dovela do vraćanja u isti lokalni optimum, bilo da je dovela do pronalaženja nekog novog lokalnog optimuma ali koji nije bolji od tekućeg, prelazi se na pretraživanje sledeće okoline tekućeg rešenja ($N_{d+1}(x^*)$) i tako na dalje (prema redosledu okolina) sve dok se u nekoj od njih ne pronađe bolji lokalni optimum. Ukoliko se ispita i poslednja okolina d_{max}, a da pri tome nije pronađeno bolje rešenje od tekućeg, ceo ciklus pretraživanja ponovo započinje od prve okoline tekućeg rešenja $N_1(x^*)$, sve dok se ne dostigne postavljeni kriterijum zaustavljanja. Pri tome se ovaj kriterijum može zadati kao maksimalno dozvoljeno vreme izvršavanja, maksimalni broj iteracija, broj iteracija (ili vreme) u kojima nije ostvareno poboljšanje itd.

Algoritam 5. Metoda promene okolina

```
function VNS (x^0, d_max , kriterijum zaustavljanja, vrsta poboljšanja)
    x* = x^0; f* = f(x^0);
    repeat
        d=1;
        repeat
            x' = RANDOM(N_d(x*));   // Shake
            x^t = LOCAL_SEARCH(x', vrsta poboljšanja);
            if f(x^t) ≤ f* then
                x* = x^t; f* = f(x^t);   // Move
                d = 1;
            else
                d =d+1;   // Promena okoline
        until d = d_max
    until kriterijum zaustavljanja
    return x*
```

Treba napomenuti da se u metodi RANDOM koristi uniformna raspodela verovatnoća za nasumično generisanje servisa (odnosno promenljive) koji će biti uključen u rešenje umesto nekog servisa koji je deo tekućeg rešenja, budući da kako je navedeno u (*Mladenović & al, 2008*) ova raspodela predstavlja očigledan izbor.

Prema (*Hansen & Mladenović, 2009*) prednost VNS metode u odnosu na većinu drugih heuristika (kao što su tabu pretraživanje, simulirano kaljenje, genetski algoritmi...) se ogleda u tome što ona ne zahteva podešavanje velikog broja dodatnih parametara, ili čak u nekim slučajevima ne mora zahtevati nijedan dodatni parametar. Naime, u opštem slučaju, pored izbora koji su zajednički za sve metode lokalnog pretraživanja (struktura okoline, način generisanja početnog rešenja, pravilo izbora sledećeg rešenja u svakoj iteraciji i kriterijuma zaustavljanja) potrebno je samo unapred zadati broj okolina d_{max}. Sa druge strane, VNS metoda se može lako primeniti na bilo koji problem ako se definiše način na koji će se dobiti početno rešenje, način na koji će se nasumično generisati rešenje u nekoj okolini („*shake*“), koja metoda će se koristiti za lokalnu pretragu i način na koji će se ispitivati da li je pronađeno rešenje bolje od prethodnog („*move*“). Pored toga, kako se navodi u (*Hansen & al, 2010*) VNS daje veoma dobre rezultate, često na jednostavniji način u poređenju sa drugim metodama, ali i što je još važnije postoji i obrazloženje za takve rezultate.

Izloženi algoritam je u ovom radu primenjen kako na MMKP model problema selekcije servisa za kompoziciju, tako i na problem modelovan predloženim pristupom zasnovanim na primeni konzistentne fazi logike i rezultati eksperimenata su dati u Poglavlju 5.2.5.

Konačno, treba napomenuti da imajući u vidu da se VNS, kako se navodi u (*Hansen & al, 2010*) uspešno primenjuje u čitavom nizu oblasti uključujući i optimizaciju na grafovima kao i probleme raspoređivanja, ona bi se mogla primeniti i za rešavanje problema selekcije servisa za kompoziciju modelovanog i na neki od drugih načina izloženih u Poglavlju 3.3. kao što su: MCOP (Multi-Constrained Optimal Path problem) ili RCPSP (Resource Constrained Project Scheduling Problem) i drugi pristupi modelovanju ovog problema koji su zasnovani na grafovima.

5.2.4. Nova hibridna metaheuristika

Prilikom definisanja predložene heuristike pošlo se od činjenice da je u slučaju globalne optimizacije neophodno obezbediti odgovarajuće mehanizme pomoću kojih će se pretraga intenzivirati u onim oblastima koje su obećavajuće i diversifikovati kako bi se usmerila u nove, do tada neispitane, oblasti dopustivog prostora. Stoga je ideja bila da se kombinuju dve poznate metaheuristike (VNS i tabu pretraživanje) koje primenjuju različite mehanizme kako za izvlačenje iz lokalnih optimuma tako i za proširivanje pretrage u neke druge oblasti ali i za intenziviranje pretrage u obećavajućim oblastima.

Stoga će u nastavku prvo biti opisana metoda tabu pretraživanja, zatim će se izložiti motivacija za kombinovanje ove dve metaheuristike i konačno će biti dat prikaz predložene metode.

5.2.4.1. TABU PRETRAŽIVANJE

Tabu pretraživanje (*eng. Tabu search*) predloženo u (*Glover, 1989; Glover, Februar 1990*) je takođe metaheuristika zasnovana na lokalnom pretraživanju (sa samo jednom strukturom okoline) koja primenjuje nešto drugačiju tehniku za „izvlačenje“ iz lokalnih optimuma. Naime, u ovoj metodi se sledi trajektorija sekvence rešenja (x^0, x^1,...,x^t,..., x^*) ali se, pored pomaka koji vode ka boljim rešenjima (tzv. silazeći pomaci), takođe dozvoljavaju i pomaci kojima se pogoršava vrednost funkcije cilja (tzv. uspinjući pomaci). Pri tome se prati istorija pretraživanja, odnosno određene informacije o sekvenci pomaka koja je dovela do tekućeg rešenja, kako bi se osiguralo da se pretraga u narednim iteracijama ne vrati u to isto rešenje (tzv. cikliranje). Naime, ukoliko se u nekoj iteraciji izabere uspinjući pomak (kako bi se pretraga izvukla iz doline u kojoj se nalazi trenutni lokalni optimum) moglo bi se desiti da se u nekoj od sledećih iteracija, praveći najbolji pomak koji je tom trenutku dostupan, pretraga vrati u isti lokalni optimum. Prema (*Pirlot, 1996*) ova opasnost je naročito prisutna kada je struktura okoline simetrična tj. kada je:

$$x^t \in N(x^{t+1}), \forall x^{t+1} \in N(x^t)$$

budući da može se desiti da će prethodno rešenje x^t biti najbolje rešenje u novoj okolini $N(x^{t+1})$ što bi vodilo ka tome da se pretraga opet vrati u prethodno rešenje x^t te bi u svim narednim iteracijama ona zapravo oscilirala između ova dva rešenja (x^t i x^{t+1}).

Shodno tome ključni aspekt tabu pretraživanje predstavlja korišćenje različitih memorijskih struktura, pomoću kojih će se pamtiti istorija pretrage, a koje pre svega imaju za cilj da spreče cikliranje po nekom malom skupu rešenja (tj. pravljenje sekvence pomaka koja bi vodila nazad u neko rešenje na trajektoriji koje je prethodno već bilo ispitano) ali i da primoraju pretragu da prati neku novu trajektoriju kako bi se pretraga iz prethodno ispitanih oblasti usmerila u neke nove oblasti. Preciznije rečeno, kao što je navedeno u (*Glover, Avgust 1990*), za razliku od metoda koje karakteriše odsustvo memorije (na primer simulirano kaljenje ili nasumične procedure) ili rigidne memorijske strukture (kao što je na primer slučaj kada je reč o metodama grananja i ograničavanja – *eng. branch and bound*) tabu pretraživanje se zasniva na upotrebi fleksibilnih memorijskih struktura, različitih vremenskih raspona, koje će omogućiti da se mnogo adekvatnije iskoriste informacije o istoriji pretraživanja kako bi se pretraga na inteligentan način usmeravala i po potrebi dodatno intenzivirala (u oblastima za koje je prethodno utvrđeno da su obećavajuće) i diversifikovala (odnosno usmerila u nove oblasti). Pri tome treba napomenuti da ne postoje neke opšte smernice u pogledu načina implementacije ovih memorijskih struktura i izbora informacija koje će se u njima čuvati te je sam proces zapravo veoma fleksibilan i može se lako prilagoditi različitim vrstama problema. U literaturi su prisutni brojni pristupi i modifikacije koje se neće detaljnije razrađivati u ovom radu ali se mogu pronaći na primer u (*Glover, 1989; Glover, Februar 1990; Glover, Avgust 1990; Glover, 1995; Glover, 1997; Glover & Laguna, 2013*).

U svom osnovnom obliku tabu pretraživanje zapravo podrazumeva postojanje samo kratkoročne memorije (*eng. short term memory*) čija je svrha da omogući strateško i sistematično usmeravanje pretrage time što će u određenom vremenskom periodu sprečiti ponovno ispitivanje nekog od prethodnih rešenja nakon čega će takva rešenja opet postati dostupna. Preciznije rečeno, ideja je da se identifikuju ona rešenja (x^t) koja su ispitana tokom nekog prethodnog vremenskog perioda i da se zatim, u nekoliko narednih iteracija, spreči povratak u njih time što će se ona označiti kao zabranjena odnosno „tabu“. Budući da se najčešće implementacija kratkoročne memorije svodi na pamćenje nekoliko poslednjih rešenja, odnosno rešenja koja su bila ispitana u nekom skorijem vremenskom periodu, ona se naziva i „recency-based“ memorija.

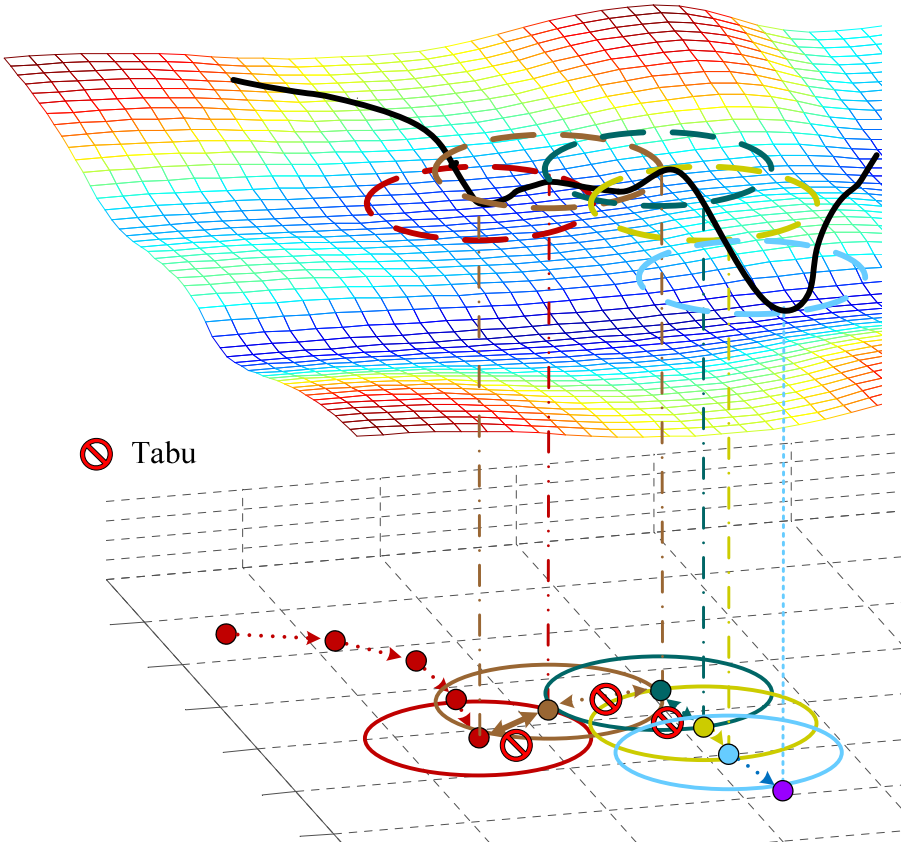
Mehanizam pomoću koga se takva rešenja pamte se naziva tabu lista (*eng. tabu list – TL*). Ova lista se pri tome konstantno ažurira pa se u noj, uopšteno govoreći, u svakom trenutku (odnosno u svakoj iteraciji t) čuva sekvenca rešenja posećenih u prethodnih s iteracija ($x^{t-s},..,x^{t-1}$). Koliko dugo će neko rešenje ostati zabranjeno, odnosno broj iteracija u kojima neće biti dozvoljen povratak u to rešenje (*eng. tabu tenure*), kao i broj rešenja koji će se zapravo pamtiti određuje se podešavanjem dužine same liste. Nakon isteka ovog roka rešenje opet postaje dozvoljeno odnosno nije više tabu. Drugim rečima, što je dužina liste veća više prethodnih rešenja će se pamtiti i ona će duže ostati zabranjena.

Na ovaj način se, u svakoj iteraciji, zapravo ograničava skup dozvoljenih (*eng. admissible*) pomaka odnosno, modifikuje se okolina tekućeg rešenja x^t koja će u datoj iteraciji biti ispitana. Konkretno, koristeći informacije sadržane u kratkoročnoj memoriji (*TL*) okolina tekućeg rešenja $N(x^t)$ se menja novom okolinom $N'(x^t)$ koja se sastoji od onih rešenja iz tekuće okoline koja nisu u datoj iteraciji zabranjena ($N'(x^t) = N(x^t) \setminus TL$). Drugim rečima u svakom koraku se formira lista kandidata (*eng. candidate list*) koja zapravo predstavlja listu rešenja koja će se u datoj iteraciji evaluirati, i u opštem slučaju, ona obuhvata sva rešenja novo-formiranoj okolini $N'(x^t)$, pri čemu se ukoliko je potrebno ova lista može i dodatno suziti[1],.

Zatim se, u ovako formiranoj novoj okolini, vrši lokalno pretraživanje evaluacijom svih raspoloživih kandidata, pri čemu se bira najbolji mogući pomak odnosno onaj pomak koje rezultuje najvećim poboljšanjem funkcije cilja, ili, ukoliko takav pomak nije moguće naći, onaj pomak koji vodi ka najmanjem pogoršanju vrednosti funkcije cilja (kako bi se pretraga usmerila u nove oblasti). Pravljenjem ovog pomaka iz tekućeg rešenja se generiše novo rešenje x^{t+1} koje postaje tekuće rešenje u sledećoj iteraciji. Pri tome se tabu lista ažurira dodavanjem tekućeg rešenje x^t u listu i izbacivanjem „najmanje nedopustivog“ rešenje iz liste tj. onog rešenja koje se najduže nalazilo u listi. Konačno, treba napomenuti da ukoliko je u nekoj iteraciji tekući skup $N'(x^t)$ prazan (tj. ukoliko su svi mogući pomaci iz tekućeg rešenja zabranjeni) tada će se nekim rešenjima u tabu listi ukinuti tabu status i to onima koja su najduže bila tabu.

[1] Kako se navodi u (*Glover, 1995*) ukoliko je data okolina suviše velika da bi se mogla efikasno ispitati ili ukoliko bi evaluacija samih rešenja u datoj okolina zahtevala prevelik računski napor i vreme, ona se može suziti primenom različitih strategija (pa čak i nasumičnog uzorkovanja kao što je slučaj kod Monte Carlo metode) a zatim po potrebi proširivati ukoliko nije moguće pronaći odgovarajući pomak.

Pretraživanje se na ovaj način nastavlja sve dok ne bude ispunjen neki unapred zadati kriterijum zasutavljanja (Slika 33.).



Slika 33. Tabu pretraživanje

Dakle, kako se navodi u (*Glover, Avgust, 1990*) tabu pretraživanje zapravo predstavlja jedan vid agresivnog lokalnog pretraživanja (u kome strukture okolina stalno evoluiraju) koje ima za cilj da omogući da se brzo stigne do lokalnih optimuma a da se zatim iz njih izvuče, na taj način što se u svakom koraku pravi najbolji mogući pomak pri čemu se zahteva da takav pomak zadovoljava određena ograničenja (zadata putem tabu listi). Dakle, može se zaključiti da je uloga tabu liste (kojom se zabranjuje pomak u neke susede iz okoline trenutnog rešenja) zapravo dvojaka, sa jedne strane ona treba da spreči, za izvesno vreme, vraćanje na neka od ranije ispitanih rešenja ali što je još važnije da omogući širenje pretraživanja u neke nove oblasti doputivog prostora X.

U skladu sa navedenim, očigledno je da će kvalitet generisanih rešenja umnogome zavisiti od načina na koji se ova lista formira i ažurira. U opštem slučaju memorija se, gledano sa aspekta informacija koje će se u njoj pamtiti, može definisati kao eksplicitna ili zasnovana na atributima (*Glover, 1995*) u zavisnosti od toga da li se

će se pamtiti cela rešenja ili samo neki podskup karakteristika tih rešenja tj. elemenata koji su promenjeni prilikom pomaka iz jednog rešenja u drugo (kao što su na primer vrednosti određenih promenljivih). Pri tome treba napomenuti da se memorija alternativno može realizovati pamćenjem sekvence poslednjih načinjenih pomaka (tj. transformacija koje su primenjene na tekuće rešenje) ili njima odgovarajućih obrnutih pomaka, odnosno samo nekih njihovih elemenata.

Prema (*Gendreau, 2003*) čuvanje kompletnog opisa poslednjih nekoliko ispitanih rešenja, a samim tim i ispitivanje da li se određeno rešenje nalazi u tabu listi, može zahtevati veoma veliki računarski napor i vreme, te se stoga ovakav pristup retko primenjuje. Sa druge strane, prema (*Pirlot, 1996*) tabu lista u kojoj se čuva samo neki podskup karakteristika (koje dato rešenje pri tome može deliti i sa drugim rešenjima) je, generalno gledano, restriktivnije nego što je potrebno jer isključuje mnogo više rešenja nego što je zapravo ispitano u poslednjih t iteracija, budući da ni neka druga rešenja (koja sa datim rešenjem dele takve tabu karakteristike) takođe neće biti dozvoljena, kao i da, u opštem slučaju, ne mora sprečiti da dođe do cikliranja. Međutim, kako autor dalje navodi ova činjenica zapravo nije od suštinskog značaja budući da je osnovna uloga tabu liste diversifikacija pretrage (odnosno proširivanje pretrage u druge oblasti dopustivog prostora koje nisu do tada ispitivane) a pre svega njeno izvlačenje iz lokalnih minimuma, i u prilog ovoj tvrdnji navodi da tabu pretraživanje zasnovano na tabu listi koja se sastoji od poslednjih ispitanih rešenja (umesto strateški identifikovanih karakteristika) najčešće daje veoma loše rezultate čak i ukoliko se zanemari vreme koje će se utrošiti na izračunavanja.

Imajući u vidu navedeno, u ovom radu je usvojen drugi princip te se u tabu listi čuvaju samo neke karakteristike rešenja odnosno, preciznije rečeno, servisi koji su u poslednih nekoliko iteracija zamenjeni. Generalno gledano, i u tom slučaju postoji nekoliko opcija. Jedna mogućnost bi bila da se pamte servisi koji su zamenjeni (tj. s_{ij}, s_{il})[1], kako bi se sprečila zamena koja bi vodila nazad ka istom rešenju, odnosno zamena servisa s_{il} sa servisom s_{ij}. Ovakav pristup je najmanje restriktivan budući da zabranjuje samo mali skup pomaka što zapravo znači da ne ograničava previše samu pretragu ali kako se navodi u (*Gendreau, 2003*) ne mora sprečiti cikliranje. Na primer,

[1] Treba napomenuti da pri tome nije neophodno pamtiti samu komponentu za koju je izvršena zamena servisa budući da se neki servis može zameniti isključivo servisom koji je na raspolaganju za istu tu komponentu.

ukoliko se za istu tu komponentu c_i zatim izabere neki novi servis s_{ik} a zatim se u sledećoj iteraciji s_{ik} zameni sa s_{ij} pretraga bi se zapravo vratila na isto rešenje. Nešto restriktivniji tabu bi zabranio vraćanje datog servisa s_{ij} u rešenje bez obzira na to kojim je servisom on u datom koraku bio zamenjen, dok bi još restriktivnija varijanta bilo pamćenje samo komponente za koju je vršena zamena (c_i) čime bi se zapravo u narednih nekoliko iteracija sprečila zamena bilo kog servisa za datu komponentu, mada ovaj poslednji pristup u opštem slučaju može biti previše restriktivan, naročito ukoliko se sama kompozicija sastoji samo od malog broja komponenti budući da bi se na taj način značajno suzila tekuća okolina, odnosno isključio prevelik broj rešenja iz razmatranja. Naime, kako se navodi u (*Gendreau, 2003*), suviše restriktivne tabu liste mogu sprečiti obećavajuće pomake (čak i kada zapravo ne postoji opasnost od cikliranja) ili dovesti do stagnacije procesa pretraživanja, pošto podrazumevaju da se implicitno zabranjuje prelazak i u neka druga rešenja koja zapravo nisu do tada ispitana a koja sa datim rešenjem dele neki broj karakteristika.

Stoga će se u ovom radu tabu status dodeliti samo onom servisu koji je u datom koraku zamenjen, te će u narednih nekoliko iteracija zapravo biti zabranjeno vraćanje ovog servisa u rešenje, što zapravo znači da su na ovaj način zabranjeni pomaci u sva ona rešenja (odnosno kompozicije) koje uključuju navedeni servis, a ne samo u dato rešenje.

