УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Nikola Dalčeković

Platforma za transformaciju softverskih rešenja pametnih elektroenergetskih mreža na *cloud* bazirani višeorganizacijski *SaaS*

DOKTORSKA DISERTACIJA

Novi Sad, 2019

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:	
Идентификациони број, **ИБР**:	
Тип документације, **ТД**:	Монографска публикација
Тип записа, **ТЗ**:	Текстуални штампани документ/ЦД
Врста рада, **ВР**:	Докторска дисертација
Аутор, **АУ**:	Никола Далчековић
Ментор, **МН**:	др Срђан Вукмировић, ванредни професор
Наслов рада, **НР**:	Платформа за трансформацију софтверских решења паметних електроенергетских мрежа на *cloud* базирани вишеорганизацијски *SaaS*
Језик публикације, **ЈП**:	Српски (латиница)
Језик извода, **ЈИ**:	Српски/Енглески
Земља публиковања, **ЗП**:	Србија
Уже географско подручје, **УГП**:	Војводина
Година, **ГО**:	2019
Издавач, **ИЗ**:	Ауторски репринт
Место и адреса, **МА**:	Факултет Техничких Наука (ФТН), Д. Обрадовића 6, 21000 Нови Сад
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)	8/155/136/13/53/0/1
Научна област, **НО**:	Електротехничко и рачунарско инжењерство
Научна дисциплина, **НД**:	Примењени софтверски инжењеринг
Предметна одредница/Кључне речи, **ПО**:	Вишеорганизацијско својство, вишеорганизацијски шаблони, вишеорганизацијске архитектуре, СОА, модел учесника, НДМС
УДК	
Чува се, **ЧУ**:	Библиотека ФТН, Д. Обрадовића 6, 21000 Нови Сад
Важна напомена, **ВН**:	

Извод, **ИЗ**:		Све чешћом употребом *cloud* окружења долази до потребе да се постојећа софтверска решења мигрирају. Методологија за миграцију на *cloud* постоји више, где се у финалним фазама планирају модификације над архитектуром софтвера тако да се искористе предности *cloud* система. За економску ефикасност услед уштеде ресурса је неопходна вишеорганизацијска особина. Сврха овог истраживања је да појасни вишеорганизацијско својство и да предложи решење за миграцију постојећих софтвера на вишеорганизацијски *SaaS* али са што мање неопходних модификација циљног софтвера. С тога је предложено решење платформа која омогућује лакшу миграцију. Након фазе истраживања и сагледавања домена паметних електроенергетских мрежа, креиран је прототип предложеног решења као и низ експеримената у складу са дефинисаним научним питањима. Експерименти су извршени у приватном *cloud* окружењу. Хипотезе су адресиране кроз визију примене решења на НДМС (Напредни дистрибутивни менаџмент систем) у случају шест организација, а донети су следећи закључци: вишеорганизацијским моделом се остварују уштеде у ресурсима од 32%, за три реда величине већа висока доступност, али уз успорења до 20 милисекунди по сваком сервисном захтеву. Такође, апликативни модел учесника модерних *PaaS* услуга није примерен где је синхронизам захтеван, нити у случајевима где се очекују одговори над скупом учесника у реалном времену. Истраживање указује на могућност примене вишеорганизацијског модела чак и у случају комплексних решења каква се срећу у домену паметних електроенергетских мрежа, а академском валидацијом потврђују начин имплементације важног финалног корака у процесу миграције софтвера на *cloud* базирани *SaaS*.	
Датум прихватања теме, **ДП**:		05.10.2018.	
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	др Владимир Стрезоски, редовни професор	
	Члан:	др Јелица Протић, редовни професор	Потпис ментора
	Члан:	др Александар Ердељан, редовни професор	
	Члан:	др Дарко Чапко, ванредни професор	
	Члан, ментор:	др Срђан Вукмировић, ванредни професор	

KEY WORDS DOCUMENTATION

Accession number, **ANO**:	
Identification number, **INO**:	
Document type, **DT**:	Monographic publication
Type of record, **TR**:	Textual material, printed/CD
Contents code, **CC**:	PhD thesis
Author, **AU**:	Nikola Dalčeković, MSc
Mentor, **MN**:	Srdjan Vukmirović, PhD, Assistant Professor
Title, **TI**:	A Platform for Smart Grid Software Solution Migration to Cloud Based SaaS
Language of text, **LT**:	Serbian (latin script)
Language of abstract, **LA**:	Serbian/English
Country of publication, **CP**:	Serbia
Locality of publication, **LP**:	Vojvodina
Publication year, **PY**:	2019
Publisher, **PB**:	Author reprint
Publication place, **PP**:	Faculty of Technical Sciences, D. Obradovića 6, 21000 Novi Sad
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)	8/155/136/13/53/0/1
Scientific field, **SF**:	Electrical and computer engineering
Scientific discipline, **SD**:	Applied software engineering
Subject/Key words, **S/KW**:	multi-tenancy, multi-tenant patterns, multi-tenant design, SOA, actor model, smart grid, ADMS, demand response
UC	
Holding data, **HD**:	Library of the Faculty of Technical Sciences, D. Obradovića 6, 21000 Novi Sad
Note, **N**:	

Abstract, **AB**:

Progressive cloud adoption requires migration of existing software solutions. Today, many cloud adoption methodologies exist. Usually, the final phase in cloud adoption include software architecture modifications to make the most of the benefits of cloud computing, like multi-tenancy which enables economic efficiency. The aim of this research is to explain the multi-tenancy and to provide a solution for migration of existing software to multi-tenant SaaS while modifying the target software as little as possible. Therefore, the research proposes a platform that enables easier cloud adoption. After the research phase focused on a smart grid domain, the prototype was created with experiments targeting formulated research questions. The experiments were conducted in a private cloud environment. Research hypotheses were analyzed using hypothetical multi-tenant ADMS (Advanced Distribution Management System) in case of six tenants, with the following conclusions: multi-tenancy saves 32% of resources, it provides three orders of magnitude higher availability, but affects performances by introducing a delay of up to 20 milliseconds per service request. Also, reliable actors programing model used in modern PaaS services is not suitable in use cases with needs for synchronous behavior, nor in in use cases where querying a set of actors is needed in real time. The research demonstrates feasibility of applying multi-tenancy even in cases of complex software solutions like the ones in the smart grid domain. The proposed solution is academically validated and it can be used as a final important step in migration of existing software to cloud based multi-tenant SaaS.

Accepted by the Scientific Board on, **ASB**:		5th of October, 2018.	
Defended on, **DE**:			
Defended Board, **DB**:	President:	Vladimir Strezoski, PhD, Full Professor	
	Member:	Jelica Protić, PhD, Full Professor	Menthor's sign
	Member:	Aleksandar Erdeljan, PhD, Full Professor	
	Member:	Darko Čapko, PhD, Associate Professor	
	Member, Mentor:	Srdjan Vukmirović, PhD, Associate Professor	

Zahvaljujem se na prvom mestu svojim roditeljima, Vidi i Peri, svom bratu Branku na usmerenju i podršci koju su nesebično i pažljivo uvek pružali. Duboko verujem da je porodica glavna gradivna ćelija društva, odakle potiče vaspitanje, obrazovanje, kao i svi ostali proizvodi treniranja uma. Na kraju krajeva, kroz sam um formiramo i same nas.

Zahvaljujem se supruzi Ivani, koja je osim konstantne podrške i potpunog razumevanja za moje vreme posvećeno nauci, direktno učestvovala u naučnim projektima zajedno sa mnom. Njena pomoć se proširila iz privatnog života u naučni.

Posebno se zahvaljujem mom mentoru prof. dr Srđanu Vukmiroviću, na svim savetima i smernicama za izradu ove doktorske disertacije kao i na ukazanom poverenju.

Zahvaljujem se svim svojim kolegama s kojima sam učestovovao u naučnim projektima i s kojima sam zajedno objavljivao naučne radove jer je iz tih timskih poduhvata uvek proizilazila inspiracija kao i međusobno usavršavanje.

Zahvaljujem se svim prijateljima na razumevanju i podršci. Takođe, svim kolegama sa kojima sam intenzivno sarađivao i s kojima sam na trnovitom putu stekao dodatne izazove, pomoć ali i nove prijatelje.

Zahvaljujem se svojim trenerima košarke, koji su u mom odrastanju utkali najvažnije životne lekcije timskog duha, discipline, vere u snove i istrajnosti.

Zahvaljujem se svim mojim studentima koji su mi tokom više godina svojim trudom, angažovanjem, kao i povratnim informacijama pomogli da budem bolji u ulozi predavača, kao i da se više angažujem u praćenju najnovijih naučnih dostignuća.

Kada postoji nepresušna energija podrške čije trajanje se ne ograničava u sadašnjosti,
najmanje štetno je prepustiti tišini da izrazi zahvalnost,
praznoj stranici,

Vidi i Peri

Sadržaj

Spisak skraćenica

Skraćenica	***Značenje skraćenice* (*kurzivom obeleženo značenje na engleskom jeziku*)**
AD	*Active Directory*
AMI	*Advanced Metering Infrastructure*
AOP	Aspektno orijentisano programiranje (*Aspect Oriented Programming*)
API	*Application Programming Interface*
ASP	*Active Server Pages*
CRM	Upravlj. odnosima sa krajnjim potrošačima (*Customer Relationship Management*)
CIS	*Customer Information System*
DDD	*Domain Driven Development*
DERMS	*Distributed Energy Resources Management System*
DNS	*Domain Name System*
DR	Upravljanje opterećenjem (*Demand Response*)
EAM	Sistem za vođenje inventara (*Enterprise Asset Management*)
ESB	Servisna magistrala (*Enterpise Service Bus*)
GIS (GIS)	Geoinformacioni sistem (*Geographic Information System*)
FB	Funkcionalni blok
HaaS	*Hardware as a Service*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IED	*Intelligent Electronic Device*
IkS (IaaS)	Infrastuktura kao Servis (*Infrastructure as a Service*)
IoT (IoT)	Internet stvari (*Internet of Things*)
IP	*Internet Protocol*
IT	Informacione tehnologije (*Information technology*)
ITIL	*Information Technology Infrastructure Library*
JSON	*JavaScript Object Notation*
LINQ	*Language Integrated Query*
MASF	*Microsoft Azure Service Fabric*
MDM	Sistem za upravljanje podacima brojila (*Meter Data Management*)
MVC	*Model View Controller*

NDMS(*ADMS*) Napredni distributivni menadžment sistem (*Advanced Distribution Mngmt. Sys.*)

NIST *National Institute of Standards and Technology*

OMS Sistem za upravljanje ispadima (*Outage Management System*)

OT Operativne tehnologije (*Operational Technologies*)

PkS (*PaaS*) Platforma kao Servis (*Platform as a Service*)

RBAC *Role Based Access Control*

RDF *Resource Description Framework*

REST *Representational State Transfer*

ROA *Resource Oriented Architecture*

RTU *Remote Terminal Unit*

SCADA *Supervisory Control and Data Acquisition*

SkS (SaaS) Softver kao Servis (*Software as a Service*)

SLA *Service Level Agreement*

SOA *Service Oriented Architecture*

SOAP *Simple Object Access Protocol*

SUBP (*DBMS*) Sistem za Upravljanje Bazama Podataka (*Database Management System*)

STS *Security Token Service*

SQL *Structured Query Language*

UML *Unified Modeling Language*

URI *Uniform Resource Identifier*

URL *Uniform Resource Locator*

VDS *Virtual Dedicated Server*

VE Virtuelne Elektrane

VM Virtualna mašina (*Virtual Machine*)

WCF *Windows Communication Foundation*

WFM *Workforce management*

XML *Extensible Markup Language*

Spisak slika

Spisak tabela

Terminologija

Usled vrlo aktivne globalne naučne zajednice, neki pojmovi su šire rasprostranjeni i u lokalnoj naučnoj zajednici ukoliko se koriste izvorni, engleski pojmovi. Iz tog razloga, u ovom poglavlju će biti opisani veoma poznati termini, uz pojašnjenje odluke za koje termine će se koristiti adekvatni prevodi, a koji će ostati u originalnom izdanju.

Multi-tenant pojam se odnosi na osobinu softvera da je u mogućnosti da istu instancu podeli na više različitih korisnika ali na taj način da korisnici nisu svesni da dele softver sa još nekim. U bukvalnom prevodu, pojam bi bio višestanarski, zbog analogije gde više domaćinstava deli zgradu u modernim gradovima. Kako se pojam odnosi na iznajmiljavanje resursa različitim klijentima, tj. firmama ili organizacijama, u ovom radu će se koristiti termin ***višeorganizacijski***. Ovde važi napomena da je organizacija sveobuhvatan pojam koji se može odnositi na jednog individualnog korisnika, ili na više njih koji pripadaju jednoj organizaciji.

Cloud. Koncept *cloud computing* se nekada prevodi kao računarstvo u oblaku. S obzirom da je termin *cloud* nastao usled svoje osobine da su resursi dostupni putem interneta, a na graficima je internet često bio predstavljan kao oblačić, sam termin računarstva u oblaku ne oslikava njegovu namenu, nego više bukvalni prevod sa engleskog. S obzirom da se u široj zajednici nije ustalio kao termin, odluka je da se u ovoj disertaciji koristi originalni pojam koji se koristi i u srpskom govornom jeziku kao takav, tj. klaud. U pisanju ovog rada će se koristiti izvorni oblik – ***cloud***.

Postoji više pojmova koji su dobili bukvalni prevod. *Big data* se odnosi na koncept modernih sistema za skladištenje podataka za koji postoji jasna definicija, a koristiće se pojam *veliki podaci* iz razloga što nije toliko često korišćen pojam u ovom radu, a iz želje da se smanji broj engleskih termina. Isto važi i za *Internet of Things – IoT,* koji će umesto možda prirodnijeg prevoda *interneta uređaja*, koristiti bukvalni prevod kao ***internet stvari***. Pod stvarima se podrazumevaju raznoliki uređaji koji su povezani putem interneta, ali zbog raznolike prirode samih uređaja, u engleskom jeziku su nazvani stvari.

Smart grid je kratak pojam i dobro definisan, ali je ipak korišćen prevod ***pametnih elektroenergetskih mreža***. Pametne mreže bi možda uvele konfuziju jer se ne bi podrazumevao domen elektroenergetike.

Sve skraćenice koje se koriste na više različitih mesta su uvedene u poglavlju „Spisak skraćenica" i korišćene su na srpskom jeziku, dok skraćenice koje nisu u težištu teze i pojavljuju se na nivou par pasusa u okviru iste strane nisu uvođene kao skraćenice nego su ostajale u svom izvornom obliku. Takođe, jako poznate skraćenice kao što je *IoT* za internet stvari su zadržane u svom izvornom obliku osim u slučaju pojmova koji su u fokusu ove disertacije, npr.: *SaaS* je SkS, *PaaS* je PkS, *IaaS* je IkS.

1. Uvod

Usavršavanjem informacionih tehnologija i povećavanjem pouzdanosti softverskih sistema, došlo je do toga da postojeći sistemi migriraju na softverski zasnovane sisteme. U savremenom svetu, čak i kritična infrastruktura kao što je energetski sistem se oslanja na softver.

Podaci koji se čuvaju u okviru računarskih sistema često postaju preobimni i praktično ih nije moguće postaviti na jednu fizičku mašinu. Stoga su distribuirani sistemi dobili još više na značaju. Evidentnim napretkom interneta po pitanju brzina i pouzdanosti, kao najnoviji trend se javljaju *cloud* sistemi o čijoj terminologiji ima detaljnije u Dodatku A.

Sve više se poslovanje zasniva na softveru, gde se javljaju potrebe za optimizacijom potrošnje. Do izražaja dolazi korišćenje infrastrukture koja je skalabilna i elastična, i time *cloud* dobija na značaju. Korišćenjem *cloud* infrasturktura, kreira se softver koji se uglavnom koristi putem udaljenog pristupa, *web* klijentima, što je dovelo do transformacije poslovnih modela na softver kao servis (SkS) (engl. *Software as a Service – SaaS*). Detaljno o aktuelnom stanju u oblasti ima više u poglavlju 2.

U SkS poslovnom modelu, moguće je ostvariti direktni profit uštedom resursa, što dovodi do favorizacije višeorganizacijskog modela softvera koji je u stanju da jednom instancom uslužuje više različitih organizacija. *Cloud* okruženja, poslovni modeli i višeorganizacijsko svojstvo su opisani u poglavlju 3. Ciljevi višeorganizacijskog softvera su ušteda računarskih resursa i omogućavanje veće dostupnosti za proizvoljan broj organizacija uz deljenje resursa.

Ovaj rad nastoji da objasni filozofiju višeorganizacijskog svojstva, da ispita moguće načine implementacije višeorganizacijskog pristupa kao i da predloži rešenje koje bi potpomoglo transformaciju postojećih softverskih rešenja na višeorganizacijska. Fokus primene rešenja će biti na vrlo kompleksnim softverskim sistemima kakvi se mogu naći u industriji elektroenergetike, gde se zahteva odgovor u realnom vremenu pri unosu velike količine podataka zbog prirode struje i razvoja pametnih elektroenergetskih mreža. Opis prirode ovakvih sistema se nalazi u poglavlju 4.

Poglavlje 5 je posvećeno predlogu rešenja ovog rada koje je rezultat istraživanja predstavljenog u ranijim poglavljima, i dizajniranog da pruži odgovor na definisane hipoteze tako da se adresiraju problemi migracije postojećih softverskih rešenja na *cloud* bazirani višeorganizacijski SkS.

Predloženo rešenje predstavlja platformu koja olakšava migraciju postojećih rešenja na višeorganizacijska. Predložena platforma je ishod iterativnog istraživanja gde su izvršena četiri različita eksperimenta na osnovu kojih su doneti zaključci za unapređenja i primenljivost ovog rešenja. Rezultati, analiza i interpretacija eksperimenata su predstavljeni u poglavlju 6, a diskusija koja dovodi do prikaza benefita i vizije rešenja na studiji NDMS sistema se nalazi u poglavlju 7.

Na kraju, iznet je zaključak ovog istraživanja u poglavlju 8, gde se može uvideti da je uvođenje višeorganizacijskog svojstva vrlo zahtevno jer se može sprovesti samo nad postojećim softverom sa određenim karakteristikama koje su predmet višegodišnjeg unapređivanja softvera. S druge strane, za moderne softvere bazirane na *cloud* okruženju, višeorganizacijsko svojstvo je neizostavno usled sve oštrijeg globalnog tržišta softvera, a i poslovanja generalno, gde i male razlike u ceni mogu selektovati kompaniju koja će opstati na tržištu.

1.1. Predmet i potreba istraživanja

Predmet istraživanja doktorske disertacije je mogućnost tehnološke transformacije sa tradicionalne isporuke softverskih rešenja na softver kao servis (SkS) (engl. *SaaS*). Potreba istraživanja se javlja iz višestrukih razloga, koji mogu biti interni kompanijski, interesi krajnjih korisnika, a mogu biti i izazvani poslovnim prilikama. Iz tog razloga je neophodno najpre definisati šta je uopšte SkS u domenu softvera sa nacionalno kritičnim infrastrukturama.

Primer softvera koji će biti razmatran je sastavni deo nacionalno kritičnih infrastruktura – NDMS (Napredni distributivni menadžment sistem) (engl. *ADMS*). NDMS je softver za upravljanje i nadgledanje distribucije električne energije u pametnim elektroenergetskim mrežama, detaljno je opisan u 4.4.

Tokom istraživanja, moraju se sagledati aspekti poslovne transformacije jer utiču na odluke u tehničkoj transformaciji, ali je predmet istraživanja transformacija na tehnološkom nivou. U skladu sa prethodno navedenim, poslovni aspekti koji direktno nameću potrebu transformacije su sledeći:

1. stvaranje novog potrošačkog segmenta,
2. odbrana pozicije od konkurencije,
3. novi tokovi zarade,
4. dodatna ponuda (mogućnost ponude funkcionalnosti kakvu je nemoguće ekonomski prihvatljivo pružiti sa tradicionalnim rešenjem).

Interni kompanijski razlozi koji nameću transformaciju na SkS:

1. kontrola razvoja proizvoda uz konstantnu povratnu informaciju (analiza podataka),
2. potencijalna monetizacija klijentskih podataka (ekstrakcija znanja),
3. postizanje efikasnosti rešenja (postoji interes da se smanji potrošnja resursa),
4. adaptacija na potrebe tržišta (poslovna i tehnološka),
5. proširenje ekosistema – mogućnost interakcije sa više poslovnih partnera,
6. standardizacija ponude,
7. zaštita intelektualne svojine.

Razlozi krajnjih potrošača koji nameću transformaciju na SkS:

1. eliminacija rizika implementacije,
2. eliminacija kompleksnosti korišćenja softverskih alata što dovodi do otežanog korišćenja,
3. smanjenje ukupnih troškova,
4. jasan uvid u korišćenje servisa,
5. fleksibilnost troškova (pretvaranje kapitalnih investicija u operacione troškove),
6. isprobavanje servisa je jeftinije i ne zahteva kapitalne investicije.

Kada se uzme u obzir potreba istraživanja i moderni trend pojave sve više rešenja u vidu SkS, prirodno se pojavljuje predmet istraživanja transformacije na SkS poslovni model. Svaka kompanija se nalazi u različitom trenutku zrelosti kada se odluči na korak transformacije i polako se pojavljuju metodologije koje su u stanju da odrede početno stanje u kom se kompanija i njen proizvod nalaze na osnovu čega predlažu niz koraka neophodnih za uspešnu transformaciju na SkS poslovni model.

Sve metodologije, i akademske, i industrijski relevantne, predlažu prelazak na višeorganizacijsku arhitekturu koja je predmet istraživanja ove disertacije sa osnovnom pretpostavkom da je svaki softverski proizvod moguće transformisati u SkS, samo je pitanje da li je to opravdano i isplativo. Jedna od najvažnijih potreba jeste motiv krajnjih potrošača da smanje ukupne troškove, što dovodi do redizajna softverskog rešenja na višeorganizacijske arhitekture koje će iz tog razloga biti predmet istraživanja iz tehnološkog aspekta.

Osim smanjenja krajnjih troškova, višeorganizacijske arhitekture dovode i do veće dostupnosti softvera generalno, jer ukupno postoji više resursa za usluživanje softverskih zahteva, što dovodi do veće robustnosti. U užem smislu, predmet i fokus ove disertacije predstavlja istraživanje osobine višeorganizacijske arhitekture, obuhvatajući analizu načina podele aplikativnog sloja sa poslovnom logikom, prezentacionog sloja kao i sloja za perzistiranje podataka na više različitih klijentskih domena. Potrebe istraživanja se mogu klasifikovati na sledeći način:

1. predlog i kreiranje višeorganizacijske platforme čiji bi cilj bio da postojeća rešenja iskoristi ali i prilagodi korišćenju od strane više korisnika,
2. ostvarivanje uštede, tj. ostvarivanje finansijske prednosti za SkS,
3. sagledavanje izazova bezbednosnog modela pri izolaciji različitih organizacija,
4. nepostojanje uniformnog pristupa za višeorganizacijsku platformu koja će omogućiti deljenje postojećih tehnologija za pametne elektroenergetske mreže,
5. ispitivanje usporenja krajnjeg rešenja koje bi koristilo prednosti višeorganizacijske platforme.

1.2. Cilj istraživanja

Potrebno je sistematično analizirati i sagledati segmente transformacije postojećih tradicionalnih softverskih rešenja na SkS model poslovanja. Nakon faze istraživanja, potrebno je kreirati platformu za tehnološku transformaciju softvera na višeorganizacijsku arhitekturu.

Platforma može da se oslanja na cloud infrastrukturu iz razloga što su višeorganizacijske arhitekture prirodne u *cloud* baziranim SkS. U tom smislu, glavni cilj je formiranje softverske platforme koja vrši podelu postojećih softverskih tehnologija na način da se mogu koristiti istovremeno od strane više različitih organizacija, ali tako da svaka od organizacija ima utisak da je SkS namenjen samo toj organizaciji. Na ciljnoj platformi, očekivani rezultati su sledeći:

1. rešenje može da pruži iste funkcionalnosti ka više organizacija sa ukupno manje resursa – prikazati egzaktno koliko resursa bi moglo da se uštedi,
2. rešenje može da omogući veću visoku dostupnost (engl. *high availability*),
3. rešenje uvodi slabljenje nefunkcionalnih karakteristika. Fokusirati se na pokušaj minimizacije penala, i egzaktno predstaviti koliki je penal korišćenja platforme.

Dobijena platforma može biti korišćena u sve prisutnijem razvoju SkS tržišta u svetu, obično izazvana sve oštrijom konkurencijom među ponuđačima softverskih rešenja. Za svakog postojećeg proizvođača softvera koji planira da otvori SkS tržište, prvo prirodno pitanje koje će se postaviti na tehničkom nivou jeste da li transformisati postojeći ili krenuti razvoj od početka?

U slučaju transformacije, što je verovatno u slučaju višegodišnje razvijanih softvera, predloženo rešenje ponudiće platformu za efikasnu migraciju na višeorganizacijski model. Sa naučne strane, definicijom celokupnog problema vezanog za višeorganizacijski model, može se otvoriti niz novih pitanja za istraživanje. U ovoj disertaciji se takođe nalazi sintetizovana kolekcija istraživanja dosadašnjeg stanja u oblasti.

1.3. Metode istraživanja

Metode koje su korišćene u fazi istraživanja su sledeće:

1. Sistematični pregled literature
2. Eksperimenti
3. Model akademsko industrijskog transfera tehnologije

S obzirom na cilj istraživanja, a to je da se sprovede sistematična transformacija softvera na višeorganizacijski, jako je važno osloniti se na prethodna iskustva u transformacijama kako bi se izvukli validni zaključci.

Kako je jedna transformacija veoma dug projekat sa konstantnim usavršavanjem, jako je teško na sopstvenom iskustvu bazirati osnovicu za istraživanje. Iz tog razloga, za izgradnju osnove istraživanja, najzastupljenija metoda je sistematični pregled literature. Cilj sistematičnog pregleda literature je svakako da se identifikuje, analizira i interpretira sav postojeći dokaz za određeno naučno pitanje. Na taj način je postignuto da se i iskustva tokom analize drugih istraživanja uvaže, kao i da se istraživanje nadogradi na već postojećim pozitivnim i negativnim iskustvima. Uz sistematičan pregled literature, korišćene su i metodologije eksperimenata kao i model akademsko industrijskog transfera tehnologije.

Sistematičan pregled literature je sproveden u tri faze: planiranje, sprovođenje i izveštavanje. Kako se naučno pitanje ove teze nalazi u preseku više oblasti, faza planiranja mora obuhvatiti sledeće oblasti istraživanja:

1. Poslovni modeli; SkS je poslovni model, i kao takav se mora prvo jasno definisati i objasniti kako bi cilj bio jasno vidljiv i poznat
2. Metodologije transformacija na SkS poslovni model
3. *Cloud* sisteme: Računarstvo u oblaku predstavlja često jedinu validnu opciju za pružanje SkS i kao takvo se ne sme izostaviti iz istraživanja. Uža oblast bi bile arhitekture za uštedu troškova.
4. Pametne elektroenergetske mreže; Uža oblast softverski alati za nadgledanje i upravljanje.
5. Sistemi velikih količina podatka (engl. *Big data*) i internet stvari (engl. *IoT – Internet of Things*)

Oblast transformacije rešenja (2.) se može smatrati najvažnija jer je naučno pitanje koje ova teza treba da adresira upravo iz oblasti metodologije transformacije na SkS poslovni model. Dakle, potreba za sistematičnim istraživanjem u ovom radu je dvojaka: da se istraži trenutni doseg nauke po pitanju metodologija transformacija, ali i da se iskoriste postojeći empirijski dokazi iz iste oblasti studije. Iz procesa sistematičnog pregleda literature, nastao je pregled aktuelnog stanja u oblasti.

U toku faze planiranja, neophodno je za svaku od oblasti istraživanja pronaći odgovarajuće radove. Strategija pretrage radova se bazira na pretrazi odgovarajućeg skupa literature iz gore navedenih oblasti istraživanja. Na grupi izdvojenih radova iz pretrage, treba da se primene kriterijumi za selekciju radova, koji su definisani u fazi planiranja. Osnovni kriterijum odabira je da radovi imaju slično naučno pitanje ili isti predmet istraživanja. Osim naučnih radova, istraživanje se može osloniti i na analize tržišta objavljene od strane referentnih istraživačkih kuća. U ovim slučajevima, usled nedostatka celokupnog izveštaja, negde su korišćene internet adrese koje sadrže isečke iz celokupnih izveštaja. Način kvalitativnog određivanja radova se vrši putem unapred određene kriterijume, a koji se odnose na kategorizaciju radova, njihovu vreme objavljivanja, kao i citiranost.

U toku faze sprovođenja sistematičnog pregleda literature, prvo se vrši pretraga radova na osnovu ključnih reči, a potom se biraju radovi na osnovu iznetih kriterijuma definisanih u fazi planiranja. Nakon čitanja svih potencijalnih radova, vrši se selekcija na osnovu kriterijuma kvaliteta.

Nad selektovanim radovima, vrši se meta-analiza radova kako bi se odredili radovi koji su najprikladniji kao osnova za dalje istraživanje. Meta-analiza se vrši po dva kriterijuma: grafičko predstavljanje opsega istraživanja koje radovi pokrivaju, kao i grafičko predstavljanje eksperimenata koje su varijable testirane. Na osnovu grafičke reprezentacije, biraju se radovi koji obuhvataju najveći zajednički opseg istraživanja ili oni čiji eksperimentalni rezultati najviše odgovaraju predmetu istraživanja. Nad uže selektovanim radovima, mogu se primenjivati pretrage radova koje su selektovani radovi referencirali kao i radova koji referenciraju dati rad. Iz dodatnih radova, opet se selektuju oni sa unapred definisanim kriterijumom selekcije. Rezultat faze izveštavanja sistematičnog pregleda literature, nalazi se u odeljku 5.3.

Druga metodologija koja je korišćena jesu eksperimenti. Eksperimenti su tehnološki bazirani, što izbacuje probleme nedeterminisanosti kao što je u slučaju sa eksperimentima koji uključuju ljudsku intervenciju. Eksperimenti su izvršavani na dva načina:

1. Poređenje dvaju pristupa (tehnologije, ili metode dolaska do cilja)
2. Poređenje ishoda pristupa rešenja u odnosu na zahteve dobijene iz industrije

Svaki eksperiment ima identifikovane varijable od interesa, tako da je uzimanje uzorka rezultata određeno na taj način da se obradom rezultata matematičkim aparatima mogu dobiti pouzdane vrednosti očišćene od grešaka. Prednost eksperimenta je što se mogu istražiti situacije u kojima su tvrdnje tačne, i na taj način na osnovu eksperimenta se mogu predložiti okruženja i uslovi u kojima date tvrdnje važe. Eksperimenti u izradi ove teze su sprovedeni kroz uobičajene faze:

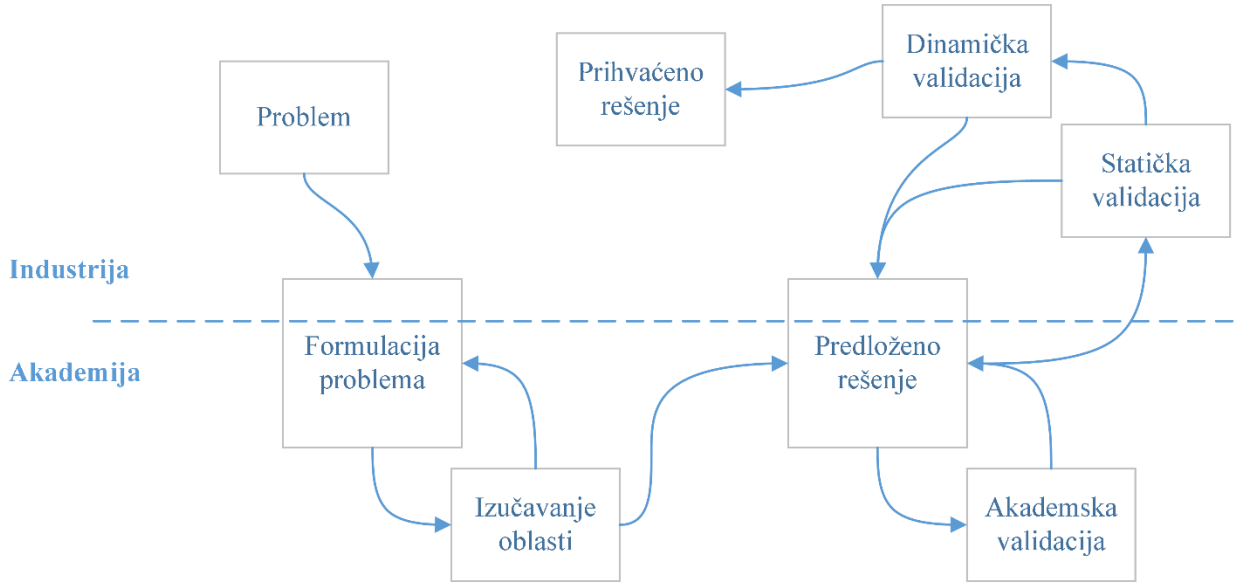
1. određivanje opsega eksperimenta
2. planiranje
3. izvršavanje
4. analiza i interpretacija
5. prezentacija rezultata

Sam opseg eksperimenta je određen u odnosu na hipotezu koju eksperiment treba da opovrgne. Planiranje je vršeno u odnosu na raspoložive resurse za istraživanje, kao i za određeni

minimum da bi se uzorci dobijeni merenjem mogli smatrati validnim. Određeni minimum zavisi od prirode eksperimenta, ali se uopšteno može reći za eksperimente izvršene na deljenoj infrastrukturi kao što je *cloud*, da je minimum izvršavanja eksperimenta dvadeset i četiri časa u kontinuitetu.

Izvršavanje eksperimenta je otpočeto po prethodno ustanovljenom planiranju uz cilj prikupljanja podataka definisanih u fazi planiranja, a određenih na osnovu opsega eksperimenta. Nakon isteka faze izvršavanja, prikupljena merenja predstavljaju ulaz za fazu analize. Analiza otpočinje vizuelizacijom prikupljenih podataka, grafički ili tabelarno.

Za obradu rezultata će se koristiti metode tabelarnog i grafičkog prikaza. Pre same vizuelizacije, eliminišu se šumovi u podacima i skup podataka se redukuje na validne uzorke. Nakon toga se podaci analiziraju tako da testiraju hipotezu eksperimenta, tačnije da li statistički rezultati zadovoljavaju pretpostavke hipoteze.



Slika 1 - Prilagođeni model transfera tehnologije [1], [2]

Softversko inženjerstvo je često primenjivano u praksi i kao takvo, očekivano je da se problemi koji se pojavljuju u industriji izučavaju u okviru akademije [1]. Ideja je da se akademija i industrija presecaju u smislu da se znanja stečena u jednom domenu mogu prenositi u oba smera – ova metodologija se naziva model transfera tehnologije. Spoj industrije i akademije pruža odličnu priliku da se znanja stečena u akademiji dokažu u industriji. Metod transfera tehnologije (Slika 1) je adaptiran u [1] a originalno je nastala u [2].

Problem u industriji u ovoj tezi je izazvan reakcijom na tržište i težnjom za inovacijama. SkS model poslovanja u elektroenergetskoj industriji još uvek nije ustaljen i postoje mnogi izazovi. Formulacija problema treba da se izvrši praćenjem industrijskih zahteva i problema vezanih za SkS i njenom paralelnom formulacijom u akademiji što će biti započeto ovom disertacijom.

Potom sledi izučavanje oblasti što je u slučaju ove teze pet navedenih oblasti na početku ovog odeljka. Na osnovu predloženog rešenja, akademska validacija je izvršena eksperimentima. Nakon analize rezultata eksperimenata, mogu se predložiti metodologije tehnološke transformacije na *cloud* bazirani višeorganizacijski SkS.

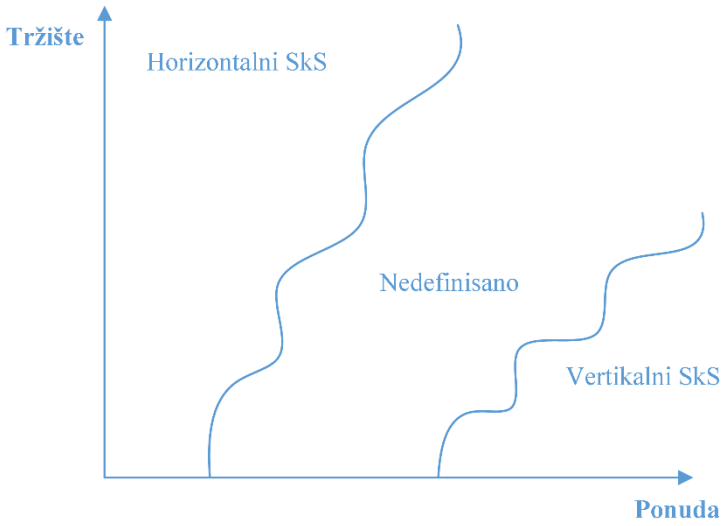
Izvršena je akademska validacija u slučaju NDMS softvera, a važno je zapaziti da se na isti način može izvesti i statička validacija u industriji. Na osnovu statičke validacije, predloženo rešenje se može korigovati, i potom da se pređe u dinamičku validaciju. U slučaju dinamičke validacije, obično se rešenje testira nad podskupom organizacije. Ovaj deo kod tehnološke transformacije bi obuhvatio podskup ciljnog softvera, a fazni pristup dinamičke validacije je opisan u poglavlju 7 koje se odnosi na akademsku validaciju.