Kao što je i uobičajeno tabu lista je na početku pretraživanja prazna dok je njena dužina (koja ukazuje na broj iteracija u kojima neće biti moguće ponovo razmatrati određeni servis) fiksna[1] i zadata unapred putem parametra koji je potrebno specificirati.

Generalno gledano, dužina tabu liste mora, sa jedne strane, biti dovoljno velika kako bi se sprečilo cikliranje[2] i kako se ne bi gubilo vreme u ponovnom ispitivanju istih rešenja, dok, sa druge strane, ne sme biti ni previše velika kako bi se osiguralo da je ipak moguće napraviti neki pomak. Pored toga treba imati u vidu da, kada je dužina liste relativno mala prethodno ispitana rešenja brže postaju ponovno dostupna, te se pretraga može usmeriti nazad ka oblastima koje sadrže rešenja za koje je prethodno utvrđeno da

[1] U opštem slučaju dužina tabu liste se može i menjati tokom pretraživanja bilo primenom određenih dinamičkih pravila koja će uzeti u obzir karakteristike i kvalitet atributa koji se u noj pamte (*Glover & Laguna, 2013*) bilo nasumično (*Gendreau, 2003*). Međutim, ovakav pristup u opštem slučaju zahtevati dodatni računarski napor.

[2] Kako se navodi u (*Glover, 1995*) ukoliko je dužina liste dovoljno velika verovatnoća cikliranja se smanjuje ili čak potpuno nestaje budući da se, sprečavanjem pomaka u neko od prethodnih rešenja u sekvenci, pretraga zapravo pomera sve dalje i dalje od rešenja ispitanih u prethodnim iteracijama.

su obećavajuća, obezbeđujući time intenziviranje pretrage u takvim oblastima, dok relativno velika dužina liste (kojom se značajno smanjuje broj dostupnih susednih rešenja odnosno izbora koje je moguće načiniti) može dovesti do usvajanja lošijih rešenja što će zapravo pospešiti diversifikaciju usmeravajući pretragu u druge oblasti.

Generalno gledano, najadekvatnija dužina tabu liste raste sa porastom dimenzija problema. Međutim, ne postoji opšte pravilo po kome odredila odgovarajuća dužina za određenu dimenziju problema, a pre svega zato što ona mora zavisiti od toga koliko je ona restriktivna. Prema (*Glover, 1989*) efektivne dužine liste su između 5 i 12 (sa klasterovanjem oko 7) nezavisno od dimenzija i strukture samog problema. Sa druge strane u (*Glover, Avgust 1990)* se navodi da empirijski podaci pokazuju da restriktivnije liste (tj. one koje više ograničavaju opseg dopustivih pomaka) obično daju bolje rezultate kada tabu liste manje dužine za razliku od onih koje su manje restriktivne.

Sam algoritam koji je u ovom radu primenjen može se predstaviti na sledeći način:

Algoritam 6. Tabu pretraživanje

function TABU_SEARCH(x^0, *TL*, *kriterijum zaustavljanja*)
  $x^t = x^0$; $x^* = x^0$; $f^* = f(x^0)$; *TL* = Ø;
  repeat
    x^t = **BEST_MOVE**(x^t); // *ne razmatraju se servisi u tabu listi*
    UPDATE_TABU_LIST(*TL*, *(i, j)*); //*ako je zamenjen servis* s_{ij}
    if $f(x^t) \leq f^*$ **then**
      $x^* = x^t$; $f^* = f(x^t)$;
  until *kriterijum zaustavljanja*
  return x^*

Pri tome treba napomenuti da se u proceduri BEST_MOVE zapravo pre provere da li je rešenja zadovoljava ograničenja proverava da li se ono nalazi u tabu listi. U proceduri UPDATE_TABU_LIST tabu lista se ažurira primenom FIFO (*eng. First-In-First-Out*) principa te će redosled isključivanja rešenja iz liste odgovarati redosledu njihovog ubacivanja u listu.

Prema (*Glover, 1995*) primena samo kratkoročne memorije može biti dovoljna da obezbedi dobijanje visoko kvalitetnih rešenja. Šta više, kako se navodi u (*Glover, Avgust, 1990*), kod mnogih problema primena samo kratkoročne memorije daje superiornije rezultate u poređenju sa drugim metodama, te kod takvih problema ne postoji potreba za dugoročnom memorijom. Pri tome autor dalje navodi da kratkoročna memorija sama za sebe takođe omogućava i intenziviranje (time što se određene lokalno atraktivne karakteristike tj. one koji pripadaju pomacima za koje je u skorijem periodu utvrđeno da su dobri zadržavaju) ali i diversifikaciju (time što primorava nove izbore da uključe ili isključe one karakteristike koje su u skorije vreme bile odbačene tj. uključene). Ukoliko je reč o težim problemima može biti neophodno pored kratkoročne memorije uključiti i neke dodatne komponente kao što su kriterijumi aspiracije, dugoročne tabu liste itd.

Naime, ukoliko je tabu lista suviše restriktivna (na primer ukoliko je realizovana na takav način da implicitno zabranjuje prelazak i u neka druga rešenja koja zapravo nisu do tada ispitana) može se uvesti i kriterijum aspiracije (*eng. aspiration criterion*) koji će omogućiti da se određenim rešenjima u toj listi poništi tabu status. Drugim rečima, pomoću kriterijuma aspiracije se zapravo formalno definišu karakteristike obećavajućih rešenja odnosno nivo aspiracije (*eng. aspiration level*) koji tabu rešenja treba da dostignu da bi se smatrala dozvoljenim. Pri tome se kriterijum aspiracije najčešće definiše tako da se dozvoljava pomak u određeno rešenje (čak iako je ono tabu) samo ako je dato rešenje bolje od bilo kog prethodno ispitanog rešenja, odnosno ukoliko je funkcija cilja u tom rešenju bolja od vrednosti funkcije cilja u najboljem do tada pronađenom rešenju, što zapravo ukazuje na to da takvo rešenje nije bilo ispitano u prethodnim iteracijama[1] Konačno, treba napomenuti da najjednostavniji oblik ovog kriterijuma (*eng. aspiration by default*) zapravo odgovara prethodno opisanoj situaciji kada se, u slučaju da su svi mogući pomaci iz tekućeg rešenja zabranjeni, bira onaj pomak koji je najmanje tabu.

[1] U opštem slučaju, ovi kriterijumi se mogu zasnivati na evaluaciji vrednosti funkcije cilja (bilo globalno gledano bilo razmatrajući samo određenu podoblast dopustivog prostora), pravcu u kome se pretraga prethodno kretala, uticaju samog pomaka (odnosno stepenu promene koju određeni pomak izaziva, što se često povezuje i sa dužinom pomaka) itd. Pri tome treba napomenuti da ovi kriterijumi mogu biti i veoma složeni kao i da se prema potrebi (uzimajući u obzir tekuće stanje pretrage) mogu dinamički aktivirati ili isključivati ili čak modifikovati. M ne zna da pomogne N u ovim matematikama a ni N M kod umetnosti.

Dodatno intenziviranje i diversifikacija se postižu time što će se identifikovati i zapamtiti neke karakteristike određenih elitnih prethodno ispitanih rešenja (koje bi mogle da upute na neka druga i bolja rešenja) kako bi se u određenom koraku tabu pretraživanja okolina tekućeg rešenja na adekvatan način suzila odnosno proširila. Drugim rečima navedene informacije će se koristiti za modifikaciju okoline tekućeg rešenja tako da se ugrade preference ka rešenjima koja dele neke od karakteristika sa tim elitnim rešenjima. odnosno koja se što je više moguće razlikuju od njih. Stoga je očigledno da se mehanizmi pomoću kojih se ove strategije implementiraju moraju zasnivati na nekom obliku dugoročnije memorije. Ova memorija se obično naziva i „frequency-based“ memorija budući da se u njoj najčešće pamti frekvencija pojavljivanja nekih događaja[1].

Intenziviranje pretrage se najčešće postiže korišćenjem srednjoročne memorije (*eng. intermediate term memory*) koja ima za cilj da omogući identifikovanje i pamćenje dobrih, odnosno obećavajućih, karakteristika prethodno ispitanih rešenja (na primer vrednosti određenih promenljivih) kako bi se obezbedila osnova za vrednovanje kvaliteta rešenja. Osnovna ideja je da se prateći nekoliko najboljih (elitnih) rešenja, pronađenih tokom određenog perioda pretrage, identifikuju one karakteristike koje su se u njima najčešće pojavljivale tj. one karakteristike koje su zajedničke većini, ili čak svim, tim rešenjima.

Sa druge strane diversifikacija pretrage se najčešće postiže korišćenjem dugoročne memorije (*eng. long term memory*) koja se prema (*Glover & Laguna, 2013*) zapravo zasniva na princpima koji su grubo rečeno inverzni principima na kojim se zasniva srednjoročna memorija. Budući da je cilj diversifikacije da omogući ispitivanje rešenja koja se u velikoj meri razlikuju od prethodno ispitanih rešenja (kako bi se pretraga usmerila ka neistraženim ili najmanje istraženim oblastima) osnovna ideja je da se zabrani pojavljivanje onih karakteristika rešenja za koje je utvrđeno da su se često pojavljivali u prethodnom procesu pretraživanja. Drugim rečima, favorizovaće se oni elementi koji nisu pripadali prethodnim rešenjima ili su se u njima ređe pojavljivali.

[1] Na primer koliko su se puta određeni elementi pojavljivali u relevantnim rešenjima ili pomacima (*eng. residence measure*) ili koliko puta je došlo do promene vrednosti nekog elementa (*eng. transition measures*). U opštem slučaju frekvencija se može meriti na različite načine na primer: brojem iteracija od početka procesa pretraživanja u kojima su se dati elementi pripadali pronađenim najboljim rešenjima (ili bili uključene u izabrane pomake) ili brojem konsekutivnih iteracija u kojima su dati elementi, bez prekida, bili prisutni u relevantnim rešenjima ili pomacima, itd.

Generalno gledano tabu pretraživanje je zamišljeno kao metaheuristika budući da ono predstavlja jednu opštu strategiju za prevazilaženje problema lokalne optimlanosti, pri čemu se za lokalno pretraživanje zapravo može koristiti bilo koja heuristika. Naime, kako se navodi u (*Glover, Avgust, 1990*), tabu pretraživanje se može koristiti za usmeravanje bilo kog procesa koji koristi skup pomaka (kojima se jedno rešenje transformiše u drugo) i koji obezbeđuje neku funkciju pomoću koje se može meriti privlačnost ovih pomaka.

U opštem slučaju prilikom primene ove metode potrebno je načiniti veliki broj izbora: da li će se koristiti eksplicitna ili atributivna memorija, da li će se koristiti samo kratkoročna ili i druge vrste memorije (srednjoročna i/ili dugoročna), na koji način će se implementirati ove memorijske strukture, da li će se za svaku vrstu memorijske strukture koristiti jedna ili više listi, da li će se i kako implementirati kriterijum aspiracije, koja vrsta lokalne pretrage će se koristiti, da li će dužina tabu liste biti fiksna ili će se dinamički menjati, da li će se primeniti i neke naprednije varijante koje iziskuju zadavanje još nekih parametara (strateška oscilacija, probabilističko tabu pretraživanje, paralelno tabu pretraživanje…) itd. Sa druge strane, kako se navodi u (*Gendreau, 2003*) ova metoda često daje rešenja koja su veoma bliska optimalnim, i spade među najefikasnije metode za rešavanje izuzetno složenih kombinatornih problema.

Konačno primena tabu pretraživanja se predlaže za selekcija servisa za kompoziciju i u (*Parejo & al 2008; Pop & al, 2011*) ali na drugačije modele problema od onih koji se u ovom radu predlažu.

5.2.4.2. PREDLOŽENA NOVA METODA

Prilikom definisanja predložene metaheuristike pošlo se od činjenice da i tabu pretraživanje i VNS obezbeđuje strategije za izvlačenje iz lokalnih optimuma. Pri tome VNS tokom lokalnog pretraživanja koristi samo silazeće pomake, pri čemu se, ako se u nekoj iteraciji takvi pomaci ne mogu naći, vrši prestruktuiranje okoline trenutnog rešenja i u tako formiranoj novoj okolini nastavlja pretraživanje. Kod tabu pretraživanja ovaj problem je prevaziđen time što ono dozvoljava i uspinjuće pomake, tj. pomake koji vode u susede sa lošijom vrednošću funkcije cilja, pri čemu se prati istorija pretraživanja, kako bi se osiguralo da se pretraga u narednim iteracijama ne vrati u to isto rešenje (tzv. cikliranje).

Pored toga, obe metaheuristike obezbeđuju različite mehanizme pomoću kojih će se pretraga dodatno intenzivirati (kako bi se oblasti oko obećavajućih rešenja temeljnije pretražile) i diversifikovati (kako bi se ispitale i neke druge prethodno neispitane oblasti). Stoga je ideja bila da se u novoj metaheuristike iskoriste neke od prednosti svake od ovih metoda.

Sa jedne strane pošlo se od činjenice da sprečavanje cikliranja nije jedina svrha tabu pretraživanja, jer u nekim slučajevima dobra putanja pretraživanja može voditi ka vraćanju u neko rešenje koje je prethodno već ispitano, te da u tom smislu tabu pretraživanje treba shvatiti kao metodu koja podstiče otkrivanje novih visoko kvalitetnih rešenja (*Glover & Laguna, 2013*). Pored toga, budući da, kako se navodi u (*Gendreau, 2003*) dodatno intenziviranje pretrage kod tabu pretraživanja najčešće nije neophodno, pošto je u najvećem broju slučajeva normalni proces pretraživanja dovoljno temeljan, u predloženoj metaheuristici se za lokalno pretraživanje u svakom od koraka zapravo primenjuje tabu pretraživanje. Međutim, autor dalje navodi da je diversifikacija jedan od većih nedostataka tabu pretraživanja. Pored toga tabu pretraživanje koristi samo jednu strukturu okoline i način na koji je ona definisana može takođe značajno da smanji efikasnost rešenja.

Sa druge strane, u opštem slučaju, postoji nekoliko različitih strategija za diversifikaciju. Jedna od često korišćenih strategija je tzv. ponovno započinjanje pretrage (*eng. restart*) koje podrazumeva da se pretraga ponovno pokrene iz nekih drugih rešenja koja se mogu odrediti bilo slučajno bilo na neki osmišljen način. Najjednostavnija tehnika, tzv. multi-start lokalno pretraživanje podrazumeva da se lokalno pretraživanje više puta pokrene iz različitih nasumično izabranih početnih rešenja. Međutim ovakav pristup može voditi ka istim optimumima odnosno može se desiti da se utroši nepotrebno vreme na ponovno određivanje već poznatih minimuma (*Törn & Žilinskas, 1989*). Za razliku od ove metode VNS omogućava kontrolisano povećanje nivoa rasejanja a pokazano je da i daje bolje rezultate od multi-start metode (*Brimberg & al, 2004*). Isto tako VNS, za razliku od tabu pretraživanja koje prilikom pretrage koristi samo jednu strukturu okoline, omogućava da se prilikom pretrage zapravo menjaju strukture okolina. U skladu sa time, izabrano je da se diversifikacija postiže putem VNS metode.

Međutim, VNS u svakom koraku fokusira pretragu na oblasti koje se nalaze oko tekućeg najboljeg rešenja pri čemu se informacije o nekim prethodnim dobrim rešenjima ne pamte. Sa jedne strane, imajući u vidu da VNS ne pretpostavlja da se čuvaju bilo kakve informacija o oblastima koje su već bile pretražene može se desiti da se nepotrebni napor uloži u ponovno pretraživanje istih oblasti. Drugim rečima lokalne pretrage mogu zapravo voditi ka istim lokalnim optimumima te će se nepotrebno utrošiti napor u njihovom ponovnom otkrivanju. Sa druge strane, kako se navodi u (*Hansen & Mladenović, 2003a*) imalo bi smisla na neki način pamtiti i informacije o ovim rešenjima budući da bi one zapravo mogle da ukažu i na obećavajuće oblasti koje nisu bile dovoljno ispitane. Autori dalje navode i da bi se karakteristike koje su zajedničke mnogim ili većini dobrih rešenja (na primer iste vrednosti određenih promenljivih) mogle iskoristiti i prilikom promene okoline tj. u „shake“ fazi. Stoga je u predloženoj metaheuristici uvedena i tabu lista u kojoj će se pamtiti i informacije dobijene u prethodnim VNS koracima. Cilj ove liste je da prilikom promena okoline omogući generisanje novih rešenja na nešto osmišljeniji umesto potpuno nasumičan način.

Dakle, predložena metaheuristika podrazumeva da se, počevši od nekog nasumično izabranog rešenja u prvoj okolini tekućeg lokalnog optimuma, primenom tabu pretraživanja traži novi lokalni optimum, penjući se i spuštajući i konstantno pri tome ažurirajući odgovarajuću tabu listu (TL_2) se sve dok se ne ispuni postavljeni kriterijum zaustavljanja (KZ_2). Pronađeno rešenje postaje tekuće rešenja ukoliko je bolje od prethodnog lokalnog optimuma. Postupak se nastavlja dokle god se generiše sekvenca koja vodi ka poboljšanju funkcije cilja, dok će se u suprotnom proširiti okolina. Dakle, diversifikacija se zapravo primenjuje samo kada se u nekom koraku dostigne lokalni optimum odnosno ukoliko se u određenom predefinisanom broju iteracija ne ostvari poboljšanje. Pre nego što se nasumično generiše rešenje u novoj okolini („shake“) ona će se modifikovati na osnovu informacija iz spoljašnje tabu liste (TL_1). U spoljašnjoj tabu listi pamti se frekvencija pojavljivanja pojedinih servisa u najboljim pronađenim rešenjima, pa se diversifikacija ostvaruje tako što se u „shake“ fazi isključuju oni servisi koji su se više puta pojavljivali u ovim rešenjima.

Postupak se iterativno nastavlja, dokle god se ne ispita i poslednja okolina (d_{max}) nakon čega ponovo polazi od prve okoline tekućeg rešenja itd. sve dok se ne dostigne kriterijum zaustavljanja (KZ_1)

Sam postupak se može predstaviti sledećim algoritmom:

Algoritam 7. Hibridna metaheuristika

function HYBRID_VNS(x^0, d_{max} , TL_1 , TL_2 , KZ_1 , KZ_2 ,)
  $x^* = x^0$; $f^* = f(x^0)$; $TL_1 = \emptyset$;
  repeat
    $d=1$;
    repeat
      $x' =$ **RANDOM**($N_d(x^*) \setminus TL_1$); // *Shake*
      $x^t =$ **TABU_SEARCH**(x', TL_2 , KZ_2);
      if $f(x^t) \leq f^*$ **then**
        $x^* = x^t$; $f^* = f(x^t)$; // *Move*
        UPDATE_TABU_LIST(TL_1, $\{(i, j)\}$);
        $d = 1$;
      else
        $d = d+1$; // *Promena okoline*
    until $d = d_{max}$
  until KZ_1
  return x^*

Konačno treba napomenuti da funkcija cilja i/ili ograničenja generalno gledano mogu biti i nelinearne funkcije. Budući da tabu pretraživanje u najčešće primenjivanoj varijanti podrazumeva da se svako rešenje u datoj okolini evaluira ponaosob, pre nego što se izabere najbolji pomak, očigledno je da u opštem slučaju ovo može iziskivati veoma veliki računski napor. Stoga, ukoliko je reč o problemima koji su linearni, kao što je MMKP problem, evaluacija rešenja se zapravo može vršiti merenjem promene u vrednosti funkcije cilja koja bi se ostvarila takvim pomakom.

Predložena metaheuristika je primenjena i na MMKP model problema kao i na model zasnovan na primeni konzistentne fazi logike i rezultati su predstavljeni u sledećem poglavlju.

Na kraju treba napomenuti da se ideja hibridizacije VNS metode i tabu pretraživanja u literaturi sve više primenjuje o čemu svedoči i pregled dat u (*Hansen & al, 2010*), pri čemu različiti autori na različite načine kombinuju ove dve metode.

5.2.5. Eksperimentalni rezultati

Svi predložni algoritmi su implementirani u programskom jeziku C# za .NET Framework 4.0. Prilikom projektovanja rešenja primenjen je objektno-orijentisani pristup i troslojna arhitektura – te se rešenje sastoji od sloja podataka, sloja logike i prezentacionog sloja. Na ovaj način je omogućeno da se sloj logike, u kome su zapravo implementirane predložene metode, lako integriše u druge aplikacije koje bi ga mogle koristiti kao komponentu. Pri tome je obezbeđeno da se podaci (tj. definicije QoS atributa i same QoS vrednosti servisa) mogu učitavati u formi XML dokumenata (kako bi se obezbedila interoperabilnost budući da je to danas de-facto standard za razmenu dokumenata na Internetu) ali i iz neke baze podataka. Za evaluaciju logičkih funkcija realizovan je jednostavan parser.

Predložene metode su testirane na 50 problema različitih dimenzija. Naime, variran je broj komponenti same kompozicije (m) i broj servisa koji su na raspolaganju za date komponente (p)[1] i za svaku takvu kombinaciju (m x p) generisano je po 10 test instanci. Razmatrani su sledeći QoS atributi:

Tabela 22. Izabrani QoS atributi

QoS atribut	*Način agregacije*	*Smer optimizacije*
Raspoloživost	$q^1(x) = \prod_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^1 x_{ij}$	max
Vreme odziva	$q^2(x) = \sum_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^2 x_{ij}$	min
Pouzdanost	$q^3(x) = \prod_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^3 x_{ij}$	max
Kašnjenje	$q^4(x) = \sum_{i=1}^{m} \sum_{j=1}^{p_i} q_{ij}^4 x_{ij}$	min
Bezbednost	$q^5(x) = \min_{i=1,\ldots,m} \sum_{j=1}^{p_i} q_{ij}^5 x_{ij}$	max
Enkripcija	$q^6(x) = \bigwedge_{i=1,\ldots,m} \sum_{j=1}^{p_i} q_{ij}^6 x_{ij}$	max

[1] Prilikom testiranja pretpostavljeno je da je broj servisa koji je na raspolaganju za svaku od komponenti isti (tj. $p = p1 = p2 = \ldots = pm$) što ni na koji način ne umanjuje mogućnost primene navedenog pristupa i u slučajevima kada se broj raspoloživih servisa za svaku od komponenti razlikuje.

Za svaku od metoda su varirani relevantni parametri (maksimalni broj okolina i dužina tabu liste), pri čemu je kao kriterijum zaustavljanja uzeto maksimalno dozvoljeno vreme izvršavanja (izraženo u milisekundama). Imajući u vidu stohastički karakter VNS metode testovi su nekoliko puta ponovljeni sa istim parametrima te su u tabelama dati prosečni dobijeni rezultati. Predložene metode su testirane na računaru sa sledećom konfiguracijom: Intel Core 2 Duo CPU E7500 processor, 2.93Ghz, 3.24GB RAM.

Najpre se rešavao MMKP model problema, kod koga je moguće uzeti u obzir samo prva četiri numerička QoS atributa (pri čemu se vrednosti *Raspoloživosti* i *Pouzdanosti* moraju prethodno logaritmovati):

$$min \sum_{i=1}^{m}\sum_{j=1}^{p_i} f_{ij}x_{ij}, \quad f_{ij} = \sum_{k=1}^{4} w_k q^k_{norm_{ij}}$$

pri ograničenjima: $\prod_{i=1}^{m}\sum_{j=1}^{p_i} q^k_{ij}x_{ij} \le Q^k,$ $k = 1,3,$

$$\sum_{i=1}^{m}\sum_{j=1}^{p_i} q^k_{ij}x_{ij} \le Q^k, \qquad k = 2,4,$$

$$\sum_{j=1}^{p_i} x_{ij} = 1, \qquad i = 1,..., m,$$

$$x_{ij} \in \{0,1\}, \qquad j = 1,..., p_i;\ i = 1,..., m.$$

Treba napomenuti da se u ograničenjima vezanim za *Raspoloživost* i *Pouzdanost* koriste originalne QoS vrednosti a ne njihovi logaritmi.

Predložene metode su prvo testirane na problemima manjih dimenzija za koje je moguće pronaći egzaktno rešenje u razumnom vremenu. Rezultati eksperimenata pokazuju da obe metode pronalaze egzaktno rešenje i to u veoma kratkom vremenskom periodu. Na primer za problem dimenzija *7*x*10* svi algoritmi su dali egzaktno rešenje za *10 ms* dok je potpuno pretraživanje svih dopustivih kombinacija u proseku trajalo *46859.4 ms*, tj. *47* sekundi.

Zatim su se razmatrali problemi većih dimenzija za koje nije moguće pronaći egzaktno rešenje u razumnom vremenu. Eksperimenti su pokazali da VNS najčešće rezultuje boljim rešenjem od tabu pretraživanja. Pored toga predložena hibridna

metaheuristika u najvećem broju slučajeva na kraćim vremenskim intervalima rezultuje boljim rešenjem nego tabu pretraživanje i VNS (Tabela 23) što se i očekivalo budući da je ideja bila da se kombinuju prednosti koje nude te dve metode.

Za problem selekcije servisa za kompoziciju koja se sastoji od 10 komponenti, pri čemu je za svaku od komponenti na raspolaganju 1000 servisa, eksperimenti su pokazali da se najbolji rezultati dobijaju kada je dužina tabu liste *7* a broj okolina *5*. Na primer, za vreme izvršavanja od *100ms* i nasumično generisano početno rešenje x^0 dobijeni su sledeći rezultati:

Tabela 23. Eksperimentalni rezultati (MMKP)

Instanca	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
x^0	8.2805	8.3289	8.1938	8.3722	8.2743	8.1835	8.4748	8.5701	8.3200	8.1153
TS	7.7588	7.7414	7.7096	7.9407	7.8528	7.5259	7.8915	7.8204	7.6861	7.5285
VNS	7.6975	7.6393	7.6254	7.6925	7.6828	7.6086	7.6545	7.8098	7.7257	7.7299
H	**7.2775**	**7.5770**	**7.3675**	**7.5870**	**7.3859**	**7.3969**	**7.5586**	**7.6452**	**7.6742**	**7.4522**

Eksperimentalni rezultati takođe pokazuju da način izbora početnog rešenja u velikoj meri utiče na uspešnost metoda te da stoga primena metode za generisanje početnog rešenja predložene u Poglavlju 5.2.2. nalazi svoje puno opravdanje.

Zatim se rešavao problem modelovan pomoću predloženog pristupa zasnovanog na primeni konzistentne fazi logike, koji omogućava da se uzmu u obzir i kvalitativni QoS atributi agregirani pomoću korisnički definisanih funkcija. Pri tome je funkcija cilja definisana kao logički izraz.

Razmatrane su tri različite funkcije:

1. $$\left(q^1(x) \wedge q^2(x)\right) \vee \left(\neg q^1(x) \wedge q^3(x)\right)\ ) \qquad (\textbf{\textit{CFLP\_1}})$$

koja se sledećim transformacijama prevodi u generalizovani atomski Bulov polinom:

$$\left(\left(q^1(x) \wedge q^2(x)\right) \vee \left(\neg q^1(x) \wedge q^3(x)\right)\right)^{\otimes} = \left(\left(q^1(x) \wedge q^2(x)\right) \vee \left(\neg q^1(x) \wedge q^3(x)\right)\right)^{\otimes} =$$
$$= \left(q^1(x) \wedge q^2(x)\right)^{\otimes} + \left(\neg q^1(x) \wedge q^3(x)\right)^{\otimes} - \left(\left(q^1(x) \wedge q^2(x)\right) \wedge \left(\neg q^1(x) \wedge q^3(x)\right)\right)^{\otimes} =$$
$$= q^1(x) \otimes q^2(x) + \left(\left(1 - q^1(x)\right) \wedge q^3(x)\right)^{\otimes} - \left(q^1(x) \wedge q^2(x)\right)^{\otimes} \otimes \left(\neg q^1(x) \wedge q^3(x)\right)^{\otimes} =$$
$$= q^1(x) \otimes q^2(x) + \left(1 - q^1(x)\right) \otimes q^3(x) - q^1(x) \otimes q^2(x) \otimes \left(\left(1 - q^1(x)\right) \otimes q^3(x)\right) =$$
$$= q^1(x) \otimes q^2(x) + q^3(x) - q^1(x) \otimes q^3(x) - q^1(x) \otimes q^2(x) \otimes q^3(x) + q^1(x) \otimes q^2(x) \otimes q^1(x) \otimes q^3(x) =$$
$$= q^1(x) \otimes q^2(x) + q^3(x) - q^1(x) \otimes q^3(x)$$

2. $\left(\left(q^1(x)\wedge q^2(x)\right)\vee\left(\neg q^1(x)\wedge q^3(x)\right)\right)\wedge q^5(x)$ (***CFLP_2***)

koji se sledećim transformacijama prevodi u generalizovani atomski Bulov polinom:

$$\left(\left(\left(q^1(x)\wedge q^2(x)\right)\vee\left(\neg q^1(x)\wedge q^3(x)\right)\right)\wedge q^5(x)\right)^{\otimes}=\left(\left(q^1(x)\wedge q^2(x)\right)\vee\left(\neg q^1(x)\wedge q^3(x)\right)\right)^{\otimes}\otimes q^5(x)=$$
$$=\left(\left(q^1(x)\wedge q^2(x)\right)^{\otimes}+\left(\neg q^1(x)\wedge q^3(x)\right)^{\otimes}-\left(\left(q^1(x)\wedge q^2(x)\right)\wedge\left(\neg q^1(x)\wedge q^3(x)\right)\right)^{\otimes}\right)\otimes q^5(x)=$$
$$=\left(q^1(x)\otimes q^2(x)+\left(\left(1-q^1(x)\right)\wedge q^3(x)\right)^{\otimes}-\left(q^1(x)\wedge q^2(x)\right)^{\otimes}\otimes\left(\neg q^1(x)\wedge q^3(x)\right)^{\otimes}\right)\otimes q^5(x)=$$
$$=\left(q^1(x)\otimes q^2(x)+\left(1-q^1(x)\right)\otimes q^3(x)-q^1(x)\otimes q^2(x)\otimes\left(\left(1-q^1(x)\right)\otimes q^3(x)\right)\right)\otimes q^5(x)=$$
$$=\left(q^1(x)\otimes q^2(x)+q^3(x)-q^1(x)\otimes q^3(x)-q^1(x)\otimes q^2(x)\otimes q^3(x)+q^1(x)\otimes q^2(x)\otimes q^1(x)\otimes q^3(x)\right)\otimes q^5(x)=$$
$$=q^1(x)\otimes q^2(x)\otimes q^5(x)+q^3(x)\otimes q^5(x)-q^1(x)\otimes q^3(x)\otimes q^5(x)$$

3. $\left(\left(q^1(x)\wedge q^2(x)\right)\vee\left(\neg q^1(x)\wedge q^3(x)\right)\right)\wedge\left(q^5(x)\vee q^6(x)\right)$ (***CFLP_3***)

koji se sledećim transformacijama prevodi u generalizovani atomski Bulov polinom:

$$\left(\left(\left(q^1(x)\wedge q^2(x)\right)\vee\left(\neg q^1(x)\wedge q^3(x)\right)\right)\wedge\left(q^5(x)\vee q^6(x)\right)\right)^{\otimes}=$$
$$=\left(\left(q^1(x)\wedge q^2(x)\right)\vee\left(\neg q^1(x)\wedge q^3(x)\right)\right)^{\otimes}\otimes\left(q^5(x)\vee q^6(x)\right)^{\otimes}=$$
$$=\left(\left(q^1(x)\wedge q^2(x)\right)^{\otimes}+\left(\neg q^1(x)\wedge q^3(x)\right)^{\otimes}-\left(\left(q^1(x)\wedge q^2(x)\right)\wedge\left(\neg q^1(x)\wedge q^3(x)\right)\right)^{\otimes}\right)\otimes\left(q^5(x)\vee q^6(x)\right)^{\otimes}=$$
$$=\left(q^1(x)\otimes q^2(x)+\left(\left(1-q^1(x)\right)\wedge q^3(x)\right)^{\otimes}-\left(q^1(x)\wedge q^2(x)\right)^{\otimes}\otimes\left(\neg q^1(x)\wedge q^3(x)\right)^{\otimes}\right)\otimes\left(q^5(x)\vee q^6(x)\right)^{\otimes}=$$
$$=\left(q^1(x)\otimes q^2(x)+\left(1-q^1(x)\right)\otimes q^3(x)-q^1(x)\otimes q^2(x)\otimes\left(\left(1-q^1(x)\right)\otimes q^3(x)\right)\right)\otimes\left(q^5(x)\vee q^6(x)\right)^{\otimes}=$$
$$=\left(q^1(x)\otimes q^2(x)+q^3(x)-q^1(x)\otimes q^3(x)-q^1(x)\otimes q^2(x)\otimes q^3(x)+q^1(x)\otimes q^2(x)\otimes q^1(x)\otimes q^3(x)\right)\otimes\left(q^5(x)\vee q^6(x)\right)^{\otimes}=$$
$$=\left(q^1(x)\otimes q^2(x)+q^3(x)-q^1(x)\otimes q^3(x)-q^1(x)\otimes q^2(x)\otimes q^3(x)+q^1(x)\otimes q^2(x)\otimes q^1(x)\otimes q^3(x)\right)\otimes\left(q^5(x)+q^6(x)-\left(q^5(x)\wedge q^6(x)\right)^{\otimes}\right)=$$
$$=\left(q^1(x)\otimes q^2(x)+q^3(x)-q^1(x)\otimes q^3(x)-q^1(x)\otimes q^2(x)\otimes q^3(x)+q^1(x)\otimes q^2(x)\otimes q^1(x)\otimes q^3(x)\right)\otimes\left(q^5(x)+q^6(x)-q^5(x)\otimes q^6(x)\right)=$$
$$=\left(q^1(x)\otimes q^2(x)+q^3(x)-q^1(x)\otimes q^3(x)-q^1(x)\otimes q^2(x)\otimes q^3(x)+q^1(x)\otimes q^2(x)\otimes q^1(x)\otimes q^3(x)\right)\otimes\left(q^5(x)+q^6(x)-q^5(x)\otimes q^6(x)\right)=$$
$$=q^1(x)\otimes q^2(x)\otimes\left(q^5(x)+q^6(x)-q^5(x)\otimes q^6(x)\right)+$$
$$q^3(x)\otimes\left(q^5(x)+q^6(x)-q^5(x)\otimes q^6(x)\right)-$$
$$q^1(x)\otimes q^3(x)\otimes\left(q^5(x)+q^6(x)-q^5(x)\otimes q^6(x)\right)$$

$$=q^1(x)\otimes q^2(x)\otimes q^5(x)+q^1(x)\otimes q^2(x)\otimes q^6(x)-q^1(x)\otimes q^2(x)\otimes q^5(x)\otimes q^6(x)+$$
$$q^3(x)\otimes q^5(x)+q^3(x)\otimes q^6(x)-q^3(x)\otimes q^5(x)\otimes q^6(x)$$
$$-q^1(x)\otimes q^3(x)\otimes q^5(x)-q^1(x)\otimes q^3(x)\otimes q^6(x)+q^1(x)\otimes q^3(x)\otimes q^5(x)\otimes q^6(x)$$

Definisanjem odgovarajućih fazi skupova za agregirane vrednosti QoS atributa, shodno razmatranjima iz poglavlja 3.5.3, i izborom algebarskog proizvoda za operator generalizovanog proizvoda dobija se sledeći model problema (na primer za Slučaj 2.):

$$min \quad q^1(x)\cdot q^2(x)\cdot q^5(x) + q^3(x)\cdot q^5(x) - q^1(x)\cdot q^3(x)\cdot q^5(x)$$

pri ograničenjima:

$$\prod_{i=1}^{m}\sum_{j=1}^{p_i} q_{ij}^k x_{ij} \le Q^k, \qquad k = 1,3,$$

$$\sum_{i=1}^{m}\sum_{j=1}^{p_i} q_{ij}^k x_{ij} \le Q^k, \qquad k = 2,$$

$$\sum_{j=1}^{p_i} x_{ij} = 1, \qquad i = 1,\ldots, m,$$

$$x_{ij} \in \{0,1\}, \qquad j = 1,\ldots, p_i;\ i = 1,\ldots, m.$$

U sledećoj tabeli dato je prosečno vreme potrebno za pronalaženje egzaktnog rešenja navedenih nelinearnih problema, potpunim pretraživanjem svih dopustivih kombinacija, na primer za problem manjih dimenzija (7 komponenti i ukupno 70 servisa).

Tabela 24. CFLP potpuno pretraživanje

Vrsta modela	*Vreme (ms)*
CFLP_1	234694.5
CFLP_2	373527.6
CFLP_3	925623.6

Obe predložene metode (VNS metoda i hibridna metaheuristika) su za najjednostavniji slučaj (CFLP_1) za samo *10ms* uvek rezultovale egzaktnim rešenjem dok je, kao što se može videti iz Tabele 24, potpuno pretraživanje svih dopustivih kombinacija trajalo skoro *4 minuta*. Iz tabele se takođe vidi da vreme potrebno za dobijanje egzaktnog rešenja u mnogome zavisi od složenosti postavljenih logičkih zahteva. U skladu sa time i primena predloženih metoda za rešavanje problema CFLP_2 i CFLP_3 je za *10ms* rezultovala egzaktnim rešenjem u *70%* odnosno *40%* slučajeva, respektivno.

Ako se vreme izvršavanja poveća na primer na *500ms* povećava se i broj slučajeva u kojima se dobija egzaktno rešenje kao što je prikazano u Tabeli 25. Za VNS i hibridnu metodu pored prosečnih dobijenih vrednosti prikazan je i procenat slučajeva u kojima je dobijena optimalna vrednost.

Tabela 25. Eksperimentalni rezultati (CFLP)

	CFLP_2					*CFLP_3*				
Inst.	x^0	*x**	*TS*	*VNS*	*H*	x^0	*x**	*TS*	*VNS*	*H*
1	0.253784	0.000232	**0.000232**	**0.000232 100%**	0.000258 **70%**	0.338576	0.000232	0.000310	0.000284 **38%**	0.000258 **80%**
2	0.086103	0.000170	**0.000170**	**0.000170 100%**	**0.000170 100%**	0.001088	0.000170	0.000390	**0.000170 100%**	**0.000170 100%**
3	0.000983	0.000231	0.000348	0.000297 0%	0.000281 **35%**	0.090180	**0.000231**	0.000231	0.000311 0%	0.000291 **40%**
4	0.001412	0.000163	**0.000163**	**0.000163 100%**	**0.000163 100%**	0.001094	0.000175	0.000185	0.000182 **34%**	0.000177 **90%**
5	0.001317	2.9E-05	**2.9E-05**	**2.9E-05 100%**	**2.9E-05 100%**	0.170916	2.9E-05	**2.9E-05**	**2.9E-05 100%**	**2.9E-05 100%**
6	0.155595	0.000106	0.000159	0.000142 **32%**	**0.000106 100%**	0.160039	0.000122	0.000397	0.000158 **41%**	**0.000122 100%**
7	0.187151	0.000327	**0.000327**	**0.000327 100%**	**0.000327 100%**	0.175492	0.000327	**0.000327**	**0.000327 100%**	**0.000327 100%**
8	0.096488	0.000104	**0.000104**	**0.000104 100%**	**0.000104 100%**	0.120205	0.000118	**0.000118**	0.000127 **85%**	**0.000118 100%**
9	0.143653	0.000148	0.000339	**0.000148 100%**	**0.000148 100%**	0.104946	0.000148	**0.000148**	0.000278 **27%**	**0.000148 100%**
10	0.000567	0.000135	0.000146	**0.000135 100%**	**0.000135 100%**	0.100446	0.000140	0.000146	**0.000140 100%**	**0.000140 100%**

Potrebno je istaći da je, kao što je to bio slučaj kod MMKP modela, hibridna metoda na problemu dimenzija *10*x*1000* dala superiornije rezultate za sva tri CFLP problema.

Dobijeni rezultati potvrđuju da se predložene metode mogu efikasno koristiti za rešavanje problema selekcije servisa za kompoziciju.

6. Mogućnosti šire primene

Na početku treba istaći da se, servisno-orijentisana paradigma zasniva na razvoju mnoštva nezavisnih softverskih komponenti (koje obuhvataju manje celine poslovne logike) a koje su razvijene na takav način da se mogu više puta koristiti i lako međusobno povezivati pri čemu je u pitanju apstraktna paradigma koja je nezavisna od načina same implementacije. Prema (*Chou & al, 2010*), bilo koja tehnologija pomoću koje je moguće razviti distribuirani sistem može koristiti i za razvoj servisno-orijentisanih rešenja. Stoga predloženi pristupi nisu ograničeni samo na selekciju web servisa, već bi se mogli primeniti i za selekciju, na osnovu nefunkcionalnih karakteristika, servisa koji su realizovani na neki drugačiji način. Pri tome je takođe važno istaći da, imajući u vidu sve veću popularnost „cloud" računarstva, čija se osnova ideja ogleda u tome da se različiti resursi (hardverski, softverski, itd.) učine dostupnim putem Interneta, nefunkcionalne karakteristike distribuiranih aplikacija postaju sve značajnije. Cloud računarstvo je takođe zasnovano na servisima. Naime, softver kao servis (*eng. Software as a Service – SaaS*) predstavlja jednu od komponenti cloud računarstva koja koristi infrastrukture cloud-a da bi jednu aplikaciju isporučila većem broju korisnika nezavisno od toga gde se oni nalaze, odnosno SaaS predstavlja jedan mogući način isporuke softvera. Servisno-orijentisani pristup omogućava da se koncept softvera kao servisa proširi kako bi se omogućila isporuka kompleksnih poslovnih procesa kao servisa.

Sa druge strane predloženo rešenje za selekciju servisa bi se moglo koristiti u okviru postojećih pristupa za razvoj aplikacija zasnovanih na SOA koji podrazumevaju postojanje centralizovane komponente – brokera za selekciju servisa. Naime realizovano softversko rešenje se može iskoristiti u fazi selekcije servisa za kompoziciju kako bi se postupak selekcije automatizovao. Pri tome bi se ono moglo integrisati u postojeća softverska rešenja ali se isto tako može izložiti i kao nezavisni web servis. Pored toga, budući da se u implementiranom softverskom rešenju sve neophodne definicije i podaci učitavaju iz XML dokumenata ono bi se moglo jednostavno prilagoditi kako bi se omogućila integracija sa pristupima zasnovanim na primeni QoS ontologija, koje predstavljaju svojevrsne rečnike za dati domen budući da omogućavaju da se formalno iskažu koncepti određenog domena kao i odnosi koji postoji između tih koncepata, čime se pospešuje razmena i ponovno korišćenje znanja o datom domenu.