Očekivano je da bi početni koraci doveli do novih praktičnih saznanja tako da je unapređenje predloženog rešenja očekivano. U svakom slučaju, statička validacija, dinamička validacija kao i industrijsko prihvaćeno rešenje su izvan opsega ove disertacije.

1.4. Mogućnost primene

U SkS rešenjima prirodno okruženje je *cloud* zbog fleksibilnosti koje pruža i finansijskog modela. Stoga su metodologije transformacije rešenja na c*loud* radi pružanja SkS usluga najintenzivnije istraživali i objavljivali upravo pružaoci *cloud* usluga. Da bi se razumela sveukupna primena očekivanih rezultata, neophodno je prikazati razlike između horizontalnih i vertikalnih SkS rešenja.

Razlike horizontalnih i vertikalnih rešenja se predstavljaju preko dve dimenizije (Slika 2) – y osa predstavlja širinu tržišta koju SkS rešenje može da zahvati dok x osa predstavlja raznolikost ponude SkS rešenja.



Slika 2 - Vertikalni i horizontalni SkS

Horizontalna SkS rešenja ciljaju da postanu najbolja za jednu usku funkcionalnost čineći svoju ponudu jako specijalizovanom. Šireći svoje tržište dostižu skalabilnost. Stoga je fokus ovakvih kompanija na proširenju tržišta što dovodi do pojačanih prodajnih i marketing strategija.

U suprotnosti, vertikalna SkS rešenja se trude da otvore tržište kod klijenata koji koriste tradicionalna dokazana rešenja i spori su adaptaciji inovacija. Zbog prirode tržišta, ovde se razviju proizvodi koji su najbolji u svojoj klasi i koji se uobičajeno prodaju širom jedne industrije. Ovo dovodi do manjeg tržišta i filozofije da nekolicina najboljih pobeđuje u establiranom tržištu.

Da bi softverski proizvodi u ovakvim uslovima opstali i postali vodeći na tržištu, neminovno je da imaju veći stepen prilagodljivosti klijentima i da razvoj funkcionalnosti bude vođen željama klijenata. Jasna granica ne postoji, tako je i na ilustraciji međuprostor predstavljen nedefinisanim jer se u praksi pokazalo da bi rešenje opstalo, mora se fokusirati na jedan od ekstrema.

Dok su postojeće metodologije transformacije rešenja na SkS model zasnovane na brojno dominantnijim horizontalnim SkS rešenjima, metodologija ove hipoteze se odnosi na vertikalna SkS rešenja. Kao takva, metodologija je pravljena da zadovolji potrebe industrija koje imaju učešće u domenu nacionalnih kritičnih infrastruktura, sa softverom namenjenim pametnim elektroenergetskim mrežama.

Vertikalna rešenja imaju striktnije zahteve od horizontalnih, tako da i sama metodologija transformacije mora da bude sveobuhvatnija u slučaju vertikalnih transformacija. Odatle sledi da i sama metodologija nije nužno limitirana na vertikalna SkS rešenja.

2. Pregled aktuelnog stanja u oblasti

Cloud okruženja su na globalnom nivou uticala na razvoj poslovanja, ali još uvek manje od polovine poslovanja koristi javne *cloud* platforme. U 2018. godini, očekuje se da će se *cloud* usvojiti u preko 50% kompanija [3]. Gartnerov ciklus promocije (engl. *Gartner Hype Cycle*) pruža grafičku analizu zrelosti i usvajanja promovisanih novih tehnologija za određeni vremenski period koji obično obuhvata par godina pre i do deset godina nakon datuma kada se izveštaj objavljuje. U avgustu 2009. godine, prema Gartneru, može se reći da je *cloud* bio na vrhuncu iščekivanih tehnologija [4]. U 2012. godini je objavljen poslednji Gartnerov ciklus što se tiče promocije *cloud* okruženja. Drugim rečima, može se smatrati da je osnovna *cloud* ponuda ušla u fazu pune produktivnosti oko 2012. godine – jasna vremenska granica naravno ne postoji. Od tada, beležen je globalni rast u pogledu zahteva za *cloud* tehnologijama što podiže važnost razvoju aplikacija koje koriste *cloud*.

Neka od ključnih Gartnerovih istraživanja tvrde da su *cloud* tehnologije u 2017. godini u svojoj punoj zrelosti i da će stoga organizacije agresivno investirati u *cloud* strategije i arhitekture [5]. Mnoge IT organizacije će i dalje gubiti interes u izgradnji i održavanju sopstvenih serverskih centara i privatnih *cloud* okruženja. Stoga je preporuka da se što pre krene u izradu strategije usvajanja *cloud* jer je transformacija teška i zahteva vreme. Procena je da će 2020. godine, 24% celokupnog IT marketa biti *cloud* [5].

Procenjuje se da će se godišnja stopa rasta konzumiranja *cloud* servisa u narednim godinama zadržati na oko 22% [3]. Iz razloga sve većeg prihvatanja *cloud* servisa, kompanije moraju biti spremne na pritisak tržišta i da se suoče sa rešenjima zasnovanim na *cloud* okruženjima i poslovnim modelima [4]. Danas, opšte prihvaćeni *cloud* modeli su infrastruktura kao servis (IkS) (engl. *IaaS*), platforma kao servis (PkS) (engl. *PaaS*) i Softver kao Servis (SkS). U literaturi [4], [6], [7] se razlika između različitih servisnih nivoa analizira u odgovornosti koju ima pružalac usluga i korisnik. Naravno, takve podele jesu tačne i danas se uzimaju kao univerzalna istina, ali to nisu definicije poslovnih modela, a svakako ne obuhvataju celokupnu sliku za dati poslovni model. Da bi se razumele tehnološke posledice, neophodno je razumeti jasne razlike između poslovnih modela i šta zapravo čini jedan SkS poslovni model.

Kako bi se definisao SkS poslovni model koji će se koristiti kao referentni, predstava poslovnog modela će se zasnivati na Ostervaldovom poslovnom platnu [8]. Tek nakon definisanja poslovnog modela, mogu se dobiti informacije od značaja za dalju analizu tehničke podrške novom načinu poslovanja. Viđenje da način poslovanja prednjači u odnosu na tehnološku podršku, tj. da se tehnološka izvedba može posmatrati u ulozi pomoći za ispunjene poslovnih ciljeva se pojavljuje u literaturi iz oblasti poslovnih modela.

U [9] su na osnovu citiranja sedam različitih izvora, autori istakli važnost aktivnog analiziranja i dizajniranja poslovnih modela. Rad se bavi analizom PkS poslovnog modela gde su autori predstavili metodologiju dijagramskog prikaza prednosti i nedostataka modela. U dobroj meri takav pristup može da bude od koristi pri izradi ove disertacije jer je tu suština za analizu poslovnog modela. Takođe, rad se bavi PkS poslovnim modelom, i to iz ugla kompanija koje bi eventualno mogle da usvoje PkS.

Postoji još radova koji dele viđenje poslovnog modela na prvom mestu i koji su istraživali na koji način se dati poslovni modeli mogu unaprediti i prilagoditi potrebama softverske industrije. U [10] su se autori bavili razumevanjem koncepta poslovnog modela iz ugla softverske industrije i došli do razvitka sopstvene metodologije za dizajn, opis i analizu poslovnog modela. Da bi to uradili, predstavili su analizu širokog spektra literature iz oblasti, i između ostalog izneli sledeće: poslovni model nije strategija sam po sebi, ali se sastoji od niza strateških elemenata [10]. Taj niz strateških elemenata praktično obuhvata transformaciju poslovnog modela u slučaju da već postoji kompanija sa razrađenim poslom. Autori takođe analiziraju Ostervalderovo poslovno platno i to iz razloga što je ono najcitiranije u akademiji.

Dodatno, u [11] autori kao deo svog istraživanja sprovode opsežnu analizu poslovnih modela i predstavljaju osam pristupa izdvojenih iz literature u tabelarnom obliku. Jedan od zaključaka rada je da je Ostervalderovo platno najobuhvatnije predstavlja opis poslovnog modela. Te činjenice su dodatno osnažila pristup da se koristi Ostervalderov pristup kao osnovica analize potencijalnog SkS modela.

Poslovni model i IT arhitekture su spregnute i ne treba da evoluiraju zasebno jer IT postaje najuticajniji segment većine poslovanja [11]. Iz ovog razloga, autori su predstavili istraživanje integracije arhitekture i poslovnog modela, ali bez jasne izjave koja komponenta prednjači evoluciji – poslovni aspekti ili tehnički. Tačnije, autori pokušavaju da daju ilustraciju jedinstvenog pogleda na dve stvari kao nerazdvojne, i kako promene u jednoj oblasti izazivaju promene u drugoj oblasti. Ovo možda jeste najpravilniji način gledanja na stvari, ali nije pristup koji će dovesti do jasnih koraka potrebnih za transformaciju. Uglavnom, ova teza deli istu premisu, a to je da su oblasti poslovnog modela i IT arhitekture nerazdvojne.

Ipak, u akademiji postoje sasvim validni pristupi koji istražuju suprotan pristup, a to je da IT arhitekture prednjače evoluciji poslovnih modela. Konkretno, IT arhitekture u modernom svetu možemo zameniti sa *cloud* okruženjima koja imaju veliki uticaj na razvoj kompanija širom sveta. Mnoga istraživanja polaze od činjenice da *cloud* podstiče evoluciju poslovnih modela [12]. Istraživanje se dalje bavi time kako *cloud* utiče na inovaciju poslovnih modela jasno predstavljajući razlike koje *cloud* donosi po sledećim oblastima: mogućnosti evolucije za kompaniju, otvaranje tržišta novim prilikama, inspiracije za promenu arhitekture i promene poslovnog modela. S obzirom na prirodu SkS poslovnog modela koji će biti jasno definisan kroz Ostervalderovo platno, prirodno komplementarno rešenje za infrastrukturu jesu *cloud* okruženja. Kao takvo, istraživanje sadrži važna sagledavanja koja direktno mogu biti primenjena u daljem radu na ovoj tezi. Istraživanje je takođe uvelo ontološki pristup kako bi se napravio koncept *cloud* poslovnog modela jer to može pomoći organizacijama da uspešno razumeju vrednosti, servise i elemente izvršavanja uz detalje kako se servisi i proizvodi profitabilno dostavljaju.

Sam pristup je veoma opravdan, pogotovo kada se uzme u obzir razlog njegovog nastajanja. Vlasnici *cloud* usluga su gigantske multinacionalne kompanije kao što su *Google*, *Amazon Web Services* (*AWS*), *Microsoft, IBM* i druge koje su otvorile svoje centre podataka za javno korišćenje. Mnoge su svoj poslovni model u međuvremenu zasnovale na prodaji svojih *cloud* servisa i kao posledica toga, za veću zaradu je bilo potrebno doći do većeg tržišta. Na taj način, nastale su mnoge metodologije transformacije na *cloud* zasnovana rešenja. Najiskusniji i najzreliji na ovom području je Amazon, koji je prvi objavio svoju metodologiju za usvajanje *cloud* usluga

(engl. *Cloud Adoption Framework*) [13]. Slično, *Microsoft* je objavio knjigu koja je praktično metodologija za transformaciju [7]. Svaki veći pružaoc *cloud* usluga sadrži neki vid saveta za prelazak na *cloud* platforme, bilo u vidu treninga, knjiga, internet materijala ili konsultantskih servisa. Čak i konsultantske kuće poput Gartnera objavljuju priručnike za tranziciju na *cloud* okruženja [5]. S obzirom na ogromno iskustvo navedenih kompanija koje su prevele mnoštvo kompanija na korišćenje *cloud* okruženja, odgovarajuća literatura [7], [13] će biti temeljno izučene.

Iako su, usled mnogostruke provere, u industriji sigurno najzrelija rešenja, ograničenja njihove primene proističu iz namene zbog koje su nastali, tačnije cilja njihovog istraživanja. Njihov cilj jeste operacionalizovati *cloud* tehnologije i time ostvariti profit. Kao takve, ove metodologije uzimaju pretpostavku da je SkS poslovni model nužno takav da maksimalno koristi *cloud* usluge. S druge strane, postoje mnoga ograničenja za razne industrije, a pogotovo za one koje rade sa nacionalno kritičnim infrastrukturama kakva je elektroenergetika. Iz tog razloga, neophodno je prvo sagledati poslovni model koji se želi postići, a potom analizirati da li postoje tehnološke ili prepreke drugog tipa za usvajanje novih tehnologija. Tek nakon studije izvodljivosti, ima smisla planirati transformaciju rešenja.

Zanimljivo je primetiti da u slučaju *cloud* okruženja čiji poslovni model je veoma jasan i u modernim rešenjima ide na naplatu resursa onoliko koliko su resursi korišćeni, finansijski aspekt mnogo utiče na arhitekturu rešenja. Kao takav, finansijski aspekt postaje integralna komponenta kriterijuma za određivanje dizajna servisa. SkS se često i bazira na *cloud* platformama pa je i finansijski model usko povezan sa odlukama pri dizajniranju servisa. Tako u SkS postoje tri različita modela zarade, gde svaki ima svoje prednosti i mane. Studija koja se bavi analizom prednosti i mana je opisana u [14], dok implementacija vrste naplate zavisi i od tehnološke zrelosti. Naplata može biti kupovina licenci, iznajmljivanje licenci na vremenskoj osnovi ili iznajmljivanje po korišćenju.

Osim što se u slučaju SkS mora voditi računa o potrošnji *cloud* resursa, postoji i momenat skalabilnosti prodaje gde se čak i na delimično male uštede postižu značajne uštede na velik broj projekata. Uštede u potrošnji *cloud* resursa dovode praktično do zarade ili do pristupačnijeg rešenja na tržištu. Iz ovog razloga je neophodno razumeti promenu poslovnog modela, ali isto tako i ekonomske aspekte kako bi praćenje finansijskog modela dovelo do pravih pokazatelja. Da bi se shvatili svi finansijski uticaji na rešenje, istraživanje se mora osvrnuti i na istraživanja iz ekonomije.

U izradi ove teze je korišćena doktorska disertacija [15] sa svojim referencama kao osnova za dalje istraživanje, kao i radovi iz oblasti tehničkih nauka koji uzimaju u obzir ekonomske aspekte. Jedan od referentnih radova se bavi analizom ukupnog troška kod isporuke tradicionalnih softverskih rešenja naspram SkS rešenja kao i različitim modelima naplate za softverska rešenja kako bi pružio alat za pomoć u odluci [16]. Na osnovu iznetih formula, može se sračunati isplativost jednog i drugog pristupa. Upravo ovakve formule mogu biti semafor odluke za neke potencijalne klijente, i kao takve moraju biti sagledane pri definiciji poslovnog modela.

Istraživanje koje je pronađeno u toku sistematične pretrage literature a koje u najvećoj meri obrađuje aspekte kompletne transformacije je objavljeno u radu [17]. Istraživanjem referenci,

izdvojeni su i sledeći relevantni radovi [18]–[21] iste grupe autora koje se nadovezuju na pomenuto istraživanje i kao celinu ćemo ih ovde posmatrati jer je veoma bitno sagledati celokupan opseg ovog istraživanja. Takođe, dodatna selektovana istraživanja nakon pretrage svih relevantnih baza podataka, a koja se bave isključivo metodologijom za celokupnu migraciju rešenja tako da obuhvataju i tehničke i netehničke faktore imaju navedene ciljeve istraživanja.

1. Pomoći organizacijama da razumeju koje veštine je potrebno razviti, gde se trenutno nalaze u dijapazonu usvajanja *cloud* tehnologija i koliko vremena će biti potrebno da se izvrši migracija na *cloud* rešenje [22].
2. Integraciju tehnoloških (engl. *TAM – Technology Acceptance Model*) i organizacionih (engl. *TOE – Technology Organization Environment*) modela sa kreiranjem intervjua koji sadrži 12 varijabli za analizu primenljivosti *cloud* modela [23].
3. Predstaviti pristup baziran na modelu za poluautomatsku migraciju tradicionalnih softverskih sistema na *cloud*. Veoma korisna strana istraživanja je opis modela *cloud* okruženja sa svim svojim ograničenjima kao i mehanizama za uočavanje prekršaja ograničenja od strane tradicionalnih softvera [24].
4. Pristup migracije tradicionalnih rešenja sa metodologijom čiji je akcenat migracija podataka. Glavna ideja u odnosu na ostale metodologije jeste način da se koristi jedinstvena baza podataka tokom migracije [25]. Praktičnost ove primene je kada se postojeće rešenje koje je već u operacijama (tj. softver se izvršava i aktivno koristi) migrira na novu tehnologiju, što je u suštini izvan opsega ove disertacije. S druge strane, principi na kojima se bazira ova metodologija mogu biti od velikog interesa kod višeorganizacijskih servisa.
5. Identifikacija podrazumevanih koraka u transformaciji kod migracije na PkS rešenja. Koriste se IkS/PkS/SkS nivoi kao primarna diferencijacija ponude da se ekstrakuju problemi vezani samo za PkS [26]. Istraživanje je bitno iz razloga što ulazi u specifičnosti *cloud* ponude i analizira migraciju iz ugla na koji sloj se ciljni softver migrira.

Analizom svih istraživanja vezanih za transformaciju arhitektura koje odgovaraju SkS modelu primene, višeorganizacijske arhitekture su izučavane kao praktično obavezna komponenta SkS. Zastupljene su jer direktno utiču na jednu od prednosti SkS poslovnog modela, a to je smanjene troškova, tj. smanjene cene što utiče na povoljniji servis ka klijentima ili veći prostor za zaradu.

S obzirom da je disertacija fokusirana na tehnološku transformaciju postojećih rešenja, disciplina koja je u fokusu u okviru transformacije jeste prevođenje arhitekture na višeorganizacijski model bez promene postojećeg rešenja, ili sa minimalnom promenom.

Jedan pristup [27] u izučavanoj literaturi se oslanjao na razvijenu softversku podršku koja se bazira na izdvajanju realizacije bezbednosnog modela i upravljanju aktivnostima u ciljnoj aplikaciji. U ovom segmentu, predloženo rešenje ima upravo isti pristup rešenju, a to je izdvajanje bezbednosnog modela za višeorganizacijski model softvera. Takođe je cilj bio da prevođenje bude primenljivo i na već postojeća rešenja. Međutim, ideja u [27] strogo podrazumeva da ne bude potreban dodatan rad na izvornom kodu samih aplikacija kako bi koristile softversku podršku. Da bi to uopšte bilo moguće, prihvataju pristup aspekt orijentisanog programiranja i dizajn šablonu injekcije zavisnosti (engl. *dependency injection*) kako bi se bezbednosni kod dinamički injektirao u funkcije u toku njenog izvršavanja. Glavna razlika u pristupu je to što se njihova softverska podrška nalazi kao omotač oko postojećih SkS tako što u nekim slučajevima presreće

komunikaciju. Preciznije, nalazi se između klijenta i SkS servisne strane i implementira sigurnosni model koristeći injekciju zavisnosti. Kao takvo, ima usku primenu na rešenja gde su komunikacione poruke poznatog formata, kao što je slučaj kod *web* servisa i ne može da utiče na internu realizaciju samih servisa.

Postoji više rešenja koja istražuju posebno domen migracije *web* servisa u višeorganizacijske servise, a jedan od izabranih predstavnika koristi ideju definisanja transformacije. Ta ideja je implementirana u [28] tako da se na osnovu analize trenutnog servisa predlažu određene tehnologije i rešenja, sa dodatkom planirane revizije sa eksperimentima koji ciljaju performanse. Rad je izdvojen jer analiza, predlog implementacije i eksperimenti za verifikaciju poprilično oslikavaju pristup predlogu rešenja kao u ovoj disertaciji.

Rešenje na koje će se osloniti ova disertacija je okrenuto tako da obuhvata gotov bezbednosni model na koji će SkS moći da se osloni, i kao takvo prirodno se implementira u servisno orijentisanu arhitekturu (SOA) softvera. S druge strane, predloženo rešenje ne može da se primeni na postojeća rešenja bez dodatnih intervencija na izvornom kodu ciljne aplikacije. U [27] se problem izmene postojećeg rešenja rešava preko injekcija zavisnosti i posebnom konfiguracijom za ciljne SkS, ali ovakav pristup može dovesti do određenih problema jer ne postoji celokupna pokrivenost nad kodom ciljne aplikacije. Rešenje je eksperimentalno evaluirano i izmereni su uticaji sigurnosnog modela na performanse. Praćenje uticaja na performanse je bitno zbog aplikacija koje koriste bezbednosni model jer će sigurno negativno uticati na servisne nivoe koje SkS mora da ispuni.

Kada je reč o generalnim performansama sistema, po teoriji je jasno da performanse moraju biti narušene pri prevođenju postojećih rešenja na višeorganizacijska. Ovde se ne uzima u obzir eventualno razvijanje istih servisa od početka sa drugačijom arhitekturom na umu, nego da se iskoriste rešenja već dokazana u praksi. Podaci su jako važan aspekt i njihova izolacija se mora obezbediti po svaku cenu. Evaluaciju performansi za različite šablone izolacije podataka uglavnom na serveru baze podataka je korisno imati, i više istraživanja je bilo fokusirano samo na ovaj segment, a [29] i [30] su primer takvog istraživanja. Doprinosi navedenih istraživanja su u teoretskom istraživanju mogućnosti izolacije podataka i daju značajan broj rezultata merenja.

U radu [31] je prikazan način za implementaciju i arhitekturu višeorganizacijskog SkS. Softver je pisan tako da se izbegne korišćenje bilo kakve softverske podrške. Razlog je to što je postojala inicijativa da se izbegne vezivanje za dodatni softver. Upravo ovaj pristup je u kontradikciji sa idejom disertacije, a to je da se sam servis brine o svim pitanjima višeorganizacijskog modela. Ideja disertacije je razvijanje softverske podrške koja će omogućiti da se transformišu tradicionalna rešenja na SkS tako da se postigne višeorganizacijska osobina, a da onaj ko ih razvija ne mora o tome da razmišlja. Uz to, nisu priloženi rezultati testiranja niti je samo testiranje sprovedeno tako da se s te strane nije mogao izvesti zaključak o efikasnosti predloženog pristupa.

Postoje i pristupi koji se oslanjaju na ponovno dizajniranje postojećih rešenja [32]. Pokušano je da se omogući automatska programska analiza i modifikacija izvornog koda ciljne aplikacije na način da se izvrši promena rešenja tako da uzima u obzir višeorganizacijski model. Ovaj pristup je skroz drugačiji od svih ostalih pristupa, i deluje preambiciozno, ako se uzmu u

obzir kompleksna softverska rešenja kakva se nalaze u industriji. Deluje teško ostvarivo da softverska podrška samostalno sprovede odgovarajuće promene u kodu. U svakom slučaju, predloženi alat koji se bavi poboljšavanjem postojećih aplikacija tako da postanu višeorganizacijska je suprotnost od pristupa koji će se ovde predložiti.

Ideja ove teze jeste da pruži pristup iskorišćenja postojećeg rešenja kako bi se transformacija dogodila što brže i efikasnije. Istraživanje sa sličnom idejom je metodologija koja pomaže pri razvoju softvera opisana u [33]. Istraživanje obuhvata kako da se tehnologija migrira sa konvencionalne klijent-server arhitekture na SkS *cloud* bazirane servise uz podršku višeorganizacijskog modela i bez ponovnog razvijanja koda i modifikacije postojećeg.

Istraživanje koje je osnova za dalji rad na višeorganizacijskim aspektima rešenja se bavi višeorganizacijskim problemom na dva nivoa: 1. prevođenje postojećih rešenja na višeorganizacijska, i 2. pravljenje prirodnih višeorganizacijskih rešenja uz kombinaciju oba pristupa [34]. Oba pristupa su važna jer kada imamo već postojeće rešenje u koje je uloženo mnogo inženjerskog truda, primorani smo da uz minimalno napora sprovedemo višeorganizacijsku paradigmu. Dalje istraživanje u okviru ove teze se oslanja na već objavljeno istraživanje [34] sa pokušajem šire primene na nivou zahtevnog distribuiranog softverskog rešenja. Benefiti moguće transformacije predloženim rešenjem biće prikazani na primeru kompleksnog softverskog rešenja u domenu pametnih elektroenergetskih mreža (engl. *Smart Grid*).

Da bi se razumele specifičnosti infrastruktura okarakterisanih kao kritičnim za nacionalnu infrastrukturu, potrebno je razumeti izazove koji su nametnuti. U ovoj tezi je fokus na industriji energetike, tačnije na pametnim elektroenergetskim mrežama koje zahtevaju veoma kompleksne informacione tehnologije. Da bi pojasnili pojam pametnih elektroenergetskih mreža i da bi se u okviru njih jasno pozicionirali na NDMS (napredni distributivni menadžment sistem) softver, neophodno je dati pregled evolucije elektroenergetskih sistema. Razumevanje NDMS pozicije je važno radi shvatanja kritičnosti koju može izazvati u okviru nacionalne kritične infrastrukture.

Današnja elektroenergetska mreža je ishod ubrzane urbanizacije koja je zahvatila većinu sveta tokom prošlog veka. Većinom su elektroprivredna preduzeća usvojila slične tehnologije. Rast elektroenergetskih sistema je bio uslovljen ekonomskim, političkim i geografskim faktorima koji su jedinstveni za svaku elektroprivredu [35]. Uprkos razlikama, osnovna topologija postojećih elektroenergetskih sistema se uopšte nije promenila.

Od osnivanja sistema, postojale su jasne granice između generacije električne energije, transmisije, i podsistema distribucije. Kroz ovu podelu, različiti nivoi automatizacije, evolucije i transmisije su postojale za svaku oblast [35]. Problem današnjih sistema je što su jednosmerni po prirodi. Dodatno, zbog svoje hijerarhijske topologije pate od lančanog ispada kada dođe do kvara na mreži.

Pametne elektroenergetske mreže treba da prevaziđu probleme postojećih mreža. U suštini, treba da omoguće elektroprivredama da imaju potpuni uvid i kontrolu nad svojim servisima i uređajima. Dodatno, potrebno je da mreže budu samostalno popravljive i otporne na predvidljive i nepredvidljive nepravilnosti sistema. Na kraju, važno je da se uključe krajnji potrošači i da im se omogući interakcija sa sistemom tako da mogu da trguju energijom kroz sistem. Jedna od najvećih kalifornijskih elektroprivreda [36] vidi pet segmenata pametnih mreža:

1. obnovljivi izvori energije i integracije sistema čuvanja energije,
2. kontrola mreže i optimizacija uređaja,
3. efektivnost radne snage,
4. pametna brojila,
5. pametna rešenja koja uključuju krajnje korisnike.

U literaturi, a koja se oslanja direktno na vodeće elektroprivrede Severne Amerike se generalno drugi segment navodi kao fundament pametnih mreža oko kog se izgrađuju dodatne funkcionalnosti [35], [36]. Svi slojevi su zasnovani na informacionim tehnologijama.

Istorijski, unutar podstanica se mogla naći automatska kontrola sa osnovnom funkcionalnošću da deenergizuje i automatski jednom ponovo energizuje mrežu kako bi proverila da li je ispad bio privremen. Primarno se ova funkcija izvodila elektromehaničkim relejima. Sledeća generacija opreme 1950-ih godina je bila bazirana na automatskoj telefonskoj kontrolnoj tabli, koja bi omogućila operatorima uvid u stanje pojedinih prekidača kao i njihovu kontrolu. Ova funkcionalnost je kasnije, u 1960-im zamenjena SCADA sistemima (engl. *SCADA - Supervisory Control and Data Acquisition System*) koji su se polako proširili 1970-im i 1980-im tako da uključe barem većinu prenosnog sistema (od 220 kilovolti i više), kao i neke podstanice distributivnog sistema. SCADA sistem je tada korišćen kao softver za centralne (engl. *mainframe*) računare i namenjen je za kontrolnu sobu sa ciljem prikupljanja podataka i upravljanjem preko udaljenih terminalskih jedinica – RTU (engl. *RTU - remote terminal unit*). Kasnije su RTU uređaji negde zamenjeni PLC uređajima (engl. *PLC – programmable logic controller*) koji došli iz proizvođačkih industrija.

Polovinom 1990-ih, konfiguracija RTU/PLC je zamenjena drugačijom arhitekturom. Danas se u Kaliforniji razmatra korišćenje mrežne konfiguracije gde uređaji mogu da razmenjuju poruke i da interaguju sa sistemom iz kontrolne sobe, pri čemu je mreža zasnovana na *IEC 61850* protokolu [36]. Važna oblast u kontroli mreža i optimizaciji uređaja u pametnim elektroenergetskim sistema je distributivni sistem jer je statistički primećeno da su ispadi sistema izazvani u distributivnom sistemu [35].

NDMS softver služi za nadgledanje distribucije u realnom vremenu i njenu optimizaciju. Osim toga, moderni NDMS softveri obuhvataju u svojim modulima i OMS (engl. *Outage Management System*) sisteme za prijavu i detekciju kvarova, WOM (engl. *Work Order Management*) za efektnije korišćenje radne snage (oblast 3.) i u skorije vreme obuhvataju funkcionalnosti koje uključuju krajnje korisnike (oblast 5.) kao što je upravljanje opterećenjem (engl. *demand response – DR*).

Tradicionalni NDMS je distribuiran softverski sistem koji je u krajnjoj produkciji veličine od više desetina virtuelnih mašina i kao takav dostiže granice vertikalne skalabilnosti rešenja. Zahtevi današnjice se povećavaju za red veličina kada se uzme u obzir da su moderni uređaji rasprostranjeni i na nivou mreže niskog napona. Primer modernog NDMS sitema koji je korišćen kao osnova u ovom istraživanju je detaljno opisan u [37]. Stvara se takozvani internet stvari (engl. *Internet of Things – IoT*) scenario, što direktno dovodi do problema velikih podataka (engl. *big data*) koji se prirodno rešava *cloud* sistemima. Čak je i jedan od glavnih motivatora za adopciju *cloud* sistema upravo *big data* problematika. Potreba za čuvanjem, procesiranjem i analizom velike količine podataka konačno tera poslovne korisnike na upotrebu *cloud* rešenja. Jedna od glavnih

prednosti je što *big data* servisi zahtevaju puno resursa za analizu podataka, ali jednom kada se izveštaj izgeneriše, ti resursi nisu potrebni do sledećeg izveštaja koji može biti i mesecima daleko [38]. Varirajući zahtevi za resursima omogućavaju ovakvim sistemima da maksimalno iskoriste prednosti fleksibilnosti *cloud* sistema.

Istraživanja u oblasti pametnih mreža u vezi sa servisima koji uključuju krajnje korisnike (oblasti 4. i 5.), obično predstavljaju ove probleme sa nazivom interneta stvari *IoT* ili pametnih gradova (engl. *Smart City*). Ideja jeste da se omogući korisnicima interakcija sa energetskim sistemom u realnom vremenu i upoznavanje krajnjih korisnika kroz analitiku. Takva istraživanja [39], [40] se često baziraju na bezbednosti i ostalim aspektima *cloud* tehnologija, jer je motivacija za *cloud* okruženjima enormna u tim scenarijima.

Razlog je dvojaki: gore navedeni motivator u vidu sistema velikih podataka i komunikacione linije koje u *cloud* slučaju omogućavaju veliku skalabilnost, tj. mogućnost konektovanja istovremeno velikog broja uređaja. S obzirom na važnost i uticaj sistema velikih podataka na *cloud*, oni su neophodni element za istraživanje jer transformacija na SkS u poslovnom modelu može imati jednu novu vrednost koju ne može ispuniti u tradicionalnom slučaju.

U slučaju ADMS softvera, funkcionalnost upravljanja opterećenjem izaziva problem velikih podataka, ali i potencijalno diferencirajuću novu vrednost za SkS. Radovi čiji će zaključci biti iskorišćeni u ovoj tezi vezano za sisteme velikih podataka su [41] i [42]. Radovi istražuju na koji način se podaci u *IoT* scenariju mogu procesuirati i čuvati tako da omoguće SkS.

S obzirom na različito početno stanje elektroprivreda širom sveta, a da su većina krenula putem pretvaranja postojećih mreža u pametne umesto investiranje u nove infrastrukture, nastala je metodologija za transformaciju mreža na pametne. Model zrelosti pametnih mreža (engl. *Smart Grid Maturity Model*) [43] je upravljački alat za elektroprivrede koji koriste, za planiranje svog razvojnog puta, određivanje prioriteta opcija i merenje progresa u realizaciji projekata pametnih mreža. Sastoji se od četiri matrice koje sadrže ocene od 1 do 5 sa deskripcijom ocenjenih koraka kako bi se elektroprivrede mogle orijentisati. Ocena 1 predstavlja početke dok ocena 5 označava prvence u oblasti.

Sama metodologija nije kompleksna, ali je proizvod radnih grupa sačinjenih od relevantnih eksperata iz oblasti. Uopšte kreiranje ove metodologije svedoči tome da su i projekti izgradnje pametnih mreža u začetku i da postoji potreba da se taj put transformacije formalno unificira na globalnom nivou. Motivi su isti kao i za potrebu istraživanja ove teze gde je težnja unificirati put od tradicionalnih softverskih isporuka do *cloud* baziranih SkS rešenja.

3. Višeorganizacijska metodologija u opštem slučaju

Za razumevanje motivacije primene višeorganizacijskog modela softvera, prvo je potrebno razumeti modele nad kojim je ona primenljiva. *Cloud* okruženja motivišu pojavu višeorganizacijskog modela, i opisani su u odeljku 3.1. Nakon toga, u odeljcima 3.2 i 3.3 se uvode pojmovi servisa i važnosti SkS poslovnog modela. Kako ovo poglavlje ima za cilj da pojasni upotrebu *cloud* rešenja kao i softverskih principa koji su proistekli iz distribuiranih sistema i *cloud* okruženja, u odeljku 3.4 su opisani moderni pravci softverskog razvoja.

Nakon pojašnjenja SkS poslovnog modela i principa korišćenja resursa u *cloud* okruženjima, odeljci 3.5 i 3.6 opisuju sam višeorganizacijski model i njegove prednosti kao i načine za implementaciju privatnosti i izolacije podataka.

Na kraju, može se definisati opis problema uz napomenu šta nije fokus ovog rada. U odeljku 3.7 se nalaze i hipoteze ovog istraživanja.

3.1. Cloud okruženja

Cloud računarstvo predstavlja širok pojam koji može da se odnosi na različite koncepte u računarstvu, a svi koncepti ukazuju na to da je *cloud* u suštini jedan visoko automatizovani, distribuirani sistem, koji je zbog svoje fizičke udaljenosti uslovljen postojanjem kvalitetnih konekcija. U dodatku A se nalazi osnovna terminologija vezana za *cloud* okruženje kao termin. U ovom odeljku je dat opis *cloud* sistema kakav će se u ovoj disertaciji podrazumevati kao referentni.

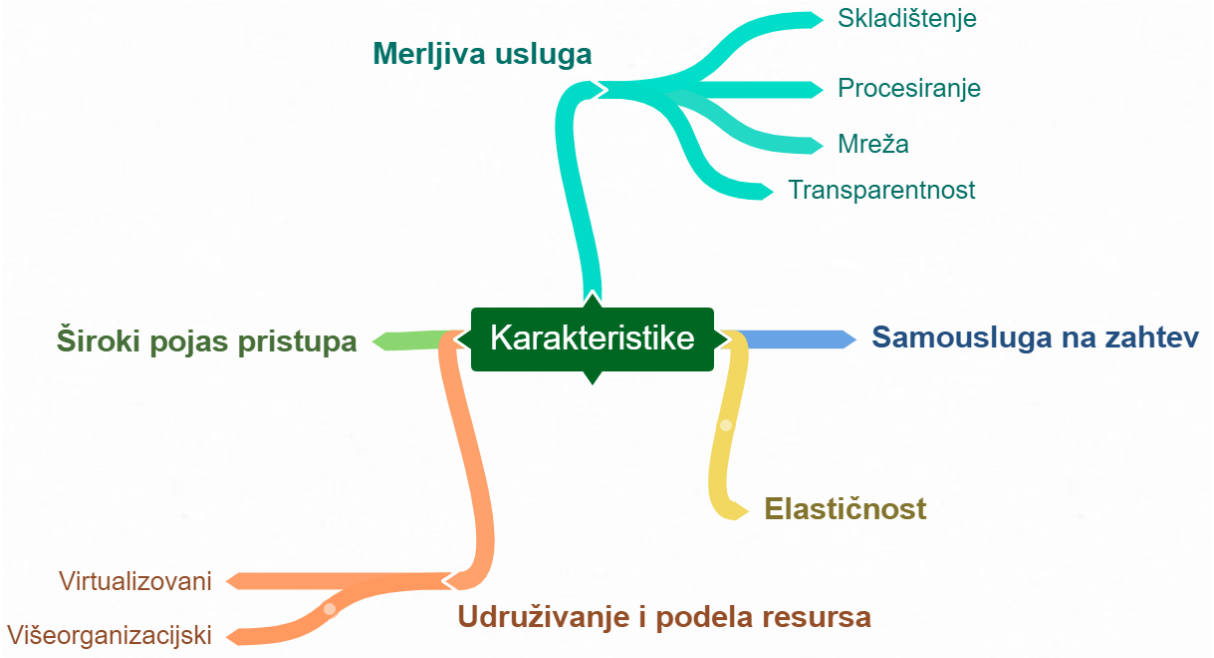
Različito korišćenje *cloud* termina je najviše izazvano potrebom raznih proizvođača da odgovore na tržište gde se traže *cloud* sistemi čak i kada ne poseduju *cloud* usluge u potpunosti. Dodatno, razlog za nepostojanje konkretne definicije jeste velik marketinški značaj samog termina *cloud*, koji se sve više upotrebljava. Marketinška upotreba termina je često u cilju odgovaranja na tržište i implicitno stavljanje do znanja da se prate moderni standardi, pa i da su SkS servisi u ponudi, iako je često reč o tradicionalnim aplikacijama koje su samo postavljene i izvršavaju se na udaljenoj lokaciji. SkS poslovni model je detaljnije opisan u poglavlju 3.3.

Cloud platforme omogućavaju da bez ulaganja u skup hardver i generalno računarske resurse postoji visoka dostupnost i visoka otpornost na greške uz veoma velik stepen skalabilnosti sistema. Upravo napretkom interneta, pojavio se i koncept *cloud* sistema, koji je omogućio s jedne strane pružanje softverskih usluga kao servisa, a s druge strane je omogućio dinamičko iznajmljivanje računarskih resursa po potrebi. Iznajmljivanje računarskih resursa je ključna komponenta uspeha *cloud* sistema jer male i srednje organizacije ne moraju da rizikuju i ulažu sredstva u veoma skupe računarske resurse.

U narednim poglavljima su prikazane osnovne karakteristike *cloud* okruženja da bi se moglo efikasno razlikovati šta obuhvata termin *hosting*, gde se isto mogu pružati resursi na zahtev na udaljenoj lokaciji, u odnosu na termin *cloud*. Odeljak 3.1.4 ukratko opisuje motivaciju za nastanak i sve masovnije korišćenje *cloud* tehnologije.

3.1.1. Esencijalne karakteristike *cloud* okruženja

Slika 3 prikazuje mapu karakteristika *cloud* okruženja koje će biti opisane. Na prvom nivou možemo razgraničiti pet esencijalnih karakteristika koje definiše naučna laboratorija *National Institute of Standards and Technology* (NIST) [44], [45].



Slika 3 - NIST karakteristike cloud okruženja

Svaka od pet karakteristika je neizostavna da bi se okruženje smatralo *cloud* okruženjem. Može se primetiti da *hosting* rešenja zadovoljavaju većinu ovih karakteristika, pogotovo ako se gleda na IkS nivou, ali elastičnost, stepen automatizacije kao i osobina širokog pojasnog pristupa (2) nisu zadovoljene. U narednim pasusima je dat opis pet NIST karakteristika koje određuju *cloud* okruženja.

1) Samousluga na zahtev (engl. *On-demand self-service*) – *Cloud* okruženja mogu da ustupe serversko vreme, mrežu i kapacitete stalne memorije po potrebi automatski bez potrebe za ljudskom intervencijom.

2) Široki pojas pristupa (engl. *Broad network access*) – Sve mogućnosti u ponudi su dostupne putem mreže, obično putem interneta i može im se pristupiti različitim tankim i debelim klijentskim platformama kao što su mobilni telefoni, tableti, lap topovi i radne stanice. Ovo omogućava inžinjerima da izvršavaju poslove sa udaljenih lokacija, bez potrebe za njihovim prisustvom na terenu.

Karakteristika širokog pojasa pristupa često izdvaja *cloud* okruženja u odnosu na *hosting* rešenja koja imaju ograničen broj centara podataka. Razlog je što *cloud* okruženja koriste mrežu centara podataka koji su geografski raspodeljeni. Distanca između klijenata i servisa je uslovljena geografskom distancom. Geografska distribucija klijentskih zahteva može biti slučajna i teška za predvideti [46]. U svakom slučaju, obična *hosting* rešenja obično nude usluge iz jednog ili više geografski bliskih centara podataka, i u tom slučaju se može smatrati da nemaju geografski širok

pristup. Današnji vodeći pružaoci *cloud* usluga poseduju sopstvene interne mreže kojima povezuju centre podataka širom sveta.

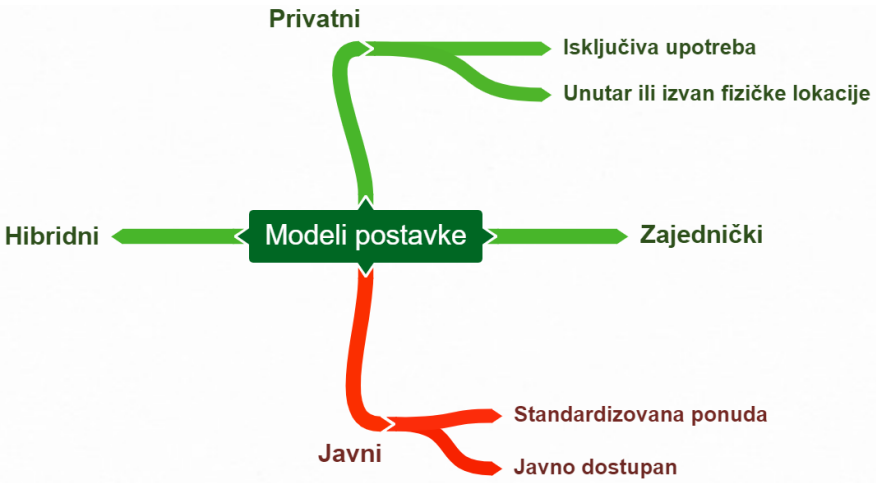
3) Udruživanje i deljenje resursa (engl. *Resource pooling*) – Računarski resursi se udružuju u višeorganizacijski model sa različitim fizičkim i virtuelnim resursima koji se dinamički alociraju različitim potrebama različitih potrošača. Primeri resursa mogu biti procesori, stalna memorija, radna memorija i mrežna komunikacija [45]. Jedan od izazova bezbednosti samog *cloud* okruženja se postavlja već u višeorganizaciskom svojstvu [47]. Svi servisni modeli imaju višeorganizacijsko svojstvo na nivou podataka, dok PkS i IkS dodatno imaju višeorganizacijsko svojstvo na nivou procesa. Procesi različitih korisnika se mogu napadati između sebe, ili čak ostale procese koje pripadaju *cloud* okruženju [47]. Više detalja o višeorganizacijskom svojstvu *cloud* okruženja će biti opisano u 3.5.

4) Brza elastičnost (engl. *Rapid elasticity*) – Ideja elastičnosti potiče od potrebe za adaptacijom potrošnje resursa na potražnju. Ukoliko je moguća automatska elastičnost, to je još bolje i većina pružaoca *cloud* usluga sadrže neki vid podrške automatizacije. Elastičnost je karakteristika koja se obično ne zahteva i ne očekuje od *hosting* rešenja.

5) Merljiva usluga (engl. *Measured service*) – Upotreba resursa može biti nadgledana, kontrolisana i izveštaji se mogu kreirati. Na ovaj način postoji transparentnost za obe strane, i za pružaoca i za korisnika usluga. Transparentnost takođe zadovoljava bazične bezbednosne zahteve jer se kroz transparentnost može primetiti neko neočekivano ponašanje.

3.1.2. Modeli postavke – od privatnog do javnog

Slika 4 prikazuje mapu modela postavke (engl. *deployment models*) za *cloud* okruženja po NIST arhitekturi [45]. Javna cloud okruženja su označena crvenom bojom, jer za razliku od ostalih modela podrazumevaju veći stepen korišćenja višeorganizacijskog svojstva, što je fokus ove disertacije, a i ukazuje na potrebu za pojačanim sigurnosnim modelom.



Slika 4 - NIST modeli postavke

NIST četiri modela postavke su sledeća:

1) Privatni cloud (engl. *Private*) – privatne infrastrukture su dodeljene za isključivu upotrebu jedne organizacije. Može biti na udaljenoj lokaciji (engl. *off premises*) ili u okviru poslovnog prostora organizacije (engl. *on premises*). Infrastruktura može biti kupljena i održavana

od strane organizacije ili treće strane, ili neke kombinacije. Privatno *cloud* okruženje podrazumeva da su prava pristupa ograničena samo na organizaciju koja je poseduje i da hardverska infrastruktura nije deljena nego pripada samo organizaciji koja je poseduje. Uobičajeno je da je privatni *cloud* fizički lociran u okviru fizičkog prostora organizacije. Inicijalna ulaganja su skuplja, ali pružaju naizgled bolju bezbednost zbog fizičke prisutnosti infrastrukture, a pružaju i bolje performanse jer infrastruktura nije deljena.

2) Zajednički *cloud* (engl. *Community cloud*) – privatne infrastrukture namenjene za korišćenje određenih zajednica koje dele iste probleme ili zahteve (npr. misija, bezbednosni zahtevi, politike i usklađenost sa regulativama). Isto, kao u slučaju privatnog *cloud* okruženja, vlasništvo i održavanje može biti podeljena uloga između organizacija u zajedništvu, ili treće strane. Zajednički *cloud* je u suštini privatno *cloud* rešenje koje je deljeno između više organizacija. Dodatni motiv za zajednički *cloud* je smanjene cene troškova privatnog *cloud* rešenja uz zadržavanje privatnosti za razliku od javnog *cloud* rešenja.

3) Javni *cloud* (engl. *Public cloud*) – javne *cloud* infrastrukture su namenjene široj javnosti. One su obično u vlasništvu određenih kompanija, akademskih ustanova ili državnih organizacija koje ih ujedno i održavaju. Naravno, moguća je i neka od kombinacija. Isključivo postoji u centrima podataka pružaoca *cloud* usluga. U slučaju javnih *cloud* infrastruktura postoji neki vid zastupljenosti višeorganizacijskog svojstva. Najčešće je hardverska infrastruktura deljena što podrazumeva stalnu memoriju, mrežu, pa čak i same fizičke servere. Izolacija se obično svodi na logičku, pa se u ovim infrastrukturama često razmišlja i o softverskim višeorganizacijskim strukturama. Takođe, smatra se da servis pripada javnom *cloud* rešenju kada je dostupan preko javne mreže, koju najčešće predstavlja Internet. Tehnički, ne postoji razlika između javne i privatne *cloud* infrastrukture, ali razlika postoji u njihovoj veličini, bezbednosti i automatizaciji procesa održavanja infrastrukture. Kod javnog *cloud* sistema postoje problemi i privatnosti jer se podaci i aplikacija nalaze u okviru centra podataka pružaoca *cloud* usluge. Takođe, s obzirom da je javni *cloud* dostupan svima na javnoj mreži, smatra se da je stoga vulnerabilniji, tj. postoji povećan rizik napada. S obzirom na postojeći rizik, mora se ozbiljnije pristupiti projektovanju bezbednosnog modela.

4) Hibridni *cloud* (engl. *Hybrid cloud*) – hibridne infrastrukture sadrže dve ili više različite *cloud* infrastrukture (privatne, zajedničke ili javne) koje su nezavisni entiteti, ali su vezane zajedno bilo standardizovanom tehnologijom ili posebno razvijenom koja omogućava portabilnost aplikacije i podataka između različitih *cloud* okruženja. Time se dobija prividna beskonačnost resursa. U slučaju da je potrebno više resursa nego što je raspoloživo u privatnom *cloud* okruženju, dinamički je moguće alocirati nove resurse u okviru javnog *cloud* okruženja. Pri testiranju implementacije koju predlaže ovaj rad, korišćen je privatni *cloud* kao okruženje, ali i javni. Međutim, s obzirom da nisu korišćeni u koordinaciji, tačno je reći da hibridno rešenje nije isprobano.

3.1.3. Nivoi usluge – IkS, PkS i SkS

Po nivou usluge koju *cloud* pružaoci usluge nude, sada već univerzalno prepoznatljiva podela, mogu se klasifikovati tri poznata različita nivoa usluge (Slika 5).

1) Softver kao Servis (engl. *Software as a service – SaaS*) – Organizacija može koristiti *cloud* bazirane aplikacije koje se izvršavaju koristeći *cloud* infrastrukturu. Sami servisi su dostupni iz mnogih klijentskih uređaja kroz tanke klijente, kao što su *web* pretraživači, ili preko programskog interfejsa. Organizacija ne održava niti kontroliše infrastrukturu uključujući mrežu, servere, operativne sisteme, skladišta podataka, pa čak ni individualne delove softvera koji koristi, osim ograničenog dela konfiguracije korisničkih podešavanja.

Navedena definicija SkS, preuzeta iz [45] je upravo ona koja uvodi nejasnoću u sam termin SkS jer je ona vrlo česta a podrazumeva *cloud* okruženje. Da bi se bolje razumeo SkS, u odeljku 3.3 je opisan SkS kao poslovni model, ono što zaista i jeste, iako se često koristi u tesnoj sprezi sa *cloud* pružaocima usluga iz prostog razloga što je *cloud* nativno okruženje za tehnologiju koja će se koristiti u SkS poslovnom modelu. Ovde je važno napomenuti da za SkS nije neophodno koristiti *cloud* okruženje.

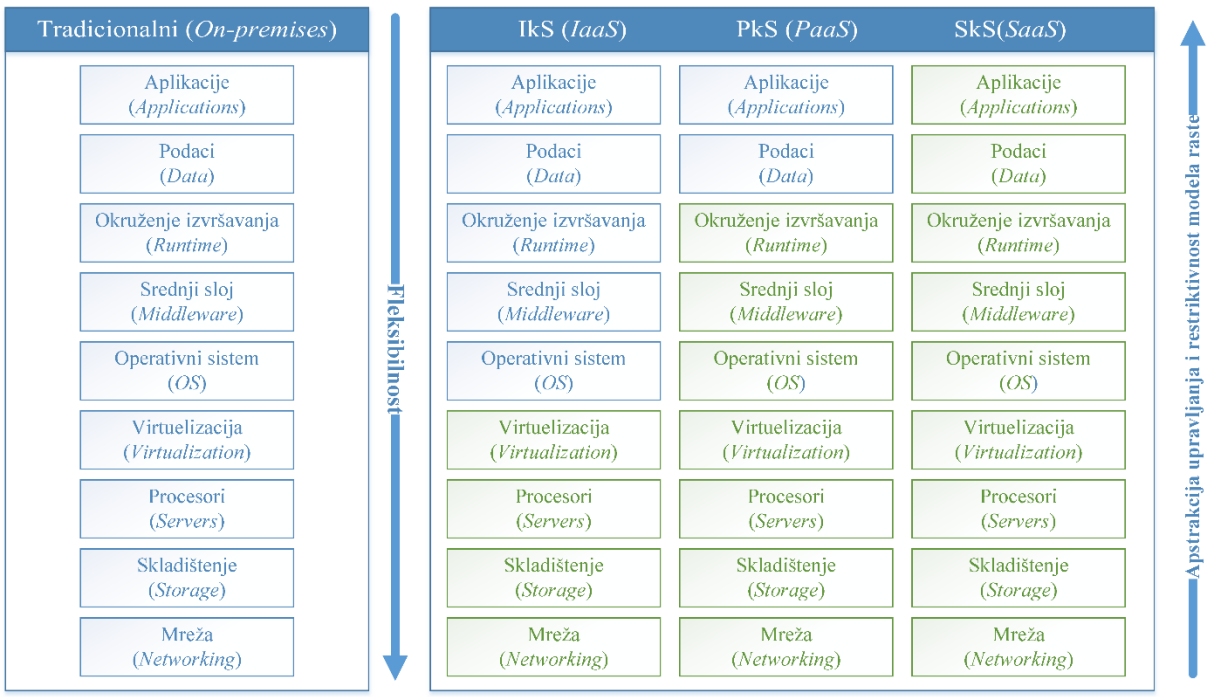
2) Platforma kao Servis (engl. *Platform as a service – PaaS*) – Organizacija može postaviti svoja softverska rešenja na *cloud* infrastrukturu koristeći programske jezike, biblioteke, servise i alate koje inače koriste. Svrha je da organizacija ne održava i ne upravlja sa *cloud* infrastrukturom uključujući mrežu, servere, operativne sisteme, ili skladišta podataka. S druge strane, organizacija ima potpunu kontrolu i odgovorna je za softver koji je postavljen na cloud infrastrukturi uključujući konfiguraciona podešavanja za izvršavanje same aplikacije.

Ono što PkS donosi u odnosu IkS jeste automatizacija procesa izvršavanja distribuiranog servisa u *cloud* okruženju. Tačnije, PkS obuhvata i održavanje operativnih sistema, postojanje srednjeg sloja (engl. *middleware*) i okruženja izvršavanja (engl. *runtime*). Srednji sloj se obično brine o životnom ciklusu aplikacije tako što osvežava operativne sisteme, automatski dodeljuje unapred konfigurisan broj resursa, automatski vrši proceduru nadgradnje servisa kada se vrši postavka nove verzije itd. Okruženje izvršavanja je zavisno od izvornog koda servisa, ali u svakom slučaju, PkS vodi računa o tome da je okruženje izvršavanja uvek osposobljeno sa primenjenim najnovijim nadogradnjama.

3) Infrastuktura kao Servis (engl. *Infrastructure as a service (IaaS)*) – Organizacija može pokrenuti procesiranje, skladištenje, mrežu i ostale fundamentalne računarske resurse gde kasnije organizacija može postaviti i pokretati bilo koji softver, što može uključiti i odabir operativnog sistema i aplikacija. Sama organizacija ne upravlja niti kontroliše *cloud* infrastrukturu, ali ima kontrolu nad operativnim sistemom, skladištem kao i postavljenim aplikacijama. Zadržava mogućnost parcijalne kontrole nad mrežnim komponentama, npr. *firewall* konfiguracije.

Nivoi usluge mogu se logički posmatrati kao slojevita struktura (Slika 5), gde određene slojeve održava pružaoc usluga (zelena boja), a korisnik preuzima odgovornost za ostatak slojeva (plava boja). Prelaskom na restriktivniji model se dobija manja fleksibilnost upravljanja *cloud* okruženjem, ali i lakše održavanje okruženja jer se određen niz poslova prepušta pružaocu usluga. Sloj ispod je zadužen za održavanje okruženja na tom sloju apstrakcije.

Slika 5 oslikava navedenu situaciju, gde se može primera radi posmatrati PkS i IkS model. PkS model je restriktivniji od IkS modela jer u poređenju sa IkS modelom, nudi mnogo manji stepen upravljanja okruženjem. IkS model nudi mogućnost proizvoljnog upravljanja okruženjem i time nudi potpunu fleksibilnost za onoga ko konzumira usluge IkS modela. S druge strane, PkS model rasterećuje korisnika, jer ne mora ni da zna na koji način je infrastrukturna komponenta rešena. Koristeći PkS model, korisnik je skoncentrisan samo na izvršavanje distribuirane aplikacije, a infrastruktura je održavana od strane IkS sloja. Tačnije rečeno, korisnik PkS usluge uopšte ne mora da razmišlja na koji način je IkS konfigurisan. Pri testiranju implementacije koju predlaže ovaj rad, korišćeni su IkS i PkS modeli *cloud* usluge.



Slika 5 - Slojevi usluge u cloud okruženjima [6]

3.1.4. Motivacija nastanka i korišćenja

Motivacija nastanka *cloud* okruženja jeste efikasnije iskorišćenje resursa. Ključna tehnologija koja omogućava razvoj *cloud* okruženja je virtuelizacija. Virtuelizacija apstrakuje fizičku infrastrukturu. Mogućnost apstrakcije fizičkog sloja omogućava da u distribuiranom sistemu učestvuju računarski resursi koji nisu hardverski identični. Upravljivost i održavanje *cloud* okruženja ne zavisi od hardverskih komponenti, a ta osobina je postignuta virtuelizacijom. Apstrakcijom fizičkog sloja dobija se na potrebnoj agilnosti i ubrzanja kada su u pitanju promene u arhitekturi rešenja. Virtuelni serveri mogu da se alociraju dinamički. Operacije promene arhitekture rešenja su ubrzane jer je problem menjanja fizičke arhitekture transliran na upravljanje softverom, što je mnogo jednostavnije. Posao rukovanja računarskom konfiguracijom je virtuelizacijom doveden u softverski domen. S tim u vezi, sada je moguće uvesti automatizaciju u proces dodele resursa u zavisnosti od trenutnih potreba u sistemu.

Virtuelizacija resursa omogućava efikasnu iskorišćenost fizičkih resursa. Računar kao resurs je virtuelizovan u formi virtuelne mašine. Problem današnjih skupih hardverskih računarskih resursa je to što hardver nije iskorišćen u potpunosti. Virtuelizacija omogućava efikasnost resursa tako što je na jednom fizičkom računaru omogućeno izvršavanje više virtuelnih mašina. Potpunim iskorišćenjem hardvera, *cloud* sistemi dobijaju na efikasnosti. Efikasnost predstavlja dodatni razlog da se virtuelizacija vidi kao ključ za postojanje *cloud* sistema.

Osim potpunog iskorišćenja hardvera, postoji i stepen više, gde se koristi više resursa nego što postoji. Pružanje više resursa nego što je realno izvodljivo se naziva *overcommitment*, u *cloud* terminologiji. Više virtuelnih resursa izvedenih od određenog fizičkog resursa ukazuje na neminovno deljenje resursa i stalnu borbu za stvarne fizičke resurse. Što je više virtuelnih resursa izvedeno od jedne fizičke jedinice, veća je verovatnoća da može doći do izgladnjivanja resursa. Određenoj fizičkoj mašini se dodeljuje više virtuelnih mašina. Ukoliko bi došlo do izgladnjivanja

resursa u slučaju kada bi sve virtuelne mašine bile pod opterećenjem, radi se o *overcommitment* procesu. Glavni oslonac *overcommitment* procesa je činjenica da se mašine, pa čak i virtuelne, ne iskorišćavaju maksimalno jer mašine nisu pod konstantnim opterećenjem. *Overcommitment* je još jedan način, na koji je u *cloud* okruženju postignuta efikasna upotreba resursa i ušteda istih.

Dakle, ispravno je zaključiti da je u srži efikasnosti *cloud* okruženja u stvari višeorganizacijsko svojstvo, a efikasnost je motivator nastanka i daljeg korišćenja *cloud* okruženja. Na isti način, pri prelasku na SkS poslovni model, koji će biti detaljnije opisan u narednom odeljku, po analogiji sa *cloud* okruženjima, moglo bi se reći da je ključna osobina softvera za SkS model upravo višeorganizacijsko svojstvo.

3.2. Servisi

Kada su u pitanju IT servisi, podrazumevaće se servisi kakvi su definisani *Information Technology Infrastructure Library* (ITIL) skupom praksi za upravljanje IT infrastrukturama. Dovoljan nivo informacija se može pronaći u [48] i [49], a suštinski predstavlja okvir za pružanje efikasnih servisa opisan u pet knjiga:

1) Servisna strategija [50] – i iz ugla ITIL-a strategija zauzima najvažnije polazno mesto, čija važnost će takođe biti naglašena u poglavlju 3.3.
2) Servisni dizajn [51] – način na koji se definišu servisi u zavisnosti od strategije, tehnologije i raspoložive organizacije,
3) Servisna tranzicija [52], tj. uvođenje servisa u fazu operacije,
4) Servisne operacije [53] – način na koji se rukovodi servisom tako da krajnji korisnik dobije ono što je očekivano i dogovoreno uz neophodnu transparentnost,
5) Kontinualno unapređivanje servisa [54] – skup praksi da se razmišlja o tome i radi na konstantnom unapređenju servisa, kao neizostavna komponenta za uspešnost jedne servisne organizacije.

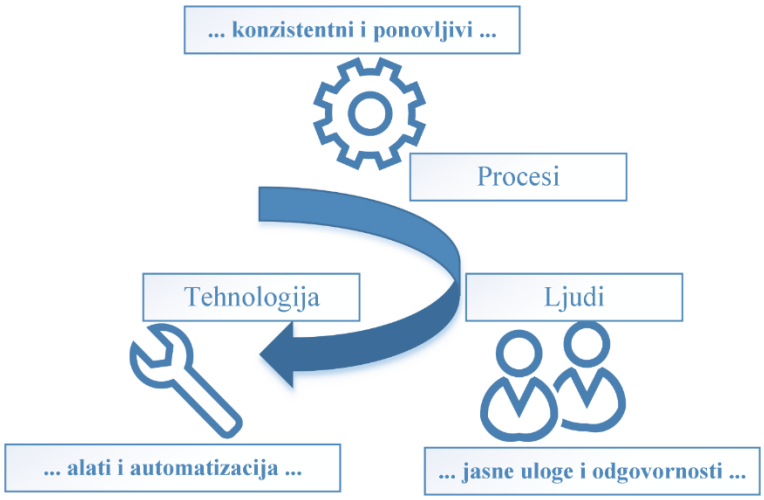
Najbitnije za izdvojiti iz ITIL-a u svrhu ove disertacije u uvođenja daljih pojmova na osnovu kojih će biti pojašnjen i SkS jeste definicija servisa:

Servis je način dostavljanja vrednosti krajnjem korisniku pružajući vrednosti koje krajnji korisnik očekuje, ali bez posedovanja rizika i asociranih troškova za samog krajnjeg korisnika.

Sama definicija se može proširiti još jednim konceptom, a to je da za pružanje svakog servisa je neminovno imati tehnologiju, procese i ljude (Slika 6), gde je smisao svake celine sledeći:

1) Tehnologija – u slučaju ove disertacije, reč je o softveru, ali servis ne mora da se pruža nad tehnologijom zasnovanom na softveru. Kod softvera je mogućnost automatizacije velika, i to je ono što čini dobar servis jer doprinosi efikasnosti. Ako se uzme *cloud* servis za primer pojašnjenja, angažovanje *cloud* resursa je potpuno automatizovano, na čemu se mogu ostvariti uštede u odnosu na manuelno konfigurisanje što je slučaj u tradicionalnim infrastrukturnim okruženjima.
2) Procesi – mogu se smatrati kao recepti, u smislu kako određene uloge postupaju sa tehnologijom u cilju ostvarenja neke akcije. Ideja je da budu ponovljivi kako bi se mogli skalirati i konzistentni.

3) Ljudi – u svakom servisu, u nekoj meri su neophodni ljudi. Zbog cene servisa, tendencija je da se ljudski manuelni posao automatizuje gde god je to moguće. Ljudi i procesi su karakteristika koja ukazuje na potrebe za organizacionim promenama tamo gde je plan da se uvodi SkS.



Slika 6 - ITIL Servis

Kada se pogleda definicija servisa, jasno je da sam softverski servis ne mora biti baziran na *cloud* okruženju, ali će u narednim poglavljima biti pojašnjeno zašto je često vezivano jedno za drugo, i u čemu su razlike. Na taj način, može se napraviti i tradicionalna isporuka softverskih rešenja uz dodatan servis koji bi omogućio za krajnjeg korisnika slično iskustvo kao što ima u SkS slučaju, iako postoje određene razlike. Najveće razlike se odnose upravo na mogućnosti deljenja resursa, tj. svrsi višeorganizacijske osobine koja dovodi do finalne efikasnosti servisa što će biti fokus ove disertacije. Različiti poslovni modeli će biti pojašnjeni u okviru poglavlja 3.3.4.

3.3. SkS poslovni model

Potrebno je pojasniti šta predstavlja SkS poslovni model, i zašto je često uziman kao neminovan spoj sa *cloud* baziranim tehnološkim rešenjima. Dalje, razumevanje šta sve promena sa poslovnog modela isporuke rešenja na SkS poslovni model podrazumeva je neophodan preduslov kako bi se ostatak ove disertacije mogao razumeti. Na osnovu opisa poslovnih modela u 3.3.1, biće prodiskutovan poslovni model SkS u 3.3.2 da bi se u 3.3.3 opisala potreba za SkS, što je ujedno i motivacija koja izaziva potrebu za ovom disertacijom. U 3.3.4 je prodiskutovan način tranzicije ka SkS iz ugla poslovnog modela, da bi u se u 3.3.5 pružio fokus na *cloud* bazirani SkS, koji je u fokusu ove disertacije.

3.3.1. Poslovni modeli

Iako se SkS koristi da označi tip modernih tehnoloških rešenja koja se baziraju na SkS poslovnom modelu, u užem smislu i pravom značenju SkS, SkS je u stvari poslovni model. S obzirom na to da je SkS poslovni model, izučena je odgovarajuća oblast koja se bavi poslovnim modelima. Nakon pregleda referentne literature detaljnije opisane u poglavlju pregleda aktuelnog stanja u oblasti [9]–[11], a koja se bavi analizom i dizajnom poslovnih modela, kao i pretragom literature, uviđa se da se Ostervalderovo platno nameće kao osnova za predstavljanje i

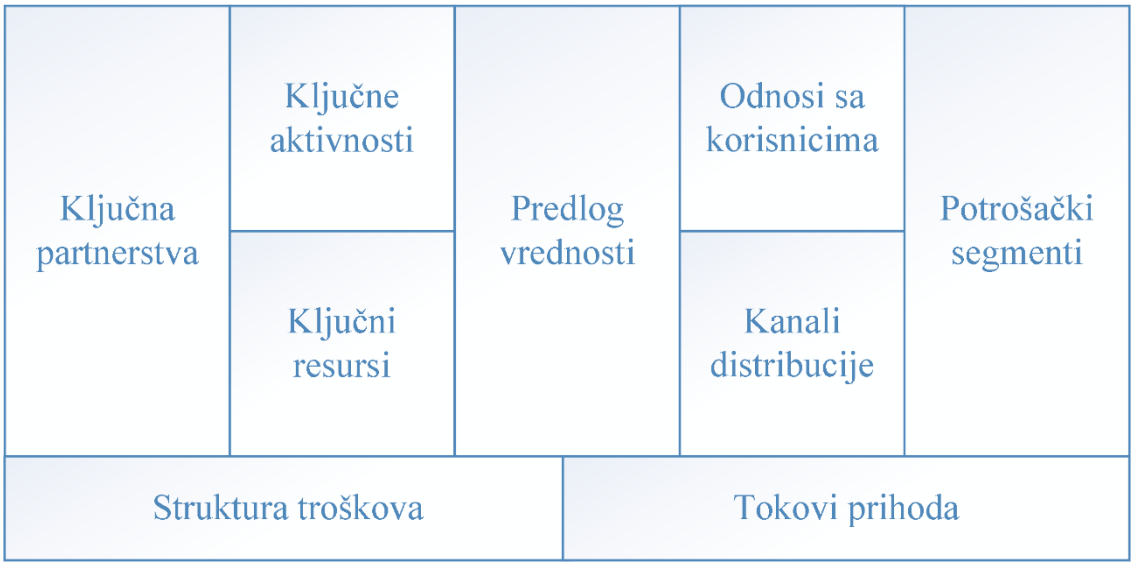
razumevanje poslovnih modela. Dodatno, Ostervalderovo platno je korišćeno i kao osnova za objašnjenje razlika SkS poslovnog modela u doktorskoj disertaciji [15] na temu analize SkS poslovnog modela. Te činjenice su dodatno osnažile pristup da se koristi Ostervalderov pristup kao osnovica analize potencijalnog SkS modela i poređenja SkS modela sa modelom tradicionalne isporuke rešenja. Ostervalderovo platno [8] se može predstaviti slikovito (Slika 7), a sastoji se iz sledećih gradivnih celina:

1. predlog vrednosti (engl. *value proposition*) – opis proizvoda i usluga koje odgovaraju određenim potrošačkim vrednostima. Predlog vrednosti je ključni gradivni blok za analizu prednosti i mana SkS poslovnog modela, i kao takav će biti detaljnije obrađen u narednim poglavljima.
2. ključni resursi (engl. *key resources*) – opisuje sve resurse, opipljive (ljudski, fizički, i finansijski resursi) i neopipljive (intelektualna svojina, patenti) neophodne za rad organizacije da funkcioniše u skladu sa ciljevima.
3. ključna partnerstva (engl. *key partnerships*) – opis lanca dobavljača kao i svih partnera neophodnih za izvršavanja poslovnog modela. U slučaju *cloud* baziranih SkS, u ključna partnerstva po pravilu spadaju pružaoci *cloud* usluga.
4. ključne aktivnosti (engl. *key activities*) – aktivnosti koje su neophodne da se kreiraju i ponude predlozi vrednosti (1). One zavise od poslovnog modela. U slučaju tradicionalne isporuke softverskih rešenja, jedna od ključnih aktivnosti je razvoj softvera.U SkS postoje dodatne ključne aktivnosti koje se odnose na isporuku usluga. Ključne aktivnosti u slučaju isporuke usluga bi se odnosile na dalju razradu strukture organizacije, ali je to u užem smislu je van opsega ovog istraživanja.
5. potrošački segmenti (engl. *customer segments*) – veoma bitna komponenta, jer od zahteva i želja potrošača se direktno kreiraju predlozi vrednosti (1). Važno je istaći da postoji više tipova potrošačkih segmenata: masovno tržište sa fokusom na široko tržište sa sličnim potrebama i problemima, tržište uske niše, segmentisano tržište gde se različita tržišta razlikuju u malim razlikama, diversifikovano gde se snabdevaju totalno različite potrošačke grupe i višestruke platforme gde organizacije moraju da povežu različite potrošačke segmente da bi dobile krajnju vrednost. Horizontalni SkS modeli se obično oslanjaju na kombinaciju masovnog tržišta, ali sa specifičnom namenom, tj. tržište uske niše. Problem nastaje kada je reč o SkS modelu koji nije namenjen za masovno tržište, jer se tada prelazi iz standardizacije rešenja u prilagođavanje, kao u slučaju softvera za elektroenergetske mreže, što izaziva poteškoće i predstavljaće jedan od problema pri izvedbi ove disertacije.
6. kanali distribucije (engl. *channels*) – način na koje se predložene vrednosti dostavljaju potrošačima. Bitno je izdvojiti da postoje direktni i indirektni putevi, kao i korišćenje svojih distributivnih mreža, ali i partnerskih. Ovde je korisno primetiti da u *cloud* baziranim SkS, *cloud* kompanije obično nude i svoje postojeće kanale za distribuciju softvera jer uglavnom poseduju lokalne partnere širom sveta.
7. odnosi sa korisnicima (engl. *customer relationships*) – obuhvata sve veze koje organizacija pokušava da uspostavi sa svojim klijentima. Za razliku od tradicionalne isporuke, u SkS je potrebno da postoji prisnija veza i više nivoa veza sa klijentima iz razloga što u pružanju usluga je potrebno dobijati češće povratne informacije, kao i voditi klijente u smeru

optimalne upotrebe softverskih usluga, kroz praćenje navika korišćenja. Dostavljanje usluga i u ovom gradivnom bloku zahteva unapređenja nad tehnologijom.

8. struktura troškova (engl. *cost structure*) – predstavlja pregled najvažnijih troškova nastalih obavljanjem poslovanja. I ova gradivna jedinica je bitna da se sagleda, jer u slučaju *cloud* baziranih SkS, gde određena, veoma primetna stavka strukture troškova se pripisuje plaćanju *cloud* usluga, kao i direktnu potrebu za istraživanjem višeorganizacijskog svojstva, ne bi li se ostvarile znatne uštede. Ovo je takođe primer da je odnos tehnološke i poslovne transformacije zatvoren krug jer izazvano poslovnim potrebama, istražuje se osobina višeorganizacijskog servisa, koji će za uzvrat neminovno prouzrokovati promenu poslovnog modela. Iz ovih razloga i postoje oprečna istraživanja, gde neka kreću od verovanja da tehnološka transformacija vodi poslovnu, do onih gde je osnovna premisa da je glavna ulazna tačka u transformaciju, bilo tehnička, bilo organizaciona, izazvana poslovnim modelom. Takvu premisu ima i ova disertacija.
9. tokovi prihoda (engl. *revenue streams*) – predstavlja pregled najvažnijih prihoda, gde se grubo mogu izdvojiti transakcioni i periodični. Već na osnovu ove podele naslućuje se velika razlika u modelu prodaje sofvera i SkS. U prodaji softvera se dobit ostvaruje transakciono sa mnogo manjim periodičnim prihodima zasnovanim na održavanju. U SkS su tokovi prihoda striktno periodični, što utiče na to kako će se naplaćivati SkS, na strategiju vezanu za zadovoljstvo krajnjih korisnika itd. Iako postoji čitava komponenta za određivanje naplate, ona neće biti fokus ove disertacije. U svakom slučaju, mora se i sagledati jer često različiti modeli naplate zahtevaju tehnološku podršku za merenje potrošnje između različitih organizacija koje predstavljaju krajnje potrošače servisa u SkS modelu.

Navedene gradivne celine obuhvataju i oblasti za uspešnu strukturu organizacije (praktično od 2 do 9) koja treba da odgovori potrebama datog poslovnog modela. Može se reći da je predlog vrednosti najvažnija stavka za određivanje strategije ogranizacije, a samim tim i poslovnog modela, jer su predložene vrednosti direktno spregnute sa tim šta je potreba korisnika kao i sa tim šta je ono što će organizaciju učiniti uspešnijom na tržištu.



Slika 7 - Ostervalderov poslovni model

Strategija se pojavljuje u svim izučavanim modelima organizacije [55]–[59]. Strategija se posmatra i kao deo organizacije, upravo zbog svoje važnosti i prirode koja diktira strategiju kao najvažniji ulazni parametar u projektovanju jedne organizacije. Iako je jako važno razumeti tržište i industriju koja je predmet strategije, na strategiju će uticati i interne vrednosti organizacije. Može se reći da se strategija nalazi između tržišta i internih struktura organizacije, te se može posmatrati kao izlazna vrednost na koju utiču faktori okruženja, ali i na nezavisnu varijablu koja utiče na organizacioni sistem. Iz tog razloga se razlikuju korporativna i poslovna strategija.

Korporativna strategija se odnosi na način na koji organizacija pokušava da maksimizuje vrednost resursa koje kontroliše, dok se poslovna strategija tiče odluke kako se takmičiti na definisanom tržištu. Detaljnije o poslovnoj strategiji i njenom kreiranju, a posebno iz ugla kreiranja SkS je opisano u 3.3.3. Za sada je neophodno iskazati da je strategija najvažniji faktor koji određuje pravac kretanja jedne organizacije pa samim tim i njenu strukturu.