Međutim, važno je istaći da bi se predloženi pristupi mogli koristiti i u drugim domenima a ne samo za razvoj aplikacija i automatizaciju poslovanja budući da servisno-orijentisani pristup predstavlja opštu paradigmu za obezbeđivanje složene funkcionalnosti povezivanjem skupa komponenti. Stoga bi se oni mogli koristiti i u bilo kom drugom polju u kome se ideja povezivanja jedinica funkcionalnosti (koje ne moraju nužno biti softverske) može primeniti.

Konačno, inovativno proširenje AHP metode (koja je jedna od najčešće korišćenih metoda višekriterijumskog odlučivanja) konzistentnom fazi logikom bi se moglo koristiti, ne samo za rešavanje problema pojedinačne selekcije web servisa već i u bilo kom drugom domenu višekriterijumskog odlučivanja u kome je potrebno doneti odluku na osnovu složenih verbalnih zahteva korisnika. Sa druge strane, predloženi pristup bi se mogao koristiti i za određivanje težina kriterijuma ukoliko je, prilikom rešavanja problema višekriterijumskog odlučivanja ili optimizacije, potrebno primeniti neku tehniku zasnovanu na težinskim koeficijentima budući da se ovakav pristup agregiranju višestrukih kriterijuma u jedinstvenu kriterijumsku funkciju najčešće sreće u literaturi. Isto tako i primena predložene hibridne metaheuristike nije ograničena samo na izložene modele problema, pri čemu bi se ona takođe mogla koristiti, i kao komponenta egzaktnih algoritama, za određivanje dobrih početnih rešenja.

7. ZAKLJUČAK

7.1. Pregled rada

U okviru ovog rada razmatrao se problem selekcije web servisa na osnovu nefunkcionalnih karakteristika kao jedan od značajnih elemenata servisno-orijentisanog pristupa razvoju aplikacija. Kao prvo su data osnovna teorijska polazišta vezana za servisno-orijentisani pristup koji omogućava fleksibilno programsko povezivanje nezavisno razvijenih softverskih komponenti (servisa). Budući da se u praksi komponente najčešće razvijaju kao web servisi ukratko je objašnjen i pojam web servisa. Zatim je pažnja usmerena na kompozicije servisa koje omogućavaju realizaciju složenijih funkcionalnosti. Drugim rečima kompozicija zapravo predstavlja skup servisa koji su komponovani na takav način da zajedno automatizuju određeni zadatak ili poslovni proces. Na kraju, je dat pregled nefunkcionalnih karakteristika servisa (*eng. Quality of Service* – QoS) sa posebnim osvrtom na način određivanja nefunkcionalnih karakteristika kompozicije servisa.

U trećem poglavlju je definisan problem selekcije servisa zasnovane na nefunkcionalnim karakteristikama. Naime, imajući u vidu da se softverske komponente nezavisno razvijaju i objavljuju na Internetu svakim danom sve je veći broj dostupnih web servisa, a kako je servisno-orijentisani pristup razvoju aplikacija sve prisutniji neminovna posledica je i porast broja web servisa koji pružaju istu funkcionalnost. Očigledno je da je problem selekcije najboljeg servisa iz skupa servisa koji pružaju istu funkcionalnost sve aktuelniji kao i da nefunkcionalne karakteristike servisa tada postaju sve značajnije. Sa druge strane, kada zahtevanu funkcionalnost ne može obezbediti pojedinačni servis ali bi se mogla realizovati izvršavanjem skupa različitih servisa, od kojih svaki obezbeđuje deo zahtevane funkcionalnosti, problem selekcije postaje još kompleksniji budući da je za svaku od komponenti kompozicije potrebno odabrati servis i to na takav način da rezultujuća kompozicija bude optimalna u pogledu relevantnih nefunkcionalnih karakteristika. Pri tome se u opštem slučaju nameću i određena ograničenja u pogledu minimalnih nefunkcionalnih karakteristika koje kompozicija mora obezbediti. Dodatni izazov predstavlja situacija u kojoj je potrebno selekciju servisa za kompoziciju vršiti dinamički (odnosno neposredno pre izvršavanja kompozicije ili čak u toku njenog izvršavanja) imajući u vidu da skup raspoloživih servisa ne mora u svakom trenutku biti isti, odnosno da se u međuvremenu mogu

pojaviti novi servisi sa boljim QoS karakteristikama ili se može desiti da u vreme izvršavanja kompozicije predviđeni servis bude nedostupan ili da se njegovi QoS nivoi značajno promene. Sa druge strane i zahtevi poslovanja se često menjaju što može iziskivati i promenu zahteva vezanih za QoS kompozicije. Stoga mogućnost brze i dinamičke kompozicije servisa postaje imperativ. Imajući u vidu heterogenost samih QoS mera (koje mogu biti kvantitativne ili kvalitativne, opšte ili domensko-specifične, međusobno korelisane ili čak konfliktne) očigledno je da definisani problemi selekcije kako pojedinačnih servisa tako i selekcije servisa za kompoziciju zapravo predstavljaju probleme višekriterijumskog odlučivanja (*eng. Multiple Criteria Decision Making* – MCDM) u kojima je potrebno iz mnoštva raspoloživih alternativa izabrati onu koja na najbolji način zadovoljava ceo skup postavljenih zahteva.

O samom značaju problema selekcije servisa i rasprostranjenosti metoda za njegovo rešavanja svedoče i veoma veliki broj radovi u ovoj oblasti. Cilj ovog rada je, pre svega, bio da se na osnovu analize postojećih pristupa modelovanju i rešavanju problema selekcije web servisa zasnovane na njihovim nefunkcionalnim karakteristikama utvrde mogućnosti definisanja novih modela i metoda koji bi unapredili rešavanje identifikovanog problema.

Modeli

U trećem poglavlju su se kao prvo razmatrali nedostaci primene metode jednostavnih aditivnih težina (*eng. Simple Additive Weighting* – SAW) za agregaciju relevatnih QoS kriterijuma u jedinstvenu kriterijumsku funkciju, budući da se ova tehnika predlaže u većini postojećih pristupa za selekciju web servisa (kako pojedinačnih servisa i tako i servisa za kompoziciju) bilo prilikom modelovanja problema, bilo prilikom njegovog rešavanja. Diskutovano je da primena SAW metode ne omogućava uzimanje u obzir činjenice da je dodeljivanje težina najčešće arbitrarno i da ne mora adekvatno odražavati korisnikove preference, niti činjenica da QoS kriterijumi na osnovu kojih se vrši selekcija mogu uticati jedni na druge, odnosno biti u korelaciji i/ili kontradikciji. Pored toga istaknuto je i da primena SAW metode zahteva i da vrednosti samih kriterijuma moraju biti pre svega numeričke a zatim i uporedive, te pored toga što značajno sužava skup QoS mera koje se mogu razmatrati ona takođe zahteva i da se same vrednosti normalizuju. U opštem slučaju način na koji će se vrednosti normalizovati predstavlja

dodatni izazov. Pri tome, problem normalizacije i određivanja težina postaje još kompleksniji kada je reč o selekciji servisa za kompoziciju ukoliko se selekcija vrši na osnovu QoS atributa koji se agregiraju putem nelinearnih funkcija.

Sa druge strane istaknuto je da, u realnosti, zahtevi korisnika mogu biti i dosta složeniji te da se često mogu izraziti samo pomoću složenih verbalnih iskaza. Ovo je posebno prisutno u situacijama kada korisnik ima potrebu da definiše zahteve u kojima se lošija vrednost po nekom od kriterijuma može nadomestiti nekim drugim kriterijumom, odnosno u kojima značaj kriterijuma zavisi od ostvarenih vrednosti drugih kriterijuma. Ovakve verbalno iskazane zahteve najčešće nije ni moguće realizovati dodeljivanjem težina, budući da SAW metoda pretpostavlja da ne postoji bilo kakva interakcija, odnosno uslovljenost, između preferenci vezanih za kriterijume.

Stoga se većina autora koja primenjuje ovaj pristup zapravo vezuje za određeni predefinisani skup QoS mera koje se mogu numerički izraziti i koje se agregiraju pomoću sume ili proizvoda (pri čemu se podrazumeva da se u tom slučaju on logaritmovanjem svede na linearnu funkciju) dok se ostali načini agregacije uglavnom ne razmatraju.

Imajući u vidu navedeno jedan od ciljeva ovog rada je bio definisanje novih pristupa za modelovanje problema selekcije servisa na osnovu nefunkcionalnih karakteristika koji ne zahtevaju dodeljivanje težina niti normalizaciju vrednosti QoS atributa, a koji će omogućiti da se na adekvatniji način uzmu u obzir specifičnosti samih mera ali i različiti odnosi koji postoje između relevantnih nefunkcionalnih karakteristika kao i zahtevi korisnika (koji mogu biti izraženi i pomoću složenih verbalnih iskaza).

U skladu sa ovim ciljem predložena su dva nova pristupa za modelovanje problema selekcije servisa a zatim je i pokazano da se oni mogu uspešno koristiti za rešavanje navedenog problema.

Prvo je predložen pristup koji omogućava da se složeni logički zahtevi vezani za relevantne QoS atribute iskažu pomoću odgovarajuća funkcionalnih zavisnosti. U tom smislu fazi logika se nametnula kao prirodno rešenje za specifikaciju zahteva korisnika budući da je usklađena sa kvalitativnim opisima koje ljudi koriste u svakodnevnoj komunikaciji.

Ovo opredeljenje je sa jedne strane bilo uslovljeno činjenicom da predstavljanje QoS atributa pomoću fazi skupova omogućava da se objedine različite vrste

nefunkcionalnih karakteristika nezavisno od toga da li su iskazane brojnim vrednostima ili lingvističkim promenljivima. Dodatno izborom oblika fazi skupova i njihovih granica moguće je svaki aspekt precizirati na najadekvatniji način. Pri tome je usvojeno da se svaki QoS atribut predstavlja samo sa po jednim fazi skupom. Ukoliko je reč o selekciji servisa za kompoziciju tada se fazi skupovi definišu za agregirane vrednosti QoS atributa a ne za pojedinačne vrednosti. Ovako predstavljeni QoS atributi imaće funkcije pripadnosti između 0 i 1 tako da normalizacija nije potrebna.

Sa druge strane, fazi logika omogućava i da se uspostavi funkcionalna zavisnost između kriterijuma, koja iskazuje valjanost pojedinačne alternative, tako da se najbolja alternativa dobija minimizacijom ili maksimizacijom date funkcije. Naime, primenom fazi logike navedena funkcionalna zavisnost se može modelovati pomoću iskaza tipa implikacija koje povezuju uslov i posledicu. Ako se odrede fazi skupovi posledice za sve dozvoljene vrednosti uslova onda se svaki od ovih skupova može defazifikovati tj. mogu se odrediti brojne vrednosti posledice u funkciji ulaznih promenljivih. Ova funkcija će tada odrediti površinu odlučivanja na kojoj potrebno pronaći optimalnu vrednost.

Međutim, pokazano je da konvencionalna fazi logika ima izvesne nedostatke pri defazifikaciji iskaza koji uključuju negaciju nekog od uslova te da je, u slučaju kada postoji negacija, u postupku određivanje brojne vrednosti izlaznog fazi skupa (ocene) usled nepoštovanja principa kontradikcije i isključenja trećeg, moguće dobiti neadekvatne vrednosti. Naime, ovako definisan pristup pretpostavlja da je svaki iskaz nezavisan, što zapravo implicira da ne postoje alternativne mogućnosti odnosno da se ni u jednom iskazu ne može sresti negacija nekog od uslova. Imajući u vidu da je donošenje odluke na osnovu većeg broj kriterijuma, po pravilu, vezano za pravljenje svojevrsnog kompromisa između različitih korelisanih, a često i suprotstavljenih kriterijuma, u cilju prevazilaženja ovog nedostatka predložena je primena konzistentne fazi logike koja koncept fazi logike smešta u okvir Bulove algebre. Naime, budući da konzistentna fazi logika konzistentno tretira problem kontradikcije i isključenja trećeg ona omogućava i da se modeluju iskazi u kojima se zamenjuje odsustvo nekog uslova nekim drugim uslovom. Treba napomenuti da, iako u literaturi postoje pristupi koji razmatraju upotrebu fazi logike prilikom definisanja zahteva vezanih za nefunkcionalne karakteristike servisa, ne postoji pristup koji razmatra mogućnost primene konzistentne

fazi logike. Konačno, potrebno je istaći da logička agregacija za rezultat ima novu strukturu komponenti što nije slučaj kada je reč o SAW pristupu kao i da u odnosu na neke druge pristupe (kao što je Šoke-ov integral) konzistentna fazi logika pruža daleko šire mogućnosti primene budući da se svaka logička funkcija može jednoznačno prevesti u odgovarajući generalizovani Bulov polinom primenom interpolativne Bulove algebre.

Drugi pristup, koji nije do sada predlagan za modelovanje problema, namenjen je slučajevima kada ne postoje logički zahtevi vezani za QoS kriterijume ali su QoS kriterijumi heterogeni i izraženi pomoću nekoliko različitih jedinica mere. Tada se problem modeluje kao diskretan problem višekriterijumskog razlomljenog programiranja (*eng. Multi-Objective Fractional Programming Problem* – MOFPP) u kome se QoS kriterijumi uparuju sa ciljem da se optimizuju njihovi količnici. Pri tome se QoS kriterijumi uparuju uzimajući u obzir njihovu prirodu (jedinicu mere i vrstu optimizacije - minimizacija ili maksimizacija) čime se dobiju razlomljene funkcije cilja koje imaju iste jedinice mere i koje se optimizuju u istom smeru. Dakle, iako početni problem zapravo ne predstavlja problem razlomljenog programiranja, navedeni pristup omogućava donosiocu odluke da uparivanjem kriterijuma iskaže kompromise koji su za njega prihvatljivi. Prednosti predloženog pristupa nad SAW pristupom proizilaze iz činjenice da MOFPP ne podrazumeva formiranje jedinstvene funkcije cilja zbog čega se ne zahteva normalizacija podataka niti dodeljivanje arbitrarnih težina QoS kriterijumima. Otuda ovaj pristup ima prednost u onim slučajevima kada se normalizacijom remeti originalni odnos između vrednosti QoS atributa. Pri tome, formirani model u opštem slučaju može biti nelinearan (ako se u brojiocima ili imeniocima količnika nađu QoS atributi koji se agregiraju pomoću nelinearnih funkcija), što znači da predloženi pristup ne nameće ograničenje u pogledu linearnosti funkcija za agregiranje QoS atributa. U tom kontekstu je od posebnog značaja činjenica da se ne zahteva normalizacija. Međutim, u radu je ilustrovano da ukoliko se, u cilju lakšeg rešavanja, želi primeniti neka metoda koja je razvijena za probleme linearnog razlomljenog programiranja nelinearni agregirani QoS kriterijumi se mogu primenom odgovarajućih transformacija (na primer, logaritmovanjem) prevesti u linearne kriterijume čime se dobija MOLFPP (*eng. Multi-Objective Linear Fractional Programming Problem*).

Metode

Za **problem selekcije pojedinačnih servisa** u četvrtom poglavlju se predlaže proširenje AHP (*eng. Analytic Hierarchy Process*) metode, kao jedne od najčešće korišćenih metoda višekriterijumskog odlučivanja. Naime AHP metoda, omogućava da se prevaziđu neki od nedostataka SAW pristupa, kao što su problem određivanja težina i normalizacije podataka, ali kao i SAW metoda podrazumeva nezavisnost samih kriterijuma. Sa druge strane iako metoda Analitičkih Mrežnih Procesa (*eng. Analytical Network Process – ANP*) dozvoljava postojanje međuzavisnih uticaja, ona ipak, zapravo jedino omogućava da se odnosi između različitih elemenata izraze putem relativnih težina, što znači da ne obezbeđuje mehanizme za eksplicitno izražavanje složenih odnosa koji mogu postojati između ovih elemenata. Stoga je predloženo njeno proširenje primenom fazi logike čime se omogućava da se prilikom selekcije uzmu obzir različiti mogući odnosi koji mogu postojati između kriterijuma kao na primer činjenica da oni mogu biti korelisani ili konfliktni ali i da donosilac odluke može želeti da specificira i različite vrste kompromisa koje je spreman da prihvati. Pri tome se fazi logika može primeniti u različitim koracima AHP metode u zavisnosti od toga da li je samo cilj definisan kao logički zahtev ili su i kriterijumi (i potkriterijumi) takođe logički zahtevi i navedeni postupak je ilustrovan odgovarajućim primerima. Posebno je istaknuto da je, u slučaju kada definisani logički zahtev uključuje negaciju nekog od elemenata, pogodnije koristiti konzistentnu fazi logiku i pokazano je da primena ove dve logike u tom slučaju neće voditi ka istom izboru. Konačno, u slučajevima kada donosilac odluke želi da donese odluku na osnovu više različitih zahteva vezanih za date kriterijume, predloženo je primena pseudo-logičke agregacije koja omogućava da se različiti zahtevi integrišu u jedinstveni globalni zahtev i navedeni pristup je takođe ilustrovan odgovarajućim primerom.

Kada je u pitanju **problem selekcije servisa za kompoziciju** u petom poglavlju predloženo je nekoliko pristupa. Imajući u vidu uspešnost primene poznatih metoda višekriterijumske optimizacije (egzaktnih i heurističkih) za rešavanje identifikovanog problema namera je bila da se razmotri mogućnost primene drugih metoda, koje nisu do sada korišćene za problem selekcije web servisa, kako za postojeće modele problema tako i za predložene nove modele.

Stoga je najpre za predloženi način modelovanja problema u slučajevima kada ne postoje logički zahtevi vezani za QoS kriterijume – MOLFPP predložena, po prvi put, primena novije tehnike za generisanje jako efikasnih rešenja koji je u ovom radu prilagođena za diskretan MOLFPP. Naime, pošto većina postojećih pristupa za rešavanje MOLFPP problema polazi od modela kod kojih je potrebno optimizovati težinsku sumu količnika, a imajući u vidu izložene probleme u pogledu određivanja adekvatnih težina ali i činjenicu da u slučaj kada je reč o kombinatornim problemima primena navedene tehnike otvara pitanje nepodržanih efikasnih rešenja, ideja je bila da se obezbedi pristup koja ne zahteva formiranje težinske sume. Pri tome je pokazano da primena izložene tehnike omogućava egzaktno rešavanje problema u veoma kratkom vremenskom roku čak i za probleme većih dimenzija.

Zatim je pažnja usmerena na primenu heurističkih metoda za rešavanje vremenski kritičnih problema većih dimenzija. Prednost se daje heuristikama imajući u vidu činjenicu da je jedan od zahteva, koji je u poslednje vreme sve aktuelniji, obezbeđivanje mogućnosti dinamičke kompozicije servisa. Naime, egzaktne metode za rešavanje problema mogu biti neadekvatne (u pogledu računarskih resursa i vremena) za dinamičku selekciju s obzirom na to da broj mogućih kombinacija servisa za datu kompoziciju eksponencijalno raste sa porastom broja komponenti i/ili raspoloživih servisa.

Sa jedne strane razmatralo se rešavanje MMKP modela problema, budući da je taj pristup modelovanju problema selekcije servisa za kompoziciju u literaturi najzastupljeniji, a kako je poznato da je MMKP problem NP težak, ideja je bila da se ispita mogućnost primene nekih metoda koje nisu do sada korišćene za problem selekcije servisa modelovanog na ovaj način kako bi se unapredilo rešavanje ovog problema kada je neophodna brza i dinamička selekcija servisa. Sa druge strane, akcenat je bio i na tome da se obezbede metode koje će omogućiti rešavanje problema bez nametanja ograničenja linearnosti funkcija za agregaciju kako bi se selekcija mogla vršiti na osnovu bilo kog skupa QoS mera uključujući i domensko-specifične QoS mere koje se agregiraju pomoću korisnički definisanih funkcija.

Budući da se predložene metode zasnivaju na principu lokalnog pretraživanja najpre je dat opis ove metode i diskutovan problem zaglavljivanja u lokalnim optimumima. Zatim je za selekciju servisa za kompoziciju predložena primena metode

promene okolina (*eng. Variable Neighborhood Search – VNS*), koja nije do sada korišćena za rešavanje ovog problema. Prednost VNS metode se ogleda u tome što zahteva podešavanje samo malog broja parametara pri čemu se može uspešno primeniti i za rešavanje nelinearnih problema.

Pored toga je razvijena i nova hibridna metaheuristika zasnovana na VNS i tabu pretraživanju koja se može primeniti u uslovima kada je neophodna brza i dinamička selekcija. Prilikom definisanja predložene heuristike pošlo se od činjenice da je u slučaju globalne optimizacije neophodno obezbediti odgovarajuće mehanizme pomoću kojih će se pretraga intenzivirati u onim oblastima koje su obećavajuće i diversifikovati kako bi se usmerila u nove, do tada neispitane, oblasti dopustivog prostora. Stoga je ideja bila da se kombinuju dve poznate metaheuristike (VNS i tabu pretraživanje) koje primenjuju različite mehanizme kako za izvlačenje iz lokalnih optimuma tako i za proširivanje pretrage u neke druge oblasti ali i za intenziviranje pretrage u obećavajućim oblastima. U predloženoj metaheuristici u okviru VNS metode se za lokalno pretraživanje u svakom od koraka primenjuje tabu pretraživanje, pri čemu je uvedena i dodatna tabu lista u kojoj će se pamtiti i informacije dobijene u prethodnim VNS koracima.