Referentni radovi koji su se bavili organizacijom uglavnom uzimaju strategiju kao jedan od najvažnijih elemenata, čak kao i ulaznu varijablu za projektovanje organizacije. U [55] je predstavljen isključivo dizajn organizacije. Šta je potrebno da bi se organizacija projektovala kako treba. Postoji strateški vrh kao deo organizacione strukture, ali generalno nije reč o tome kako se dolazi do efikasne strategije. Takođe, 7S model [56] se bavi projektovanjem organizacije ali se strategija prepoznaje kao jedan od elemenata koji mora biti usklađen. Daftov model [57] najslabije prepoznaje strategiju i njenu važnost, iako postoji kao dimenzija namenjena projektovanju organizacije. Međutim, u ovom predlogu modela projekcije organizacije, strategija se čini da ne dobija centralnu ulogu pokretača projektovanja organizacije. S druge strane, u Simonsovom modelu [58], poslovna strategija predstavlja jedan od osnovnih elemenata organizacije, ima centralnu poziciju i predstavlja polaznu tačku u projektovanju organizacije. Isto tako, noviji, Galbrajtov model, stavlja stavlja strategiju kao glavnu ulaznu tačku u proces kreiranja organizacije [59].

Sagledavanjem relevantnih istraživanja vezano za projektovanje organizacija, jasno je da promena strategije može da zahteva, i utiče na promenu same organizacije. Odlukom da se implementira novi poslovni model kao što je SkS automatski podrazumeva sledeće segmente:

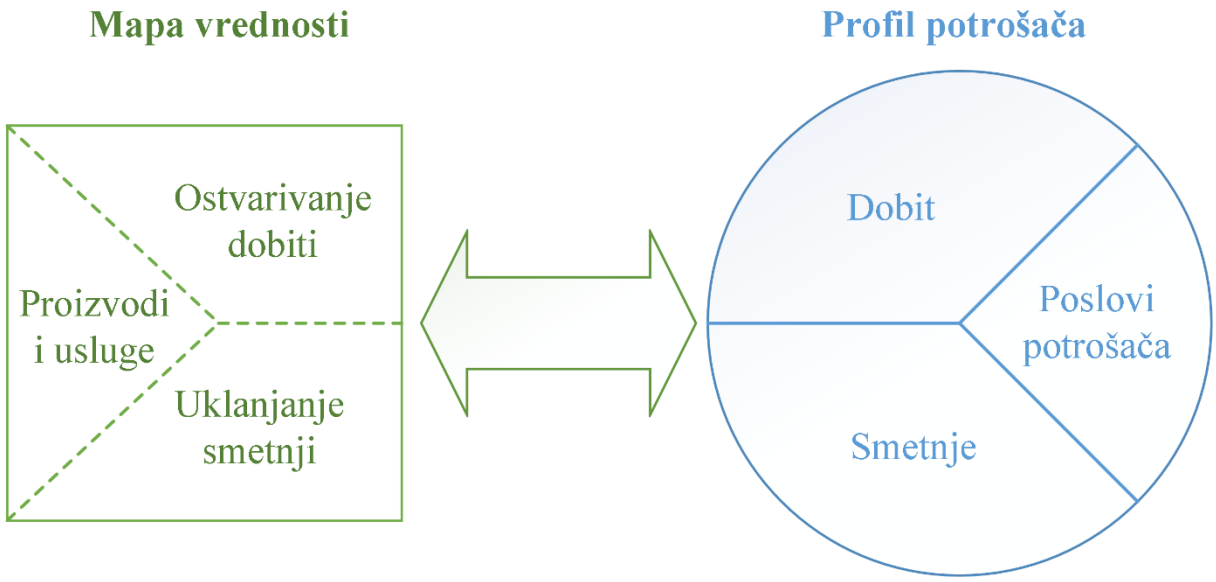
1. transformaciju poslovnog modela – tačnije, kreira se novi poslovni model što je SkS,
2. izazvano prvim segmentom, dolazi do transformacije organizacije. Zbog svoje kompleksnosti, i netehničke prirode, transformacija organizacije nije predmet ove disertacije. Iako se na nekim mestima u tekstu može naići na pojašnjenja zbog čega se zahtevaju određene organizacione promene, tema i fokus se zadržavaju na tehničkoj transformaciji, tj. trećim segmentom,
3. tehnološka-tehnička transformacija podrazumeva modifikaciju postojećih softverskih rešenja tako da odgovaraju promenama izazvanim poslovnom modelu, ali i modifikacije koje će omogućiti (barem u finansijskom smislu) da novi poslovni model bude funkcionalan za organizaciju.

Tehnička transformacija je fokus ove disertacije, ali da bi se njeni motivi mogli razumeti, neophodno je dati širi kontekst, koji podrazumeva objašnjenje potreba za transformacijom i na ostalim nivoima. Ipak, da bi se fokus zadržao na tehničkoj transformaciji, u narednom poglavlju

će biti obrađen samo gradivni blok predloženih vrednosti po Ostervalderu, jer upravo taj gradivni blok se preslikava na strategiju koja će direktno ukazati na potrebe za tehničkim transformacijama, kao i na to zašto dolazi do veoma čestog vezivanja SkS pojma za *cloud* bazirane usluge.

3.3.2. Predlog vrednosti za SkS

Izgradnja vrednosti ponude je opisana u [60] gde je autor ponovo Ostervalder. Takođe je napravljen slikoviti sistem (Slika 8) koji bi trebao da pomogne da se sistematično razume šta krajnji potrošači žele kako bi se napravili proizvodi koji odgovaraju na potrebe. Ideja je u skupljanju informacija na jednostavan način koje će pomoći da se kreira poslovni model. Na kraju, pravi poslovni model će voditi do profitabilnosti a razvoj funkcionalnosti koje neće biti neophodne potrošačima će dovesti do velikih troškova.



Slika 8 - Kreiranje predloga vrednosti [60]

S obzirom da praviti poslovni model u opštem slučaju nije moguće, ovde treba podrazumevati tržište opisano u odeljku 1.4 gde je opisana primena, a to podrazumeva softver namenjen industriji kao kupcu, gde se softverske instalacije protežu na velik broj virtuelnih mašina i održavanje takvog rešenja je teško i zahteva ekspertizu kakva postoji samo u kompaniji koja je razvila rešenje.

Pitanje koje se postavlja jeste koje poslove krajnji potrošač želi da uradi. Poslovi su nizovi zadataka koji moraju da se urade i zbog kojih se krajnji potrošač može obratiti dobavljaču za pomoć. U hipotetičkom slučaju koji analiziramo, poslovi bi mogli biti definisani službom koja se bavi održavanjem samog sistema.

Primeri poslova su: monitoring sistema, istraživanje problema u radu, rešavanje incidenata, vođenje dnevnika promena na sistemu, primena softverskih zakrpi i nadogranja sistema, upravljanje virtuelnim mašinama, komunikacija sa ostalim dobavljačima u lancu dobavljača, vođenje računa o sigurnosnim aspektima rešenja kao što su anti-virus definicije i pravila, bezbednosne zakrpe operativnih sistema, definicija i upravljanje pravima pristupa, razrešavanje problema u slučaju komprovitovanog sistema. Uz sve ovo, postoji i aspekt kontinualnog

unapređenja koji se dobija efikasnim evidentiranjem naučenih lekcija i sistematičnim sprovođenjem novousvojenog znanja.

Kada su opisani poslovi koji su krajnjem potrošaču neophodni da radi, sledeći korak je da se opiše šta je ono što je smetnja krajnjim potrošačima. Pod smetnjom se podrazumeva onaj deo posla koji predstavlja dodatni posao koji krajnjem potrošaču ne donosi dodatnu vrednost niti mu odgovara na bilo koji način. Tačnije, ono što krajnji potrošač doživljava kao negativnu kolateralnu dobit. Ovo mogu biti bočni efekti koji donose negativna iskustva. U hipotetičkom slučaju sagledavanja prelaska na SkS, smetnja potrošačima su sledeće: potreba za razvijanjem nove kompetencije, povećani trošak, veliki rizici, kašnjenje projekata, a ostvarenje rizika dovodi do negativnih emocija i atmosfere.

Krajnje pitanje nakon sagledavanje poslova i smetnji, jeste šta je dobit krajnjeg potrošača. Dobit je bilo koji pozitivni ishod, nešto što se očekuje ali i što bi bilo iznenađujuće. U dobiti mogu spadati uštede, lakoća korišćenja, usluga, saveti i udobnost rada sa krajnjim rešenjem.

Vidi se da SkS na svakom polju, tako što pružalac usluga preuzima sve navedene poslove može učiniti da dođe do zadovoljstva kod krajnjeg korisnika, ali i više od toga. U SkS poslovnom modelu je za razliku od tradicionalne dostave softvera veza između krajnjih potrošača i proizvođača prisnija i komunikacija je češća, što dovodi do toga da se predložene prednosti mogu konstantno evaluirati. Takođe, osim evaluacije prednosti direktnom komunikacijom sa krajnjim potrošačima, s obzirom da je pružalac usluge u nadležnosti tehnologije, može se uraditi analiza upotreba i na tehnološkom nivou, što je velika prednost kod SkS poslovnog modela.

3.3.3. Potreba za promenom poslovnog modela

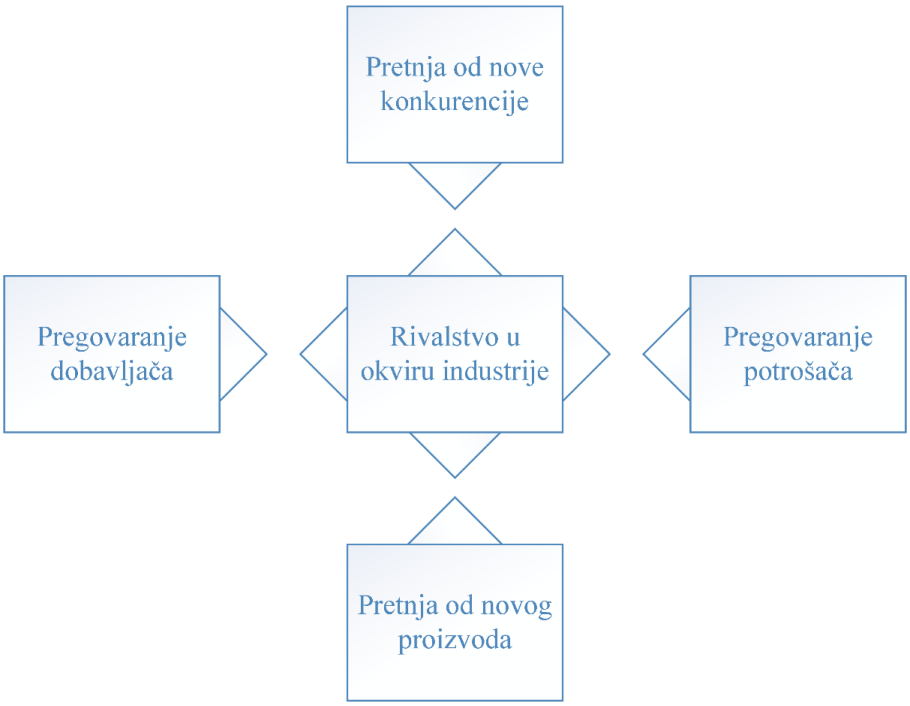
Usponi i padovi organizacija na tržištu su često izazvani poslovnim modelom, gde se mogu ponuditi čak i isti proizvodi ali sa različitim poslovnim modelom. U prethodnom poglavlju su opisani predlozi vrednosti, koji su zapravo glavni uzrok da se krajnji potrošači odluče na promenu. Osim potreba za promenom koje proističu iz razlika između poslovnih modela opisanih u prethodnom poglavlju, neophodno je razumeti i pravu motivaciju za promenom.

Porter [61] je opisao u svom modelu pet kompetitivnih sila koje mogu influencirati atraktivnost određene industrije. Razlog za promenom poslovnog modela je ujedno pokretač potrebe za transformacijom, a razlozi se mogu sagledati na visokom nivou putem Porterovih kompetitivnih sila (Slika 9).

Pet kompetitivnih sila vode strukturu profita jedne industrije tako što određuju ekonomsku vrednost za svaku od sila. Vrednost profita može naravno biti raspodeljena između postojećih konkurencija, ali takođe može biti smanjena kroz diktiranje cene od strane potrošača koji su dobri u pregovaranju, diktiranja cene od strane dobavljača, ili biti ugrožena zbog pretnje nove konkurencije koja pokušava da otvori tržište kao i proizvoda koji mogu zameniti nuđeni proizvod.

Strategija može biti viđena kao izgradnja odbrane uzimajući u obzir svih pet sila, ili tražeći poziciju gde su sile slabije. Uzimajući u obzir sve sile, Porter objašnjava kako industrije sa naglim rastom nisu uvek profitabilne, kako eliminacija današnjih suparničkih organizacija kroz akvizicije može smanjiti potencijal za profit, kako vladine regulative igraju ulogu tako što utiču na relativnu snagu sila i kako da se sile iskoriste kako bi se razumele određene prednosti. Razumevanjem sila,

može se oformiti strategija koja uzima u obzir uslove industrije u kojoj se proizvod ili usluga nalaze.



Slika 9 - Porterove potrebe za promenom poslovnog modela [15]

Analizom sila, može se zaključiti da li je transformacija na SkS potrebna, na svakoj od mogućih sila:

1) Pretnja od nove konkurencije – ukoliko se ne uvede SkS poslovni model, to je čitava oblast na kojoj postoji velika verovatnoća da će se pojaviti konkurencija, posebno u vreme dominacije *cloud* okruženja.
2) Pregovaranje dobavljača – kao najveći dobavljač u SkS poslovnom modelu, mogu se svrstati *cloud* okruženja. Dobra stvar za Porterovu silu je što je ovo tržište već dobro definisano i opcije postoje.
3) Pregovaranje potrošača – uvođenjem SkS, pružaoc servisa je u odličnoj poziciji za pregovaranje jer ima višestruku ponudu. Ukoliko potrošači ne cene dovoljno SkS, uvek postoji ponuda tradicionalne isporuke rešenja.
4) Pretnja od novog proizvoda – uvođenjem SkS, slušanje samih potrošača je frekventnije i obavezno, tako da rizik od pojave novog proizvoda koji će zameniti postojeći se smanjuje.
5) Rivalstvo u okviru industrije – u slučaju da je kompanija koja prva uvodi SkS, stavlja se u poziciju inovatora i lidera u oblasti. U slučaju da već postoji konkurencija sa SkS modelom, to je verovatno znak da bi se trebalo što pre odgovoriti uvođenjem SkS

Saglevanjem svih pet sila iz ugla uvođenja SkS, jasno je da postoje samo benefiti. Naravno, trošak transformacije i mogućnosti za transformaciju je nešto što predstavlja eventualne negativne posledice za uvođenje SkS, ali cilj ovog odeljka je da prikaže motivaciju za transformacijom. Pod mogućnošću za transformaciju, ne misli se samo na tehnološku, nego i na politički aspekt gde

razne regulative možda otežavaju uvođenje SkS modela jer ipak krajnja kontrola nad softverom i podacima uvek ostaje na strani pružaoca usluga.

3.3.4. Od tradicionalnog modela do SkS

Da bi se povukla krajnja jasnija razlika između poslovnih modela, kao i prikazao paradoks koji dovodi do zabune, različiti modeli od tradicionalnog do SkS su poređeni po svom logičnom sledu (Tabela 1). Tačnije, tranzicija od tradicionalnih servisa ka SkS bi mogla da se dogodi u smeru kada se tabela čita od levo ka desno. Čitajući tabelu, jasno je da u tradicionalnoj isporuci rešenja, krajnji korisnik mora da investira u infrastrukturu, licencu i da plaća održavanje softvera. Uz sve to, mora da investira u razvoj internih veština i da zaposli ljude koji se bave operacijama novog softvera. U tom slučaju, skroz sami nose rizik, ali je sve to nagrađeno potpunom kontrolom nad podacima i samim softverom.

Softver i servisi predstavlja model nadogradnje nad tradicionalnom isporukom rešenja, gde je ideja da se proizvođač softvera ponudi servis operacija nad softverom. U ovom slučaju, praktično sve ostaje kao i u tradicionalnom slučaju, s tim da procesi i ljudi organizacije koja je proizvela softver se bave operacijama, opisanim kao spisak poslova u poglavlju 3.3.2. Na ovaj način nastaju servisi kako su opisani 3.2. Ideja je da je rizik prebačen na pružaoca servisa, a kontrola i bezbednost ipak nisu skroz prepušteni pružaocu servisa jer se i dalje fizički nalaze na istom mestu, tj. kod korisnika, pa se određenim kontrolama bezbednosti korisnik može osigurati da se bezbednost neće narušiti.

Tabela 1 - Modeli dostave softvera iz korisničkog ugla

Model	Tradicionalna isporuka softvera	Softver i servisi	SkS	Cloud SkS
Investicija u infrastrukturu	X	X		
Investicija u licencu	X	X		
Troškovi održavanja	X	X		
Investicija u interne veštine	X			
Razvoj sopstvenih procesa	X			
Dostupnost	Interna mreža	Interna mreža	Interna mreža	Internet
Kontrola softvera	X	Parcijalno	Parcijalno	
Kontrola podataka podataka	X	Parcijalno	Parcijalno	
Rizik	X			

Ukoliko bi se model softvera i servisa ponudio tako da se infrastruktura, licenca i troškovi održavanja ponude kao ponavljajući redovan trošak korisniku, a ne kao transakcioni trošak, moglo bi se reći da je reč o SkS. *Cloud* bazirani SkS bi zahtevao da se softverski servisi dostavljaju putem interneta, jer je *cloud* udaljena lokacija u odnosu na klijenta, i podrazumevao bi prepuštanje kontrole nad softverom i podacima u potpunosti pružaocu usluga. Ovo prepuštanje kontrole predstavlja ogromnu, nekada nepremostivu, razliku između poslovnih modela jer ukazuje na isključivo poverenje u pružaoca usluge. Da bi se steklo takvo poverenje, potrebno je više faktora, a to su jačina brenda, zastupljenost takve usluge u industriji, regulative, politički odnosi, ali i razne sertifikacije organizacije koje potvrđuju da organizacija implementira određene procese po određenim standardima.

Ukoliko je procena da krajnji potrošači nisu spremni u određenoj industriji da daju taj nivo poverenja da se odluče na *cloud* bazirani SkS, tranzicioni put bi mogao da podrazumeva pružanje prvo servisnih usluga uz softver, potom SkS model bez prelaska na *cloud* a finalno migraciju na *cloud* bazirani SkS. Prednost ovog tranzicionog puta leži u tome što se razvijeni servisi mogu upotrebiti u svim modelima isporuke softvera osim u slučaju tradicionalnog.

Migracija na *cloud* omogućava prednosti, kao što su višeorganizacijsko svojstvo, ali zato i unosi niz ograničenja koji se moraju sagledati. Iz tog razloga, vrlo je verovatno da se SkS može isplativije ponuditi u *cloud* izvedbi nego bez *cloud* okruženja. Još jedna krucijalna distinkcija jeste u mogućem otkazivanju servisa od strane krajnjeg potrošača. U slučaju SkS, iako su troškovi podeljeni na određene intervale, u slučaju otkaza tu investiciju u infrastrukturu treba nekako premostiti, dok je *cloud* prirodno elastično okruženje. Iz ovog razloga, SkS koji nije baziran na *cloud* opciji, često zahteva ugovorne obaveze na duži period što utiče na to da se smatra istinitim SkS modelom. Ovo su najčešći razlozi zašto se u užem smislu SkS *cloud* okruženje podrazumeva, a u narednom poglavlju je dat detaljniji pregled.

3.3.5. SkS kao *cloud* bazirani servis

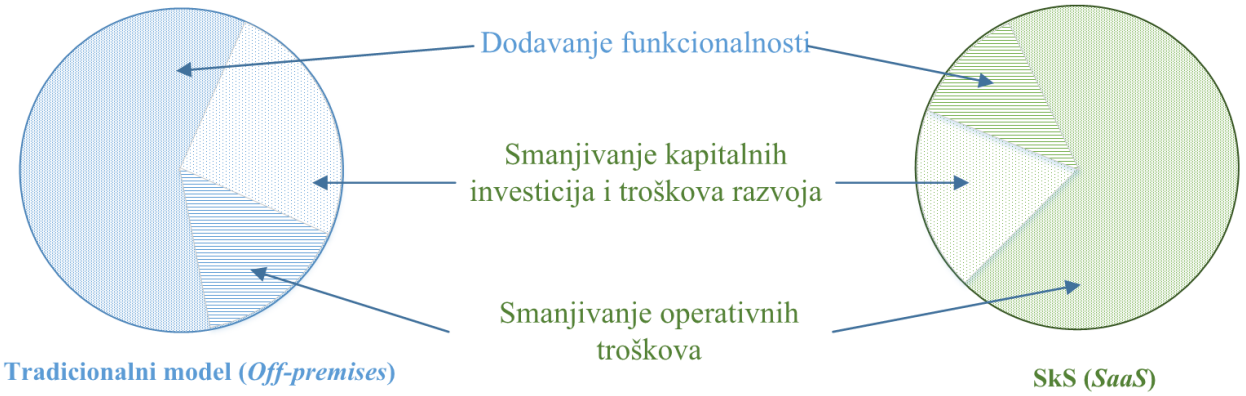
U užem smislu reči SkS, obično se misli na *cloud* bazirani tehnološki servis. U literaturi se čak može pronaći i kao definicija SkS što suštinski nije tačno. U tom smslu, u ovom odeljku je reč o sve zastupljenijem pojmu SkS, koji se odnosi na *cloud* bazirani servis, ali s obzirom na nesveobuhvatnost takvog pojma, uvek bi se prvo trebalo proveriti o čemu je reč kada se spominje SkS.

SkS je moderan pristup za dostavljanje velikog i skalabilnog softvera kao servis putem interneta [31]. Postoje mnogi tehnički izazovi vezani za razvoj SkS servisa. Jedan od izazova je svakako višeorganizacijski pristup, koji bi omogućio da jedna instanca softvera bude korišćena od strane više različitih organizacija. Idealno, individualni zahtevi različitih organizacija su izmešteni u konfiguraciju. Ovde se polako uvodi pojam standardizacije rešenja gde se svaka pojedinost pokriva konfigurabilnošću. SkS tehnička arhitektura koja uključuje višeorganizacijsku osobinu implicitno zahteva konfiguraciju i određen nivo prilagodljivosti aplikacije.

Kao razlog nastanka SkS modela servisa možemo uzeti evidentan napredak Interneta. Internet je omogućio da se vlasništvo i vođenje računa o aplikaciji izmesti tako da bude u nadležnost pružaoca servisa [62]. Odgovornost nad servisom je jasno naznačena time što se pružaoc servisa nalazi u nadležnosti održavanja, a tehnološki baziran servis je postavljen uglavnom na udaljenoj lokaciji, što je podstaknuto sve većim razvojem *cloud* okruženja.

Slika 10 ilustruje različite ciljeve tradicionalnog softvera i SkS softverskog modela. Tradicionalni softver se najčešće izvršava kod klijenta, i licence predstavljaju mehanizam naplate.

Kod SkS softverskog modela, za održavanje je zadužen pružaoc softverske usluge kao servisa. Prioriteti za dizajn i razvoj SkS servisa se u mnogome razlikuju od prioriteta kod razvijanja tradicionalnog softvera. Fokus kod tradicionalnog softvera je generalno dodavanje funkcionalnosti, koje su obično produkt klijentskih zahteva. Prilagođavanje klijentskim zahtevima posledično utiče na povećanje ukupnih troškova aplikacije. U suprotnosti s tradicionalnim pristupom, fokus SkS aplikacija je generalno na smanjenju ukupnog troška servisa, ali po cenu ređeg i znatno smanjenog dodavanja novih funkcionalnosti [62]. Ovo predstavlja još jedan od razloga zašto SkS model zahteva zrelo tržište, kao i mogućnost da standardizovano rešenje zadovolji većinu korisnika.



Slika 10 - Ciljevi pri razvoju [62]

Kao tranzicioni korak, kod kompleksnih rešenja koja su predmet ovog istraživanja, mogu se gledati različiti podsistemi, gde bi se neki podsistemi mogli standardizovati i ponuditi u višeorganizacijskoj konstrukciji, a visoko prilagodljivi delovi ostavili da svakom krajnjem korisniku odgovara posebna instanca tog dela sistema.

Prelaskom na jednu bazu izvornog koda, na osnovu promene funkcionalnosti u slučaju jednog korisnika, automatski i istovremeno funkcionalnost se menja i kod ostalih korisnika servisa. Eleminiše se potreba primenjivanja istih unapređenja ili popravki na više mesta. Da bi servis podržavao prilagodljivost na osnovu konfiguracije, često je najlakše da se izvrši ponovno projektovanje arhitekture aplikacije. Posebno ako je u pitanju tradicionalna monolitna aplikacija čija se prilagodljivost zasniva na menjanju izvornog koda aplikacije.

Model SkS aplikacija podrazumeva da se klasični model plaćanja softverskih usluga sa licenciranja prebacuje na pretplate (engl. *subscription*) [31]. Korisnici servisa se pretplaćuju na korišćenje servisa i kontrola pristupa SkS servisu je na ovaj način centralizovana, i nalazi se u samoj logici servisa.

SkS vizija pokušava da napravi okruženje u kom će korisnici moći da iznajmljuju prilagođen softver kao što to čine sa strujom ili vodom. Očekuje se da će ovakav način iznajmljivanja softvera dovesti do opadanja cene informacionih tehnologija [63]. Stoga je čest model pretplate plaćanje usluge servisa po korišćenju (engl. *pay per use*), iako postoje i drugi modeli, kao što su [14] iznajmiljavnje softvera i licenciranje softvera. Modeli pretplate predstavljaju ulazni parametar za kreiranje pravog prodajnog i naplatnog modela.

Korisnik koji je pretplaćen, koristi servise tako što koristi tanke klijente u vidu *web* pretraživača ili posebno napisanih klijentskih aplikacija koje koriste udaljene servise. SkS model je sve zastupljeniji, i postoji tendencija migracije postojećih rešenja informacionih tehnologija na SkS model [62].

SkS model nastaje kao novi trend oslanjajući se na *cloud* okruženja i napretkom interneta. Napredak interneta se može smatrati potrebnim i dovoljnim uslovom za postojanje SkS modela jer SkS kao koncept nije moguć bez komunikacije klijenta i udaljenog centra na kom se tehnološki bazirani servis izvršava. Isto tako, spora konekcija bi dovela do lošeg iskustva u korišćenju servisa što bi dovelo do neuspeha SkS modela. *Cloud* okruženje i sve češći pružaoci *cloud* usluga omogućavaju manjim i srednjim organizacijama da iznajme potrebnu infrastrukturu za razvoj SkS model servisa. Na ovaj način organizacije ne moraju ulagati u skup distribuirani sistem. *Cloud* pružaoci usluga su omogućili i podstakli razvoj SkS aplikacija zbog mogućnosti dinamičkog iznajmljivanja resursa. Moguće je skaliranje čitavog sistema po potrebi, što utiče direktno na efikasnost finansija.

Skalabilnost aplikacije i njena elastičnost su dva kvaliteta koja izdvajaju SkS model i predstavljaju prednost u odnosu na tradicionalni model. *Cloud* omogućava i skalabilnost i elastičnost, ali nije dovoljno da se servisi samo izvršavaju u *cloud* okruženju. Potrebno je da servisi budu spremni za *cloud* (engl. *cloud-ready*). Pod spremanjem servisa se podrazumeva da arhitektura servisa bude distribuirana, sa jasno definisanim celinama čije se instance mogu menjati po broju. Tehnički gledano, *SOA* jeste način za pisanje aplikacija spremnih za *cloud*. Skalabilnost aplikacije je ključna, jer u SkS modelu, isplativost se ostvaruje ukoliko je broj klijenata veoma velik.

Cilj SkS aplikacije jeste sniženje cene za krajnjeg korisnika, uz korišćenje *cloud* okruženja. Da bi sniženje cene bilo moguće, potrebno je da postoji velik broj korisnika servisa. Zbog očekivanja velikog broja korisnika, skalabilnost i elasticitet aplikacije su dve veoma bitne osobine SkS modela.

Uvažavajući tranzicioni put od tradicionalne isporuke rešenja do *cloud* baziranog SkS, opisanog u 3.3.4, jasno je da je u prirodi SkS *cloud* koji omogućava višeorganizacijsko svojstvo, a time i efikasnost servisa. Upravo ovakva tehnološka migracija je fokus ove disertacije. Iz tog razloga, u ovoj disertaciji, od ovog mesta pa na dalje, **pod SkS će se podrazumevati *cloud* bazirani SkS.**

SkS model aplikacija se može posmatrati kao servis koji je u potpunosti razvija po *SOA* principima, prateći filozofiju poslovnog SkS modela koji je opisan u ovom odeljku. U odeljku 3.4 je opisano na koji način mehanizmi *cloud* okruženja potpomažu razvoj višeorganizacijskih servisa, dok se u odeljku 3.5 opisuje samo višeorganizacijsko svojstvo.

3.4. Softverski dizajn za cloud bazirane sisteme

Razvojem cloud okruženja, PkS sloj je doživeo najveću raznolikost zbog različitih pristupa problemu, kao i različitih tehnologija za koje je u principu namenjen. Glavni problem kod prelaska na PkS predstavlja potencijalno zaključavanje na određenu platformu, što u nekim slučajevima dovodi i do zaključavanja na određeno *cloud* okruženje. Ako se ovaj problem interoperabilnosti

sagleda, jako dugo su postojale važne odluke za organizacije koje razvijaju softver u smislu koji programski jezik i koji ciljni operativni sistem će se koristiti. Iako ova odluka ima i tehničku dimenziju, sam odabir zavisi od više uticaja: političkih faktora sagledavajući tržište koje je ciljno, ekonomskih, veštine koje organizacija već poseduje, pa takođe i postojeća baza izvornog koda koja može uticati na odluku. Može se reći da bi odabir ciljne PkS zavisio i od odluka koji operativni sistem rešenje koristi, kao i koji razvojni programski jezik.

Iz ovih razloga, ovde se neće diskutovati o tome koja je najpogodnija PkS, ali u odeljku 3.4.1 će se dati pregled postojećih platformi, kako bi se pokazalo da se ne gubi u opštosti u radu, i kako bi se prikazale sličnosti i razlike. U rešenju i eksperimentima za ovaj rad je korišćena PkS *Windows Azure ServiceFabric*. U odeljku 3.4.2 je dat kratak pregled uopštenog softverskog dizajna koji je najprirodniji kada je reč o razvijanju *cloud* baziranih servisa, dok je u odeljku 3.4.3 prikazan sve zastupljeniji programski model u modernim PkS.

3.4.1. PkS rešenja

Kada bismo pokušali da izdvojimo relevantne PkS, moglo bi se početi sa izdvojenim iz rada [64] koji se bavi analizom i poređenjem komercijalno dostupnih PkS. Tamo se nalaze sledeće PkS: *Manjrasoft Aneka, Google's AppEngine, AppScale, OpenShift, WSO2 Stratos*(ukinut 2017. godine), *COSCA* (pokrenuta 2011. ali još nije implementirana), *AWS BeansTalk, CloudBees, Cloud Foundry, Cloudify, Cumulogic* (Ne postoji više), *Engine Yard*. U [65] je rađena analiza PkS, gde je višeorganizacijsko svojstvo jedan od glavnih parametara za analizu. Tu bismo mogli izdvojiti još sledeće PkS: *Heroku, IBM Bluemix, Microsoft Azure,* i *Force.com.*

Jako je bitno istaći da se PkS mogu implementirati u dve kategorije, koje su poprilično različite, ali obe zadovoljavaju definiciju PkS, i time se punopravno tako i nazivaju:

1) PkS v1: Platforme izgrađene obično da pruže automatizaciju, i lakšu manipulaciju i monitoring nad IkS slojem. Npr. današnji predominantni otvoreni projekat za implementaciju IkS je *OpenStack*, nad kojim se često nadograđuju platforme. Ovo je razlog i zašto ih postoji toliko puno na tržištu.
2) PkS v2: Platforme namenjene za određeni softverski dizajn, obično za mikroservisnu arhitekturu. Iz ovog razloga, migracije servisa zahtevaju redizajn, i samim se teško migriraju. Obično se migracija zaniva na tome da se krene od monolita koji će koristiti benefite PkSv1 modela, a potom se izgrađuju mikroservisi koji koriste benefite PkSv2. Benefiti PkS v2 su obično rešeni mehanizmi za visoku dostupnost servisa, ponuda dodatnih servisa za skladištenje, komunikaciju i procesiranje, automatizacija životnog ciklusa ciljnog servisa, gde je platforma u stanju da automatski primeni nadogradnju servisa, kao i ponuda dodatnih servisa koji omogućavaju nove modele programiranja, kao što je model učesnika (engl. *Actor model*) koji je opisan u 3.4.3.

Imajući u vidu razliku između PkSv1 i PkSv2 (Tabela 2), gde su najbitnije sledeće dimenzije:

1) Da li je platforma zamišljena kao v1 ili v2?
2) Višeorganizacijsko svojstvo – da li postoje ugrađeni mehanizmi podele?
3) Interoperabilnost – da li može da se izvršava nad bilo kojom IkS?

4) Godina – godina u kojoj je objavljena inicijalna verzija

Tabela 2 - PkS platforme

	v1 ili v2	**Višeorganizacijsko svojstvo**	**Godina**	**Interoperabilnost**
Windows Azure Cloud Services	1	NE	2010	NE
Microsoft Azure ServiceFabric	2	DA	~2014	DA
Google AppEngine	2	DA	2008	NE
IBM Bluemix	2	DA	~2007	DA
Heroku	2	DA	~2007	DA
CloudFoundry	2	DA	~2008 [66]	DA
AWS BeansTalk	1	NE	~2012	NE
Manjrasoft Aneka	1	NE	2009 [67]	DA
AppScale	1	NE	2010 [68]	DA
RedHat OpenShift	1	NE	2011	DA
CloudBees	1	NE	2010	DA
Cloudify	2	DA	~2012	DA

Cloud Foundry [66], [69] je nastao nakon nastanka *Heroku* platforme pri čemu ove dve imaju veliku sličnost, i zbog svoje otvorenosti zadobio pažnju šire programerske zajednice. *IBM* je iskoristio *Cloud Foundry* nad kojim je izgradio svoju *Bluemix* platformu, *HP* izgradio *Hellion* platformu, dok je *VMWare* otkupio *Cloud Foundry* i napravio svoju verziju, dok se može zaključiti da je *Windows Azure Service Fabric* nastao kao odgovor na *Cloud Foundry*. *Heroku* kao inicijator PkS v2, barem kada se sagleda iz istorijske perspektive je otkupljen 2012. godine od strane kompanije *Salesforce.com*. *Cloud Foundry* je osnovica i za *Predix* platformu za *IoT* u industriji, koju je razvio *General Electric*.

Kao što se vidi, neke PkS nisu doživele 2018. godinu, a neke nisu uspele ni da započnu svoj ciklus. U prethodnih deset godina, tržište je bilo dosta nedefinisano u smislu da nije postojala sigurnost koje platforme će opstati. U tim slučajevima je bitno uvažiti i ko je vlasnik platforme, jer one PkS iza kojih stoje velike korporacije će verovatno opstati. Da ni to nije garancija je dokaz *Microsoft* koji je u trenutku objavljivanja PkS v2 najavio ukidanje podrške za sopstveni PkS v1.

PkS v1 ne zahteva neke promene u softverskom dizajnu, niti se bave na posebnom nivou višeorganizacijskim modelom, već doprinose automatizaciji. Iz ugla ove disertacije, čiji je cilj da predloži rešenje za migraciju na *cloud* bazirani SkS model, PkS v1 se može zanemariti, dok PkS v2 treba istražiti kako se ne bi izostavila važna paradigma kada su pitanju *cloud* okruženja.

Ako se sagleda PkS v2, *Google AppEngine* je namenjen specifičnom scenariju, *Heroku* je preuzet od strane kompanije *SalesForce.com*, dok je *Cloud Foundry* postao de fakto standard gde se poredi [66] sa *OpenStack* distribucijom za IkS, tj. najrasprostranjenija PkS v2 sa otvorenim kodom koja je preuzeta da bude osnova mnogih distribucija. Ako uzmemo analogiju, u operativnim sistemima bi takvu sudbinu imao *Linux*, dok bi u IkS taj nivo predstavljao *OpenStack*. U predloženom rešenju je korišćen *Microsoft Azure*, koji je baziran na istim konceptima kao i *Cloud Foundry*, ali ima veoma dobru integraciju sa ostalim *Microsoft* tehnologijama uključujući *.NET* okruženje.

U tabeli se nisu našle PkS koje su u uvodnom delu spomenute: *Force.com* i *Engine Yard*. Odnose se na pružanje specifičnog skupa funkcionalnosti, koje gube na generičnosti toliko da se ne mogu razmatrati za migraciju proizvoljnog rešenja, a niti onog u industriji pametnih elektroenergetskih mreža. Ovakve platforme bismo mogli deklarisati kao PkS za SkS jer upravo to i pokušavaju – da omoguće lak razvoj određenih SkS baziranih na toj platformi.

Postoje i nove PkS, tzv. bezserverske (engl. *serverless*) PkS koje nudi svaka veća *cloud* kompanija, ali takođe ne predstavljaju dovoljno opšti model da bi se moglo diskutovati o migraciji postojećih rešenja, nego više upotpunjuju ponudu *cloud* okruženja. Naime, cilj ovih platformi je da ponude izvršavanje programiranih funkcionalnosti baziranih na nekom događaju. Iako je zaključak da bi se trebale koristiti za neke manje zahtevne scenarije zbog ekonomičnosti [70], ovakve platforme dobijaju na značaju i mogli deklarisati kao PkS v3. PkS v3 neće biti razmatrane u okviru ove disertacije.