Predložene metode su primenjene, kako za rešavanje problema modelovanog kao MMKP (budući da je ovaj pristup modelovanju problema selekcije u literaturi najzastupljeniji), tako i za rešavanje problema iskazanog pomoću predloženog novog pristupa modelovanju zasnovanog na primeni konzistentne fazi logike koji u opštem slučaju može biti nelinearan. Uz to je predložena i jedna nova metoda koja se može koristiti za efikasno pronalaženje početnog dopustivog rešenja MMKP modela problema a koja je zasnovana na principu pohlepnog dodavanja (*eng. greedy add approach*) sa vraćanjem unazad (*eng. backtracking*).

Eksperimentalni rezultati pokazuju da se predloženi pristupi mogu efikasno koristiti za rešavanje problema selekcije servisa za kompoziciju bez nametanja ograničenja linearnosti. Na ovaj način se omogućava da se selekcija vrši na osnovu bilo kog skupa QoS mera uključujući i domensko-specifične QoS mere koje se agregiraju pomoću korisnički definisanih funkcija.

7.2. Osvrt na postavljene hipoteze

Sa gledišta postavljene hipoteze:

> *„Moguće je definisati nove modele za formulisanje problema selekcije web servisa zasnovane na njihovim nefunkcionalnim karakteristikama koji će na adekvatan način uzeti u obzir složene verbalne zahteve korisnika, različite odnose koji postoje između relevantnih nefunkcionalnih karakteristika kao i specifičnosti samih mera."*

rezultati rada se mogu sumirati na sledeći način:

- U radu je pokazano da predstavljanje preferenci korisnika vezanih za nefunkcionalne karakteristike putem arbitrarnog dodeljivanja težina najčešće ne mora na adekvatan način odražavati preference korisnika te da je adekvatnija primena fazi logike. Pored problematičnosti u izražavanju značaja pojedinog QoS atributa pomoću težine, dodatni problem je heterogenost samih QoS mera koje zahteva da se izvrši i normalizacija njihovih vrednosti. Sa druge strane primena fazi logike prevazilazi ovaj problem izražavanjem QoS atributa pomoću stepena pripadnosti koji direktno izražava značaj i budući da je u opsegu [0,1] ne zahteva normalizaciju. Šta više na ovakav način se mogu mnogo prirodnije tretirati kvalitativne QoS mere.
- Isto tako složeni verbalni zahtevi korisnika (na primer situacije u kojima korisnik želi da nadomesti lošiju vrednost po nekom od kriterijuma nekim drugom kriterijumom, odnosno u kojima značaj kriterijuma zavisi od ostvarenih vrednosti drugih kriterijuma) ne mogu se predstaviti dodeljivanjem težina ali se prirodno mogu izraziti pomoću logičkih funkcija.
- Primenom konzistentne fazi logike moguće je složene verbalne zahteve korisnika prevesti u logičke funkcije, definisane u Bulovom okviru, koje će uzeti u obzir i korelaciju koja može postojati među nefunkcionalnim karakteristika. Šta više, primena konzistentne fazi logike ne mora rezultovati istim izborom servisa kao primena konvencionalne fazi logike naročito ako se uzme u obzir da zahtev može uključivati negaciju koja u klasičnoj fazi logici narušava princip kontradikcije i isključenja trećeg.
- Pokazano je i da se, u slučajevima kada ne postoje logički zahtevi vezani za nefunkcionalne kriterijume ali su dati kvantitativni kriterijumi heterogeni,

problem može modelovati kao diskretni problem razlomljenog programiranja MOFPP (Multi-Objective Fractional Programming Problem) kao i da je ovakav pristup takođe adekvatniji od formiranja jedinstvene kriterijumske funkcije pomoću metode jednostavnih aditivnih težina (SAW).

Sa gledišta postavljene hipoteze:

„Metode višekriterijumskog odlučivanja i optimizacije se mogu efektivno i efikasno koristiti za rešavanje problema selekcije, kako pojedinačnih web servisa tako i servisa za kompoziciju, na osnovu nefunkcionalnih karakteristika."

rezultati rada se mogu sumirati na sledeći način:

- Pokazano je da je moguće proširiti AHP (Analytic Hierarchy Process) metodu kako bi se omogućila selekcija pojedinačnih servisa na osnovu logičkih zahteva vezanih za njihove nefunkcionalne karakteristike.
- Pokazano je da je moguće primeniti noviju tehniku za generisanje efikasnih rešenja za egzaktno rešavanje problema selekcije servisa za kompoziciju modelovanog kao diskretni MOLFPP.
- Pokazano je da je moguće primeniti VNS (Variable Neighborhood Search) za rešavanje problema selekcije servisa za kompoziciju, modelovanog kako pomoću postojećeg pristupa kao što je MMKP (Multi-dimensional Multiple-choice 0-1 Knapsack Problem) tako i pomoću predloženog modela zasnovanog na primeni konzistentne fazi logike.
- Razvijena je nove heuristika koja omogućava efikasno rešavanje problema kada je neophodna brza i dinamička selekcija servisa za kompoziciju.

Na kraju, na osnovu iznete analize, može se zaključiti da je u radu dokazana i opšta hipoteza:

„Primenom različitih savremenih pristupa mekog računarstva i optimizacije moguće unaprediti modelovanje i rešavanje problema selekcije i kompozicije web servisa zasnovane na njihovim nefunkcionalnim karakteristikama."

7.3. Ostvareni doprinosi

Među ostvarenim doprinosima se izdvajaju:

- Povezivanjem oblasti mekog računarstva i optimizacije i oblasti razvoja informacionih sistema ostvarena je multidisciplinarnost;
- Dat je kritički osvrt na dosadašnja istraživanja u predmetnoj oblasti sa posebnim osvrtom na relevantne pristupe;
- Definisani su novi modeli za formulisanje problema selekcije web servisa zasnovane na njihovim nefunkcionalnim karakteristikama koji na adekvatan način uzimaju u obzir složene verbalne zahteve korisnika, različite odnose koji postoje između relevantnih nefunkcionalnih karakteristika kao i specifičnosti samih mera [*novi modeli*];
- Predloženo je inovativno proširenje AHP metode (kao jedne od najčešće korišćenih metoda višekriterijumskog odlučivanja) konzistentnom fazi logikom koje se može koristiti, ne samo za rešavanje problema pojedinačne selekcije servisa već i u bilo kom drugom domenu odlučivanja u kome je potrebno doneti odluku na osnovu složenih verbalnih zahteva korisnika. [*nova metoda*];
- Pokazana je mogućnost efikasne primene postojećih metoda višekriterijumske optimizacije koje nisu do sada korišćene za problem selekcije web servisa za kompoziciju na osnovu nefunkcionalnih karakteristika, kako za postojeće modele problema tako i za predložene nove modele [*nova primena*];
- Razvijena je nova metaheuristika koja unapređuje rešavanje problema kada je neophodna brza i dinamička selekcija servisa za kompoziciju[*nova metoda*];
- Razvijeno je prototipsko softversko rešenje kojim se obezbeđuje implementacija predloženih metoda [*novi softver*];

Društveni doprinos se ogleda u mogućnosti primene predloženih pristupa u raznim domenima a ne samo za razvoj aplikacija i automatizaciju poslovanja. Pre svega treba imati u vidu da servisno-orijentisani pristup razvoju aplikacija predstavlja opštu paradigmu za obezbeđivanje složene funkcionalnosti povezivanjem skupa komponenti, koji se ne mora nužno zasnivati na web servisima, iako se u praksi najčešće na ovaj način realizuje. Dakle predloženi pristupi bi se mogli primeniti i za selekciju komponenti realizovanih na druge načine. Šta više predloženi pristupi bi se mogli koristiti i u bilo kom drugom polju u kome se ideja povezivanja jedinica funkcionalnosti (koje ne moraju nužno biti softverske) može primeniti.

Tokom izrade ove disertacije objavljeno je nekoliko radova:

Dragović, I., Turajlić, N., Radojević, D. (2012). *Extending AHP with Boolean Consistent Fuzzy Logic and Its Application in Web Service Selection.* In Proc. of the 10th International FLINS Conference on Foundations and Applications of Computational Intelligence - FLINS'12, (Istanbul, Turkey), pp. 576-591. (ISBN 978-981-4417-73-0). [M33]

Turajlić, N., Nešković, S., (2012). *Variable Neighborhood Search and Tabu Search for the Web Service Selection Problem*. Electronic Notes in Discrete Mathematics, Vol. 39, pp. 177–184. (ISSN 1571-0653, DOI: 10.1016/j.endm.2012.10.024). [M33]

Turajlić, N., Dragović, I., (2012). *A Hybrid Metaheuristic Based on Variable Neighborhood Search and Tabu Search for the Web Service Selection Problem*. Electronic Notes in Discrete Mathematics, Vol. 39, pp. 145-152. (ISSN 1571-0653, DOI: 10.1016/j.endm.2012.10.020). [M33]

Dragović, I., Turajlić, N., Radojević, D., Petrović, B. (2012). *Korišćenje logičke agregacije za selekciju web servisa*. In Proc. of SYM-OP-IS 2012, (Tara, Srbija), pp. 377-380. (ISBN-978-86-7488-086-9). [M63]

Stanojević, M., Stanojević, B., Turajlić, N. (2013). *Solving the Web Service Selection Problem Using Multi-Objective Linear Fractional Programming.* In Proc. of BALCOR 2013 (Beograd i Zlatibor, Srbija), pp. 617-622. (ISBN-978-86-7680-285-2. [M33]

Dragović, I., Turajlić, N., Radojević, D., Petrović, B. (2013). *Combining Boolean Consistent Fuzzy Logic and AHP Illustrated on the Web Service Selection Problem.* International Journal of Computational Intelligence Systems, Vol. 7, Supplement 1, pp. 84-93. (ISSN 1875-6891 (Print), 1875-6883 (Online), DOI: 10.1080/18756891.2014.853935). [M23, eSCI, IF: 0.451]

Stanojević, M., Stanojević, B., Turajlić, N. (2014). *Optimization of multiple-objective web service selection using fractional programming*. Annals of Data Science. (ISSN 2198-5804 (Print), 2198-5812 (Online)). [M23] – accepted for publication.

7.4. Mogući dalji pravci istraživanja

Dalje istraživanje u ovoj oblasti moglo bi ići u nekoliko pravaca:

- Rešavanje problema modelovanih pomoću predloženih novih pristupa pomoću drugih metoda.

  S obzirom na uspešnost rešavanja datih modela primenom metoda koje su u ovom radu izložene mogla bi se ispitati i mogućnost primene nekih drugih metoda za njihovo rešavanje. Naime za predloženi MOFP model problema mogla bi se ispitati mogućnost primene neke metode višekriterijumskog razlomljenog programiranja koja omogućava rešavanje problema u slučaju kada su kriterijumske funckije nelinearne. Predloženi model zasnovan na primeni konzistentne fazi logike mogao bi se rešavati i primenom drugih heurističkih metoda kao što su genetski i drugi evolutivni algoritmi.

- Primena predloženih metoda i na druge modele problema.

  Imajući u vidu da se VNS uspešno primenjuje u čitavom nizu oblasti uključujući i optimizaciju na grafovima kao i probleme raspoređivanja, ona bi se mogla primeniti i za rešavanje problema selekcije servisa za kompoziciju modelovanog i na neki od drugih načina koji se predlažu literaturi kao što su: MCOP (Multi-Constrained Optimal Path problem) ili RCPSP (Resource Constrained Project Scheduling Problem) i drugi pristupi modelovanju ovog problema koji su zasnovani na grafovima. Isto tako bi se mogla ispitati mogućnost primene predložene nove metaheuristike za rešavanje takvih modela problema.

- Ispitivanje mogućnosti unapređenja predloženih metoda za rešavanje problema modelovanih pomoću pristupa koji se u ovom radu predlažu.

  Kao prvo mogla bi se ispitati mogućnost da se u predloženom proširenju AHP metode koriste fazi preference umesto Satijeve skale 9 tačaka.

  Zatim bi se mogla ispitati mogućnost primene neke druge varijante VNS metode, kao što je na primer metoda promene okolina sa najboljim poboljšanjem (*eng. Best Improvement VNS – BIVNS*) koja podrazumeva pre izbora rešenja u koje će se u određenom koraku preći ispitaju sve definisane okoline, metoda promena okolina sa dekompozicijom (*eng. Variable Neighborhood Decomposition Search – VNDS*) kod koje se prilikom

definisanja strukture okolina određenog rešenja fiksira određeni broj elemenata (promenljivih) datog rešenja, bijasirani VNS (*eng. Skewed VNS – SVNS*) u kome se razdaljina između tekućeg rešenja i pronađenog novog lokalnog optimuma, koristi za definisanje nove strukture okoline kako bi se pretraga u određenim trenucima usmerila ka okolinama koje su daleko od tekućeg rešenja itd. Posebno bi bilo interesantno ispitati mogućnost primene paralelnih VNS metoda (*eng. Parallel VNS – PVNS*) u uslovima kada je neophodna brza dinamička selekcija servisa budući da one omogućavaju bilo paralelno pokretanje više lokalnih pretraga iz istog rešenja, bilo paralelno pokretanje lokalnih pretraga iz većeg broja rešenja nasumično izabranih iz tekuće okoline rešenja, pri čemu je moguće i da se ispitivanje svake od okolina dodeli drugom procesoru.

Sa druge strane, kada je reč o predloženoj metaheuristici mogla bi se ispitati mogućnost unapređenja tabu pretraživanja, uvođenjem kriterijuma aspiracije ili nekih drugih mehanizama za intenziviranje ili diversifikaciju pretrage (kao što su veći broj tabu listi, druge vrste memorijskih struktura, relaksacija ograničenja pomoću strateške oscilacije ili uvođenjem kaznenih i podsticajnih faktora itd). Sa druge strane i kod tabu pretraživanja postoji mogućnost paralelizacije koja bi se takođe mogla iskoristiti za unapređenje procesa pretraživanja u slučaju dinamičke selekcije servisa.

- Primena predloženih pristupa u druge svrhe.

  Mogla bi se ispitati mogućnost primene predloženih pristupa i za određivanje optimalne konfiguracije kompozicije servisa, odnosno optimalnog redosleda njihovog izvršavanja, u slučajevima kada dati redosled nije predodređen prirodom procesa koji se automatizuje. Na primer, kada je reč o automatizaciji procesa ekstrakcije, transformacije i punjenja podataka (*eng. Extract, Transform and Load Process – ETL*) skladišta podataka (*eng. data warehouse*) mogla bi se ispitati mogućnost primene predloženih pristupa za određivanje optimalnog plana izvršavanja definisanog ETL procesa.

8. Literatura

Aarts, E. H., and Lenstra, J. K. (Eds.). (2003). *Local Search in Combinatorial Optimization*. Princeton University Press, Princeton, NJ, USA.

Agarwal, S., and Lamparter, S. (2005). User preference based automated selection of web service compositions. *In Proc. of ICSOC Workshop on Dynamic Web Processes* (Amsterdam, Netherlands), (pp. 1-12).

Agarwal, S., Lamparter, S., and Studer, R. (2009). Making Web services tradable: A policy-based approach for specifying preferences on web service properties. *Web Semantics: Science, Services and Agents on the World Wide Web, 7*(1), pp. 11-20.

Aggarwal, R., Verma, K., Miller, J., and Milnor,W. (2004). Constraint driven web service composition in METEOR-S. *In Proc. of the IEEE International Conference on Service Computing - SCC'04*, (Shanghai, China), pp. 23-30.

Ai, L. (2011). *QoS-Aware Web Service Composition Using Genetic Algorithms*, PhD Thesis, Queensland University of Technology.

Ai, L., and Tang, M. (2008). A penalty-based genetic algorithm for QoS-aware Web service composition with inter-service dependencies and conflicts. *In Proc. of the IEEE International Conference on Computational Intelligence for Modelling Control & Automation - CIMCA'08* (Wien, Austria), pp. 738-743.

Almulla, M., Almatori, K., and Yahyaoui, H. (2011). A QoS-based fuzzy model for ranking real world web services. *In Proc. of the IEEE International Conference on Web Services – ICWS* (Washington, DC, USA), pp. 203-210.

Alrifai, M. and Risse ,T. (2009). Combining global optimization with local selection for efficient QoS-aware service composition. *In Proc. of the 18th International Conference on World WideWeb (WWW)* (Madrid, Spain), pp. 881-890.

Andresen, K. (2006). *Design and Use Patterns of Adaptability in Enterprise Systems*. In N. Gronau (Ed), Reihe Wirtschaftsinformatik (Vol. 5). GITO mbH Verlag, Berlin.

Ardagna, D. and Pernici, B. (2005). Global and Local QoS Guarantee in Web Service Selection.In C.J. Bussler & A. Haller (Eds.), *Business process management workshops, LNCS 3812*, pp. 32-46.

Ardagna, D. and Pernici, B. (2007). Adaptive service composition in flexible processes. *IEEE Transactions on Software Engineering, 33*(6), pp. 369-384.

Barros, A. I. (1998). *Discrete and Fractional Programming Techniques for Location Models*. Series: Combinatorial Optimization (Vol. 3). Kluwer Academic Publishers, Dordrecht.

Barzilai, J., and Lootsma, F. A. (1997). Power relations and group aggregation in the multiplicative AHP and SMART. *Journal of Multi-Criteria Decision Analysis, 6*(3), pp. 155-165.

Bass, L., Clements, P., and Kazman, R. (2003). *Software Architecture in Practice* (2nd ed.). Series: SEI series in software engineering. Addison-Wesley Professional, Boston, MA, USA.

Berbner, R., Spahn, M., Repp, N., Heckmann, O., and Steinmetz, R. (2006). Heuristics for QoS -aware web service composition. *In Proc. of the International Conference on Web Services - ICWS'06* (Chicago, IL, USA), pp. 72-82.

Botella, P., Burgués, X., Carvallo, J. P., Franch, X., Grau, G., Marco, J., and Quer, C. (2004). ISO/IEC 9126 in practice: what do we need to know?. *In Proc. of the First Software Measurement European Forum - SMEF* (Rome, Italy), pp. 297-306.

Brathwaite, K. (1990). *Application Development Using Case Tools*. Academic Press. San Diego, CA, USA.

Brimberg, J., Hansen, P., and Mladenović, N. (2004). Convergence of variable neighborhood search. *Les Cahiers du GERAD G-2002-21*.

Buckley, J. J., Feuring, T., and Hayashi, Y. (2001). Fuzzy hierarchical analysis revisited. *European Journal of Operational Research, 129*(1), pp. 48-64.

Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. (2004). A lightweight approach for QoS-aware service composition. *In Proc. of 2nd International Conference on Service Oriented Computing - ICSOC'04* (New York, USA).

Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. (2008). A framework for QoS-aware binding and re-binding of composite web services. *Journal of Systems and Software, 81*(10), pp. 1754-1769.

Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. (Jul, 2005). Qos-aware replanning of *composite web services. In Proc. of the IEEE International Conference on Web Services - ICWS 2005* (Orlando, Florida), pp. 121-129.

Canfora, G., Di Penta, M., Esposito, R., and Villani, M. L. (Jun, 2005). An approach for QoS-aware service composition based on genetic algorithms. *In Proc. of the 7th annual conference on Genetic and evolutionary computation - GECCO'05* (Washington, DC, USA), pp. 1069 – 1075.

Canfora, G., Di Penta, M., Esposito, R., Perfetto, F., and Villani, M. L. (2006). Service composition (re) binding driven by application–specific QoS. In A. Dan & W. Lamersdorf (Eds.), *Service-Oriented Computing - ICSOC 2006, LNCS 4294*, pp. 141-152.

Cao, H., Feng, X., Sun, Y., Zhang, Z., and Wu, Q. (2007). A service selection model with multiple QoS constraints on the MMKP. *In Proc. of the IFIP International Conference on Network and Parallel Computing Workshops NPC Workshops*, (Liaoning, China), pp. 584-589.

Cardoso, A. J. S. (2002). *Quality of Service and Semantic Composition of Workflows*. PhD Thesis, University of Georgia.

Chang, D. Y. (1996). Applications of the extent analysis method on fuzzy AHP. *European Journal of Operational Research, 95*(3), pp. 649-655.

Chen, M. F., Tzeng, G. H., and Ding, C. G. (2003). Fuzzy MCDM approach to select service provider. *In Proc. of the 12th IEEE International Conference on Fuzzy Systems - FUZZ'03 (Vol.1)* (St Louis, MO, USA), pp. 572-577.

Chin, K. S., Tang, D. W., Yang, J. B., Wong, S. Y., and Wang, H. (2009). Assessing new product development project risk by Bayesian network with a systematic probability generation methodology. *Expert Systems with Applications, 36*(6), pp. 9879-9890.