3.4.2. Distribuiranost, SOA i mikroservisi

U *cloud* okruženjima, podrazumeva se arhitektura distribuiranih sistema, a u ovom radu se osnovno znanje o distribuiranim sistemima podrazumeva. Savremeni distribuirani sistemi se baziraju na *SOA* (engl. *Service-Oriented Architecture*) principima. Tip *SOA* arhitekture je baziran na slaboj povezanosti komponenata i na interoperabilnošću među servisima [71]. U suprotnosti nedistribuiranim aplikacijama, najveći izazov je razdvojiti odgovornosti različitih servisa kako bi svaki servis imao jasno određenu odgovornost. Iako postoje očigledni problemi kod ovakvog pristupa, kao što je gubitak konektivnosti između servisa, distribuiranost aplikacije je često i zahtev nametnut od strane prirode problema. Servisna orijentisanost i distribuiranost omogućavaju skalabilnost aplikacije. Poslovni procesi se često izvršavaju na različitim platformama, pa čak i tehnologijama, a čest je i zahtev za razmenom informacija između servisa koji ne pripadaju istoj organizaciji.

SOA je u suštini dizajnerski princip koji pokušava da omogući razvoj agilnih i lako adaptivnih aplikacija. Može se posmatrati kao skup dobro definisanih servisa, gde svaki individualni servis može biti menjan nezavisno od ostalih servisa [71]. Cilj mogućnosti nezavisne izmene servisa je da se omogući implementacija novih zahteva. Evidentno je da se klijentski zahtevi brzo menjaju i nastaju novi. Za razliku od tradicionalnih monolitnih arhitektura, *SOA* implementacija se sastoji od skupa servisa koji su interoperabilni i međusobno nezavisni. Mogu se nezavisno menjati i evoluirati vremenom. Osobina implementacione nezavisnosti između servisa se naziva slabom povezanošću servisa.

Iako su navedeni aspekti slični razvoju baziranom na komponentama, gde komponente imaju jasno definisane interfejse, postoji jasna razlika. *SOA* pruža način komunikacije putem poruka koje su deo standarda. Visok nivo interoperabilnosti između različitih platformi i tehnologija je postignut koristeći standarde i generičke poruke. Poruke ne predstavljaju ni jednu konkretnu platformu, niti određuju programski jezik. Upravo zasnovanost komunikacije na porukama je omogućila servisima i *SOA* principima da u potpunosti zamene komponentno bazirane distribuirane sisteme.

Svaki servis je autonoman i pruža jednu ili više funkcija koju čitav poslovni sistem obrađuje. Dodatno, bilo kakva promena implementacije u servisu ne izaziva promene u ostalim servisima u sistemu ukoliko se interfejs servisa nije menjao. Implementacioni detalji su skriveni od korisnika servisa. Opisana nezavisnost omogućava *SOA* aplikacijama da brže implementiraju nove zahteve.

Kod *SOA* pristupa, glavnu prednost za sobom povlače poruke. Poruke omogućavaju slabu povezanost servisa koji mogu biti na različitim operativnim sistemima. Nad porukom se može naći kontekst poruke u okviru kog se može implementirati bezbednosni model.

Servis se može posmatrati kao autonomni sistem koji prihvata zahteve i vraća odgovore tim zahtevima putem objavljenih i dobro definisanih interfejsa [71]. Autonomnost servisa čini servis pogodnim kandidatom za deljenje između više različitih korisnika. Najpogodnije je da se servis od početka projektuje tako da uzima u obzir mogućnost rada sa više različitih klijenata, da bude spreman za višeorganizacijsko okruženje. Razumevanjem granica i odgovornosti servisa, može se zaključiti da *SOA*, za razliku od tradicionalnih spregnutih arhitektura, implementira skup servisa koji zajednički vrše određenu funkcionalnost.

Servis u *SOA* principima treba da zadovolji četiri pravila. Četiri pravila nakon primene višeorganizacijskog modela treba da ostanu zadovoljena, a to su sledeća pravila:

1) granice servisa su jasno definisane,
2) servis je autonomna celina,
3) servisi dele šeme i interfejse, a ne klase,
4) kompatibilnost servisa se bazira na politici servisa.

Prilikom dizajniranja servisa, potrebno je obratiti pažnju na to da se ne naruši neko od četiri pravila. Isto tako, prilikom uvođenja svesti pri dizajniranju višeorganizacijskog servisa, ova četiri pravila je potrebno slediti i ne narušavati.

Prednost servisno orijentisanih arhitektura je to što se novi servisi mogu konstruisati od već razvijanih servisa. Postojeći servisi su često već potvrđeni kao ispravni u produkciji. Prateći *SOA* principe, ponovna upotreba razvijenih servisa je pojednostavljena. Ponovno korišćenje postojećih servisa sprečava dugačak, spor i sklon greškama proces ponovnog razvijanja funkcionalnosti koje su se već više puta razvijale [72]. Tehnologija *web* servisa kao jedna od tehnologija koja implementira *SOA* dozvoljava definisanje ponovno iskoristivih komponenti zvanih *web* servisi [72].

Mikroservisi su pojam koji se počeo pojavljivati od 2014. godine, uglavnom iz organizacija koje su zagovarale agilan razvoj softvera [73]. Istraživanjem literature i razumevanjem

mikroservisa, nesporno je da oni ne predstavljaju novi arhitekturalni stil u odnosu na SOA. Ovakvo viđenje se može pronaći i u [73], pod tvrdnjom da su mikroservisi u stvari SOA. Sa ovom rečenicom bi se mogla staviti tačka na uvod pojma mikroservisa i poistovetiti ga sa SOA. Ipak, osim marketinškog efekta nazivati sve vezano za *cloud* okruženja mikroservisnom arhitekturom, ovde će se izdvojiti i principi izvedeni iz [73] koji uže određuju da je reč o mikroservisnoj arhitekturi:

1) Fino definisane granice interfejsa – može se primetiti da je ovo preuređena izjava iz SOA principa, ali sa izričitom napomenom da servisi treba da imaju što jednostavniju svrhu i da enkapsuliraju svoje podatke.
2) Dizajn vođen domenom (engl. *Domain Driven Design* - DDD) – za identifikaciju i koncepcionalizaciju servisa. S obzirom da pojam proističe iz agilnih metodologija razvoja procesa, ideja je da se domen razvoja ograniči na minimalne celine, ali tako da se dozvoli autonomija manjim timovima da rade na razvoju određenog mikroservisa.
3) *Cloud* nativni dizajn principi – kao što je u 3.4.1 pomenuto, PkS v2 su pravljenja tako da se benefiti maksimizuju uz predodređeni šablon razvijanja servisa. Da bi se ta pravila stavila pod veo novog termina, mikroservisi su poslužili toj svrsi. Odličan primer je sada već vrlo poznati princip 12 faktora razvoja aplikacija (engl. *The twelve-factor app*), koji je proizišao iz prve PkS v2 platforme, *Heroku*.
4) Poliglotsko programiranje i strategija konačne konzistencije (engl. *eventual consistency*) – svaki mikroservis može koristiti svoju tehnologiju, ponovo isto kao i SOA, s tim da se kod skladištenja podataka može raditi o relacionim bazama podataka, NoSQL, ili bilo kojim drugim. Umesto stroge konzistencije, konačna konzistencija je dozvoljena. Ovakav princip nije eksplicitno naglašen u SOA, mada suštinski ništa ne menja, nego više naglašava šta se očekuje od pojma mikroservisa.
5) Decentralizovano kontinualno dostavljanje (engl. *continious delivery*) – promocija velikog stepena automatizacije i autonomije. Ovo je isto u potpunosti u skladu sa SOA.
6) *DevOps* – automatizovani pristup konfiguraciji, upravljanju performansama i otkazima koji proširuju prakse agilnog razvoja softvera i uključuju monitoring servisa.

Vidljivo iz navedenih principa mikroservisne arhitekture jeste da se u principu radi o SOA arhitekturalnom stilu, s tim da se niz osobina podrazumeva kako bismo jedinstveno odredili taj dodatni niz karakteristika koje se očekuje od takvog rešenja. Kao važan dodatak, nešto što se ne spominje eksplicitno u SOA principima, a utiče na to kako će se servisi projektovati i kreirati, jeste izazvano time što tehnološki gledano, mikroservisi su najčešće pravljeni sa SkS modelom na umu. Ovo direktno znači da će klijent biti *web* baziran. To dalje omogućava da se od mikroservisa može očekivati da implementira, za svoj domen logike, deo korisničkog interfejsa, perzistenciju podataka, kao i poslovnu logiku. Ova filozofija inspiriše da se arhitektura kreće tako što se celine dele vertikalno, a ne horizontalno. Pod vertikalnim se podrazumeva da servis sadrži u sebi sva tri sloja jedne troslojne aplikacije, dok bi horizontalno značilo da se servisi uvezuju u niz međuzavisnih servisa koji svaki za sebe obavlja određeni deo poslovne logike.

3.4.3. Model učesnika

Problem koji se manifestuje u PkS v1, opisanim u 3.4.1, jeste to što ne rešavaju probleme koji se sreću u distriburanom programiranju za softver koji će se nad njima izvršavati. Pod glavnim

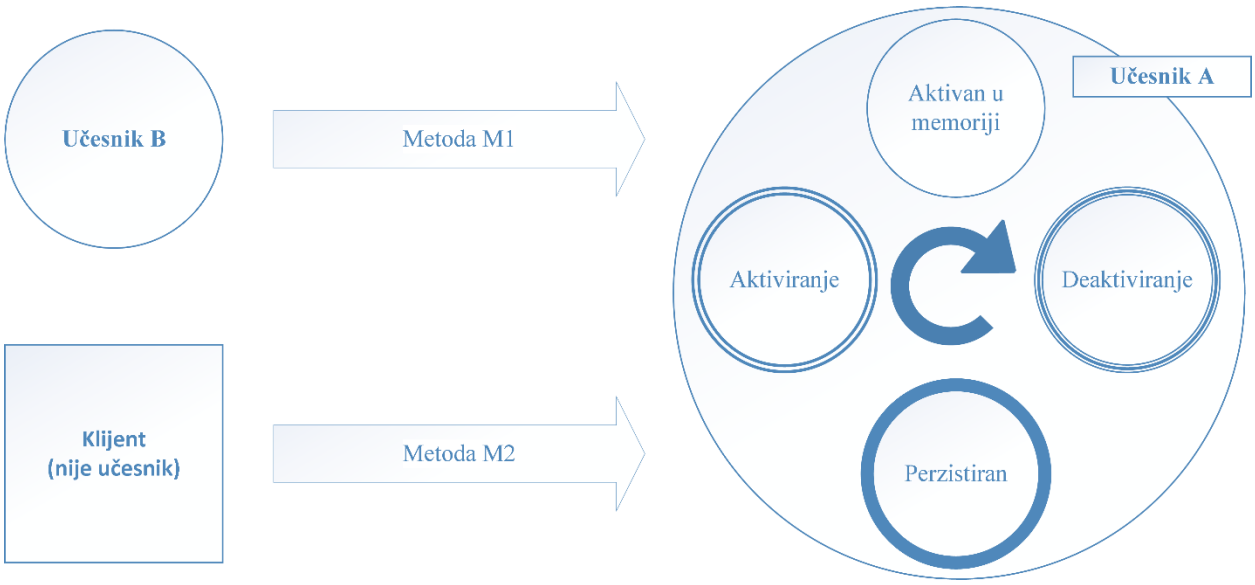
problemom se smatra distribuiranje stanja, koje je i dalje ostalo da bude rešeno na dizajnu aplikacije. *Cloud* kompanije su popularizovale tzv. pristup bez stanja (engl. *stateless*), gde se favorizuje eksternalizacija stanja gde god je to moguće. Ideja je da se podrazumeva da se jedan proces izvršava na više virtuelnih mašina, koje će se nazivati instancama, gde te instance treba da imaju interno stanje poravnato kako bi mogle izvršavati poslovnu logiku. Pod eksternalizacijom se podrazumeva čuvanje svih promena u nekom eksternom servisu, što bi omogućilo da u slučaju otkaza jedne od instanci, druga instanca može da učita trenutno stanje i nastavi sa radom.

Nažalost, srednji sloj sa eksternalizovanim stanjem je često nezadovoljavajuće rešenje. Ako se svi zahtevi moraju preusmeriti na servis za čuvanje podataka, tada čitav sloj gubi na skalabilnosti jer skladištenje izaziva latenciju i limite na propustnost [74]. U ovim slučajevima keširanje može da pomogne, ali onda keš predstavlja stanje svih instanci, što znači da same postaju instance sa stanjem (engl. *stateful*). Keševi uvode dodatni niz problema, čiji detalji će biti preskočeni jer ne utiču na svrhu, a to je da PkS v1 ne rešava glavne probleme koji se sreću u distribuiranom programiranju.

PkS v2 su napravljene tako da omoguće instance sa stanjem, tj. da se sloj za replikaciju internog stanja pruži kao deo platforme. Ovo je upravo i glavni razlog zašto PkS v2 zahtevaju razvoj aplikacija u unapred određenom šablonu.

Iako ne nov koncept, model učesnika (engl. *Actor model*) je uveden 1970. godine sa *Erlang* jezikom kao najpopularnijom implementacijom. Danas, u kontekstu PkS v2 dobija na značaju jer inherentno omogućava **horizontalnu skalabilnost** i rešava čuvanje stanja među instancama.

Sumiranjem [74], [75] i [76], dolazi se do zaključka da su učesnici zamišljeni kao singularni, jasno definisani objekti koji su izolovani jedni od drugih. Komuniciraju putem sloja za razmenu poruka koji je asinhron. Dinamički se kreiraju nad skupom resursa (Slika 11) koji je obično klaster virtuelnih mašina, i ne dele memoriju. Mnoge današnje aplikacije imaju prirodno particionisanje koje odgovara učesnicima, kao što su industrija video igara, telemetrija (eksperiment će biti kreiran tako da se pokrije mogućnost kreiranja SCADA funkcionalnosti sa učesnicima), socijalne mreže i IoT, koji će biti obuhvaćen eksperimentom.



Slika 11 - Model učesnika [74]

Izolovanost po dizajnu učesnika ih inherentno čini spremnim za implementaciju višeorganizacijskog svojstva, jer svaki učesnik može biti namapiran na drugu organizaciju. **Uz ispravnu implementaciju konteksta organizacije što je primarni fokus ove disertacije, učesnici mogu biti korišćeni u okviru višeorganizacijskog rešenja**.

Microsoft Azure Service Fabric (MASF) implementacija modela učesnika koristi virtuelne učesnike. Učesnici su *.NET* klase koji implementiraju *.NET* interfejse. Skup učesnika koji je aktivan u radnoj memoriji je uvek podskup učesnika koji su logički kreirani i sačuvani u okviru stalne memorije. Ovaj koncept je sličan virtuelnoj memoriji, i stoga se učesnici nazivaju virtuelnim. MASF donosi odluku kada aktivira učesnike ili da ih deaktivira. Na taj način, okruženje ima pristup svim učesnicima na logičkom nivou, bez obzira da li su aktivni, tj. u radnoj memoriji ili nisu. Na ovaj način upravljanje učesnicima je izmešteno u samo MASF okruženje.

Svaki učesnik predstavlja jednu nit, i izolovani objekat koji je dovoljan sam po sebi čije se stanje perzistira odvojeno od drugih učesnika. Ovo pojednostavljuje pisanje visoko paralelizovanog koda. S obzirom da je učesnik uvek izvršavan u okviru jedne niti, na nivou učesnika se ne mora razmišljati o zaključavanjima i ostalim mehanizmima pristupa konkurentno deljenim podacima.

Učesnici se aktiviraju uvek kada je za njih namenjena neka poruka, a komuniciraju isključivo asinhronim mehanizmom. Upravo ovaj asinhronizam omogućava efikasno iskorišćenje resursa, ali sa druge strane, ideja eksperimenata u ovom radu će biti da se izmeri koliko asinhronizam utiče na problem gde je potrebno stanje svih učesnika ili velikog dela učesnika u određenom momentu.

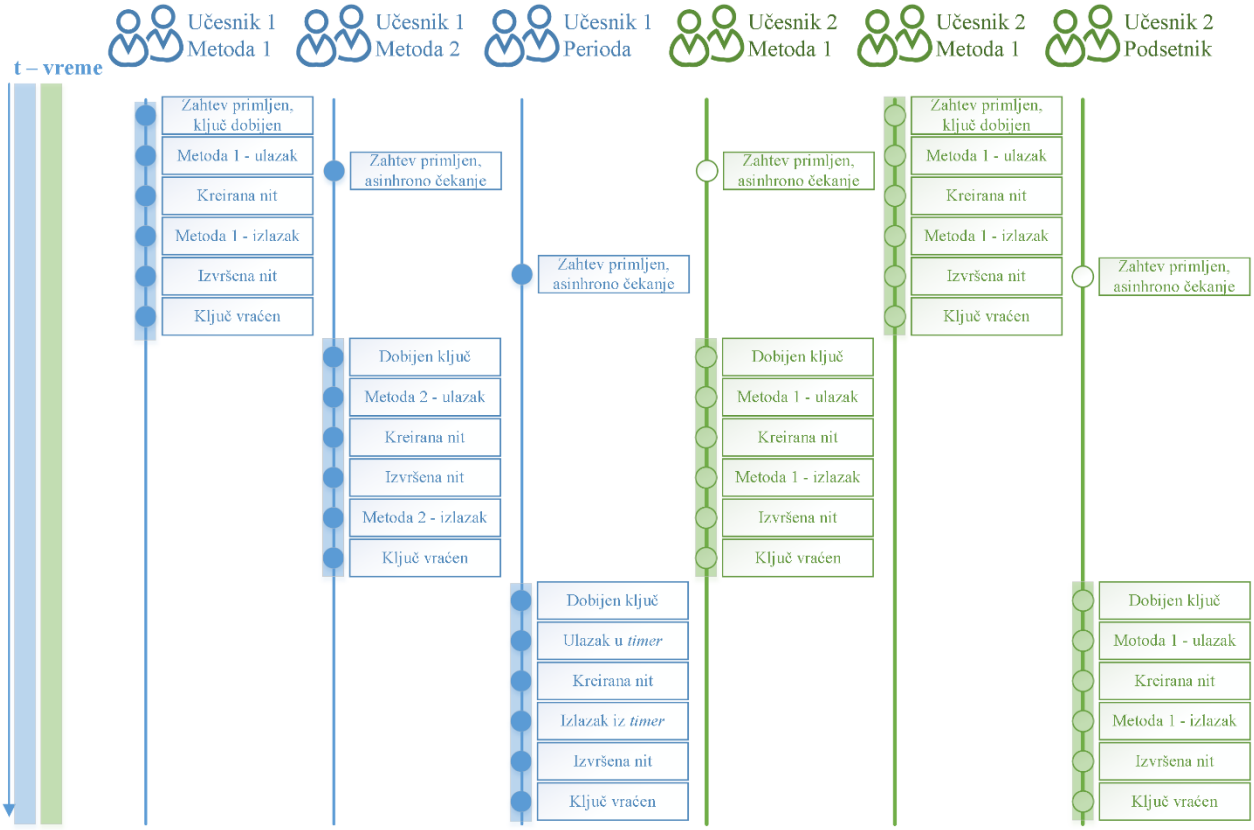
Osobina lokacijske transparentnosti je zadovoljena, i fizičku adresu određenog učesnika određuje MASF, pri čemu su podržani i geodistribuirani učesnici.

Slika 11 prikazuje životni ciklus učesnika. Učesnik se aktivira na poruku, koju je mogao poslati drugi učesnik ili klijent koji nije učesnik. Pri aktiviranju, može se dogoditi da je učesnik već aktivan i obrađuje poruku ili se instanca učesnika kreira iz prethodno sačuvanog stanja. Učesnik je aktivan u memoriji konfigurabilno dugo gde se gleda poslednje procesiranje poruke, i u slučaju deaktivacije, njegovo trenutno stanje se čuva, i učesnik se oslobađa iz radne memorije. Bitna je napomena da učesnik u toku svog aktivnog perioda je osiguran od gubitka podataka tako što jedna instanca ima više svojih replika koje primaju sve promene koje se dešavaju u okviru primarne instance.

Model učesnika omogućava transparetnost zbog sledećih osobina:

1) Implicitno particionisanje internog stanja – svaki učesnik sadrži svoje stanje i jasne granice definisane metodama, tako da je interno stanje podeljeno različitim brojem učesnika. Zbog efikasnije skalabilnosti, preporučuje se veći broj učesnika sa manjim ingerencijama nad delom stanja.
2) Adaptivno upravljanje resursima – usled lokacijske transparentnosti, MASF odlučuje o migraciji učesnika u zavisnosti od šablona njihove interne komunikacije i aktivnosti.
3) Multipleksirana komunikacija – MASF koristi jednu TCP konekciju između servera u klasteru, i na taj način postiže da se veliki broj učesnika adresira koristeći relativno mali broj TCP konekcija.

4) Efikasno planiranje – MASF vodi računa o izrvšravanju učesnika, dok se čitav MASF izvršava u korisničkom režimu rada procesora. Ovo implicira da promena učesnika ne zahteva promenu konteksta niti (engl. *thread*) na procesoru, što dovodi do velike iskorišćenosti samog procesora.



Slika 12 - Konkurentnost učesnika [77]

Može se zaključiti, kao u [75], da postoji asinhrona komunikacija i visoka konkurencija oko učesnika, ali u samim učesnicima, pozivi metoda su sinhroni. Učesnik se može posmatrati kao običan objekat sa asinhronom komunikacijom.

Slika 12 ilustruje konkurentnost na nivou učesnika, i prilagođena je iz zvanične MASF dokumentacije. Na primeru učesnika 1 i 2 se vidi da novopristigli zahtevi čekaju da se prethodni posao izvrši, i tek onda dobijaju svoju priliku za potpunim izvršenjem metode gde je prioritet određen vremenskim trenutkom kada je zahtev učesniku upućen. Učesnici mogu zakazivati periodične poslove sami sebi koristeći periode (engl. *timer*) i podsetnike (engl. *reminder*). Oba se koriste tako da se ispoštuje konkurentnost poziva kao što je na slici i prikazano, s tim da je razlika u tome što se podsetnici pozivaju uvek, dokle god ih učesnik eksplicitno ne zaustavi ili se sam učesnik ne obriše.

3.5. Višeorganizacijsko svojstvo

Svojstvo deljenja softvera izaziva konfuzije u smislu preciznosti pojma kod javnih *cloud* okruženja iz razloga što su ona podrazumevano deljena. Postoji više nivoa apstrakcije i više uglova iz kojih se može sagledavati da li je kranji SkS višeorganizacijski ili ne. Iako naizgled trivijalno, često je vrlo konfuzno i pojašnjenje se nalazi u odeljku 3.5.1.

U odeljku 3.5.2 je detaljno opisan sam pojam višeorganizacijskog svojstva u softveru, i predstavljeni su modeli zrelosti. Skalabilnost, kao neizostavna komponenta rešenja koje se bazira na deljenu softverskih instanci je približena u odeljku 3.5.3 iz ugla višeorganizacijskog softvera.

3.5.1. Na nivou *cloud* usluge

Podela usluga u okviru *cloud* okruženja po nivou usluge je detaljno opisano u 3.1.3. U ovom odeljku je cilj da se opiše naizgled jednostavna, ali poprilično zbunjujuća i kompleksna pojava deklaracije da li je neka usluga višeorganizacijska (VO) ili posvećena jednoj organizaciji (PO).

Tabela 3 se odnosi na javna *cloud* okruženja, gde redovi prikazuju nivoe apstrakcije u uslugama, dok kolone predstavljaju različiti pogled, u zavisnosti od nivoa usluge *cloud* okruženja. Npr. iz ugla IkS, ne može se pričati o osobinama operativnog sistema, jer operativni sistemi ne spadaju u taj nivo ponude. Vrednosti u ćelijama označavaju podrazumevanu vrednost u većini današnjih komercijalnih ponuda, uzimajući u obzir ponude javnih *cloud* okruženja kompanija *Microsoft, Amazon,* i *IBM.*

Hardver kao servis (HkS) gotovo i nije u *cloud* ponudi, osim u slučaju *IBM cloud* okruženja, ali se npr. u *Amazon* katalogu može odabrati opcija da HkS bude PO. Podrazumevano je deljen između više organizacija. Mreža je uvek VO i drugačija izvedba ne postoji, osim ako se ne sagleda jedan nivo apstrakcije više. Mreža se može podeliti na mrežnu konekciju od ulaska u centar podataka, do samog hardvera gde se naši resursi nalaze – ta mreža, jasno je, uvek je VO i drugačije ne može biti. Mreža u okviru iznajmljenog hardvera teoretski može biti posvećena, ako i L3 uređaji pripadaju samo našim resursima i saobraćaj ne napušta perimetar naših resursa. Ovo je praktično veoma teško izvodljivo, i ekonomski neisplativo. Zaključak je da je mreža uvek VO, pogotovo ako pričamo o regularnim javnim *cloud* okruženjaima.

Tabela 3 - Višeorganizacijska osobina na nivou usluge

Nivo apstrakcije	IkS	PkS v2	SkS
Aplikacija			VO (PkS?)
Aplikativni podaci			VO (PkS?)
Izvršno okruženje		PO	
Softverski paketi		PO	
Operativni Sistem		PO	
Virtualizacija	VO (PO)	VO	
Serveri	VO (PO)		
Stalna memorija	VO (PO)		
Mreža	VO		
Hardver kao Servis	VO (PO)		

Stalna memorija se oslanja na niz servisa koji su VO, ali se može implementirati sopstvena verzija skladištenja na samim zakupljenim serverima, stoga u zagradi stoji (PO). Serveri isto tako, mada je interesantno tumačenje javnih *cloud* pružaoca usluga, da je CPU resurs posvećen organizaciji, ako je u okviru *cloud* okruženja statički dodeljeno CPU jezgro virtuelnim mašinama u našem domenu, tj. ako se CPU ne deli između više virtuelnih mašina. Jasno je da matična ploča ostaje deljena, kao i sve ostale komponente u serveru. Virtuelizacija je definisana u zavisnosti od osobine servera.

Dakle, da li je moguće i pod kojim uslovima se usaglasiti da je IkS posvećena organizaciji? Ovo pitanje otvara mnoge filozofske debate, i u slučaju javnih *cloud* usluga se mora pojam redefinisati tako da logička izolacija predstavlja posvećenost određenoj organizaciji (npr. VLAN u okviru mrežnog saobraćaja). Ukoliko se takvo stanovište prihvati, i logička izolacija prestavlja prihvatljivu definiciju, šta sprečava bilo kog SkS vlasnika da izjavi da je SkS posvećen organizaciji, a da pri tome izveden višeorganizacijski?

Da bi se našlo prihvatljivo obrazloženje, neophodna je analiza tabele iz ugla PkS. Ako se prihvati da je IkS višeorganizacijska, iz ugla PkS se može sagledati da li je jedan operativni sistem, softverski paketi i izvršno okruženje posvećeno organizaciji. Iz ugla PkS, uvek jeste, ali ugao PkS je irelevantan za ovu diskusiju jer će to pitanje odrediti ugao SkS koji se oslanja na tu platformu. Iako su sve tri stvari PO iz ugla PkS, sam PkS kao skup softverskih alata može biti VO ili PO. Tačnije, pitanje je da li sami pružaoci SkS uzimaju PkS kao softver, instaliraju je na IkS i održavaju ili koriste, ili koriste gotovu platformu koju nude *cloud* okruženja, a koja je često VO. Pošto se tabela odnosi na tumačenje ponuda javnih *cloud* usluga, odgovor je da je sama PkS višeorganizacijska, iako su i operativni sistem, i softverski paketi i izvršno okruženje posvećeni organizaciji.

Da li ima smisla i da li se može reći za SkS da je posvećen organizaciji ukoliko se oslanja na IkS i PkS koji su višeorganizacijski? U praksi se sreću izjave da je SkS posvećen iako je zasnovan na VO IkS ili VO PkS, ali uz adekvatno pojašnjenje se može zaključiti da to stanovište ima smisla. Ukoliko vlasnik SkS tvrdi da je SkS posvećen organizaciji, to znači isključivo samo da se na nivou aplikacije i podataka ne uvodi višeorganizacijsko svojstvo. Ovim se jasno razgraničava stepen odgovornosti u tržištu gde su klijenti prihvatili i navikli na činjenicu da *cloud* okruženja jesu dovoljno bezbedna po ovom pitanju, tj. imaju poverenja u brendirana *cloud* okruženja. U slučajevima gde se nema poverenja u *cloud* okruženja, ili postoje regulative koje striktno nalažu određene stvari kao npr. da je stalna memorija posvećena, ima smisla da se zalazi i u detalje kako su pojedinosti implementirane na nivou IkS. Ukoliko je SkS višeorganizacijski, tu je zaista veoma mali uticaj da li je i IkS višeorganizacijska iako je i dalje validno pitanje zbog sagledavanja bezbednosnih pretnji.

Zbog gorenavedenih otvorenih pitanja, i diskusije koja bi trebalo da pomogne u razumevanju odgovora, u nastavku se navode opšteprihvaćeni stavovi iz prakse, uz napomenu da je reč o javnim *cloud* okruženjima:

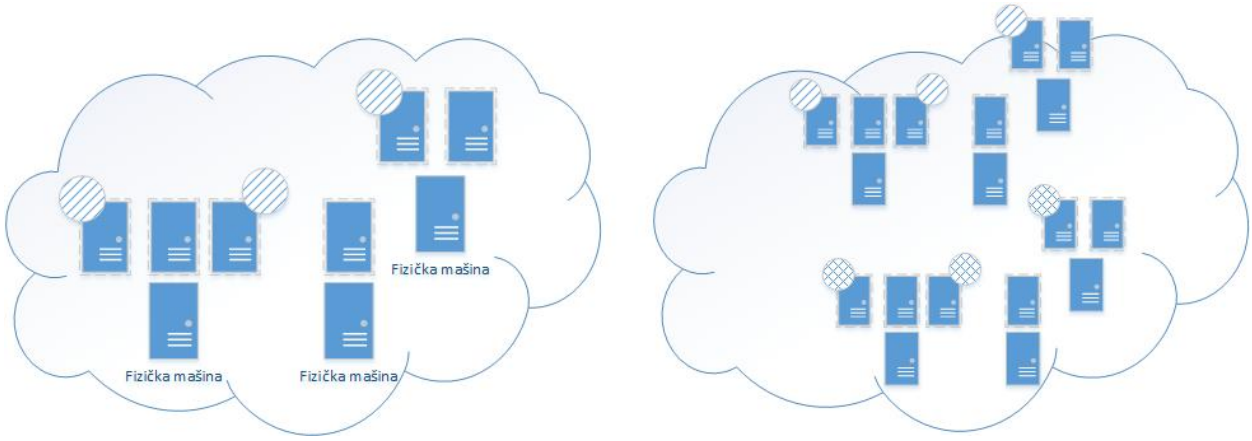
1) IkS se smatra višeorganizacijskom, pri čemu se određeni nivoi apstrakcije mogu razmatrati za izolaciju zbog potrebe usaglašavanja sa regulativama, ili bezbenosnim politikama klijenata u datom tržištu.

2) PkS ide uz IkS, a obično se debata postavlja na nivou sigurnosti operativnih sistema i da li je reč o virtuelnim mašinama ili kontejnerima. U oba slučaja, bitna osobina je da li su elementi PkS posvećeni po klijentu ili nisu – ovaj odgovor iz ugla krajnjih klijenata može se pružiti samo uz sagledavanje implementacije SkS. Iz ugla PkS ka proizvođačima SkS, sva tri elementa platforme su posvećena organizaciji, iako je sama platforma višeorganizacijska.
3) Iz ugla SkS, odgovor na višeorganizacijsko pitanje se odnosi na razgraničavanje IkS odgovora, i odgovora vezana za aplikativni nivo i podatke. U slučaju da sam SkS jeste višeorganizacijske prirode, vrlo verovatno će biti potrebna izgrađena jaka veza poverenja koja se gradi ispunjavajući:
    a. Raniji odnos sa klijentom je bio vanredno dobar,
    b. Postojanje visokog stepena bezbednosti u organizaciji koja pruža SkS – većina pitanja će biti usmerena na prakse razvoja softvera, informacionu bezbednost kompanije i sertifikacije u domenu bezbednosti,
    c. Jačina brenda kompanije u domenu SkS.

3.5.2. Modeli zrelosti

Višeorganizacijski model je jedna od ključnih karakteristika servisno orijentisane arhitekture, a posebno za SkS model jer podiže ekonomiju skaliranja spuštajući ukupan trošak i za korisnika servisnih usluga i za pružaoca servisa [29]. Cilj ovog poglavlja je da objasni zašto je višeorganizacijski model u tesnoj sprezi sa SkS modelom i zbog čega bi SkS model trebao da teži ka tome da ujedno bude i višeorganizacijski model.

Konceptualno, *cloud* bazirani SkS bi u tradicionalnom modelu postavke zahtevao linearno više resursa po broju klijenata (Slika 13). Kružići različitog šablona na slici predstavljaju različite organizacije.



Slika 13 - SkS model sa jednom i više organizacija

Višeorganizacijski model pomaže pružaocima servisnih usluga da smanje troškove, poboljšaju upotrebu raspoloživih resursa, i smanje vreme održavanja i prilagođavanja aplikacije jer organizacije dele istu instancu servisa [32]. Za podržavanje višeorganizacijskog modela, potrebno je da SkS servis ima mogućnost čuvanja, procesiranja i čitanja podataka različitih organizacija (korisnika) u okviru iste softverske instance.

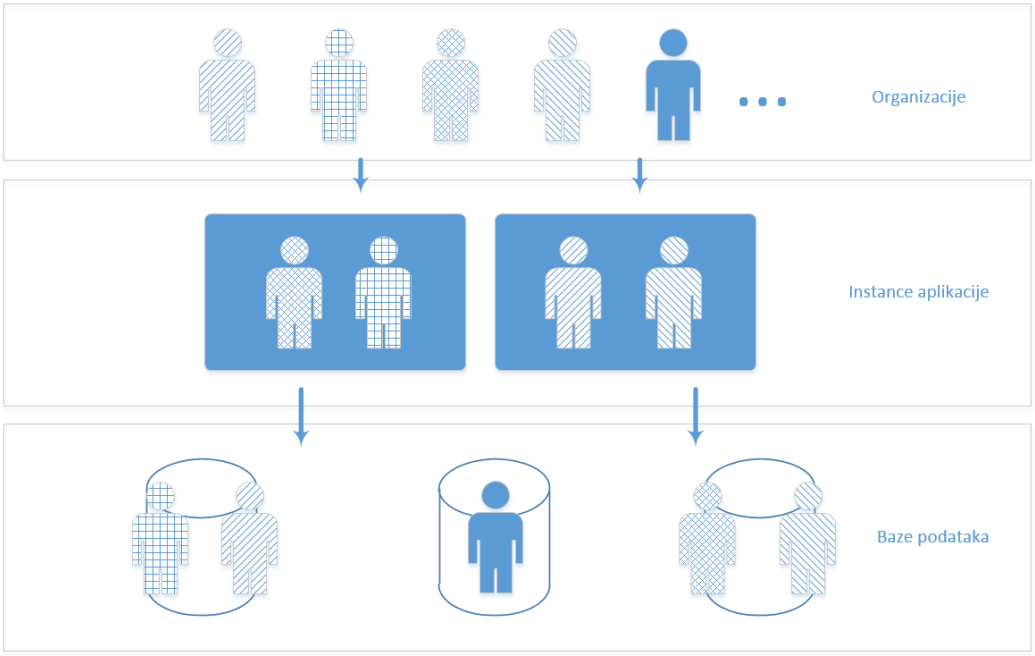
Deleći jedan proces servisa kroz mnoge organizacije, višeorganizacijski model pokušava da zameni pojavu mnogo malih instanci jedne aplikacije sa jednom ili više velikih instanci koje su deljene između više organizacija i da na taj način smanji ukupne troškove infrastrukture informacionih tehnologija [63].

Na više mesta se opisuje pojam višeorganizacijskog modela kao pravac deljenja resursa između različitih organizacija a sve u cilju uštede na resursima. U [78] se daju dve definicije:

- Definicija 1: Višeorganizacijski servis pruža korisnicima (organizacijama) deljenje hardverskih resursa, pružajući deljenu instancu aplikacije i deljenu instancu baze podataka, dopuštajući im konfigurisanje aplikacije tako da pokriva njihove potrebe kao da se aplikacija izvršava u posebnom okruženju.
- Definicija 2: Organizacija je organizacioni entitet koji iznajmljuje višeorganizacijski SkS. Tipično, organizacija grupiše više korisnika.

Definicija 2 definiše pojam organizacije, što će u ovom radu na dalje biti označeno pod pojmom organizacije. Definicija 1 na prvi utisak deluje previše restriktivno jer se pominje deljenje hardverskih resursa. Ako se uzme u obzir da deljenje softverskih instanci implicira da je potrebno manje hardvera za hostovanje instanci softvera, može se smatrati da je tvrdnja tačna. Stoga, definicija 1 će se uzimati kao relevantna definicija višeorganizacijske aplikacije. Za kompletnost definicije 1, potrebno je dodati da se uz deljenje instance servisa i deljenje instance baze podataka, može podrazumevati bilo kakvo deljenje resursa kao što je sloj za razmenu poruka između procesa. Deljenje bilo kakvog softverskog sloja znači da se određeni softverski sloj mora projektovati tako da podrži višeorganizacijski model. Slika 14 ilustruje višeorganizacijski model po definiciji 1.

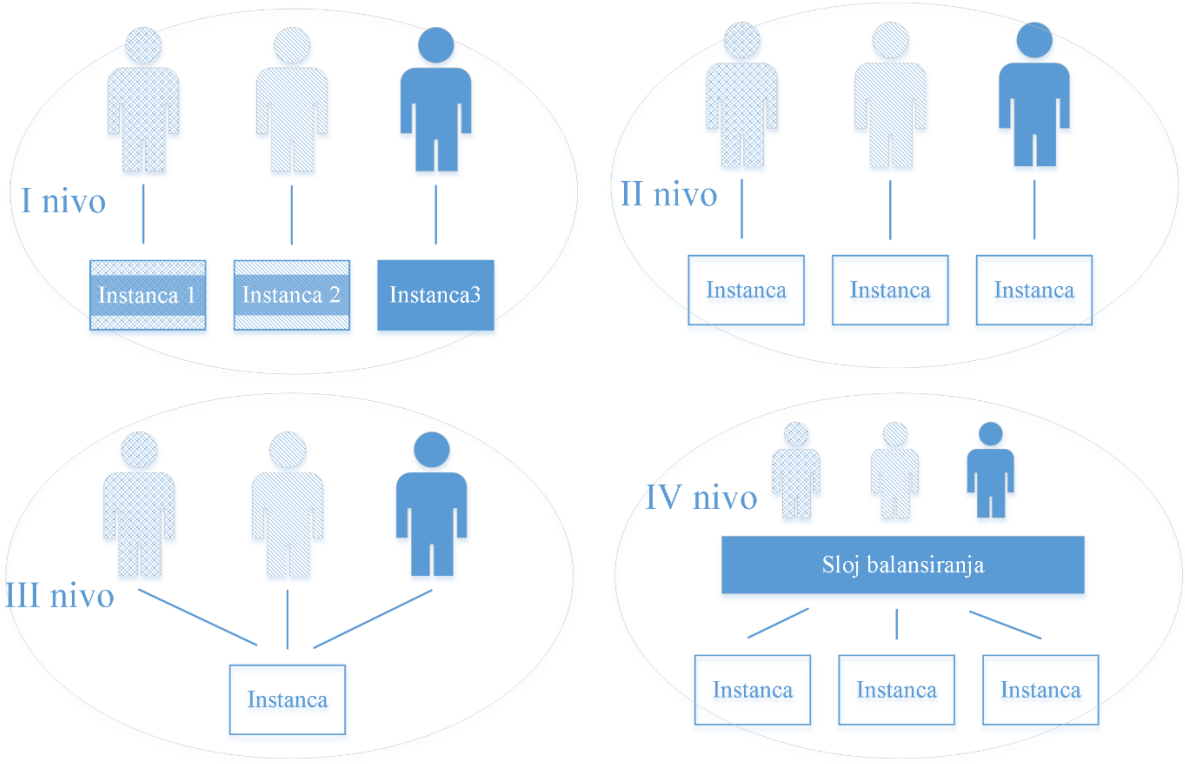
Ako se uzme virtuelizacija kao ključna tehnologija za uvođenje *cloud* okruženja, može se iskoristiti osobina izolacije među virtuelnim mašinama ili kontejerima, za potrebe višeorganizacijskog modela.



Slika 14 - Višeorganizacijski model [29]

U IkS modelu, višeorganizacijski model je postignut koristeći hipervizore koji virtuelizuju servere kao resurse. Sam operativni sistem ne mora biti modifikovan. Još više, svaki korisnik IkS usluge poseduje posebnu instancu virtuelne mašine (VM) [32]. Kako god, apstrakcije koje pruža *cloud* okruženje su često neadekvatne za pružanje velikih ušteda u toku životnog veka infrastrukture informacionih tehnologija [63].

U SkS, višeorganizacijski model ima više modela postavke (engl. *deployment*) uključujući različite instance za svaku organizaciju do jedne instance za različite organizacije. Ukupno, razlikuju se četiri nivoa postavke (Slika 15). Treći i četvrti nivo se odnose na višeorganizacijski model, dok prvi i drugi model i nisu višeorganizacijski nego višeinstancni.



Slika 15 – Nivoi zrelost višeorganizacijskog modela [79]

Prvi nivo je veoma sličan modelu tradicionalnih aplikacija. Na ovom nivou, svaka organizacija ima svoju prilagođenu instancu aplikacije. Po arhitekturi, softver se i ne razlikuje puno u odnosu na tradicionalni. Razlika je u tome što se aplikacija izvršava u *cloud* okruženju koje je samo po sebi deljeno, opisano u odeljku 3.5.1.

Drugi nivo predstavlja nivo konfigurabilnosti servisa. Zasebna instanca servisa se izvršava po organizaciji. Razlika u odnosu na prvi nivo je to što se koristi ista implementacija u slučaju svih organizacija, a posebni zahtevi se zadovoljavaju na nivou konfiguracije.

Uvod u konfigurabilnost aplikacije i njena važnost je opisan u odeljku 3.3.5, dok će u ovom odeljku će biti detaljnije iznete prednosti konfigurabilnosti aplikacije. Slično prvom nivou, pružaoc softverskih usluga mora biti u stanju da pruži dovoljno hardvera i skladišta podataka tako da podrži potencionalno velik broj instanci aplikacije. Uštede u pogledu deljenja resursa i dalje ne postoje, ali se ostvaruju na polju automatizacije pri kreiranju sistema za nove organizacije, kao i na nivou održavanja koda jer svaka organizacija sada poseduje istu softversku osnovu.

Treći nivo podrazumeva konfigurabilni višeorganizacijski servis. Bezbednosni model mora obezbediti izolaciju podataka jer u ovom modelu organizacije po prvi put zaista dele istu softversku instancu. Treći nivo je dovoljan da se servis smatra višeorganizacijskim. U slučaju trećeg nivoa, skalabilnost rešenja je narušena jer je moguće samo vertikalno skaliranje.

Četvrti nivo adresira nedostatak trećeg nivoa, a to je da omogućava horizontalno skaliranje. Skalabilno višeorganizacijsko rešenje je optimalno po pitanju uštede resursa. Na ovaj način se postiže efikasnost ekonomije kompletnog rešenja.

Treći i četvrti nivo vrše uštedu resursa. Četvrti model postavke zahteva da arhitektura SkS servisa bude projektovana tako da izoluje različite organizacije. Međutim, osim uštede u resursima, četvrti model zrelosti dovodi do veoma važnog benefita višeorganizacijskih sistema, a to je veća visoka dostupnost rešenja, iz razloga što se instance uvezuju u paralelni sistem računanja otkaza datog servisa. O računanju paralelno vezanih instanci detaljno ima u odeljku 7.4.2.

Virtuelne mašine su međusobno u potpunosti izolovane. Ovo je jedan logički nivo na kom se može ispoljiti osobina višeorganizacijskog modela. Ukoliko se različite virtuelne mašine izvršavaju na jednoj fizičkoj mašini, i pri tome, virtuelne mašine pripadaju različitim organizacijama, može se smatrati da je fizička mašina deljena između više organizacija. Podela na nivou virtuelnih mašina je na najnižem nivou apstrakcije, ali samim tim je i najsigurnija jer se izolacija na nivou hipervizora smatra dovoljno dobrom.

Što je nivo apstrakcije viši u pogledu višeorganizacijskog modela, veće su i uštede. Najveće uštede nastaju usled mogućnosti da se na aplikativnom nivou vrši podela. Tako je na vrhu apstrakcije SkS model.

Softver mora biti napisan tako da zna da deli jednu instancu aplikacije na više organizacija. Što je model deljenja na višem logičkom nivou, sve je potrebniji dobar bezbednosni model fokusiran na izolaciju podataka različitih organizacija. Apstrakcija i bezbednosni model su povezani obrnutom proporcionalnošću. Razlog za sigurnosne probleme je to što se izolacija organizacija prebacuje na aplikativni nivo. Tačnije, pisanje softvera mora da podrazumeva delove funkcionalnosti koji se bave podelom resursa između organizacija.

SkS servis mora održati izolaciju po pitanju bezbednosti i performansi između svih organizacija koje su pretplaćene. Ovaj zahtev primorava na razmatranje višeorganizacijskog modela kao ključni zahtev počevši od rane faze razvoja softvera [9].

Uvođenjem višeorganizacijskog modela, najveći problem predstavlja svakako bezbednosni model i izolacija organizacija, ali isto tako važna stvar je podrška konfigurabilnosti rešenja. Ideja konfigurabilnosti je da se klijentski zahtevi mogu lako implementirati menjajući konfiguraciju softvera. Višeorganizacijski model teoretski treba da obezbedi da ista instanca aplikacije ponudi delimično različit softver različitim organizacijama. Upravo prilagodljivost je najteži izazov pri projektovanju višeorganizacijske aplikacije.

Generičnost jednog softvera se ne može diskretno predstaviti, nego je reč o logičkom kontinuumu (Slika 16). Na jednom kraju kontinuuma postoji generična aplikacija što podrazumeva aplikaciju razvijenu tako da su proširenja moguća menjanjem određenih metapodataka, tj.

promenom konfiguracije. Na suprotnoj strani kontinuuma, postoji konkretna aplikacija čije promene se postižu direktnim menjanjem modela podataka ili menjanjem izvornog koda softvera.

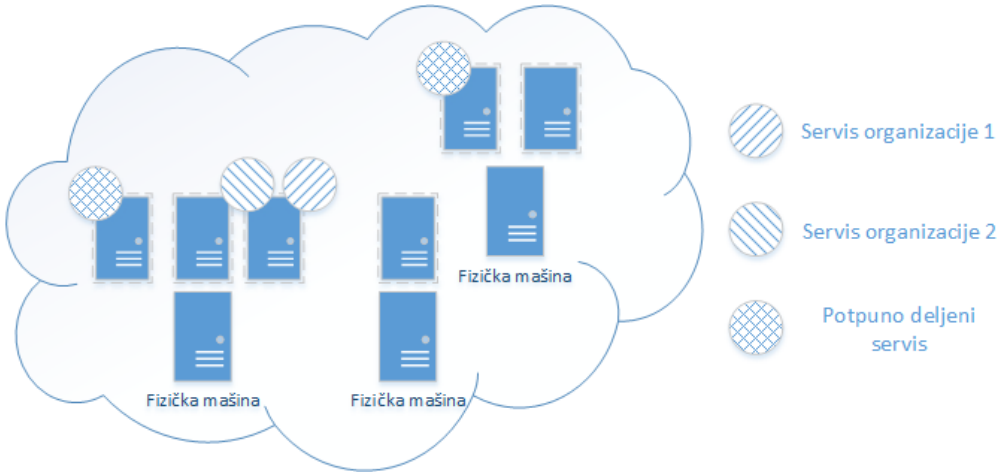
Slika 16 - Izazovi višeorganizacijskog modela

Kontinuum između je prostran i zavisi od vrste softvera koji se razvija. Problem kod višeorganizacijskog modela je pozicionirati se na pravilnom mestu u okviru kontinuuma i razviti softver tako da zadovolji što više mogućih potreba organizacija.

Strelica obeležena kao teže izmene, sugeriše na to da se generični softver jako teško prilagođava u slučaju da promene nisu predviđene konfiguracijom. Konfiguracija obično podrazumeva mogućnost za primenom unapred predviđenih promena. Tako softver postaje veoma kompleksan pa je nepredviđene promene teško sprovesti.

Strelica obeležena kao teža prilagodljivost, sugeriše na nemogućnost prilagođavanja softvera na više različitih organizacija. Nemogućnost prilagođavanja nastaje u slučaju konkretnog softvera bez generičnosti. U tom slučaju, softver će biti odgovarajući za jednu organizaciju, ali je velika verovatnoća da već novu funkcionalnost druge organizacije neće moći da zadovolji usled svoje konkretnosti.

Višeorganizacijski model nad konfigurabilnim softverom zavisi najviše od domena problema. Svaki softver, čak i u istom domenu je individualan jer njegov razvoj zavisi od procene sveobuhvatnih klijentskih zahteva. Uopšteno, pri razvijanju softvera se vodi računa o tome da se sagleda što više mogućih zahteva klijenata i potrebnih funkcionalnosti koju softver treba da zadovolji. Iako postoje šabloni vezani za konfigurabilni model podataka, aspekt konfigurabilnog višeorganizacijskog modela neće biti pokriven u implementaciji ovog rada.



Slika 17 - Željeni višeorganizacijski model

Postavka višeorganizacijskog modela (Slika 17) na konceptualnom nivou ukazuje na prednost u odnosu na postavku softvera posvećenog jednoj organizaciji (Slika 13).

Mogu se definisati i formule po kojim se vidi odnos uštede uvođenjem višeorganizacijskog modela na aplikativnom nivou.

Formula 1, bez aplikativnog višeorganizacijskog modela: $n_{vm} = t * si$

Formula 2, sa aplikativnim višeorganizacijskim modelom: $n_{vm} = \frac{t}{mtdg} * si$, gde je:

nvm – broj virtuelnih mašina, *t* – broj organizacija, *si* – instance servisa, *mtdg* – stepen višeorganizacijske iskorišćenosti na aplikativnom nivou koji je jednak broju organizacija koje se mogu u isto vreme izvršavati u okviru jedne instance aplikacije.

Ukoliko se formula 2 pomnoži sa *mtdg*, tada se formula 1 i formula 2 mogu izjednačiti po desnoj strani. Ukoliko se formule izjednače, vidi se da je ušteda virtuelnih mašina sam broj *mtdg*. Iz ovoga se može zaključiti da će ušteda resursa zavisiti od servisa do servisa, jer konstanta *mtdg* zavisi od opterećenosti servisa, implementacije višeorganizacijskog modela na aplikativnom nivou i kritičnosti konkretnog servisa.

Može se uzeti kao primer servis koji vrši autentifikaciju korisnika na sistem. Ovakav *web* servis, ukoliko se napiše tako da podržava višeorganizacijski model, može postići visoku vrednost za *mtgd*.

Na kraju, potrebno je napomenuti da je glavni poslovni cilj *cloud* baziranog višeorganizacijskog SkS da servis bude korišćen od strane velikog broja krajnjih korisnika koji pripadaju različitim organizacijama. Na osnovu masovnog broja korisnika, postoji mogućnost za uštedu instanci aplikacije, što implicira manje potrebnog hardvera, licenci, i održavanja. Ukupne uštede dovode do sniženja cena za krajnje korisnike i veću zaradu za pružaoce usluga.

3.5.3. Skalabilnost

Skalabilnost je neophodna osobina u slučaju višeorganizacijskog softvera. Da bi se pojasnili razlozi, prvo je potrebno definisati skalabilnost. Definicija koja obuhvata tri dimenzije generalne skalabilnosti se može naći u [80]:

1) Po veličini: sistem može biti skalabilan po svojoj veličini, što se odnosi na to da se uvek mogu dodati korisnici u sistem, ali i resursi.
2) Geografski skalabilan: sistem gde se korisnici i resursi mogu nalaziti na velikim udaljenostima.
3) Administrativno: sistem može biti administrativno skalabilan u smislu da se može lako upravljati sistemom čak i ako se odnosi na nezavisne administrativne organizacije.

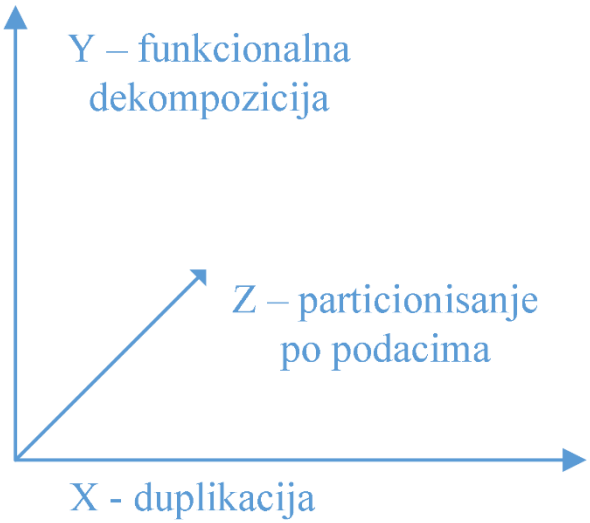
Što se više dimenzija skalabilnosti zadovolji, performanse sistema opadaju kako se sistem skalira. Što se tiče višeorganizacijske platforme koja treba da podrži transformaciju softvera na višeorganizacijski model, ona bi trebalo da zadovolji sve tri dimenzije, što će biti razmatrano u pristupu za realizaciju višeorganizacijskog softvera.

Ako se uzme u obzir skalabilnost po veličini, arhitektura samih rešenja određuje da li je ta dimenizija skalabilnosti zadovoljena. Skalabilnost može biti zadovoljena horizontalnom ili vertikalnom distribucijom.

Karakteristike vertikalne distribucije su da se ostvariju tako što se logičke celine postavljaju na druge fizičke mašine. Ovim tipom distribucije je uslovljena višeslojna arhitektura aplikacija koja se uobičajeno sastoji od korisničkog interfejsa, komponente koje procesiraju poslovnu logiku, i sloja podataka. Različiti slojevi se odnose na logičke organizacije softvera. Iz ugla održavanja sistema, vertikalna distribucija može pomoći jer su logički odvojene funkcije na različitim fizičkim komponentama. Međutim, vertikalna distribucija je jedan način organizacije klijent-server arhitektura, i ona ima svoja ograničenja koja su direktno određena fizičkom komponentom nad kojom se izvršava. Stoga u modernim arhitekturama se ide na distribuciju klijenata i servera koja se naziva horizontalnom distribucijom.

U horizontalnoj distribuciji, klijent ili server mogu biti fizički podeljeni na logički jednake celine, ali svaki deo se izvršava nad svojim skupom podataka što omogućava da se ukupno opterećenje balansira na više fizičkih mašina. Horizontalnu distribuciju je mnogo teže postići u praksi jer podrazumeva arhitekture koje rešavaju probleme distribuiranog programiranja, zadovoljavajući sledeće karakteristike: mašine nemaju kompletno stanje, mašine donose odluke bazirane na lokalnim informacijama i ispad bilo koje mašine ne dovodi do rušenja sistema. Nekada bezbednost informacija ili generalne performanse sistema zahtevaju određenu centralizaciju serverskih programa ili podataka, što onemogućava horizontalnu skalabilnost.

Kako je vertikalna skalabilnost primamljivija opcija, moderne arhitekture idu na dekompoziciju problema na minimalne celine, kreirajući popularno zvanu arhitekturu mikroservisa opisane u odeljku 3.4.2, vodeći računa o minimizaciji servisnog funkcionalnog opsega. Da bi se mikroservisi definisali i napravili, obično se pribegava dimenzijama za dekompoziciju (Slika 18).



Slika 18 - Dimenizije za mikroservisnu dekompoziciju

Da bi se razumela skalabilnost mikroservisa, mora se pojasniti da mikroservisi u stvari dovode do izbegavanja implementacije horizontalne skalabilnosti, nego čine sistem skalabilnim tako što će se dovoljno male celine moći da se skaliraju vertikalnim skaliranjem. Sledeće dimenzije omogućavaju simplifikaciju servisa:

1) Y osa – funkcionalna dekompozicija. Osigurati da se SOA primeni na minimalne logičke jedinice koje imaju smisla za krajnjeg korisnika, ili barem vrše važne funkcije za ostale mikroservise u sistemu.
2) X osa – duplikacija, tj. kloniranje omogućava kreiranje servisa koji zadovoljavaju osobinu visoke dostupnosti, balansiranje opterećenja, a i geografske skalabilnosti. Moderne PkS omogućavaju trijade mikroservisa koje su po stanju poravnate, a izvršavaju se na tri različite fizičke mašine. Konfiguracija je aktivni-pasivni-pasivni.
3) Z osa – particionisanje po podacima. Idealan kandidat za višeorganizacijsku platformu jeste razdvajanje resursa po organizacijama.

Upravo je Z osa prirodna prednost za višeorganizacijske sisteme na kojoj se baziraju moderne PkS. *Cloud* sistemi su višeorganizacijski po svojoj prirodi, ali su i namenjeni višeorganizacijskim softverskim rešenjima jer jedino ušteda troškova dovodi do velike prednosti korišćenja neutralne treće strane za izvršavanje višeorganizacijskih servisa. Iz tog razloga, PkS se obično baziraju na skalabilnosti mikroservisa. Nesreća je što mikroservisne arhitekture nisu uvek primenljive, i nisu uvek najbolji odabir za rešenje određenih problema.

U skladu sa ciljem ovog istraživanja, a to je migracija postojećih softverskih rešenja, ne može se razmatrati idealan slučaj da je softver sačinjen od mikroservisa. Čak se može reći da je mikroservisna arhitektura popularizovana razvojem ponude PkS raznih vodećih *cloud* pružaoca usluga, što je trend u poslednjim godinama (do 10 godina). U tom smislu, verovatnoća da je softversko rešenje koje treba migrirati na *cloud* platforme bazirano na mikroservisima je minimalna. Isto tako, predlagati kompletna rearhitektuisanja rešenja na mikroservis baziranu je neozbiljan pristup iz dva razloga: nekada rešenja nisu podložna za ovaj tip arhitekture, a nekada je toliko uloženo u osnovni softver da prosto nije isplativo praviti proizvod iz početka.

Neko međurešenje jeste migrirati postojeće rešenje takvo kakvo jeste, a kasnije delove konvertovati u mikroservise i novi razvoj bazirati na mikroservisima. Konvertovanje rešenja iz tradicionalnog u mikroservisnu arhitekturu je prikazano u radu [81]. Kako načiniti da postojeće rešenje postane višeorganizacijsko nije nešto što i jedan PkS pružalac usluge može univerzalno odgovoriti, i stoga je to predmet ovog istraživanja koje ima za cilj da predloži platformu koja će upravo to omogućiti.

3.6. Pristupi za realizaciju višeorganizacijskog svojstva

U ovom poglavlju je pružen teorijski uvod koji bi trebalo da opiše osnovne šablone, koncepte, i pristupe pri realizaciji višeorganizacijskog modela rešenja. Opisani su ključni šabloni za višeorganizacijsku podelu na sloju podataka i bezbednosnog modela koji je od ključne važnosti za implementaciju rešenja koje predlaže ovaj rad.

Višeorganizacijska infrastruktura treba da uzme u obzir sledeće ključne aspekte [29]:

1. Izolacija resursa: Razdvojiti alokaciju i upotrebu resursa između različitih organizacija.
2. Bezbednost: Sprečiti nedozvoljen pristup resursima i potencionalni zlonamerni napad.
3. Prilagodljivost: Podržati zahteve specifične za određene organizacije.
4. Skalabilnost: Skalirati SkS tako da se podrži rastući broj organizacija.

Iako je uobičajeno za SkS aplikaciju da se sastoji od instance aplikacije i baze podataka, ovo poglavlje objašnjava višeorganizacijski model na nivou PkS v2 mogućnostima, a potom na nivou sloja podataka.

Instanca servisa može da sadrži korisničke interfejse, poslovnu logiku, kao i ostale razne procese. Razlog što je poglavlje orijentisano na PkS v2 i sloj podataka je to što većina postojećih servisa čuva podatke u relacionoj bazi podataka i moguće je praviti analizu u generalnom slučaju, a ideja je migracija na javna *cloud* okruženja gde je neophodno analizirati PkS v2. Postoje razvijeni dizajn šabloni koji predlažu moguća rešenja na nivou sloja podataka.

Servisi sadrže uglavnom poslovnu logiku koja se implementira u zavisnosti od domenske logike tako da se ne može opisati generalni pristup rešenja višeorganizacijskog modela na nivou predefinisanih šablona. Kada su servisi bez stanja, tada je skoro trivijalno implementirati ih tako da budu višeorganizacijski orijentisani, ali je u realnosti ciljni proizvod mešavina servisa bez stanja i sa stanjem.

Kada su baze podataka u pitanju, postoje generalno tri pristupa za rešavanje izolacije podataka i oni se mogu kombinovati. Pristupi su: odvojene baze podataka, deljena baza podataka sa zasebnim šemama po organizaciji i potpuno deljena baza podataka.

3.6.1. PkS v2 višeorganizacijski mehanizmi

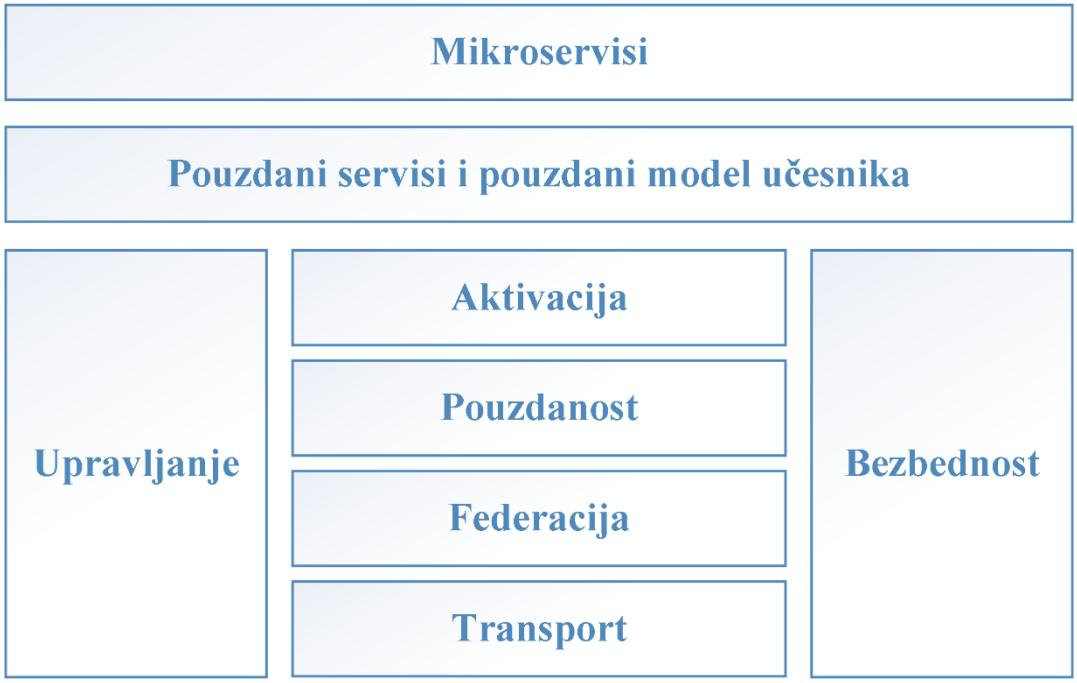
U poglavlju 3.4.1 su opisana PkS rešenja, sa akcentom na PkS v2 rešenja kao novim pravcom koji diktira i određeni tip dizajna samog rešenja, a i veću integraciju sa *cloud* okruženjima. Može se posmatrati i kao visok stepen automatizacije distribuiranog okruženja sa dodacima koji rešavaju upravo glavne probleme distribuiranih sistema kao što su deljeno stanje između više procesa, proces nadogradnje servisa i komunikacioni sloj. Negde se sreću termini da PkS v2 u stvari predstavlja *cloud* operativni sistem, iako se ovaj pojam nije proširio u industriji niti u akademiji.

Kako je već pojašnjeno, bez gubitka na opštosti, u ovom radu će se u svrhu eksperimenata koristiti konkretna PkSv2 *Microsoft Service Fabric*, a u ovom poglavlju će biti analizirana iz ugla višeorganizacijske osobine.

Pri migraciji tradicionalnih rešenja na *cloud*, retko je moguće kompleksnija rešenja prilagoditi PkS v2 modelu jer obično su razvijana sa drugim dizajnom i drugim potrebama na umu.

Međutim, prednost jednom migriranog softvera na *cloud* je da njegov novi razvoj može biti razvijan koristeći benefite PkS v2, što je i u ovom radu iskorišćeno na primeru proširenja NDMS softvera sa IoT baziranim funkcijama. NDMS je opisan u 4.4, a IoT proširenje u vidu funkcionalnosti upravljanja opterećenjem u 4.5.

U opštem slučaju, rad [82] sumira razloge zbog čega IoT prosto zahteva hibridno *cloud* okruženje. Koliko god da mreže postanu brze, latencija će uvek postojati. Zaključak je da većina problema performansi rešenja se neće rešiti novom opremom, infrastrukturom nego novim dizajnom. Iz tog razloga, javna *cloud* okruženja sa svojim PkS v2 nekada ne mogu da pruže rešenje. IoT zahteva prisutnost procesne moći blizu uređaja, ili barem postavku takvu da između javnih *cloud* okruženja i stvari na internetu postoje procesori podataka. Kod *Microsoft Service Fabric* (MASF) opcije je dobro što podržava i privatni *cloud*, a samim tim se može i napraviti hibridna verzija platforme.



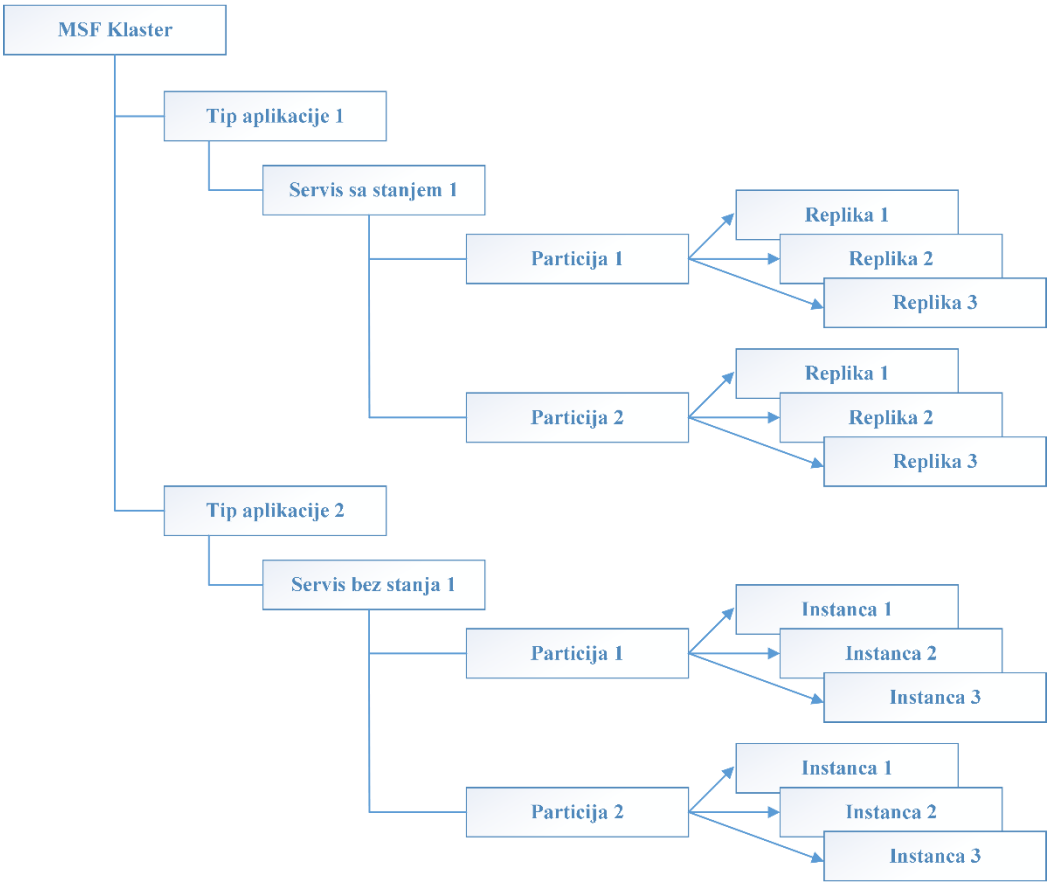
Slika 19 - *Service Fabric* Arhitektura

Arhitektura MASF (Slika 19) se sastoji iz pet gradivnih celina, opisanih u narednim pasusima.

1) Transport: omogućava pouzdanu razmenu poruka između svih komponenti.
2) Federacija: koordinacija resursa u klasteru. Klaster predstavlja skup čvorova koji mogu biti virtuelne mašine ili fizičke, i povezane su mrežom. Čvor je adresabilna jedinica u klasteru, koja se može dodati i skloniti iz klastera. Fedracija se brine i o visokoj dostupnosti tako što ukoliko primeti da je čvor ili servis rezultovao otkazom, poziva ponovno pokretanje, prebacivanje na sekundarnu repliku ili prebacivanje mikroservisa na druge čvorove.
3) Pouzdanost: servis pouzdanosti omogućava replikaciju stanja između čvorova kao i proglašavanjem glavne instance (uvek postoje tri instance servisa, gde je jedna glavna a ostale dve sekundarne). Replikacija stanja rešava problem distribuiranih mikroservisa sa stanjima i stoga predstavlja integralnu komponentu modernih PkS v2. Model učesnika, opisan u 3.4.3 se oslanja na ovaj servis, a baziran je na komponenti (4).

4) Aktivacija: servis aktivacije omogućava logičku distribuciju paketa aplikacija nad svim čvorovima i participama, gde se mogu pokretati više instanci, različite konfiguracije kao i različite verzije servisa u okviru istog klastera.
5) Upravljanje: servis upravljanja se brine o čitavnom životnom veku aplikacije, kako obezbediti resurse, izvršiti automatske postavke nadogradnji servisa, vraćanje verzija unazad (engl. *rollback*), kao i nadgledanje.

Svaki čvor u klasteru može imati više aplikacija koje predstavljaju izolovane celine životnog ciklusa. Jedna aplikacija je sačinjena od jednog ili više servisa, a servis se sastoji od izvornog koda, podataka i konfiguracije. Na osnovu paketa, MASF postavlja servis uvek u tri instance, pri čemu je jedna glavna, tj. aktivna instanca a druge dve predstavljaju redundansu. U PkS v2, tri instance su uvek u različitim domenima otkaza što omogućava visoku dostupnost.



Slika 20 - Klaster i particionisanje

Particionisanje nije jedinstveno za MASF PkS v2, nego predstavlja poznati šablon za kreiranje skalabilnih servisa. U širem smislu, predstavlja koncept podele stanja i procesiranja u manje jedinice kako bi se poboljšale performanse i skalabilnost. Ovo je idealan kandidat za potrebe višeorganizacijskih postavki jer je stanje po dizajnu odvojeno. Dakle, ukoliko su servisi dovoljno veliki da su potrebni jednoj organizaciji, particionisanje je pravi način. U suprotnom, višeorganizacijsko svojstvo se mora implementirati u samom servisu.

Svaka particija (Slika 20) je izvedena kao tri instance servisa. Isti je slučaj i sa servisima koji upravljaju internim stanjem i onih koji svoje stanje eksternalizuju ili ga nemaju. Ova karakteristika omogućava da višeorganizacijsku osobinu posmatramo kao potpuno zadovoljenom,

jer su odvojeni i podaci i procesori podataka, ali s druge strane to dovodi do kreiranja dodatnih sekundarnih procesa. U tom smislu, particionisanje u svrhu obezbeđivanja višeorganizacijske osobine treba koristiti onda kada sa stanovišta granulacije organizacije ima smisla. Npr. milioni organizacija/korisnika koji po nekada koriste servis, verovatno zahteva da se višeorganizaciono svojstvo implementira na nivou servisa, dok za manji broj organizacija koje relativno stalno koriste servise, particionisanje je pravi izbor, i podrška koju pruža *Service Fabric*.

Particije se smeštaju na različite čvorove u *cloud* okruženju. Ovo im omogućava da rastu do limita jednog čvora. Kako podaci rastu, particije rastu i okruženje rebalansira particije preko svih čvorova. Ovo dovodi i do efikasne upotrebe hardverskih resursa.

Particije se adresiraju preko svog indentifikatora, a sam identifikator predstavlja u svetlu ove disertacije kontekst organizacije, koji se kao centralni fokus načina implementacije istražuje u ovoj disertaciji. Zaključak je da se upravljanje kontekstom organizacije direktno uklapa sa mehanizmima koje nude moderni PkS v2.

3.6.2. Nivo podataka

Jedna od glavnih karakteristika cloud okruženja jeste uska povezanost sa sistemima za skladištenje velikih podataka (engl. *Big data*). Kako je istaknuto u radu [38], zapravo paradigma velikih podataka utiče na bržu adopciju *cloud* okruženja, a razlog je najviše to što one predstavljaju idealan scenario gde je potrebna velika količina računarskih resursa u retkim vremenskim intervalima. Ovaj scenario važi za neki vid pretrage velike količine podataka, bilo zbog traženja novih informacija, bilo zbog redovnih izveštaja koji se zahtevaju u određenim vremenskim periodima. U ovom slučaju, više virtuelnih mašina bi bilo zatraženo od *cloud* okruženja samo za potrebe analize podataka i vraćeno odmah nakon toga, tj. efikasno se koristi elasticitet koji je omogućen.

Ipak, sistemi velikih podataka su obično zasnovani na modelima koji nisu relacioni, jer po definiciji moraju da zadovolje veliku količinu podataka, brz protok podataka, ali i veliku raznolikost u strukturi podataka. U ovom poglavlju, akcenat će biti na relacionim bazama podataka, a iz više razloga, opisanih u narednom pasusu.

Relacione baze podataka su starija tehnologija a samim tim i zrelija, i metode izolacije podataka kod relacionih baza su na neki od načina podržane i u mlađim tehnologijama kao što su sistemi velikih podataka. Tokom migracije postojećih rešenja na *cloud*, vrlo verovatno je reč o relacionim bazama podataka jer bi dizajn današnjih tehnologija verovatno odmah uzeo u obzir i sisteme velikih podataka ukoliko je priroda problema takva. S druge strane, usled prirode problema, i dalje su relacione baze neophodne u slučajevima informacionih sistema sa prirodno relacionim modelom podataka.