Chou, D., deVadoss, J., Erl, T., Gandhi, N., Kommalapati, H., Loesgen, B., Schittko, C., Wilhelmsen, H., and Williams, M. (2010). *SOA with .NET and Windows Azure: Realizing Service Orientation with the Microsoft Platform*. Prentice Hall/PearsonPTR, Upper Saddle River, NJ, USA.

Claro, D. B., Albers, P., and Hao, J. K. (2005). Selecting web services for optimal composition. *In Proc. of the ICWS International Workshop on Semantic and Dynamic Web Processes - SDWP 2005* (Orlando, USA), pp. 32-45.

Costa, J. P. (2007). Computing non-dominated solutions in MOLFP. *European Journal of Operational Research, 181*(3), pp. 1464-1475.

Craven, B. D. (1988). *Fractional Programming*. Sigma Series in Applied Mathematics (Vol. 4). Helderman Verlag, Berlin.

Cvetković, D., Čangalović, M., Dugošija, D., Kovačević-Vujčić, V., Simić, S., and Vuleta, J. (1996). *Kombinatorna Optimizacija – Matematicka Teorija i Algoritmi*. Društvo operacionih istraživača Jugoslavije – DOPIS, Beograd.

Dijkstra, E. W. (1982). On the role of scientific thought. *Selected Writings on Computing: A Personal Perspective* (pp. 60-66). Series: Texts and Monographs in Computer Science. Springer New York.

Eeles, P. (2006). *What is a software architecture?*. Preuzeto Aprila 2014 sa http://www.ibm.com/developerworks/rational/library/feb06/eeles/

Ehrgott, M. (2005). *Multicriteria Optimization* (2nd ed.). Springer Berlin Heidelberg.

Erl, T. (2008). *SOA Principles of Service Design*. Prentice Hall/PearsonPTR, Upper Saddle River, NJ, USA.

Fan, X., Jiang, C., and Fang, X. (2009). An efficient approach to web service selection. In W. Liu, X. Luo, F. Lee & J. Lei (Eds), *Web Information Systems and Mining, LNCS 5854* , pp. 271-280.

Fang, H. L. (1994). Genetic Algorithms in Timetabling and Scheduling. PhD Thesis, University of Edinburgh.

Figueira, J., Greco, S., and Ehrgott, M. (Eds.). (2005). *Multiple Criteria Decision Analysis: State of the Art Surveys*. International Series in Operations Research & Management Science (Vol. 78). Springer, New York.

Finkelstein, C. (2004). *Systems Development Strategies for 21st Century Enterprises*. The Enterprise Newsletter, Iss. 24, pp. 1-13. Preuzeto Aprila 2014 sa http://www.ies.aust.com/PDF-papers/ten24.pdf

Frenk, H., and Schaible, S. (2009). Fractional programming. In C.A. Floudas & P. Pardalos (Eds.), *Encyclopedia of Optimization* (2nd ed.) (pp. 1080-1091). Springer New York.

Frenk, J. B. G., and Schaible, S. (2005). Fractional programming. In N. Hadjisavvas, S. Komlósi & S. Schaible (Eds.), *Handbook of Generalized Convexity and Generalized Monotonicity* (pp. 335-386). Series: Nonconvex Optimization and Its Applications (Vol. 76). Springer New York.

Gao, C., and Ma, J. (2013). A Collaborative QoS-Aware Service Evaluation Method for Service Selection. *Journal of Networks, 8*(6), pp. 1370-1379.

Gao, Y., Na, J., Zhang, B., Yang, L., and Gong, Q. (2006). Optimal web services selection using dynamic programming. *In Proc. of the 11th IEEE Symposium on Computers and Communications - ISCC'06* (Cagliari, Italy), pp. 365-370.

Gao, Z. P., Chen, J., Qiu, X. S., and Meng, L. M. (2009). QoE/QoS driven simulated annealing-based genetic algorithm for Web services selection. *The Journal of China Universities of Posts and Telecommunications, 16*, pp. 102-107.

Garg, S. K., Versteeg, S., and Buyya, R. (2013). A framework for ranking of cloud computing services. *Future Generation Computer Systems, 29*(4), pp. 1012-1023.

Gendreau, M. (2003). An Introduction to Tabu Search. In F. Glover & G. Kochenberger (Eds.), *Handbook of Metaheuristics* (pp 37-54). Series: International Series in Operations Research & Management Science (Vol. 57). Springer New York.

Gillmann, M., Weikum, G., and Wonner, W. (2002). Workflow management with service quality guarantees. *In Proc. of the 2002 ACM SIGMOD international conference on Management of data* (Madison, WI, USA), pp. 228-239.

Glover, F. (1989). Tabu search – part I. *ORSA Journal on Computing, 1*(3), pp. 190-206.

Glover, F. (1995). *Tabu Search Fundamentals and Uses*. Working paper, Graduate School of Business, University of Colorado, Boulder.

Glover, F. (1997). Tabu search and adaptive memory programming – advances, applications and challenges. *Series: Interfaces in computer science and operations research (Vol. 7 Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies, R.S. Barr, R.V. Helgason & J.L. Kennington (Eds.))*. pp. 1-75.

Glover, F. (Avgust, 1990). Tabu search: A tutorial. *Interfaces, 20*(4), pp. 74-94.

Glover, F. (Februar, 1990). Tabu search – part II. *ORSA Journal on Computing, 2*(1), pp. 4-32.

Glover, F., and Laguna, M. (2013). Tabu Search*. In P.M. Pardalos, D.-Z. Du & R.L. Graham (Eds.), *Handbook of Combinatorial Optimization (2nd ed.)*, (pp. 3261-3362). Springer New York.

Godse, M., Sonar, R., and Mulik, S. (2008). The Analytical Hierarchy Process approach for prioritizing features in the selection of web service. *In Proc. of the IEEE Sixth European Conference on Web Services - ECOWS'08* (Dublin, Ireland), pp. 41-50.

Grønmo, R., and Jaeger, M. C. (Decembar, 2005). Model-driven semantic web service composition. *In Proc. of the IEEE 12th Asia-Pacific Software Engineering Conference - APSEC'05* (Taipei, Taiwan), pp. 79-87.

Grønmo, R., and Jaeger, M. C. (Jun, 2005). Model-driven methodology for building QoS -optimised web service compositions. In L. Kutvonen & N. Alonistioti (Eds.), *Distributed Applications and Interoperable Systems, LNCS 3543*, pp. 68-82.

Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal, 29*(2), pp. 147-160.

Hanafi, S., Lazić, J., Mladenović, N., Wilbaut, C., and Crévits, I. (2010). New hybrid matheuristics for solving the multidimensional knapsack problem. *Hybrid Metaheuristics, Lecture Notes in Computer Science, 6373*, pp. 118-132.

Hansen, P., and Mladenović, N. (1999). An introduction to variable neighborhood search. In S. Voss, S. Martello, I. H. Osman & C. Roucairol (Eds.), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization* (pp. 433-458). Springer New York.

Hansen, P., and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3), pp. 449-467.

Hansen, P., and Mladenović, N. (2003a). A tutorial on variable neighborhood search. *Les Cahiers du GERAD G-2003-46*.

Hansen, P., and Mladenović, N. (2003b). Variable Neighborhood Search. In F. Glover & G. Kochenberger (Eds.), *Handbook of Metaheuristics* (pp 145-184). Series: International Series in Operations Research & Management Science (Vol. 57). Springer New York.

Hansen, P., and Mladenović, N. (2005). Variable Neighborhood Search. In E.K. Burke & G. Kendall (Eds.), *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* (pp. 211-238). Springer Science+Business Media, New York.

Hansen, P., and Mladenović, N. (2009). Variable neighborhood search methods. In C.A. Floudas & P. Pardalos (Eds.), *Encyclopedia of Optimization* (2nd ed.) (pp. 3975-3989). Springer New York.

Hansen, P., Mladenović, N., and Moreno Pérez, J. A. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research, 175*(1), pp. 367-407.

Harmon, P. (2010). The scope and evolution of business process management. In J. vom Brocke & M. Rosemann (Eds.), *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems* (pp. 37-81). Series: International Handbooks on Information Systems. Springer Berlin Heidelberg.

Havey, M. (2005). *Essential Business Process Modeling*. O'Reilly Media, Inc., Sebastopol, CA, USA.

Herssens, C., Jureta, I. J., and Faulkner, S. (Januar, 2008). Capturing and using QoS relationships to improve service selection. In Z. Bellahsène & M. Léonard (Eds.), *Advanced Information Systems Engineering, LNCS 5074*, pp. 312-327.

Herssens, C., Jureta, I. J., and Faulkner, S. (Jun, 2008). Dealing with quality tradeoffs during service selection. *In Proc. of the International Conference on Autonomic Computing - ICAC'08*. (Chicago, IL, USA), pp. 77-86.

Huang, Z., Jiang, W., Hu, S., and Liu, Z. (2009). Effective pruning algorithm for QoS-aware service composition. *In Proc. of the IEEE Conference on Commerce and Enterprise Computing - CEC'09* (Vienna, Austria), pp. 519-522.

Hwang, C.-L., and Yoon, K. (1981). *Multiple Attribute Decision Making Methods and Applications: A State-of-the Art Survey*. Lecture Notes in Economics and Mathematical Systems (Vol. 104). Springer-Verlag Berlin Heidelberg.

IEEE 1471:2000 – *Recommended practice for architectural description of software intensive systems*. (2000). Preuzeto Aprila 2014 sa IEEE Standards: http://standards.ieee.org/findstds/standard/1471-2000.html

Ishizaka, A., and Labib, A. (2011). Review of the main developments in the analytic hierarchy process. *Expert Systems with Applications, 38*(11), pp. 14336-14345.

ISO 9126 *Software Quality Characteristics*. (2001). Preuzeto Aprila 2014 sa http://www.iso.org; http://www.sqa.net/iso9126.html

ISO/IEC 12207 *Systems and software engineering - Software life cycle processes*. (2008). Preuzeto Aprila 2014 sa International Organization for Standardization: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43447

Jaeger, M. C. (2006). *Optimising Quality-of-Service for the Composition of Electronic Services*, PhD Thesis, Technische Universität Berlin.

Jaeger, M. C., and Ladner, H. (2006). A Model for the aggregation of QoS in WS compositions involving redundant services. *Journal of Digital Information Management, 4*(1), pp. 44-49.

Jaeger, M. C., and Mühl, G. (2007). QoS-based selection of services: the implementation of a genetic algorithm. *In Proc. of ITG-GI Conference Communication in Distributed Systems - KiVS* (Bern, Switzerland), pp. 1-12.

Jaeger, M. C., Mühl, G., and Golze, S. (2005). QoS-aware composition of web services: an evaluation of selection algorithms. In R. Meersman & Z. Tari (Eds.), *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, LNCS 3760*, pp. 646-661.

Jaeger, M. C., Rojec-Goldmann, G., and Mühl, G. (2004). QoS aggregation for web service composition using workflow patterns. *In Proc. of the IEEE Eighth International Enterprise Distributed Object Computing Conference - EDOC 2004* (Monterey, California, USA), pp. 149-159.

Josuttis, N. M. (2007). *SOA in Practice: The Art of Distributed System Design*. O'Reilly Media, Inc., Sebastopol, CA, USA.

Kahraman, C., Ulukan, Z., and Tolga, E. (1998). A fuzzy weighted evaluation method using objective and subjective measures. *In Proc. of the International ICSC Symposium on Engineering of Intelligent Systems - EIS'98 (Vol. 1)* (University of La Laguna, Tenerife), pp. 57-63.

Karim, R., Ding, C., and Chi, C. H. (2011). An enhanced PROMETHEE model for QoS-based web service selection. *In Proc. of the IEEE International Conference on Services Computing - SCC* (Washington, DC, USA), pp. 536-543.

Karp, R. M. (1972). Reducibility among combinatorial problems. In R.E. Miller, J.W. Thatcher & J.D. Bohlinger (Eds.), *Complexity of Computer Computations* (pp. 85-103). Series: The IBM Research Symposia Series (1972). Plenum Press, New York.

Kattepur, A., Benveniste, A., and Jard, C. (2011). Optimizing decisions in web services orchestrations. In G. Kappel, Z. Maamar & H.R. Motahari-Nezhad (Eds), *Service-Oriented Computing, LNCS 7084*, pp. 77-91.

Keeney, R. L., and Raiffa H. (1976). *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. Wiley series in probability and mathematical statistics. John Wiley, New York.

Klein, A., Ishikawa, F., and Honiden, S. (2011). Efficient heuristic approach with improved time complexity for QoS-aware service composition. *In Proc. of the IEEE International Conference on Web Services - ICWS 2011* (Washington, DC, USA), pp. 436-443.

Kosko, B. (1993). *Fuzzy Thinking: The New Science of Fuzzy Logic*. Hyperion, New York.

Krčevinac, S., Čangalovic, M., Kovačević-Vujičić, V., and Martić, M. (2004). *Operaciona Istraživanja*. Fakultet Organizacionih Nauka, Beograd.

Kruchten, P. (1995). Architectural blueprints – the "4+1" view model of software architecture. *IEEE Software, 12* (6), pp. 42-50.

Leymann, F. (2003). Web Services: Distributed Applications without Limits. In *Proc. of Database Systems For Business, Technology and Web - BTW'03*, (Leipzig, Germany), pp. 26-28.

Lin, M., Xie, J., GuoJ, H., and Wang, H. (2005). Solving QoS-driven web service dynamic composition as fuzzy constraint satisfaction. *In Proc. of the IEEE International Conference on e-Technology, e-Commerce and e-Service - EEE'05* (Hong Kong , China), pp. 9-14.

Linthicum, D (2007). *Service Oriented Architecture (SOA) in the Real World*. Preuzeto Aprila 2014 sa http://msdn.microsoft.com/en-us/library/bb833022.aspx

Liu, D., Shao, Z., Yu, C., and Fan, G. (2009). A heuristic QoS-aware service selection approach to web service composition. *In Proc. of the Eighth IEEE/ACIS International Conference on Computer and Information Science - ICIS 2009* (Shanghai, China) pp. 1184-1189.

Lootsma, F. A. (1993). Scale sensitivity in the multiplicative AHP and SMART. *Journal of Multi-Criteria Decision Analysis, 2*(2), pp. 87-110.

Lotfi, F. H., Noora, A. A., Jahanshahloo, G. R., Khodabakhshi, M., and Payan, A. (2010). A linear programming approach to test efficiency in multi-objective linear fractional programming problems. *Applied Mathematical Modelling, 34*(12), pp. 4179-4183.

Luo, Y. S., Qi, Y., Hou, D., Shen, L. F., Chen, Y., and Zhong, X. (2011). A novel heuristic algorithm for QoS-aware end-to-end service composition. *Computer Communications, 34*(9), pp. 1137-1144.

MacCrimmon, K. R. (1968). *Decisionmaking Among Multiple-Attribute Alternatives: A Survey and Consolidated Approach* (RAND Corporation Memorandum No. RM-4823-ARPA). RAND Corporation, Santa Monica, CA, USA.

MacDonald, M. (2003). *Microsoft .NET Distributed Applications: Integrating XML Web Services and .NET Remoting*. Microsoft Press, Redmond, WA, USA.

Menascé, D. (2002). QoS issues in web services. *IEEE Internet Computing, 6*(6), pp. 72-75.

Microsoft Patterns & Practices Team (2009). *Microsoft Application Architecture Guide* (2nd ed.). Microsoft Press, Redmond, WA, USA.

Ming, C., and Zhen-wu, W. (2007). An approach for web services composition based on QoS and discrete particle swarm optimization. *In Proc. of the IEEE Eighth ACIS*

International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing - SNPD 2007 (Vol. 2) (Qingdao, China) pp. 37-41.

Mladenović, N., and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*(11), pp. 1097-1100.

Mladenović, N., Dražić, M., Kovačevic-Vujčić, V., and Čangalović, M. (2008). General variable neighborhood search for the continuous optimization. *European Journal of Operational Research, 191*(3), pp, 753-770.

Moody, D. L. (2005). Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. *Data & Knowledge Engineering, 55*(3), pp. 243-276.

Nguyen, H. T., and Walker, E. A. (2005). *A First Course in Fuzzy Logic* (3rd ed.). Chapman & Hall/CRC Press, Taylor & Francis Group, Boca Raton, FL, USA.

Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. *In Proc. of the 4th International Conference on Web Information Systems Engineering - WISE 2003*, (Rome, Italy), pp. 3-12.

Parejo, J.A. , Fernandez, P. and Cortes, A.R. (2008). QoS-aware services composition using tabu search and hybrid genetic algorithms. *Actas de los Talleres de las Jornadas de Ingeniera del Software y Bases de Datos, 2*(1), pp. 55-66.

Parnas, D. L. (1972). On the criteria to be used in decomposing systems into modules. *Communications of the ACM, 15*(12), pp. 1053-1058.

Pedrycz, W. (1993). *Fuzzy Control and Fuzzy Systems* (2nd ed.). Research Studies Press Ltd, John Wiley & Sons, Inc. New York.

Pejman, E., Rastegari, Y., Esfahani, P. M., and Salajegheh, A. (2012). Web service composition methods: a survey. *In Proc. of the International MultiConference of Engineers and Computer Scientists - IMECS* (Hong Kong).

Pirlot, M. (1996). General local search methods. *European Journal of Operational Research, 92*(3), pp. 493-511.

Pop, C. B., Vlad, M., Chifu, V. R., Salomie, I., and Dinsoreanu, M. (2011). A tabu search optimization approach for semantic web service composition. *In Proc. of the 10th International Symposium on Parallel and Distributed Computing – ISPDC* (Cluj Napoca, Romania) pp. 274-277.

Prokopyev, O. (2009). Fractional zero-one programming. In C.A. Floudas & P. Pardalos (Eds.), *Encyclopedia of Optimization* (2nd ed.) (pp. 1091-1094). Springer New York.

Puschner, P. P., and Schedl, A. V. (1997). Computing maximum task execution times - a graph-based approach. *Real-Time Systems, 13*(1), pp. 67-91.

Qi, L., Tang, Y., Dou, W., and Chen, J. (2010). Combining local optimization and enumeration for QoS-aware web service composition. *In Proc. of the IEEE International Conference on Web Services – ICWS 2010* (Miami, FL, USA)pp. 34-41.

Radojević, D. (1999). Logical interpretation of discrete Choquet integral defined by general measure. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 7*(06), pp. 577-588.

Radojević, D. (2000). [0,1]-valued logic: a natural generalization of Boolean logic. *Yugoslav Journal of Operations Research, 10*(2), pp. 185-216.

Radojević, D. (2008a). Fuzzy set theory in Boolean frame. *International Journal of Computers, Communications & Control (IJCCC), 3*(Suppl. issue), pp. 121-131.

Radojević, D. (2008b). Interpolative realization of Boolean algebra as a consistent frame for gradation and/or fuzziness. *Series: Studies in Fuzziness and Soft Computing (Vol. 104 Forging New Frontiers: Fuzzy Pioneers II, M. Nikravesh, J. Kacprzyk & L. A. Zadeh (Eds.)*), pp. 295-317

Radojević, D. (2008c). Logical aggregation based on interpolative Boolean algebra. *Mathware & Soft Computing*, *15*(1), pp. 125-141.

Radojević, D. (2010). Logical Aggregation – Why and How. *In Proc. of the 9th International FLINS Conference on Foundations and Applications of Computational Intelligence - FLINS'10*, (Chengdu, China), pp. 511-517.

Radojević, D. (2013). Real-valued realizations of Boolean algebras are a natural frame for consistent fuzzy logic. *Series: Studies in Fuzziness and Soft Computing (Vol. 229 On Fuzziness: A Homage to Lotfi A. Zadeh* – Vol. 2, R. Seising, E. Trillas, C. Moraga & S. Termini *(Eds.)*), pp. 559-565.

Radzik, T. (1998). Fractional combinatorial optimization. In D.-Z. Du & P.M. Pardalos (Eds.), *Handbook of Combinatorial Optimization* (Vol. 1) (pp. 429-478). Kluwer Academic Publishers, Dordrecht.

Rao, J., and Su, X. (2005). A survey of automated web service composition methods. In J. Cardoso & A. Sheth (Eds), *Semantic Web Services and Web Process Composition, LNCS 3387*, pp. 43-54.

Rosen, M., Lublinsky, B., Smith, K. T., and Balcer, M. J. (2008). *Applied SOA: Service-Oriented Architecture and Design Strategies*. Wiley Publishing, Indianapolis, IN, USA.

Saaty, T. L. (1980). *The Analytic Hierarchy Process: Planning, Priority Setting, Resources Allocation*. McGraw-Hill, New York.

Saaty, T. L. (1988). *Multicriteria Decision Making: The Analytic Hierarchy Process*. RWS Publications, Pittsburgh, PA, USA.

Saaty, T. L. (1990). How to make a decision: The analytic hierarchy process. *European Journal of Operational Research, 48*(1), pp. 9-26.

Saaty, T. L. (2004). Fundamentals of the analytic network process – Dependence and feedback in decision-making with a single network. *Journal of Systems science and Systems engineering, 13*(2), pp. 129-157.

Sathya, M., Swarnamugi, M., Dhavachelvan, P., and Sureshkumar, G. (2010). Evaluation of QoS based web-service selection techniques for service composition. *International Journal of Software Engineering, 1*(5), pp. 73-90.

Schaible, S. (1995). Fractional programming. In R. Horst & P.M. Pardalos (Eds.), *Handbook of Global Optimization* (pp. 495-608), Kluwer Academic Publishers, Dordrecht.

Schaible, S. (2002). Fractional programming: a recent survey. *Journal of Statistics and Management Systems, 5*(1-3), pp. 63-86.

Schaible, S., and Shi, J. (2003). Fractional programming: the sum-of-ratios case. *Optimization Methods and Software, 18*(2), pp. 219-229.

Seo, Y. J., Jeong, H. Y., and Song, Y. J. (2005). Best Web service selection based on the decision making between QoS criteria of service. In L.T. Yang & al, (Eds), *Embedded Software and Systems, LNCS 3820*, pp. 408-419.

Shahid, S. (2008). *A Business-Driven Evaluation of Distributed-Computing Models*. Preuzeto Aprila 2014 sa http://msdn.microsoft.com/en-us/library/cc984967.aspx

Shaw, M., and Garlan, D. (1996). *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, Upper Saddle River, NJ, USA.

Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S., and Xu, X. (2014). Web services composition: A decade's overview. *Information Sciences, 280*, pp. 218-238.

Shuai, J. J. (2009). A fuzzy MCDM partnership selection model-case of the IC design house. *In Proc. of the Fourth IEEE International Conference on Innovative Computing, Information and Control - ICICIC* (Kaohsiung, Taiwan), pp. 1452-1455.

SOA Manifesto. (2009). Preuzeto Aprila 2014 sa http://www.soa-manifesto.org/

Sohrabi, S., Prokoshyna, N., and McIlraith, S. A. (2006). Web service composition via generic procedures and customizing user preferences. In I. Cruz & al (Eds), *The Semantic Web - ISWC 2006, LNCS 4273*, pp. 597-611).

Sora, I., Todinca, D., and Avram, C. (2009). Translating user preferences into fuzzy rules for the automatic selection of services. *In Proc. of the 5th International Symposium on Applied Computational Intelligence and Informatics - SACI'09* (Timisoara, Romania), pp. 497-502.

Stancu-Minasian, I. M. (1997). *Fractional Programming: Theory, Methods and Applications*. Series: Mathematics and Its Applications (Vol. 409). Kluwer Academic Publishers, Dordrecht.

Stancu-Minasian, I. M. (2006). A sixth bibliography of fractional programming. *Optimization, 55*(4), pp. 405-428.

Stanojević, B., and Stanojević, M. (2013). On the efficiency test in multi-objective linear fractional programming problems by Lotfi et al. 2010. *Applied Mathematical Modelling, 37*(10), pp. 7086-7093.

Stanojević, B., and Stanojević, M. (2014). *Generation of the non-dominated set for a bi-objective linear fractional programming problem*. – work in progress.

Stevens, W., Myers, G., and Constantine, L. (1974). Structured design. *IBM Systems Journal, 13*(2), pp. 115-139.

Strunk, A. (2010). QoS-aware service composition: A survey. *In Proc. of the IEEE 8th European Conference on Web Services - ECOWS10* (Ayia Napa, Cyprus), pp. 67-74.

Sun, S. X., and Zhao, J. (2012). A decomposition-based approach for service composition with global QoS guarantees. *Information Sciences, 199*, pp. 138-153.

Tang, M., and Ai, L. (2010). A hybrid genetic algorithm for the optimal constrained web service selection problem in web service composition. *In Proc. of the IEEE World Congress on Computational Intelligence – WCCI 2010* (Barcelona, Spain), pp. 268-275.

Tong, H., and Zhang, S. (2006). A fuzzy multi-attribute decision making algorithm for web services selection based on QoS. *In Proc. of the IEEE Asia-Pacific Conference on Services Computing -APSCC'06* (Guangzhou, China), pp. 51-57.

Torfi, F., Farahani, R. Z., and Rezapour, S. (2010). Fuzzy AHP to determine the relative weights of evaluation criteria and Fuzzy TOPSIS to rank the alternatives. *Applied Soft Computing, 10*(2), pp. 520-528.

Törn, A. and Žilinskas, A. (1989). *Global Optimization*. Lecture Notes in Computer Science (Vol. 350), Springer-Verlag, New York.

Tran, V. X., Tsuji, H., and Masuda, R. (2009). A new QoS ontology and its QoS-based ranking algorithm for Web services. *Simulation Modelling Practice and Theory, 17*(8), pp. 1378-1398.

Tzeng, G.-H., and Huang, J.-J. (2011). *Multiple Attribute Decision Making: Methods and Applications*. Chapman and Hall/CRC Press, Taylor & Francis Group, Boca Raton, FL, USA.

Van Laarhoven, P. J. M., and Pedrycz, W. (1983). A fuzzy extension of Saaty's priority theory. *Fuzzy sets and Systems, 11*(1), 199-227.

Vujošević, M., Stanojević, M., and Mladenović, N. (1996). *Metode Optimizacije: Mrežni, Lokacijski i Višekriterijumski Modeli*. Društvo operacionih istraživača Jugoslavije – DOPIS, Beograd.

W3C (2003). *QoS for Web Services: Requirements and Possible Approaches*. Preuzeto Aprila 2014 sa http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/.

W3C (2004). *Web Services Architecture*. Preuzeto Aprila 2014 sa http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

Wada, H., Champrasert, P., Suzuki, J., and Oba, K. (2008). Multiobjective optimization of SLA-aware service compositions. *In Proc. of the IEEE Congress on Congress on Services -Services'08*, (Honolulu, Hawaii, USA), pp. 368-375.

Wada, H., Suzuki, J., Yamano, Y., and Oba, K. (2012). E³: A Multiobjective Optimization Framework for SLA-Aware Service Composition. *IEEE Transactions on Services Computing, 5*(3), pp. 358-372.

Wang, P., Chao, K. M., Lo, C. C., Huang, C. L., and Li, Y. (2006). A fuzzy model for selection of QoS-aware web services. *In Proc. of the IEEE International Conference on e-Business Engineering - ICEBE'06* (Shanghai, China), pp. 585-593.

Wang, W., Sun, Q., Zhao, X., and Yang, F. (2010). An improved particle swarm optimization algorithm for QoS-aware web service selection in service oriented communication. *International Journal of Computational Intelligence Systems, 3*(sup01). pp. 18-30.

Weske, M. (2012). *Business Process Management: Concepts, Languages, Architectures* (2nd ed.). Springer-Verlag Berlin Heidelberg.

White, S. A. (2008). *BPMN Modeling and Reference Guide: Understanding and Using BPMN*. Future Strategies Inc., Lighthouse Point, FL, USA.

Wolfslehner, B., Vacik, H., and Lexer, M. J. (2005). Application of the analytic network process in multi-criteria analysis of sustainable forest management. *Forest Ecology and Management, 207*(1), pp. 157-170.

Xiong, P., and Fan, Y. (2007). QoS-aware web service selection by a synthetic weight. *In Proc. of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery - FSKD'07 (Vol. 3)* (Haikou, China), pp. 632-637.

Ye, X., and Mounla, R. (2008). A hybrid approach to QoS-aware service composition. *In Proc. of the IEEE International Conference on Web Services - ICWS'08* (Bejing, China), pp. 62-69.

Yoon, K. P., and Hwang, C.-L. (1995). *Multiple Attribute Decision Making: An Introduction*. Series: Quantitative applications in the social sciences (Vol. 104). Sage Publications, Thousand Oaks, CA, USA.

Yu, T. and Lin, K.-J. (Decembar, 2005). Service selection algorithms for composing complex services with multiple QoS constraints. In B. Benatallah, F. Casati & P. Traverso (Eds.), *Service-Oriented Computing - ICSOC05, LNCS 3826*, pp. 130-143.

Yu, T., and Lin, K.-J. (Jul, 2005). Service selection algorithms for web services with end-to-end QoS constraints. *Information Systems and E-Business Management, 3*(2), pp. 103-126.

Yu, T., Zhang, Y., and Lin, K.-J. (2007). Efficient algorithms for web services selection with end-to-end QoS constraints. *ACM Transactions on the Web, 1*(1), pp. 1-26.

Yulu, S., and Xi, C. (2011). A survey on QoS-aware web service composition. *In Proc. of the Third International Conference on Multimedia Information Networking and Security - MINES'11* (Shanghai, China), pp. 283-287.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control, 8*(3), pp. 338-353.

Zadeh, L. A. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, 3(1), pp. 28-44.

Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., and Sheng, Q. Z. (2003). Quality driven web services composition. *In Proc. of the 12th international conference on World Wide Web* (Budapest, Hungary), pp. 411-421.

Zeng, L., Benatallah, B., Ngu, A. H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). QoS-aware middleware for web services composition. *IEEE Transactions on Software Engineering, 30*(5), pp. 311-327.

Zhang, Y., Zhang, S., and Han, S. (2006). A new methodology of QoS evaluation and service selection for ubiquitous computing. In X. Cheng, W. Li & T. Znati (Eds.), *Wireless Algorithms, Systems, and Applications, LNCS 4138*, pp. 69-80.

Zimmermann, O., Krogdahl, P., and Gee, C. (2004). *Elements of Service-Oriented Analysis and Design, An interdisciplinary modeling approach for SOA projects*, Preuzeto Aprila 2014 sa http://www.ibm.com/developerworks/library/ws-soad1/

9. Prilog

U ovom poglavlju je predstavljeno originalno softversko rešenje *WSSSolver* za problem selekcije web servisa. Opisane su osnovne funkcionalnosti softvera i dato je kratko uputstvo za korišćenje.

Aplikacija *WSSSolver* je razvijena kao tehnološka podrška predloženom pristupu rešavanju problema selekcije web servisa. Pre svega, implementiran je veći broj funkcionalnosti kojima je omogućeno definisanje odnosno modelovanje problema i njegovo rešavanje korišćenjem predloženih metoda.

Predloženo softversko rešenje je realizovano na .NET platformi korišćenjem Microsoft Visual Studio alata. Korišćena je Windows Forms tehnologija.

Uzimajući u obzir prirodu problema koji se rešava, aplikacija je organizovana kao serija koraka (*eng. wizard*) kojima je omogućeno modelovanje problema koji treba da se reši, definisanje načina na koji je problem potrebno rešiti (tj. izbor odgovarajuće metode), zatim njegovo rešavanje i na kraju pregled dobijenog rešenja. Uopšte, definisani su sledeći koraci:

1. Modelovanje problema
    1.1. Definisanje atributa
    1.2. Definisanje modela problema
    1.3. Definisanje ograničenja
    1.4. Učitavanje QoS vrednosti
2. Rešavanje problema
    2.1. Izbor metode za rešavanje problema
    2.2. Izvršavanje izabrane metode
    2.3. Pregled rešenja

Predstavljeni koraci su u nastavku detaljno objašnjeni.

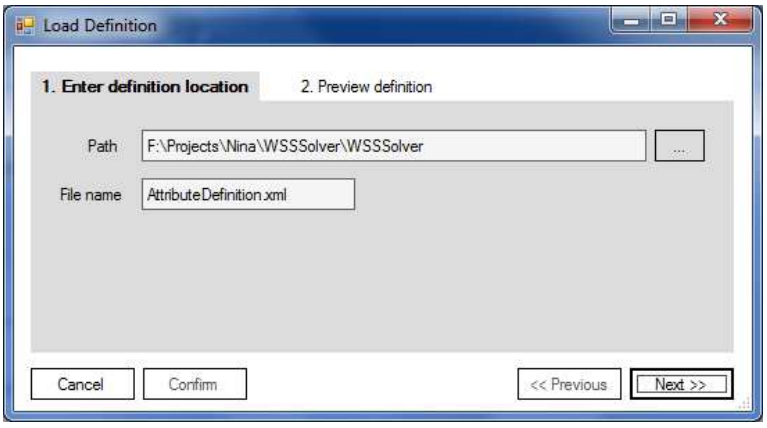
1. Modelovanje problema

Po pokretanju aplikacije potrebno je da se definiše problem koji se rešava. Ovaj korak podrazumeva definisanje odgovarajućih atributa, zatim funkcije cilja i na kraju

ograničenja. Shodno tome, implementirana su četiri potkoraka koja su opisana u nastavku.

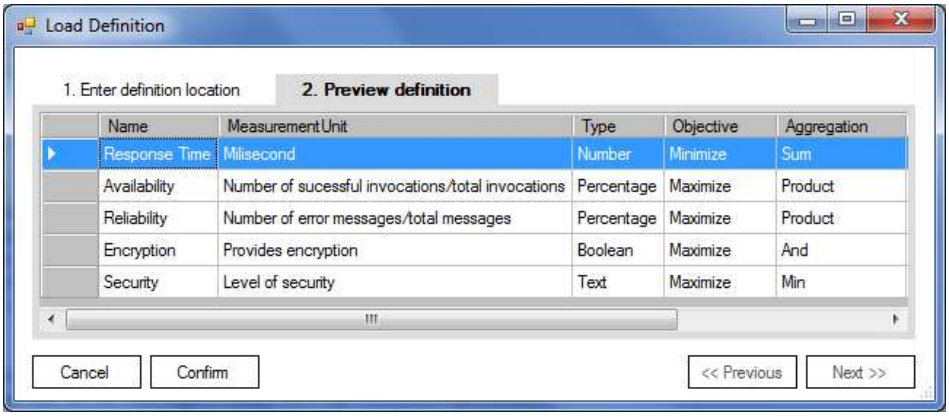
1.1. Definisanje atributa

Definisanje atributa podržano je odgovarajućim *xml* datotekama. Naime, definisana je odgovarajuća *xml* schema u skladu sa kojom se kreiraju *xml* datoteke, odnosno definicije atributa. Shodno tome, implementirana je funkcionalnost kojom se vrši automatsko učitavanje definicije atributa iz izabranog *xml* dokumenta. Učitavanje definicije atributa započinje unosom, tj. izborom putanje do *xml* datoteke, nakon čega se bira opcija *Next* čime se zapravo inicira učitavanje definicije atributa.



Slika 34. Učitavanje definicije atributa – izbor xml datoteke

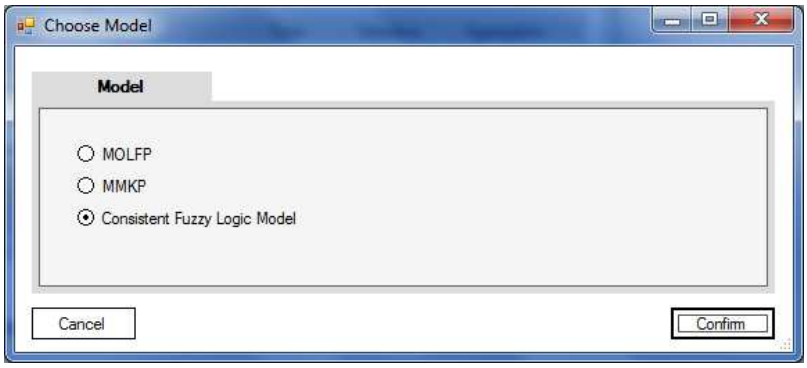
Ukoliko je operacija učitavanja uspešno završena, prikazuje se definicija atributa i bira se opcija *Confirm* kojom se potvrđuje učitana definicija.



Slika 35. Učitavanje definicije atributa - Prikaz definicija atributa

1.2. Definisanje modela problema

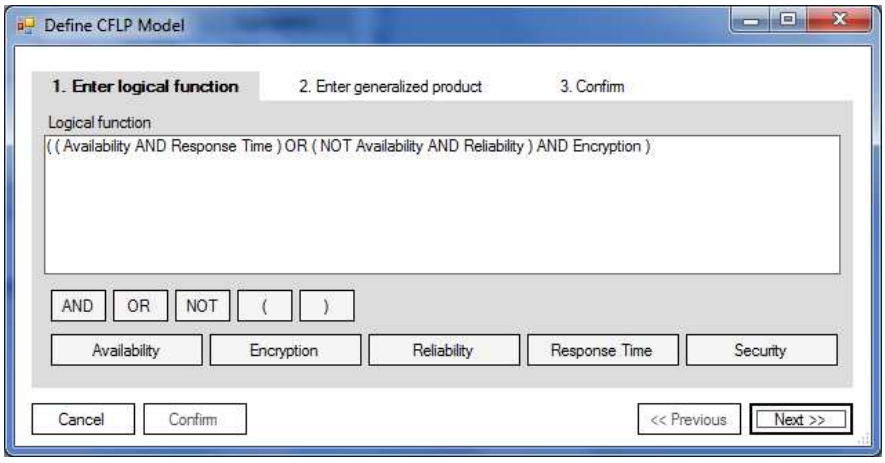
Nakon što je učitana definicija atributa, potrebno je da se definiše model problema. Definisanje modela problema započinje izborom jednog od ponuđenih tipova modela, a zatim se vrši njegovo konfigurisanje. Softverskim rešenjem su implementirana tri tipa modela: MOLFP, MMKP i CFLM.



Slika 36. Definisanje modela problema – izbor tipa modela

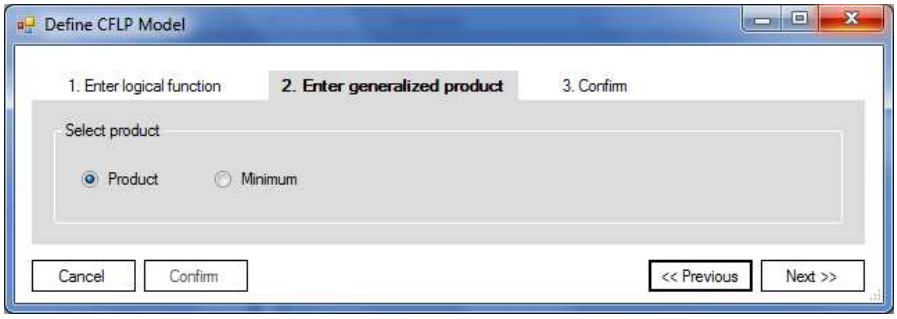
Bira se željeni tip modela problema, a zatim opcija *Confirm* kojom se inicira njegovo podešavanje. Sa svaki od ponuđenih tipova modela implementiran je poseban postupak podešavanja. U nastavku je prikazano podešavanje za izabrani tip modela CFLM, koje se sastoji od tri koraka.

Prvim korakom se definiše logička funkcija. Implementiran je odgovarajući tekstualni editor, kojim su, pored standardnih logičkih operatora *AND*, *OR* i *NOT*, ponuđeni i odgovarajući atributi.



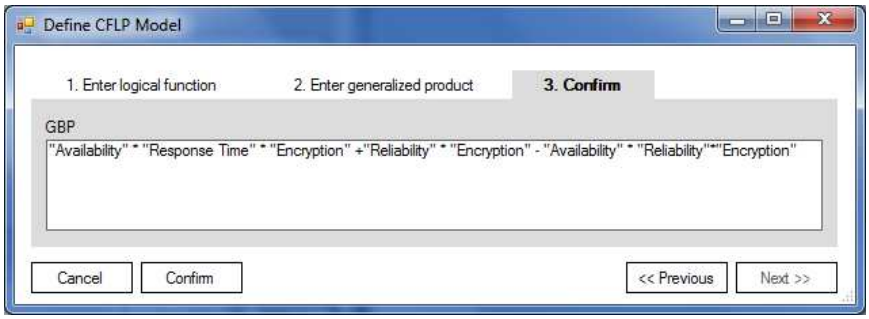
Slika 37. Definisanje CFLM modela – prvi korak

Atributi se dodaju dinamički u odnosu na prethodno učitanu definiciju atributa. U drugom koraku se bira operator generalizovanog proizvoda. Implementacijom su obuhvaćena dva tipa operatora: (1) *product* i (2) *minimum*.



Slika 38. Definisanje CFLM modela – drugi korak

Treći korak podrazumeva da se izvrši odgovarajuća transformacija i zatim se daje pregled kreirane funkcije cilja. Bira se opcija *Confirm* kojom se potvrđuje kreirana funkcija cilja i ujedno prelazi na definisanje ograničenja.

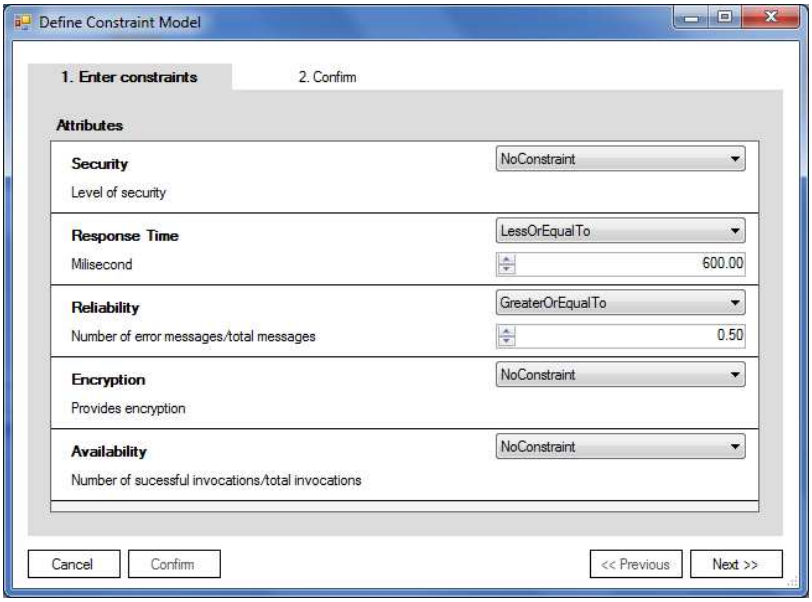


Slika 39. Definisanje CFLM modela – treći korak

1.3. Definisanje ograničenja

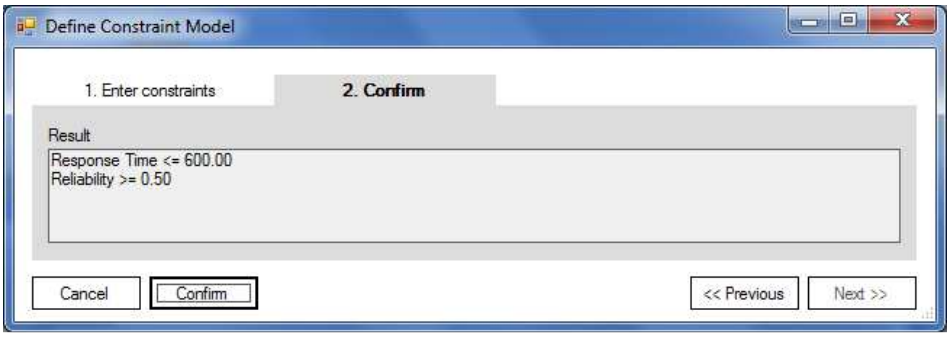
Ograničenja sa zadaju za prethodno definisane atribute i za svaki atribut je moguće definisati najviše jedno ograničenje. Potrebno je napomenuti da definisanje ograničenja nije obavezno.