Migracija rešenja na nivou sloja podataka gde je ideja da se postignu benefiti prelaskom na sisteme velikih podataka je opisana u radu [41], ali to je van fokusa ove disertacije.

U slučaju relacionih baza podataka, šabloni uključuju tri scenarija: odvojene baze podataka, deljena baza podataka sa zasebnim šemama i potpuno deljena baza podataka. U narednim odeljcima, svaki pristup će biti detaljnije razrađen, a biće i prokomentarisan iz ugla velikih sistema podataka.

3.6.3. Odvojene baze podataka

Čuvanje podataka organizacija u zasebnoj bazi podatka je najjednostavniji pristup izolacije podataka. Računarski resursi i instance aplikacija su generalno deljeni među svim organizacijama, dok se baze podataka alociraju po organizaciji. Bitno je napomenuti da podaci svake organizacije ostaju logički izolovani od podataka ostalih organizacija. Svaka organizacija je povezana sa svojom bazom podataka putem metapodataka. Ostaje odgovornost aplikativnog ili bezbednosnog sloja da spreči organizacije od slučajnog ili zlonamernog pristupa podacima koji pripadaju nekoj drugoj organizaciji podataka.

Slika 21 - Baza podataka po organizaciji

Pristup odvojenih baza podataka (Slika 21) je lako proširiv model podataka za svaku organizaciju posebno, i rešava probleme generičkog pristupa projektovanju baze podataka. U slučaju otkaza, proces oporavka podataka je jednostavan jer je isti kao i klasični pristup oporavka.

Pristup ne dovodi do benefita višeorganizacijskog svojstva jer i dalje postoje troškovi za licence, a i troškovi infrastrukture su takođe veći nego u ostalim pristupima jer se server baze podataka mora skalirati ka najvišim zahtevima koji su u opticaju za datu organizaciju.

Odvajanje podataka korisnika u zasebne baze podataka je najskuplji pristup, ali može imati smisla u kombinaciji sa ostalim pristupima. Posebno kada organizacija želi da integriše već postojeću bazu podataka u postojeće SkS rešenje. Pristup zasebnih baza podatka može biti i opcija u slučaju striktnih bezbednosnih regulativa ili eksplicitnih korisničkih zahteva. Još jedan od razloga gde je ovaj pristup podržan je u slučaju servisa kojima brzina nije kritičan faktor a organizacija insistira da se server baze podataka nalazi lokacijski tamo gde je organizacija i fizički postojana.

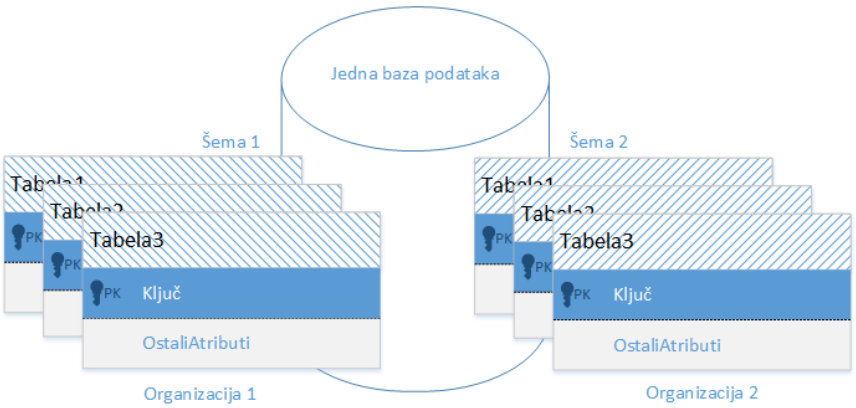
Kada se pogleda iz ugla sistema velikih podataka, jasno je da je ovaj šablon primenljiv, gde bi se čitavi klasteri sistema velikih podataka bili posvećeni samo jednoj organizaciji.

3.6.4. Deljena baza podataka sa zasebnim šemama

Čuvanje podataka u okviru iste baze podataka može se vršiti na više načina u zavisnosti od logičke organizacije podele podataka različitih organizacija. Slika 22 prikazuje pristup u kom su podaci međusobno izolovani tako što se za svaku organizaciju kreira posebna šema u okviru baze podataka.

U pristupu deljenih šema, pri dodavanju novih organizacija, potrebno je da podsistem zadužen za dodavanje novih organizacija kreira skup tabela koje će pripadati novokreiranoj šemi nove organizacije. Sledeće što treba da se uradi, jeste kreiranje novog korisnika baze podataka i

dodeljivanja prava rada nad novom šemom novom korisniku. Čitav postupak se može izvršiti prosleđivanjem odgovarajućih *SQL* komandi sistemu za upravljanje bazama podataka.



Slika 22 - Deljena baza podataka sa zasebnom šemom

Pristup zasebnih šema podataka je relativno lak za implementaciju, a prednost je što organizacije i dalje mogu proširivati svoj model podataka nezavisno od drugih organizacija. Jednom kada se kreira podrazumevajući skup tabela, moguće je vršiti izmene na nivou šeme, što znači na nivou organizacije. Čak je moguće menjati i tabele iz početnog skupa jer tabele iz različitih šema, iako istoimene, su u suštini različite tabele. Pristup zasebnih šema omogućava logički stepen izolacije. Iako su podaci i dalje u okviru iste baze podataka, moguće je razdvojiti pristup različitim šemama i postići odgovarajući stepen izolacije.

U ovom pristupu, katastrofalna greška može nastati pri dodeljivanju podrazumevane šeme određenom korisniku koji predstavlja organizaciju. Ista vrsta greške u pristupu zasebnih baza podataka može nastati ukoliko se metapodaci na aplikativnom nivou ne konfigurišu dobro.

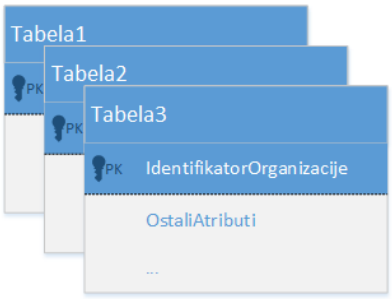
Loša strana pristupa zasebnih šema je to što se podaci organizacije teže mogu oporaviti u slučaju da dođe do otkaza. Ukoliko svaka organizacija poseduje svoju bazu podataka, proces oporavka baze podataka je uobičajen, tako što se podaci oporave iz poslednje sačuvane arhive. U pristupu zasebnih šema, podrazumevano je da ukoliko se vrši oporavak podataka, to ima uticaja na sve organizacije, pa i na one kojima su podaci u konzistentnom stanju. Zbog ovoga, dolazi do komplikovanijeg procesa oporavka. Administratori baze podataka bi morali da oporave podatke na privremeni server, i potom da podatke prekopiraju u produkcioni server. Proces je komplikovan i zahteva vremena, ali se može automatizovati dodatnim razvojem softvera.

Iz ugla sistema velikih podataka, koji su u suštini bazirani na nepostojanju jasne šeme, na logičkom nivou imaju podršku za analognu stvar. Tačnije, svaki od njih zbog optimizacije sistema ima podršku za različite particije na kojima se podaci čuvaju, što se može iskoristiti za logičku podelu na različite organizacije. Npr. u [41] je opisan *HBase* i mehanizam familija kolona, koji omogućava optimizaciju skladištenja podataka koji nisu ista logička celina.

Ekonomski aspekt je zadovoljen, jer se licence sistema za upravljanje bazama podataka u ovom pristupu mogu deliti, kao i infrastruktura koja je potrebna.

3.6.5. Potpuno deljena baza podataka

Poslednji pristup obuhvata korišćenje iste baze podataka i istog skupa tabela u kojima se čuvaju podaci od više organizacija istovremeno. Slika 23 prikazuje pristup potpuno deljenih baza podataka, s tim da svaka tabela u bazi podataka mora biti proširena dodatnom kolonom koja bi identifikovala organizaciju.



Slika 23 - Potpuno deljena baza podataka

Stepen iskorišćenja višeorganizacijske osobine je isti kao i u prethodnom slučaju (deljenje po šemama) s tim da je način implementacije drugačiji. Pristup dozvoljava da se usluži najveći broj organizacija po serveru baze podataka jer je u praksi pokazano da jako velik broj šema utiče loše na administraciju i performanse baza podataka.

Svi podaci su izmešani u istim tabelama, tako da se zahteva više vremena i pažnje za razvijanje bezbednosnog modela. Mora se obezbediti mehanizam da jedna organizacija nikako ne može pristupiti podacima druge organizacije, čak i u slučaju neočekivanih grešaka u aplikaciji ili zlonamernih napada. Mehanizam se bazira na presretanju upita, i detaljno je opisan u odeljku 5.9.

Procedura oporavka podataka organizacije u slučaju otkaza je slična kao i u pristupu deljenih baza podataka sa zasebnom šemom. Međutim, postoji dodatna komplikacija da se torke u produkcionoj bazi moraju izbrisati pre nego što se ponovo upišu preuzimajući ih iz privremene baze podataka. Ako je broj torki u oštećenim tabelama veliki, oporavak ovakvih tabela može izazvati značaj pad performansi za sve organizacije koje dele istu bazu podataka.

Potpuno deljena baza podataka je pristup namenjen onda kada je važno da aplikacija može da usluži veliki broj organizacija sa što manjim brojem serverskih instanci. Naravno, preduslov je da organizacije imaju poverenja i želju za ovako naprednim višeorganizacijskim pristupom. Obično je prirodan izbor u situacijama kada je besmisleno da se odvajaju baze po organizacijama usled količine podataka, njihove senzitivnosti kao i finalne cene servisa.

3.6.6. Izolacija podataka

Pod bezbednosnim modelom za višeorganizacijske servise, podrazumeva se bezbednost izolacije podataka između različitih organizacija. Najveći problem koji nastaje usled mešanja jeste bezbednost podataka, i najveći rizik koji se direktno asocira za višeorganizacijske arhitekture jeste da ne dođe do pristupa podacima različitih organizacija.

Cilj je na nivou svake organizacije obezbediti nivo bezbednosti minimalno kakav je postojao i kod tradicionalnih aplikacija. Zbog poverenja koje se teško zadobija u slučaju višeorganizacijskih modela, realnost je da često treba da postoji veći stepen bezbednosti nego u slučaju tradicionalnih servisa.

Postoje dva izvorna sistema koja mogu dovesti do greške u izolaciji podataka:

1) Na nivou greške u aplikativnom sloju za rukovanjem kontekstom organizacije, i
2) Na nivou sloja za baze podataka

Kada su izvori problema poznati, mogu se postaviti adekvatne bezbednosne kontrole koje će sprećiti da dođe do bezbednosnh propusta. Iz tog razloga se i javljaju dva primarna šablona bezbednosti kod višeorganizacijskih servisa.

Dva pristupa implementiranja izolacije podataka kod višeorganizacijskog modela [29] su: šablon baziran na filtriranju podataka na aplikativnom nivou i šablon baziran na permisijama na nivou sistema za upravljanjem bazama podataka.

Šablon baziran na filtriranju podataka na aplikativnom nivou podrazumeva dodavanje filtera u svaki zahtev za resursom. Na ovaj način, može se osigurati da će podaci organizacije biti dostupni samo organizaciji čiji su podaci. Da bi se mogle staviti efikasne bezbednosne kontrole, bilo bi dobro biti u mogućnosti izdvojiti sav kod vezan za aplikativno filtriranje. Upravo ovo dovodi do pojma konteksta organizacije, što će biti integralna ideja za pravljenje platforme za migraciju postojećih rešenja na višeorganizacijska.

U slučaju pristupa odvojenih baza podataka, filter se zamenjuje adresiranjem korektne baze podataka. U slučaju deljene baze podataka sa zasebnim šemama, filter se bazira na imenu šeme. Za pristup potpuno deljene baze podataka, filter se bazira na *where* SQL klauzuli pri čemu svaka tabela ima dodatni atribut koji se naziva *IdentifikatorOrganizacije*. Vrednost *IdentifikatorOrganizacije* polja je diskriminanta koja određuje različite organizacije.

Šablon baziran na permisijama na nivou sistema za upravljanjem bazama podataka (SUBP) podrazumeva dodeljivanje korisničkog naloga baze podataka za svaku organizaciju posebno. Na nivou SUBP se kreiraju privilegije tako da svakom korisniku odgovara čitanje podataka samo jedne organizacije. Ovaj pristup je primenljiv na pristup odvojenih baza podataka i pristup deljenih baza podataka sa zasebnim šemama. U pristupu potpuno deljenih baza podataka, većina SUBP ne podržava dodelu prava na osnovu vrednosti atributa relacija, tako da se implementacija mora osloniti na prvi pristup.

Izolacija podataka na nivou servisa se mora definisati u okviru same poslovne logike servisa. Ideja u ovom radu jeste da se kontekst organizacije eksternalizuje kako bi se smanjili mogući propusti pogrešnog adresiranja podataka. Tu postoje dva scenarija: servis je bez stanja, tj. stanje je eksternalizovano, kao i servis koji interno čuva stanje.

U slučaju servisa bez stanja, intergracija sa platformom je jednostavna jer pruža čitav kontekst organizacije, i adresiranje pravih podataka se odvija preko predložene platforme. U slučaju servisa sa stanjem, postoje dva pristupa: instance servisa postoje po organizaciji, gde adresiranje i pronalazak već podržava platforma, i sami servisi u svojoj logici implementiraju izolaciju podataka, gde *IdentifikatorOrganizacije* dobijaju od platforme, ali internu podelu stanja je nemoguće rešiti platformski.

3.7. Opis problema i otvorenih pitanja

Migracijom tradicionalnih monolitnih softvera na SkS model, arhitektura softvera mora pretrpeti ozbiljnije promene u svojoj arhitekturi kako bi bila ekonomski isplatljiva. Potrebne promene zavise od početnog stanja softvera, da li je klijent tanak ili debeo, *web* baziran, da li je servisna strana implementirana prateći SOA principe ili nije.

Put tranzicije će podrazumevati kreiranje SOA od monolita, prelazak na *web* klijente u slučaju *cloud* baziranog SkS, a potom na višeorganizacijski model detaljno opisan u poglavlju 3.5. Ukoliko je višeorganizacijski model pravilno implementiran, ostvaruju se uštede u resursima usled veće iskorišćenosti instanci servisa.

Fokus ove disertacije u najužem smislu je olakšavanje tranzicije na višeorganizaijsko svojstvo, tako se u poglavlju 3.7.1 opisuje problem koji je potrebno rešiti platformom za transformaciju rešenja na višeorganizacijski *cloud* bazirani SkS. S obzirom da tranzicija može obuhvatiti mnogo problema koje je neophodno sagledati pri projektovanju platforme, oni su opisani kao ograničenja ove disertacije u poglavlju 3.7.2 jer ti problemi ne predstavljaju fokus analize i rešenje prikazano ovim istraživanjem. Na kraju je data formulacija teza u poglavlju 3.7.3.

3.7.1. Platforma za migraciju na cloud bazirani SkS

Zbog svojih navedenih prednosti, a veoma kompleksne implementacije višeorganizacijskog svojstva, pristup koji se uzima u ovoj disertaciji je kreiranje servisa i softverske podrške za lakšu tranziciju postojećih softverskih rešenja. Zbog svoje pomoćne uloge, naziva se platformom za migraciju na *cloud* bazirani SkS.

Mane višeorganizacijskog svojstva su kompleksnost, veoma težak razvoj, smanjenje fleksibilnosti čitavog rešenja i na kraju najvažnije, veoma visok rizik iz ugla bezbednosti i izolacije podataka. Najveći rizik, a ujedno i razlog za strah kod korisnika koji su svesni da koriste višeorganizacijski servis, jeste narušavanje privatnosti podataka. Dakle, problem izolacije podataka mora biti rešen.

Greška u višeorganizacijskom SkS može koštati neprihvatljivo mnogo. Tačnije, ukoliko dođe do greške u prikazu podataka, može se smatrati da je to kraj poverenja krajnjeg korisnika. Kraj poverenja kod korisnika implicira da je to kraj i za dalju upotrebu konkretnog servisa, a najverovatnije i za organizaciju koja je proizvela softversko rešenje. Pod greškom u prikazu podataka, podrazumeva se greška u kojoj bi se jednoj organizaciji prikazali podaci druge organizacije usled greške u radu višeorganizacijskog servisa.

Jasno je da greška u određivanju organizacije u višeorganizacijskom SkS može biti katastrofalna. Upravo smanjenje rizika za pojavu ovakve greške u višeorganizacijskom rešenju jeste osnovni, ali i najteži problem koji je potrebno rešiti. Sledeće, potrebno je logiku bezbednosti izmestiti u servis, ili implementirati kao softversku podršku, koja bi se koristila kao bezbednosni okvir višeorganizacijskih arhitektura.

Izmeštanje upravljanja kontekstom organizacije donosi više prednosti. Prvo, odgovornost vezana za bezbednost privatnosti u smislu određivanja prave organizacije u svakom zahtevu je jasno podeljena. Razvijani servis za podršku bezbednosnog modela mora da eliminiše mogućnost pojave katastrofalne greške. Tek kada ovaj uslov bude zadovoljen, razvijani bezbednosni servis će

imati jaku upotrebnu moć, jer će tek tada imati smisla njegova višestruka upotreba u više višeorganizacijskih SkS servisa.

Iako je glavni problem i predmet ovog rada jeste tehnološka migracija postojećih softverskih rešenja na *cloud* bazirani višeorganizacijski SkS, ta tranzicija povlači dodatne probleme. Glavna stavka koja bi omogućila razvijanje višeorganizacijskog SkS je postojanje platforme koja podrazumeva bezbednosne mehanizme izolacije podataka. Naredni problem koji je potrebno rešiti je sama podela instance softvera između različitih organizacija. Sumarno, probleme koje je potrebno rešiti jesu:

1. eleminisanje mogućnosti pojave greške koja bi dovela do narušavanja privatnosti podataka,
2. izdvajanje bezbednosnog modela, zbog mogućnosti ponovnog korišćenja i zbog jasno definisanih odgovornosti servisa u servisno orijentisanom okruženju,
3. implementirati efikasnu podelu instanci aplikacije i podataka aplikacije.

S obzirom da navedeni problemi predstavljaju najuži fokus za efikasnu transformaciju na višeorganizacijski model, u narednom poglavlju će biti predstavljena ograničenja koja neće biti razmatrana u okviru ovog rada i iz kojih razloga. Ograničenja je potrebno sagledati da se ne bi izostavili parametri koji bi rešenje onemogućili ili zahtevali značajnu izmenu, ali sami po sebi predstavljaju odvojeni problem koji nije fokus ovog istraživanja.

3.7.2. Ograničenja i pravci daljeg istraživanja

Migracijom na višeorganizacijski model, najviše se otvara pitanje bezbednosti i promene arhitekture. Iako se obe osobine prožimaju u ovom istraživanju, rad sadrži limitacije u pogledu njihove sveobuhvatnosti, te stoga postoje ograničenja u obe oblasti koje će biti iznete u nastavku.

Tranizicijom sa tradicionalnih rešenja koja se postavljaju kod krajnjih korisnika na cloud okruženja, bezbednosni model se menja. U ovom radu će biti obraćena pažnja na delove od interesa za predloženo rešenje, ali rad ne prikazuje prizmu bezbednosti u njenoj potpunosti. Čak i kada se dođe do razlika koje se uvode samo višeorganizacijskim modelom, rad ne obrađuje svaki aspekt, iako se tokom istraživanja jeste obratilo pažnje na ovu oblast. Bezbednost u *cloud* okruženjima uvodi nova polja o kojima treba voditi računa [83]:

1. kompleksnost sistema je veća što unosi nove površine za napad,
2. geopolitičke granice moraju da se poštuju zbog zakona o podacima,
3. deljeno (višeorganizacijsko) okruženje – otvara polja bezbednosti višeorganizacijskog modela | izolacija podataka je obrađena, ali komšijski napadi npr. nisu,
4. korisnici koji ne poštuju politiku *cloud* okruženja – „nezgodne komšije“,
5. bezbednost samih *cloud* servisa kojih je puno i koji se koriste u *cloud* servisima,
6. maliciozni insajderi – teško se identifikuju u *cloud* okruženju zbog pristupa velikog broja korisnika usled višeorganizacijske osobine. Mogu upotrebiti razne tehnike za preuzimanje virtuelnih mašina jer su hipervizori deljeni, kao što je opisano u radu [84],
7. gubljenje ili curenje podataka usled višeorganizacijske prirode *cloud* okruženja,
8. model autorizacije je drugačiji – uključuje više tela koja se bave identifikacijom korisnika, pa se autorizacija svodi na generisanje bezbednosnih žetona tvrdnji. U ovom rešenju je

iskorišćen sličan pristup, ali se sam rad ne bavi istraživanjem najpogotnijeg modela za višeorganizacijsku implementaciju.

Dodatno na rizike koje unose *cloud* okruženja, iako većina njih izazvana upravo zbog višeorganizacijske osobine, unose se rizici kao što su prikazani u [85] a odnose se na bezbednosne aspekte prelaska na višeorganizacijski nivo. Izneti rizici su sledeći:

1. izolacija podataka – tehnike izolacije su obrađene u odeljku 3.6.2, ali ciljno rešenje se ne bavi izolacijom podataka na nivou ciljnih servisa, nego pružanju informacija o organizaciji za svaki poseban zahtev (kontekst organizacije),
2. međusobno ometanje organizacija usled svojih opterećenja – ovaj aspekt nije obrađen u samom rešenju, jer bi suštinski predstavljao odvojenu komponentu koja se bavi bezbednošću ovog aspekta,
3. dodeljivanje resursa korisnicima koji su neprovereni – davanje autorizacije korisnicima servisa se odvija od strane samih organizacija, kojih pružaoc servisa nije svestan. Ovaj problem se rešava modulima koji omogućavaju reviziju, što nije obrađeno u ovom radu,
4. nekordinisane i opasne promene konfiguracije – zbog svoje kompleksnosti, višeorganizacijske postavke moraju imati jasno dokumentovan proces promene, i mapu zavisnosti komponenti kako bi se softverom moglo upravljati,
5. enkripcija podataka se odvija sa istim algoritmom za sve organizacije – aspekti enkripcije nisu obrađivani u ovom radu,
6. model autorizacije obuhvata mehanizme sa velikim brojem korisnika, autorizacionih pravila kroz konfliktne domene politike – nije obrađivan ovaj aspekt u ovom radu.

Navedeni problemi predstavljaju pravac istraživanja za sebe, uglavnom baziran na polje bezbednosti višeorganizacijskih sistema. Bezbednost je uvek u fokusu višeorganizcijskog modela iz razloga što strah od problema privatnosti i bezbednosti je najveća prepreka u usvajanju višeorganizacijskog modela. Iako se ovaj rad bavi eksternalizacijom bezbednosti u pogledu izolacije organizacija, ne bavi se zasebno navedenim poljima bezbednosti. Suštinski, svaki od ovih nedostataka predstavlja moguće buduće istraživanje koje bi upotpunilo predloženo rešenje.

Druga oblast je svakako tranzicija na višeorganizacijski model. Predloženo rešenje omogućuje kontekst organizacije i time olakšava prevođenje postojećih softverskih rešenja na višeorganizacijski model. Ono čime se rešenje ne bavi jeste delineacijom poslovne logike na više organizacija samih ciljnih servisa koji treba da se modifikuju. Istraživanja koja se bave ovim aspektom praktično i ne postoje iz prostog razloga što je svaki softver konkretizacija problema u domenu, sa implementacijom koja je određena znanjem i iskustvom one organizacije koja je razvijala softver. Dakle, ne mogu se izvesti generalizacije nad postojećim softverskim rešenjima, nego se mogu izdvojiti šabloni za neke slučajeve koji bi bili opisani na visokom nivou. Svakako, ta migracija bi se oslonila na predloženo rešenje ovim radom, i svelo bi se na inženjersku disciplinu razvoja koda u skladu sa novim ograničenjima.

Na osnovu ovog odeljka, izneti su problemi koji svaki za sebe predstavlja eventualne pravce daljeg istraživanja, a ujedno predstavljaju i odvojene naučnoistraživačke oblasti. Za potrebe ove disertacije važe sledeće pretpostavke:

1. *cloud* okruženja su bezbedna,

2. bezbednost *web* klijenata se implementira na isti način kao i kod tradicionalnih servisa,
3. perimetar unutar *cloud* okruženja gde se predloženo rešenje izvršava je bezbedno,
4. za ciljno rešenje, problem malicioznih korisnika se ne rešava predloženim rešenjem,
5. servis identifikacije i autorizacije je implementiran da podrži tražene funkcionalnosti,
6. proizvoljna modifikacija ciljnih arhitektura nije obrađivana, nego se kreira platforma koja će omogućiti prilagođavanje postojećih arhitektura.

Shodno dosadašnjem izlaganju i opisu problematike višeorganizacijskog modela, u narednom odeljku će biti definisane hipoteze ove disertacije u skladu sa onim što ciljno rešenje, tj. platforma za migraciju postojećeg softvera na *cloud* bazirani višeorganizacijski SkS želi da postigne.

3.7.3. Hipoteze

Glavna korist višeorganizacijskog modela je njegova ekonomska isplativost. Ona se oslikava na uštedi resursa, ali i na uštedi vremena koje se troši u održavanje servisa. Kako je fokus ove disertacije na tehnološkoj transformaciji, od interesa su uštede koje nastaju na tehničkom nivou, tačnije resursi koji se plaćaju u okviru *cloud* okruženja. Iz tog razloga, proističe prva hipoteza ove disertacije.

H1: Predloženo rešenje može da pruži iste funkcionalnosti ka više organizacija sa ukupno manje resursa – može se egzaktno prikazati koliko resursa bi moglo da se uštedi

Na osnovu predloženog rešenja, moguće je prevoditi postojeće softverske sisteme na višeorganizacijski model. Da bi postojao fokus i merljivi pokazatelj potencijalnih ušteda, u skladu sa temom ove disertacije, uzeće se postojeći sistem za upravljanje distribucijom električne energije opisan u odeljku 4.4, gde će se potencijalne uštede izmeriti i prikazati u odeljku 7.4.5.

Osim ekonomske dobiti, ono što čini višeorganizacijske sisteme interesantnom opcijom, jeste postizanje veće visoke dostupnosti i ukupnu robustnost rešenja. Odatle proizilazi druga hipoteza ove disertacije.

H2: Predloženo rešenje može da omogući veću visoku dostupnost (engl. *high availability*)

S obzirom na to da je visoka dostupnost osobina koja se može pripisati arhitekturi kao takvoj, ova hipoteza neće biti eksperimentalno testirana, nego će se na teorijskom nivou obraditi u odeljku 7.4.2.

Uz benefite koje donosi višeorganizacijski model, mane modela su kompleksnost, bezbednost i ukoliko se ne dizajnira nova arhitektura rešenja, sasvim sigurno se uvodi slabljenje performansi iz razloga što se postojeće rešenje proširuje proverama vezanim za organizaciju. Kako je opisano u odeljku 3.7.2, ovaj rad se ograničava od istraživanja celokupne bezbednosti višeorganizacijskog pristupa. Kompleksnost i slabljenje performansi formulišu treću hipotezu disertacije.

H3: Predloženo rešenje uvodi slabljenje nefunkcionalnih karakteristika – može se egzaktno odrediti koliko je slabljenje performansi usled korišćenja platforme

U svakom eksperimentu, od ukupno četiri, merene su performanse kao jedan od izlaznih rezultata kako bi se u skladu sa hipotezom 3 pokazalo slabljenje performansi. U odeljku 5.4.5 je prikazano egzaktno koliko su performanse lošije kada se koristi predloženo rešenje na primeru višeorganizacijskog *web* servisa.

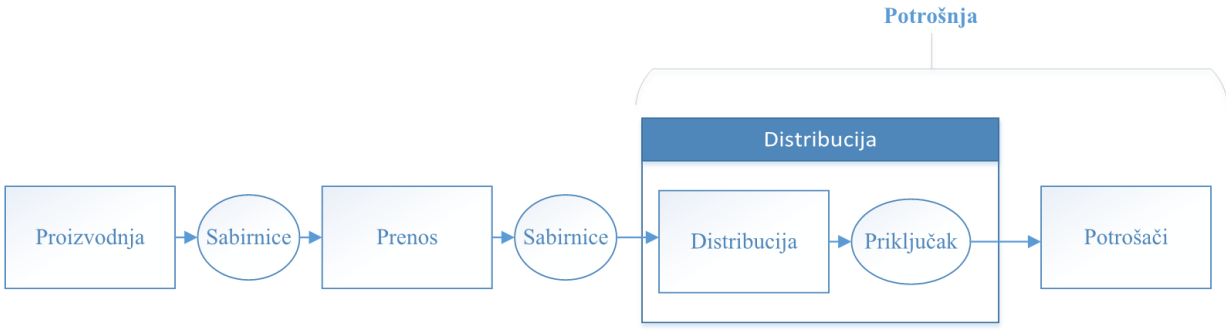
4. Pametni elektroenergetski sistemi

Pametni elektroenergetski sistemi predstavljaju sledeći korak u evoluciji elektroenergetskih sistema. Da bi se pravilno pozicionirali, a posebno da se razumele potrebe za softverom koji bi bio fundamentalni deo modernih pametnih mreža, u ovom poglavlju će se opisati prvo tradicionalne elektroenergetske mreže i njihovi problemi. Potom će se dati opis pametnih elektroenergetskih mreža, da bi se fokusirali na trenutno fundamentalne informacione tehnologije (IT) u službi operativnih tehnologija (OT) namenjenih vođenju distribucije, prvensteveno NDMS sistemu, kao i na nadolazeću funkcionalnost upravljanja opterećenjem.

4.1. Tradicionalne tehnologije elektroenergetskih mreža

Za istorijski početak tradicionalnih elektroenergetskih mreža kakve bi i danas bile da nije došlo do napretka IT tehnologija, može se uzeti završetak rata struja, gde je pobedu osvojio Nikola Tesla naspram Tomasa Edisona, 1890-ih godina. Može se uzeti sajam u Čikagu koji je osvetljen pomoću naizmenične struje 1983. godine kao datum pobede naizmenične struje na osnovu koje je nastala topologija elektroenergetskih mreža kakvu i danas imamo. Današnje elektroenergetske mreže su ishod ubrzane urbanizacije koja je zahvatila većinu sveta tokom prošlog veka. Većinom su elektroprivredna preduzeća usvojila slične tehnologije. Rast elektroenergetskih sistema je bio uslovljen ekonomskim, političkim i geografskim faktorima koji su jedinstveni za svaku elektroprivredu [35]. Uprkos razlikama, osnovna topologija postojećih elektroenergetskih sistema se uopšte nije promenila. Od osnivanja sistema, postojale su jasne granice između generacije električne energije, transmisije, i podsistema distribucije. Kroz ovu podelu, različiti nivoi automatizacije, evolucije i transmisije su postojale za svaku oblast [35]. Slika 24 prikazuje pojednostavljenu strukturu tradicionalnih tehnologija elektroenergetskih mreža, koja sadrži sledeće podsisteme [86]:

1. podsistem proizvodnje čine elektrane, koje se na podsistem prenosa priključuju preko generatorskih sabirnica, tj. sabirnica elektrana,
2. podsistem prenosa čini (upetljana) prenosna mreža, koja se priključuje na generatorske sabirnice, do potrošačkih sabirnica,
3. podsistem distribucije čini distributivna mreža, koja se priključuje na potrošačke sabirnice, do tačaka na koje se priključuju individualni potrošači,
4. podsistem neposredne potrošnje čine individualni potrošači svih distributivnih mreža razmatranog elektroenergetskog sistema, koji se priključuju u priključnim tačkama.



Slika 24 - Tradicionalne elektroenergetske mreže [86]

Problemi tradicionalnih mreža se mogu ukratko rezimirati:

1. unidirekciona priroda dovodi do lančanih ispada kada dođe do kvara na mreži,
2. uvid i kontrola nad servisima i uređajima zahteva prisustvo IT/OT tehnologija,
3. uvid u potrošnju energije kako bi se mogla optimizovati potrošnja i naučiti navike,
4. potrebno je da mreže budu samopopravljive i otporne na anomalije sistema,
5. uključivanje distribuiranih električnih izvora,
6. uključivanje krajnjih potrošača uz omogućenu interakciju sa sistemom tako da mogu da trguju energijom kroz sistem,
7. čuvanje energije ne postoji što dovodi do neophodnog bilansa snaga.

4.2. Pametne elektroenergetske mreže

Tradicionalne elektroenergetske mreže dostavljaju struju od generacije do krajnjih potrošača, bilo individualnih ili industrijskih. Ovaj proces se izvrši kroz podsistem transmisije i podsistem distribucije. Elektroenergetske mreže su se odnosile na interkonektovani sistem prenosa koristeći isključivo analogne tehnologije. U odnosu na tradicionalne elektroenergetske mreže, termin pametne elektroenergetske mreže je često korišćen u različitim kontekstima obuhvatajući mnoge funkcionalnosti vezane za modernizaciju elektroenergetskih mreža. U svojoj srži, pametne elektroenergetske mreže koriste digitalne komunikacije i kontrolne sisteme radi nadgledanja i upravljanja tokovima snaga, sa ciljem da se mreža unapredi u pogledu otpornosti, efikasnosti i isplativosti [87]. Pametne elektroenergetske mreže povećavaju konektivnost, automatizaciju i koordinaciju između proizvođača, distributera, potrošača i mreže tako što uvode nove funkcionalnosti kao što je upravljanje opterećenjem, naplatom u realnom vremenu i pametnim brojilima [87].

Moderna pametna elektroenergetska (engl. *Smart Grid*) mreža (Slika 25) nije više unidirekciona kao što je to bio slučaj u tradicionalnim tehnologijama. Može se smatrati da je kroz uključivanje krajnjih potrošača u proizvođače, distributivni podsistem postao spoj generacije (zelena boja za izvore obnovljive energije i crna za distribuirane generatore) i potrošnje. Osim potrošnje i generacije, pametne elektroenergetske mreže uvode skladištenje energije (svetlo zelena) i kroz napredne funkcije upravljanja opterećenjem (žuta boja) virtuelne elektrane (VE).

Upravljanje opterećenjem (engl. *Demand Response* – DR) je funkcionalnost detaljno opisana u 4.5, ali se u suštini svodi na smanjivanje potrošnje u periodima kada sistem ne može da izdrži opterećenje tako što se krajnji potrošači uključe u proces. Krajnji potrošači zarad nekih benefita mogu svesno smanjiti potrošnju u momentima kada je za opstajanje elektroenergetskog sistema to najbitnije, ili čak mogu da prodaju energiju koju generišu nazad u sistem. Čitav koncept dovodi do toga da se može upravljati određenom količinom snage, pa se ovaj koncept naziva i virtuelnim elektranama. Tako, postoje istraživanja koja se bave istraživanjem samih DR programa kako bi povećali efikasnost VE. U [88] se ide na limitiranje fleksibilnosti krajnjih potrošača da učestvuju u DR programu u realnom vremenu, kako bi se napravilo optimalno raspoređivanje potrošnje na nivou virtuelne elektrane. Da je koncept uvođenja VE sve više zastupljen, pokazuju i istraživanja koja pokušavaju da predstave potencijalne generičke protokole za komunikaciju uređaja sa virtuelom elektranom, kao što je *IEC 61850* [89]. Ono što se zaključuje, jeste da se u neku ruku očekuje da elektroprivrede i dalje zadrže centralnu ulogu u sistemu pametnih

elektroenergetskih mreža. Iako će postojati više nivoa integracije i interoperabilnosti, pretpostavka je da će se logički centralizovane odluke odvijati iz pozicije distributivnih kompanija.

Slika 25 - Pametna elektroenergetska mreža [90]

Uključivanje krajnjih potrošača uz postojeće NDMS sisteme detaljno opisane u 4.4, dovodi do količine i kvaliteta podataka koji odgovaraju opisu paradigme velikih podataka:

1. brzina (engl. *velocity*) – moderna pametna brojila očitavaju merenja na svakih 15 minuta, što dovodi do 96 miliona čitanja dnevno za svakih milion brojila,
2. raznovrsnost (engl. *variety*) – raznovrsnost se povećava. Osim senzorskih podataka, podaci koji ulaze u celokupnu sliku su podaci o vremenu, korisničkim navikama, električnim automobilima itd;
3. veličina (engl. *volume*) – Ukoliko SCADA uzorci budu prikupljani svake 3 sekunde nad 10.000,00 signala, veličina takvih uzoraka može dostići i veličine od 1 TB [91].

S obzirom na potrebu za velikim količinama podataka za rad virtuelnih elektrana, u akademiji postoje istraživanja koja se fokusiraju baš na ovaj konkretan problem [92], [93]. Ono što je logičan sledeći korak, jeste baziranje softverskih rešenja na *cloud* tehnologiji čiji jedan od uspešnijih slučajeva korišćenja jeste upravo za potrebe velikih podataka [38]. Posebno, u slučaju upravljanja opterećenjem, postavlja se problem unosa ogromne količine podataka u vremenu, što je takođe motivacija za *cloud* platforme. Iz tog razloga, pravac istraživanja je sve češći u *cloud* smeru. PkS bazirana platforma se predlaže u [39] jer su idealne za potrebe skalabilnosti i

savladavanja velike količine podataka. Iz tih razloga, i u ovom istraživanju je stavljen poseban akcenat na *cloud*, a i na PkS u užem smislu. S obzirom na veliku količinu podataka, sve je češći osvrt na veštačku inteligenciju zbog nemogućnosti čoveka da ekstrakuje informacije na osnovu ogromne količine podataka. To su isto ukazatelji na isplativost *cloud* sistema za pametne elektroenergetske mreže budućnosti [94].