Definisanje ograničenja nad atributom podrazumeva izbor jednog od tri ponuđena tipa ograničenja: *GreaterOrEqualTo, LessOrEqualTo, EqualTo*. Pošto se izabere tip ograničenja, potrebno je da se definiše i vrednost ograničenja. Ukoliko atribut nema ograničenja prikazuje se vrednost *NoConstraint*.



Slika 40. Definisanje ograničenja

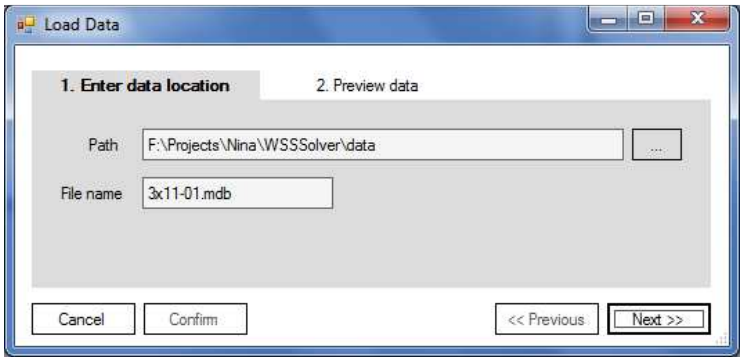
Nakon što su definisana ograničenja bira se opcija *Next* i prelazi se na potvrdu definisanih ograničenja



Slika 41. Definisanje ograničenja - potvrda

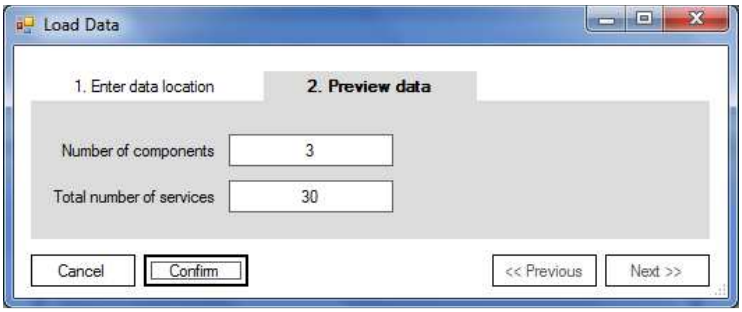
1.4. Učitavanje QoS vrednosti

Poslednji korak tokom definisanja problema je učitavanje QoS vrednosti. Učitavanje započinje unosom, tj. izborom putanje do datoteke u kojoj su smešteni podaci (*xml* ili *mdb* datoteka), nakon čega se bira opcija *Next* čime se zapravo inicira učitavanje podataka.



Slika 42. Učitavanje QoS vrednosti

Ukoliko je operacija učitavanja uspešno završena, prikazuje se broj komponenti i ukupan broj servisa, a zatim se bira opcija *Confirm* kojom se potvrđuju učitani podaci.



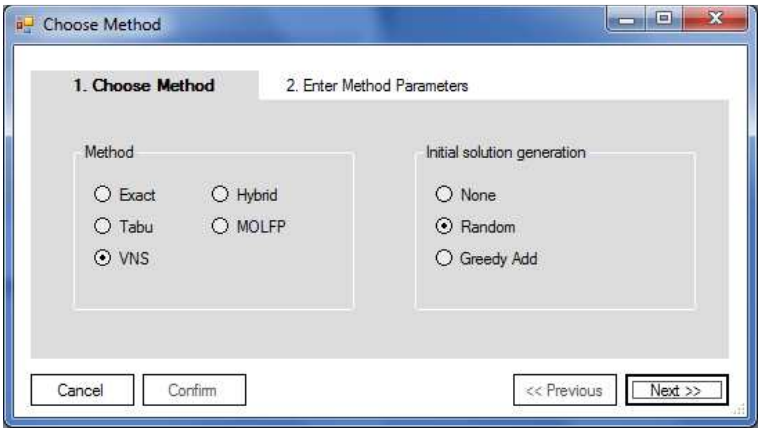
Slika 43. Pregled dimenzija problema

2. Rešavanje problema

Nakon što je u prvom koraku definisan problem, u drugom koraku se prelazi na njegovo rešavanje. Pre svega, vrši se izbor metode kojom će se problem rešavati, zatim se zadaju vrednosti parametra u skladu sa izabranom metodom rešavanja i na kraju se izvršava izabrana metoda sa zadatim parametrima. Shodno navedenom, implementirana su tri potkoraka koji su opisani u nastavku.

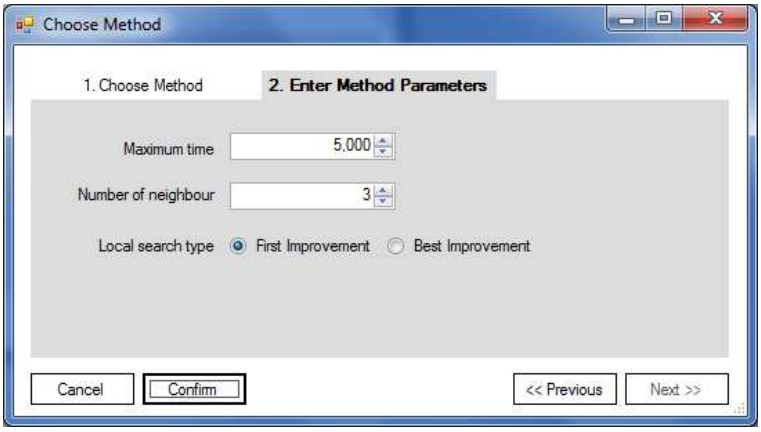
2.1. Izbor metode za rešavanje problema

Za rešavanje prethodno definisanog problema implementiran je veći broj metoda. Ukoliko izabrana metoda zahteva kreiranje početnog rešenja moguće je izabrati *Random* ili *Greedy Add* metodu, u suprotnom postavljena je vrednost *None*.



Slika 44. Izbor metode za rešavanje problema – prvi korak

Izborom opcije *Next* prelazi se na definisanje vrednosti parametara izabrane metode. Za svaku metodu definisan je poseban skup parametara. U nastavku su predstavljeni parametri za metodu *VNS*.



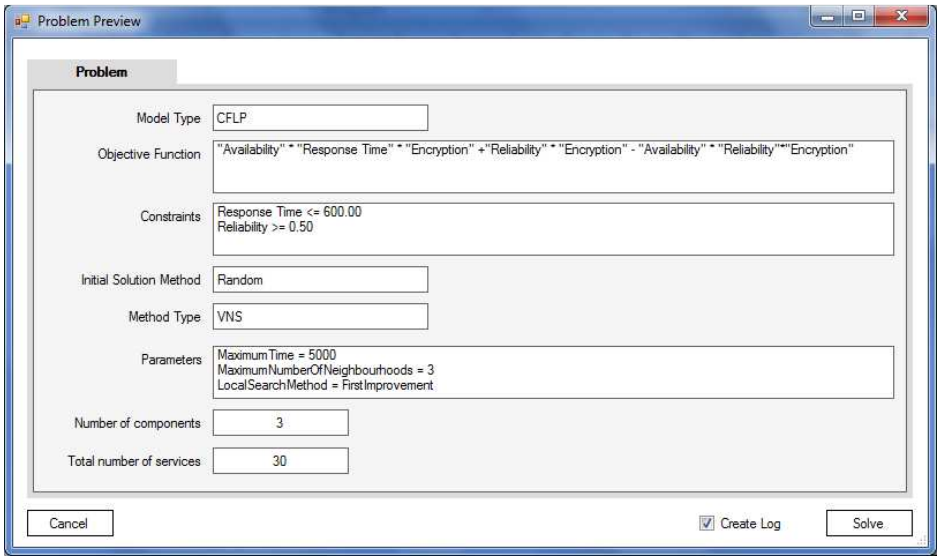
Slika 45. Izbor metode za rešavanje problema – drugi korak

Nakon što su zadate vrednosti svih zahtevanih parametara bira se opcija *Confirm* kojom se potvrđuje izabrana metoda za rešavanje problema.

2.2. Izvršavanje izabrane metode

Pre nego što se pokrene odgovarajući algoritam za rešavanje definisanog problema prikazuju se svi relevantni podaci o problemu koji se rešava (tip modela, funkcija cilja i ograničenja), zatim podaci o izabranom načinu njegovog rešavanja (tip metode, metoda za kreiranje početnog rešenja, parametri izabrane metode), kao i broj komponenti i ukupan broj servisa.

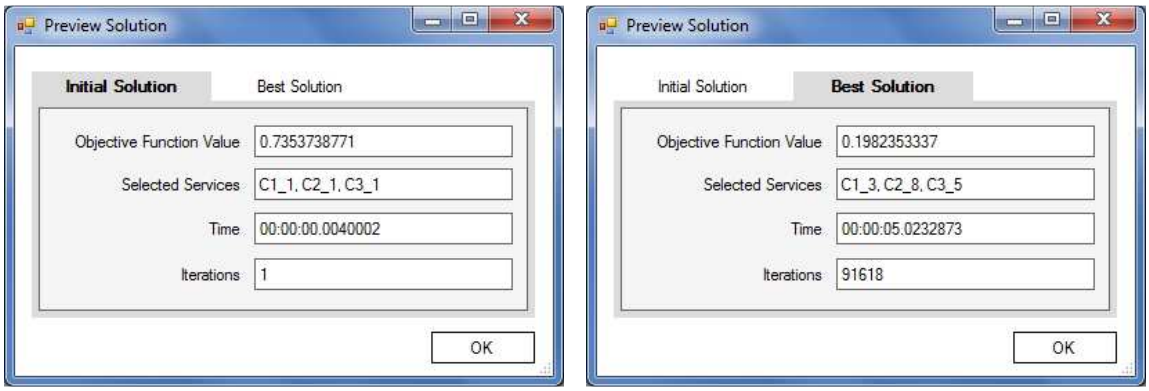
Rešavanje problema inicira se izborom opcije *Solve*. Ukoliko je potrebno voditi evidenciju o toku rešavanja problema potrebno je prethodno izabrati opciju *Create Log*.



Slika 46. Pregled definisanog problema i izabrane metode za njegovo rešavanje

2.3. Pregled rešenja

Poslednji korak predstavlja prikaz dobijenog rešenja. Prikazuje se dobijena vrednost funkcije cilja, zatim spisak izabranih servisa, kao i broj iteracija i vreme potrebno za rešavanje definisanog problema. Prikaz rešenja obuhvata i prikaz početnog rešenja (*initial solution*) ukoliko je izabranom metodom rešavanja problema definisano i kreiranje početnog rešenja.



Slika 47. Pregled rešenja

Biografija

Nina Turajlić je rođena 05. marta 1978. godine u Beogradu. Osnovnu školu pohađala je u Pacific Grove-u (Kalifornija, SAD) i Beogradu. Završila je gimnaziju „Sveti Sava“ u Beogradu. Nakon završene gimnazije, 1996. godine upisala je Fakultet organizacionih nauka, smer Informacioni sistemi. Školske 1999/2000 godine, za uspeh na studijama, nagrađena stipendijom Norveške vlade. Tokom 2000. godine boravila na praksi na Ecole Polytechnique Federale de Lausanne (Švajcarska) u Laboratoriji za arhitekturu procesora. Osnovne studije završila je 2006. godine. Diplomski rad pod nazivom „Realizacija rekurzivnog parsera XML šeme“, koji je razvijen za potrebe projekta ADMIS Laboratorije za Informacione sisteme Fakulteta organizacionih nauka, odbranila je sa ocenom 10. Školske 2006/2007. godine upisala je poslediplomske (doktorske) studije na Fakultetu organizacionih nauka, smer Operaciona istraživanja.

Radno iskustvo

- 2007-2009 godine zaposlena je na Fakultetu organizacionih nauka u zvanju saradnik u nastavi.
- Od 2009. godine zaposlena je Fakultetu organizacionih nauka u zvanju asistent.

Nastavne aktivnosti

Još tokom studija (2003-2007) radila je kao demonstrator na sledećim predmetima: Programski jezici i prevodioci, Projektovanje programa, i Principi programiranja. Od školske 2006/2007 godine, prvo kao saradnik u nastavi, a zatim kao asistent angažovana je predmetima: Programski jezici, Osnove informaciono komunikacionih tehnologija i Uvod u informacione sisteme. Školske 2007/2008 godine učestvovala je u izvođenju nastave za studente Vojne Akademije na predmetu Programski jezici.

Istraživačko iskustvo

2014- . *Predviđanje budućih stanja pacijenata: Razvoj i primena brzih, efektivnih i interpretabilnih algoritama za zdravstvo*, Zajednički projekat Švajcarske Nacionalne Naučne Fondacije SCOPES 2014-2016. Broj projekta: IZ73Z0_152415.

2008-2011. *Napredne metode za integraciju poslovnih procesa u složenim informacionim sistemima*, naučno-istraživački projekat u okviru Programa tehnološkog razvoja Ministarstva nauke Republike Srbije, Beograd.

2007-2008. *Strategija i dugoročni plan razvoja IS MUP Srbije*, Beograd.

Pregled projekata

Pregled idejnih i glavnih projekata i tehničkih rešenja (prototip, softver) u čijoj je izradi kandidatkinja učestvovala kao član ili vođa tima:

- Projektovanje i izrada informacionog sistema za potrebe realizacije projekta Ministarstva Zdravlja „*Kontrola tuberkuloze u Srbiji kroz sprovođenje strategije direktno opservirane terapije*“ Faza I i faza II, Beograd, 2004-2007.
- „*FilatelIS*“, Motiv d.o.o., Beograd, 2003-2005.
- „*BelVille*“,Blok 67 Associates d.o.o., Beograd, 2008-2010.
- „*TiCat*“, Telekom Srbija, Beograd, 2007-2009.

Pregled objavljenih radova

Nina Turajlić objavila je više naučnih radova u časopisima međunarodnog i nacionalnog značaja, kao i u zbornicima sa domaćih i međunarodnih konferencija.

Kategorija M23:

1. Dragović I., **Turajlić N.**, Radojević D., Petrović B.: Combining Boolean Consistent Fuzzy Logic and AHP Illustrated on the Web Service Selection Problem, *International Journal of Computational Intelligence Systems*, Vol. 7, Supplement 1, pp. 84-93, 2013 (**IF=0.451**) (ISSN 1875-6891 (Print), 1875-6883 (Online)).
2. Vučković M., Petrović M., **Turajlić N.**, Stanojević M.: The Specification of ETL Transformation Operations based on Weaving Models, *International Journal of Computers, Communication and Control*, Vol.7, No.5, pp. 968-975, 2012 (**IF=0.441**) (ISSN 1841-9836).
3. Stanojević M., Stanojević B., **Turajlić N.**: *Optimization of multiple-objective web service selection using fractional programming*. Annals of Data Science. (2014) (ISSN 2198-5804 (Print), 2198-5812 (Online)).– accepted for publication.

Kategorija M33:

1. **Turajlić N.**, Dragović I.: A Hybrid Metaheuristic Based on Variable Neighborhood Search and Tabu Search for the Web Service Selection Problem, *Electronic Notes in Discrete Mathematics*, Vol. 39, pp. 145-152, 2012 (ISSN 1571-0653) (DOI:10.1016/j.endm.2012.10.020).
2. **Turajlić N.**, Nešković S.: Variable Neighborhood Search and Tabu Search for the Web Service Selection Problem, *Electronic Notes in Discrete Mathematics*, Vol. 39, pp. 177–184, 2012 (ISSN 1571-0653) (DOI: 10.1016/j.endm.2012.10.024).
3. Dragović I., **Turajlić N.**, Radojević D.: Extending AHP with Boolean Consistent Fuzzy Logic and Its Application in Web Service Selection, *zbornik radova X međunarodne FLINS konferencije - FLINS 2012*, (Istanbul, Turska), pp. 576-591, 2012 (ISBN 978-981-4417-73-0).

4. Stanojević M., Stanojević B., **Turajlić N.**: Solving the Web Service Selection Problem Using Multi-Objective Linear Fractional Programming, *zbornik radova XI Balkanske konferencije o operacionim istraživanjima - BALCOR 2013*, (Beograd i Zlatibor, Srbija), pp. 617-622, 2013 (ISBN-978-86-7680-285-2).
5. **Turajlić N.**, Petrović M., Vučković M., Dragović I.: Groundwork for Presentation Pattern Metamodels, *zbornik radova XII međunarodnog naučno-stručnog Simpozijuma INFOTEH-JAHORINA - INFOTEH-JAHORINA 2013*, (Jahorina, Bosna i Hercegovina), Vol. 12, Ref. RSS-3-11, pp. 731-736, 2013 (CD Izdanje: ISBN 978-99955-763-1-8).
6. **Turajlić N.**, Petrović M., Vučković M.: Analysis of ETL Process Development Approaches: Some Open Issues, *zbornik radova XIV međunarodne konferencije - SymOrg'14*, (Zlatibor, Srbija), pp. 45-51, 2014 (ISBN 978-86-7680-295-1).

Kategorija M52:

1. Petrović M., **Turajlić N.**, Dragović I.: Pregled i uporedna analiza prezentacionih paterna, *Journal of Information technology and multimedia systems Info M*, Vol. 9, No. 34, pp. 35-41, 2010 (ISSN 1451-4397).

Kategorija M63:

1. Dragović I., **Turajlić N.**, Radojević D., Petrović B.: Korišćenje logičke agregacije za selekciju web sevisa, *zbornik radova XXXIX Simpozijuma o operacionim istraživanjima - SYM-OP-IS 2012*, (Tara, Srbija), pp. 377-380, 2012 (ISBN-978-86-7488-086-9).
2. Dragović I., Jednak S., **Turajlić N.**: Predviđanje stope ekonomskog rasta korišćenjem neuronskih mreža i ANFIS-a, *zbornik radova XXXVIII Simpozijuma o operacionim istraživanjima - SYM-OP-IS 2011*, (Zlatibor, Srbija), pp. 407-410, 2011 (ISBN-978-86-403-1168-7).
3. **Turajlić N.**, Nešković S., Vučković M.: Mesto mera performansi u modelima poslovnih procesa, *zbornik radova VIII naučno-stručnog Simpozijuma INFOTEH-JAHORINA - INFOTEH-JAHORINA 2009*, (Jahorina, Bosna i Hercegovina), Vol. 8, Ref. E-III-18, pp. 598-602, 2009 (CD Izdanje: ISBN-99938-624-2-8).
4. **Turajlić N.**, Vučković M.: Realizacija rekurzivnog parsera XML šeme, *zbornik radova XI internacionalnog Simpozijuma iz projektnog menadžmenta - YUPMA 2007*, (Zlatibor, Srbija), pp. 205-209, 2007 (ISBN-978-86-86385-02-04).
5. Nešković S., Vučković M., **Turajlić N.**: Transformacija XML šeme u relacioni model zasnovana na OMG MDA pristupu i apstraktnom modelu, *zbornik radova VI naučno-stručnog Simpozijuma INFOTEH-JAHORINA - INFOTEH-JAHORINA 2007*, (Jahorina, Bosna i Hercegovina), Vol. 6, Ref. E-II-3, pp. 343-347, 2007 (CD Izdanje: ISBN-99938-624-2-8).

Прилог 1.

Изјава о ауторству

Потписани-а Нина Турајлић

број индекса 15/2006

Изјављујем

да је докторска дисертација под насловом

Нови модели и методе за селекцију и композицију веб сервиса на основу нефункционалних карактеристика

- 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, 26.08.2014

Нина Турајлић

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Нина Турајлић

Број индекса 15/2006

Студијски програм Операциона истраживања

Наслов рада Нови модели и методе за селекцију и композицију веб сервиса на основу нефункционалних карактеристика

Ментор Проф. др Ненад Младеновић

Потписани/а Нина Турајлић

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, 26.08.2014

Нина Турајлић

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Нови модели и методе за селекцију и композицију веб сервиса на основу нефункционалних карактеристика

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, 26.08.2014

Јелена Шћепановић