Tabela 4 - Pregled pametnih elektroenergetskih mreža

Značaj	Ime sistema/Opis	D	L
Nadogradnja Mogućnosti Budućnost	**Upravljanje opterećenjem (DR)**	X	X
	Volt-Var optimizacija*	X	
	Distribuirana generacija**	X	X
	Skladištenje energije**	X	X
	Električna vozila**	X	X
	Kućna povezanost (engl. *Home Area Network* - HAN)	X	X
	Mikrogridovi	X	X
Fundamentalne aplikacije	**NDMS (uključujući i EMS)**	X	
	Automatizacija distribucije	X	X
	Automatizacija podstanica	X	
	Pametna brojila	X	X
	Sistem za upravljanje podacima brojila (engl. *Meter Data Management* – MDM)	X	
Osnova	IT Infrastruktura u koju spadaju: Servisna magistrala (engl. *Enterprise Service Bus*), GIS i Upravljanje odnosima sa krajnjim potrošačima (engl. *Customer Relationship Managment*)	/	/
	Toplogija elektroenergetske mreže (Podstanice, napojnice itd.)		
	Telekomunikaciona infrastruktura		
Temelj	Upravljanje inventarom (engl. *Enterprise Asset Management*)	/	/

*Volt-var optimizacija spada u NDMS prema [95] što će se ovde podrazumevati kao tačno

** Ove tri funkcionalnosti spadaju u DERMS koji je deo NDMS sistema, opisano u odeljku 4.4

Tabela 4 daje odličan prikaz vizije pametnih elektroenergetskih mreža, a dobijena je sintetizovanjem istraživanja predstavljenog u [35]. Kolona „Značaj“ predstavlja opis hijerarhijske povezanosti i u jednu ruku se može posmatrati kao kolona zavisnosti, gde bi pravilan redosled čitanja tabele u tom slučaju bio od dole ka gore. Kolona „Ime sistema/Opis“ predstavlja opis sistema koji se nalazi u nekoj od kategorija određene kolonom „Značaj“. Kolone „D“ i „L“ su veoma važne za razumevanje kretanja razvoja pametnih elektroenergetskih mreža jer ukazuju na

uključenost krajnjih potrošača. Znak X označava u koloni „L“ da su krajnji potrošači uključeni u sistem, dok znak X onzačava u koloni „D“ da je distributivna kompanija uključena u sistem.

Vidi se da je fundamentalni softver za pametne elektroenergetske mreže NDMS sistem koji objedinjuje dosta funkcionalnosti navedenih u značaju nadogradnje i budućeg razvoja. Tačnije, značaj NDMS sistema raste u procesu dodavanja novih funkcionalnosti u njegovu objedinjenu platformu. Takođe, može se primetiti da je uključenost krajnjih potrošača sve veća kako se ide ka modernijim funkcijama koje se očekuju u budućnosti.

IT komunikacije koje povezuju sistem pametnih elektroenergetskih mreža su generalno izvan okvira ove teme, jer je fokus na centralizovanoj softverskoj platformi, ali je neophodno zbog eventualnih softverskih modifikacija imati svest o zastupljenim pristupima. U [96] su poredili performanse u četiri pristupa za arhitekture mreže bežičnih senzora u dva scenarija pametnih elektroenergetskih mreža – DR i naplata. Karakteristike poređenih arhitektura su se razlikovale od potpuno centralizovanog do potpuno distribuiranog pristupa. Komparacije su bile bazirane na performansama mreže u smislu potrošnje energije, procesiranja poruka, propusni opseg mreže, skladištenje podataka i finansijski trošak. Istraživanje je potvrdilo da preferabilni pristup ne postoji. Izbor arhitekture će zavisiti od ciljeva distributivnih kompanija, a implementacija će zavisiti od benefita i finansijskog troška jedne arhitekture. Uzimajući ovo u obzir, sva potencijalna softverska rešenja bi trebala da budu u stanju da budu geografski skalabilna.

Gledajući prognoze i IEEE viziju [97] za pametne elektroenergetske mreže 2030. godine i pravac razvoja [98], može se zaključiti da je centar budućnosti postavljen na potrošače. Iz tog razloga, poseban fokus u nastavku će biti na NDMS sistemima i DR funkcijama.

4.3. Informacioni sistemi u modernim distribucijama

Pametne elektroenergetske mreže se razvijaju i trenutno su distribucije širom sveta u procesu transformacije. Sve je više modela za proveru zrelosti distributivnih kompanija i njihovih spremnosti na buduće elektroenergetske mreže. Jedan od primera je dat u [43], a pregled različitih metodologija za ispitivanje segmenata je dat u [87]. Da bi se razumela pozicija očekivanja u budućnosti, u ovom poglavlju je dat sumarni pregled danas zastupljenih tehnologija što je opisano detaljno u [99] odakle je preuzeta i većina opisa data u ovom odeljku.

Informacioni sistemi u modernim distribucijama se mogu grubo okarakterisati sledećim karakteristikama:

I. Monitoring i upravljanje distributivnom mrežom (celokupno ili delimično)
II. Planiranje pogona distributivne mreže i pomoć pri izvođenju radova
III. Očekivan rad u realnom ili blizu realnog vremena
IV. Isključivo prikupljanje podataka vezanih samo za distributivnu mrežu
V. Isključivo prikupljanje podataka vezanih za celokupnu distributivnu kompaniju
VI. Prikupljanje pomoćnih podataka namenjenih za ostale IT sisteme

Tabela 5 prikazuje kako se sistemi koji se mogu sresti u današnjim prosečnim distribucijama mogu okarakterisati. Analizom tabele dolazimo da jasnog razgraničavanja namene informacionih sistema.

Tabela 5 - Karakteristike IT sistema u OT

IT / karakteristika	I	II	III	IV	V	VI
SCADA	X		X			
OMS	X		X			
NDMS	X	X	X			
GIS				X		
CIS					X	
EAM					X	
Vreme						X
MDM				X		
AMI				X		
WMS		X				
WFM		X				

Svi navedeni informacioni sistemi su opisani u nastavku ovog odeljka, ali analizom tabele možemo izdvojiti one sa karakteristikom III jer je za njih interesantno ispitati primenljivost višeorganizacijskih servisa. Dodatno, OMS se smatra integralnim delom modernog NDMS sistema [37], [100] što nas ostavlja sa NDMS i SCADA sistemima. S obzirom da NDMS poseduje više karakteristika od SCADA, moguće je ispitati primenu parcijalne višeorganizacijske arhitekture što čini NDMS najinteresantnijim za primer istraživanja. U nastavku su dati kratki opisi svih sistema.

SCADA – predstavlja sistem koji se koristi za nadzor i upravljanje telemetrisanim tačkama mreže u realnom vremenu. Skoro sva distributivna preduzeća danas imaju implementiran neki od SCADA sistema u kontrolnoj sobi (dispečerskom centru), a uvođenje ovih sistema počelo je još 1960ih godina dok je proširenje bilo tokom 1970-ih i 1980-ih godina [36].

OMS (engl. *Outage Management System*) – predstavlja sistem za detekciju, analizu i upravljanje svim neplanskim ispadima tj. kvarovima, ali i određenim problemima u polju koji ne rezultuju prekidima napajanja. Takođe, jedna od osnovnih svrha ovog sistema je upravljanje ispadima i problemima u uslovima velikih vremenskih nepogoda, oluja, tornada, uragana i slično. Za razliku od SCADA sistema koji uključuje samo telemetrisane tačke, OMS sistem se koristi na nivou cele distributivne mreže uključujući i netelemetrisane tačke. Većina distributivnih kompanija u svetu danas imaju implementiran neki od OMS sistema u kontrolnoj sobi [99].

NDMS – predstavlja sistem za sveobuhvatan nadzor režima distributivne mreže (uključujući i netelemetrisane delove), optimizaciju pogona u realnom vremenu, i planiranje pogona u proširenom realnom vremenu. Suština DMS sistema jeste estimiranje stanja na nivou cele mreže, kao i proračun niza korektivnih i preventivnih upravljačkih akcija na bazi proračunatog režima mreže. DMS je relativno nova vrsta informacionog sistema i svega mali broj preduzeća ima ovaj sistem implementiran, ali promene u dizajnu i pogonu distributivnih mreža ubrzano

podižu potrebu za postojanjem DMS sistema u distributivnim kompanijama [99]. Ako se analizira vodeća literatura, kao što je [35], [37], [99], [100] može se primetiti da će NDMS biti idealna početna tačka za napredne funkcionalnosti pametnih elektroenergetskih mreža što potvrđuje i svrstavanje NDMS sistema u fundamentalni sloj u [35].

GIS – predstavlja sistem koji je osnovni izvor podataka o konektivnosti mreže. Naime, geografska lokacija svakog uređaja, elementa, fazna konektivnost mreže i pripadnost izvodu je smeštena u ovom sistemu. Svako moderno distributivno preduzeće na svetu ima implementiran GIS sistem, s obzirom da je to osnova softverske vidljivosti distributivne mreže u preduzeću [99].

CIS (engl. *Customer Information System*) – predstavlja informacioni sistem u koji se smeštaju svi privatni podaci individualnih potrošača, odnosno kupaca električne energije. Svaki priključak novog potrošača se beleži prvo u CIS sistemu. Svi privatni i kontakt podaci se smeštaju u ovaj sistem. Neretko CIS u sebi sadrži i modul koji je odgovoran za naplatu električne energije, pa se podaci o potrošnji električne energije posledično nalaze ovde. Podaci o potrošnji mogu biti na mesečnom, sedmičnom, dnevnom, satnom ili čak 15-minutnom nivou, zavisno od mogućnosti akvizicije individualne potrošnje. Svako distributivno preduzeće na svetu ima implementiran neki oblik CIS sistema [99].

EAM (engl. *Enterprise Asset Management*) – predstavlja sistem u koji se skladište svi podaci o svakom individualnom objektu koji postoji u distributivnoj mreži. Ovi podaci se naknadno koriste u upravljanju radovima prilikom zamene/popravke elemenata, ili u modelovanju same distributivne mreže.

MDM (engl. *Meter Data Management*) – predstavlja sistem koji je odgovoran isključivo za akviziciju i skladištenje podataka o individualnoj potrošnji električne energije. Podaci iz ovog sistema (kada postoji) se direktno koriste naknadno u obračunu potrošnje električne energije. MDM nije široko implementiran u distributivnim preduzećima, ali se može reći da svako veće moderno preduzeće je opredeljeno za implementaciju MDM sistema.

AMI (engl. *Advanced Metering Infrastructure*) – predstavlja infrastrukturu (hardversku i softversku) za daljinsko očitavanje potrošnje električne energije, daljinsko očitavanje statusa napajanja individualnog potrošača, a sve češće i daljinsko očitavanje vrednosti napona na mestu potrošača. AMI se sastoji od pametnih brojila, koncetratora podataka i softverskog modula koji te podatke prikuplja i smešta na centralno mesto. Kada je u pitanju daljinsko očitavanje, AMI se u preduzećima razlikuju po vremenskom horizontu. Na primer, svi podaci se mogu očitavati na nivou 24 sata, 1 sat, 15 minuta, 5 minuta, ili 1 minut. Što je vremenski horizont manji, to je implementacija skuplja. Za podatke prikupljene na ovaj način se kaže da su kvazi-telemetrisani, iz razloga što vremenski horizont ne odgovara realnom vremenu kao kod SCADA sistema. Status napajanja potrošača se, po pravilu, automatski i direktno prikuplja. Jako mali broj preduzeća ima implementiran AMI na nivou cele mreže, dok je trend implementacija na pilot oblastima. Vrednost potrošnje/proizvodnje sa AMI, neretko se koriste u svrhu ažuriranja prognozirane potrošnje/proizvodnje.

Sistemi vremenske prognoze (engl. *Weather systems*) – predstavljaju sisteme koji prikupljaju trenutne vremenske pokazatelje, ali su u stanju i da pruže prognozirane vremenske prilike. Samo velika preduzeća u razvijenim zemljama imaju implementirane ove sisteme.

WMS (engl. *Work Management System*) – predstavlja sistem kroz koji se vode, beleže i izveštavaju svi planirani radovi na distributivnoj mreži, pri čemu ti radovi ne moraju biti na primarnim elementima mreže i ne moraju da rezultuju prekidom napajanja. Na primer, provera postrojenja, instalacija brojila, radovi izgradnje, i zamena brojila se vode kroz ovaj sistem. Preduzeća koja imaju veliki broj radova na dnevnom nivou, odnosno preduzeća koja upravljaju velikim mrežama, pribegavaju implementaciji ovih sistema iz razloga optimalnog planiranja radova.

WFM (engl. *Workforce Management System*) – predstavlja sistem za smeštanje podataka o ekipama koje rade u polju, odnosno izvode radove. Neretko, ovaj sistem direktno proizvodi platni spisak za ekipe u polju, te je adekvatan nadzor ekipa u ovom sistemu izuzetno bitan. Preduzeća koja imaju veliki broj radova na dnevnom nivou i samim tim veliki broj aktivnih ekipa u polju, pribegavaju implementaciji ovih sistema iz razloga optimalnog planiranja ekipa i njihovih troškova.

S obzirom na istaknutu važnost NDMS sitema, kao i uključivanje krajnjih potrošača u pametne elektroenergetske mreže, u naredna dva odeljka će biti opisani NDMS sitemi i DR funkcionalnosti.

4.4. NDMS

NDMS je vrsta softverskog paketa koji je prepoznat kao ključni softverski paket širokog spektra koji podržava zahteve modernih preduzeća za efikasnim upravljanjem modernih distributivnih mreža [99]. Suštinska karakteristika NDMS sistema je postojanje jednog integrisanog softverskog paketa koji se koristi za sve aktivnosti upravljanja (nadzora telemetrisanih i netelemetrisanih delova mreže, kontrole telemetrisanih i netelemetrisanih upravljačkih uređaja, proračuna režima mreže u realnom vremenu, upravljanja prekidima napajanja, upravljanja mrežom u ekstremnim vremenskim uslovima, upravljanja planskim radovima, optimizacije tehničkih gubitaka, balansiranja faza, redukcije opterećenja, održavanja strujne rezerve, regulacije napona, regulacije reaktivne snage, upravljanja faktorom snage, optimizacije kvaliteta električne energije, održavanja pouzdanosti napajanja, planiranja razvoja mreže, upravljanja radovima proširenja mreže itd). Kako NDMS predstavlja jedan integrisan paket, sve navedene aktivnosti upravljanja baziraju se na zajedničkom jedinstvenom električnom modelu distributivne mreže [99].

Ako sagledamo da je suština NDMS sistema estimiranje stanja na nivou cele mreže, kao i proračun niza korektivnih i preventivnih upravljačkih akcija na bazi proračunatog režima mreže, možemo uzeti da je NDMS sistem za upravljanje elektrodistributivnom mrežom. Upravljanje se vrši kombinovanjem statičkih podataka preuzetih iz *GIS* sistema, dinamičkih podataka prikupljenih iz SCADA sistema na osnovu kojih se vrši proračun tokova snaga, estimacija stanja korišćena pri rukovanju mrežom, optimalno operativno planiranje, uzročno-posledične analize i simulacije [95]. NDMS je kvalitetan onoliko koliko su podaci kojima raspolaže kvalitetni jer nekompletni podaci mogu voditi do netačnih kalkulacija i prouzrokovati rezultate niskog kvaliteta u naprednim energetičarskim funkcijama, kao što je funkcija tokova snaga, proračuni kvarova i analiza relejne zaštite [101]. Kvalitetnijim podacima u GIS sistemu, tako da osim lokacije uređaja sadrži još dodatnih podataka, mogu se postići i kvalitetniji ulazi za NMDS sisteme. GIS

tradicionalno je korišćen u kontrolnim sobama da podržava OMS (engl. *Outage Management Sistem*) sisteme, ali sa više GIS podataka, može se obuhvatiti i podrška za NDMS [101]. Iz tih razloga, GIS igra važnu ulogu kada je reč o integraciji NDMS sistema sa ostalim sistemima u okviru distribucije. Dodatno, evolucijom NDMS sistema su se srodne funkcionalnosti integrisale u okviru NDMS sistema, pa je tako OMS integrisan u NDMS dok se NDMS integriše sa SCADA sistemima, EMS sistemima i AMI sistemima [95], [100]. Relativno nova funkcionalnost koju NDMS uključuje je DERMS (engl. *Distributed Energy Resources Management System*).

DERMS – Sistem namenjen da servisira milione individualnih distribuiranih energetskih resursa i da ih prezentuje operateru sistema kao manji i upravljiviji broj virtualnih resursa. Ove akcije DERMS izvodi u cilju upravljanja mrežom uz minimalne troškove i maksimalni kvalitet energije [95]. Analizom (Slika 25) modernih pametnih elektoenergetskih mreža, vidi se da DERMS zahvata sve osim DR funkcionalnosti:

1. distribuirani generatori:
    a. generatori obnovljivih izvora energije (solarni i vetrogeneratori) – zelena boja,
    b. klasični generatori (koriste goriva) – crna boja,
    c. manje jedinice iza pametnog brojila (npr. solarne ploče na domaćinstvima).
2. skladišta energije – svetlo zelena boja (uključuje i električna vozila).

Iako manje važno iz ugla primene višeorganizacijske osobine na NDMS sistem, ovde ćemo izdvojiti funkcionalnosti koje se u literaturi smatraju da pripadaju NDMS sistemima [37], [95], [99], [100]:

- nadzor (tokovi i estimacija stanja),
- optimizacija (rekonfiguracija mreže, kontrola aktivne i reaktivne snage),
- operativno planiranje (prognoze i skaliranja opterećenja),
- analitičke aplikacije (gubici, kvarovi i proračuni pouzdanosti),
- razvojno planiranje (regulatori napona, kondenzatori u mreži, konekcije potrošača).

U novijoj literaturi se kristališe DR kroz niz različitih pojmova, i obično se predstavlja kao ekstenzija NDMS sistema. Npr. u [102] se istraživanje bavilo računanjem rezerve tako što se u realnom vremenu računanju moguće uštede koje bi se ostvarile od strane krajnjih potrošača, u slučaju hitne potrebe. Na taj način bi se uštedela potrošnja strujnih generatora koji troše goriva. Ukazuje se na obaveznu participaciju krajnjih potrošača zarad postizanja benefita na nivou čitavog elektroenergetskog sistema. Rad je fokusiran na tehničke aspekte, ali se čak pominje i ekonomski aspekti koji su neizostavni deo upravljanja opterećenjem.

U analizi ovog istraživanja, koristiće se NDMS rešenje detaljno opisano u doktorskoj disertaciji [95], dok će relevantni aspekti IT arhitekture biti opisani u poglavlju 7.

4.5. Upravljanje opterećenjem

Upravljanje opterećenjem pruža krajnjim potrošačima mogućnost da učestvuju u elektroenergetskom mrežnom sistemu tako što će smanjiti ili pomeriti opterećenje u vreme najveće potrošnje. DR postiže to da čitav sistem preživi tako što će se rasteretiti opterećenja svesnim učešćem krajnjih potrošaća i njihovim smanjivanjem opterećenja u kritičnim vremenima.

Da bi potrošači bili motivisani za učešće u tom pristupu, čitavo upravljanje opterećenjem se svodi na davanje određenih stimulansa koji se obično svode na zajednički ekonomski prosperitet. Potrošači bivaju plaćeni da pomere svoju potrošnju ili da odustanu od nje, a s druge strane distributivno preduzeće lakše podnosi dodatni trošak nego dodatne investicije u mrežnu infrastrukturu radi slučajeva koji zahtevaju smanjenje opterećenja i koji su generalno retki.

Postoje metode da se krajnji potrošači uključe u DR i to je opisano u različitim programima za DR. Programi su formalizovani kroz ugovore koji definišu učešće potrošača: šta će oni dobiti ako prilagode svoju potrošnju kao odgovor na molbu u realnom vremenu od strane distribucije, a šta će se desiti u suprotnom slučaju, da li će doći do penala ili samo izostanka beneficija [103].

Postoji više izazova kod uključivanja krajnjih korisnika u DR programe. Prvi je razvoj pametnih tehnologija – pametni uređaji mogu značajno unaprediti komunikaciju između krajnjih potrošača i distribucije. Ono što usporava razvoj pametnih elektroenergetskih mreža je nedostatak infrastrukture kao i nedostatak standardizacije. Bez adekvatne infrastrukture, potrošači ne mogu koristiti DR u realnom vremenom i iskoristiti pun potencijal. S druge strane, ulaganje u infrastrukturu je regulisano i bez dodatnih pogodnosti je praktično teško izvodljivo za distribucije.

Osim tehničkih preduslova, postoji još veći izazov, a to je kako uključiti krajnje potrošače. Istraživači [104] se slažu u vezi važnosti dobre definisanosti ugovora koja će dovesti do učešća krajnjih potrošača. Da bi se predložio adekvatan ugovor, esencijalno je razumeti navike potrošača, pogotovo iz ugla potrošnje električne energije. Istraživanjem potrošnje krajnjih potrošaća, može se doći do predloženih ugovora koji bi bili od interesa za krajnje potrošače.

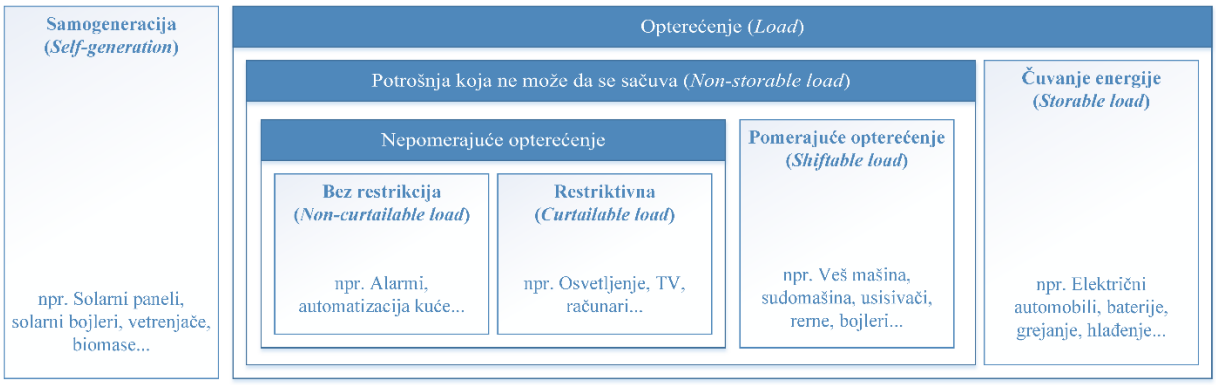
Postoji više ekonomskih modela kao što su: vreme korišćenja (engl. *Time of Use – TOU*) [105], upravljanje opterećenjem u hitnim slučajevima (engl. *Emergency Demand Response Program – EDPR*) [106], naplaćivanje u vremenima kritične potrošnje (engl. *Critical Peak Pricing – CPP*), naplaćivanje u realnom vremenu (engl. *Real Time Pricing – RTP*) i programi direktne kontrole opterećenja (engl. *Direct Load Control – DLC*), kao i programi koji podrazumevaju penale [107].

U vremensko baziranim programima (TOU, RTP, CPP), cena struje je drugačija u drugačijim periodima, npr. najskuplja je u periodima visoke potrošnje i najjeftinija kada je potrošnja mala. Takođe, ne postoje penali u ovim programima. Što se tiče programa baziranim na podsticajima (DLC, EDRP, potrošačka berza), koriste se da ohrabre kupce da troše energiju u vremenima kada je inače slabija potraga za energijom. Neki programi uključuju i penale. Slika 26 prikazuje pet različitih grupa potrošača:

1. čuvanje energije (engl. *storable load*) – potrošnja i krajnja usluga su razdvojeni čuvanjem energije koje može biti u formi baterija ili termalne inercije,
2. pomerajuće opterećenje (engl. *shiftable load*) – potrošnja koja može biti pomerena u vremenu bez da se naruši krajnja usluga,
3. restriktivna potrošnja (engl. *curtailable load*) – potrošnja koja ne može biti pomerena bez afektovanja krajnjeg servisa, ali servis može biti prekinut u bilo kom momentu,
4. stalna potrošnja (engl. *non-shiftable non-curtailable load*) – potrošnja koja ne može biti pomerena bez afektovanja krajnjeg servisa, ali ne bi smelo da dođe ni do prekida servisa,

5. samogeneracija (engl. *self-generation*) – predstavlja generaciju električne energije u neposrednoj blizini potrošača, smanjujući ukupnu potrošnju i time opterećenje distributivne mreže. Može se koristiti i za slučaj rezervnog izvora energije.

Potošaći sa najviše potrošača iz grupe 1 i 5 su najfleksibilniji dok su potrošači sa grupama 4, 3, i 2 manje fleksibilni, u redosledu navođenja grupa [108].



Slika 26 - Tipovi opterećenja [108]

Platforme koje nude potrošačima učešće u elektroenergetskom ekosistemu su danas veoma ograničene. Ovo takođe otvara nove izazove i prilike u razvoju pametnih elektroenergetskih mreža. Da bi se omogućili ciljevi modernih pametnih mreža, mora se staviti poseban akcenat na potrošačke navike i predložiti način na koji će se odvijati komunikacija. U tom smislu, svaki krajnji potrošač, učesnik u sistemu, jeste jedna odvojena organizacija koja uključuje potrebu za višeorganizacijskim pristupom rešenja.

5. Realizacija višeorganizacijske platforme

U ovom poglavlju je dat detaljan opis predloga rešenja postavljenog problema. Izdvojeni su problemi koji su se pojavili u toku implementacije rešenja, opisana njihova priroda i razjašnjeno je zbog čega je izabran određen pravac pri implementaciji rešenja.

Da bi se razumeo raspored ostalih odeljaka ovog poglavlja, potrebno je prvo pojasniti pravac istraživanja. S obzirom da je prelaskom na javna *cloud* okruženja podrazumevano da je IkS višeorganizacijska, kao što je opisano u poglavlju 3.5.1, potrebno je bilo prvo ispitati da li postojeće PkS nude zadovoljavajuću uslugu u pogledu višeorganizacijskog modela što je opisano u poglavlju 3.6.1. U svakom slučaju, za zaokruženu sliku, pravac istraživanja je tekao u sledećem smeru:

1) predložiti rešenje koje omogućuje višeorganizacijsko svojstvo bez vezivanja za konkretna PkS rešenja ili bilo koja druga komercijalno dostupna rešenja,
2) ispitati višeorganizacijske mogućnosti modernih PkS, jer su neizostavna komponenta javnih cloud okruženja, i stoga ih ne treba ignorisati.

Poglavlje je organizovano u više zasebnih odeljaka. U odeljku 5.3 je dat pregled literature, sa fokusom na radove koji su imali najviše uticaja ili se u najvećoj meri mogu porediti sa predloženim tehnološkim rešenjem. Zatim, odeljak 5.4 sažima implementaciju višeorganizacijskog svojstva i u nekoj perspektivi predstavlja sumiranje doprinosa izučavanja oblasti migracije na *cloud* bazirani SkS. Odeljak 5.3 se odnosi na ispitivanje višeorganizacijskog svojstva na nivou PkS v2 i analiza mogućnosti korišćenja PkS za razvoj novih funkcionalnosti nakon migracije postojećeg softvera na *cloud*. Od odeljka 5.4 do kraja poglavlja je dat opis platforme koja predstavlja predloženo rešenje i konkretnim servisima koji je čine. Svaki od servisa rešava pojedinačne probleme.

5.3. Lekcije iz akademije

Literatura je selektovana prateći metodologiju opisanu u odeljku 1.3. Aktuelno stanje u oblasti je opisano u poglavlju 2, i ono sumira deo vodeće literature koja je korišćena i u redosledu koji je prirodan za ovo istraživanje.

U ovom odeljku je zbog velikog opsega radova cilj da se predstave samo oni radovi koji imali direktan uticaj na ishod predloženog rešenja i na način implementacije platforme za migraciju postojećeg softvera na višeorganizacijski *cloud* bazirani SkS. Iz tog razloga, relevantna istraživanja su predstavljena tabelarno (Tabela 6), gde se grupe radova izdvajaju u smislu lekcija u najkraćim crtama koje su proistekle iz te grupe radova.

Tabela 6 - Lekcije iz akademije

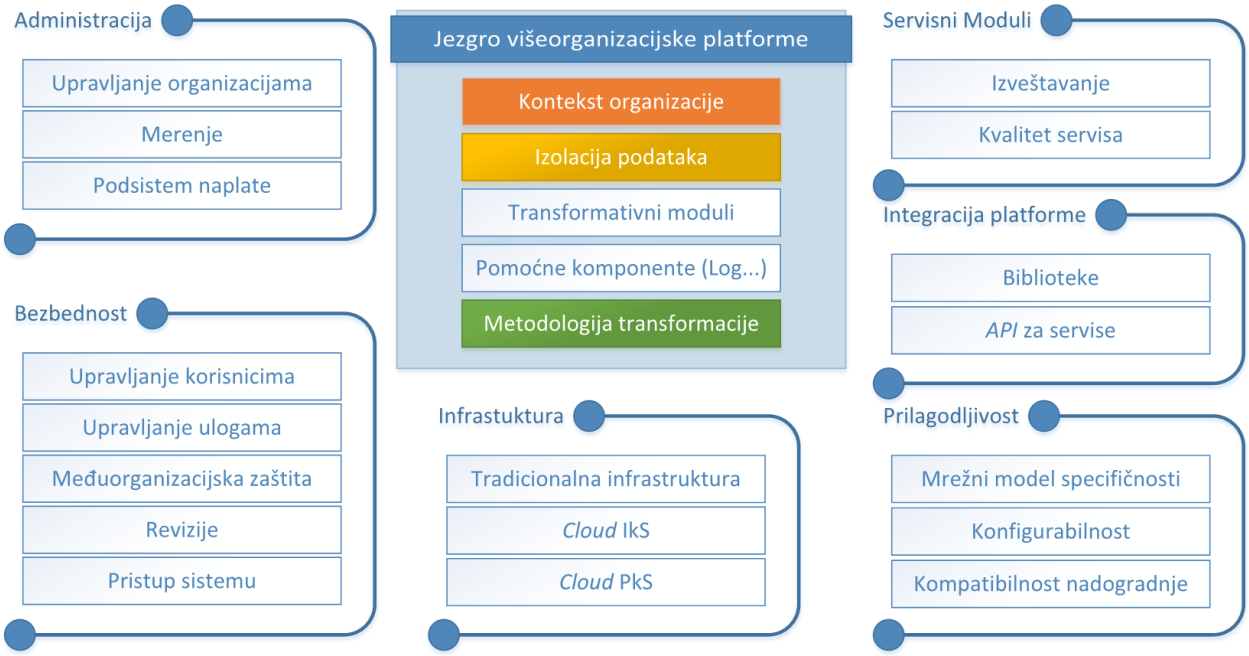
Pravac istraživanja	Radovi	Lekcije
***Cloud* adaptacija rešenja**	[109] [33] [26] [22] [18] [19] [20] [21] [17]	Migracija na *cloud* okruženja ne zavisi od tehnoloških karakteristika samo (postoje i spremnost organizacije koja razvija softver kao i poslovni model). Put do *cloud* baziranog SkS zavisi od početnog stanja softvera. Gotovo uvek u planiranju transformacije je

		cilj da rešenje bude u nekoj meri višeorganizacijsko zbog ekonomske efikasnosti.
Višeorganizacijski dizajn	[110] [27] [31] [32] [111] [112] [113] [30] [114] [34] [115] [116] [117] [118]	Osnovni koncepti višeorganizacijskog dizajna. Izdvajanje bezbednosnog modela. Minimalne modifikacije postojećih softverskih rešenja. Implementacija konteksta organizacija na nivou komunikacije između servisa. Uz kontekst organizacije, ulogu otkrivanja servisa i njihovog balansiranja treba da preuzme neka platforma. Na aplikativnom nivou mora postojati mogućnost izolacije servisnih resursa. Istraživanja u pravcu modifikacije postojećih baza podataka tako da podrže više organizacija gde se kontekst organizacija podrazumeva što sledi da je fokus ovog rada dobar jer takva platforma mora da postoji.
Višeorganizacijska svojstva	[27] [31] [32] [119] [111] [112] [120] [121] [115]	Rad nad podacima više organizacija ne sme da ima vidljive rezultate iz ugla samih organizacija, ali se tehnike mogu koristiti radi dodatnih ušteda u višeorganizacijskom modelu. Modeli zrelosti. Najzrelija rešenja zahtevaju osobinu skalabilnosti. Nivoi usluge moraju biti u korelaciji sa važnošću organizacija odakle sledi da je potrebno pratiti upotrebu svih resursa po organizaciji.
Izolacija	[110] [29] [111] [112] [120] [117]	Izolacija podataka je od ključnog značaja za opstanak višeorganizacijskog modela. Primena šablona nad relacionim bazama podatka. Izolacija na aplikativnom nivou je takođe važna i postiže se uslovima korišćenja servisa. Načini za postizanje aplikativne izolovanosti. U užem smislu bezbednost višeorganizacijskog modela se svodi na osiguravanje izolovanosti podataka.
Prilagodljivost	[27] [62] [122] [123] [124] [125]	Smanjiti potrebe za modifikacijom rešenja generičnošću. Standardizacijom doći do smanjenja razlika između organizacija. Podrška prilagodljivošću putem razbijanja mikroservisa. Kreiranje mapa gde su prilagodljivosti postavljeni tako da se u zavisnosti od organizacije mogu pozvati različite instance servisa. Ovaj pristup ukazuje takođe na oslanjanje na servis koji pruža kontekst organizacije koji je fokus ovog rada.

Uvažavajući prethodna istraživanja i naučene lekcije, u narednim odeljcima ovog poglavlja se razrađuje platforma za višeorganizacijsku transformaciju, koja je najuži fokus ove disertacije. Čitava ideja se zasniva na eksternalizaciji bezbednosnog modela, gde uz što jednostavniju integraciju, postojeći softver može da integriše višeogranizacijsko svojstvo uz proširenje postojećih ulaznih tačaka u sistem, tj. interfejsa. Naravno, upravljanje višeorganizacijskim kontekstom se odvija kroz predloženu platformu.

5.4. *Pregled platforme za višeorganizacijsku transformaciju*

Platforma koja omogućava transformaciju postojećih rešenja na višeorganizacijski SkS se može podeliti na više modula (Slika 27). Prikaz sveobuhvatne platforme je rezultat izučavanja oblasti i literature koja se bavi različitim aspektima podrške za višeorganizacijski servis.



Slika 27 - Sveobuhvatna platforma

Fokus ovog istraživanja je isključivo na jezgru višeorganizacijske platforme. Naravno, da bi se jezgro dobro definisalo i bilo sveobuhvatno, istraživanje svih pratećih oblasti je bilo neophodno, a posebno iz ugla razumevanja celokupnih zahteva koje jedna platforma treba da zadovolji.

Jezgro višeorganizacijske platforme obuhvata sledećih pet modula:

1) Kontekst organizacije – Primarni fokus ovog istraživanja. U literaturi se retko obrađuje problem konteksta organizacije, koji je od suštinskog značaja za transformaciju. U višeorganizacijskim arhitekturima uvek mora biti poznato o kojoj organizaciji se radi, što se ovde naziva kontekst organizacije. Od momenta ulaska zahteva u sistem od jednog od klijenata, mora postojati identifikator organizacije kako bi se aplikativni resursi uključujući i sloj podataka mogli podeliti na zadovoljavajući način.

2) Izolacija podataka – svaki tehnološki servis u jednom rešenju može egzistirati na dva načina: svestan je da radi u višeorganizacijskom režimu gde mu je odmah neophodan kontekst, ili

je nesvestan da je komponenta višeorganizacijske arhitekture. U drugom slučaju, platforma koristeći kontekst može neke zahteve rutirati na sam servis koji je posvećen baš za tu organizaciju čiji zahtev treba da se usluži. U oba slučaja, bez konteksta, servis se ne bi mogao koristiti. Ukoliko je reč o prvom slučaju, sam servis mora biti u stanju da razlikuje podatke različitih organizacija. Za svaku tehnologiju postoje istraživanja, i metodologije podele. U ovom istraživanju su obrađene metodologije raslojavanja podataka u slučaju relacionih baza podataka kao najzastupljenijih kod uveliko postojećih softverskih rešenja. U svakom slučaju, neophodno je razumeti načine izolacije, tako da je izolacija podataka već u sekundarnom fokusu ovog istraživanja.

3) Transformativni moduli – u principu se odnosi na tehničke dodatke ili specifikacije koje postojeće softversko rešenje mora upotrebiti kako bi iskoristilo jezgro višeorganizacijske platforme. Transformativni moduli su prvenstveno biblioteke koje u principu koriste kontekst organizacije u svojim eksternim pozivima servisa.

4) Pomoćne komponente – sve podrazumevane uobičajene komponente treba da budu sastavni deo platforme. Npr. *logging* komponenta mora biti u stanju da razlikuje posebne organizacije. S obzirom na svoju jednostavnost, otočne stvari za podršku nisu u fokusu ovog istraživanja.

5) Metodologija transformacije – U suštini, jedina netehnološka stvar koja je bitna da ide uz platformu. Celo istraživanje se može shvatiti kao deo metodologije transformacije. Jako je važno uvideti sva ograničenja, prvenstveno iz ugla regulativa, bezbednosti, potrebama za prilagođavanje određenog dela sofvtvera, poslovnog modela kao i analize isplativosti da bi se predložio pravi način za kreiranje višeorganizacijskog softvera. Veoma je važno razumeti da ne moraju sve komponente rešenja podržati višeorganizacijsku arhitekturu, ukoliko postoji adekvatno jezgro platforme. Npr. moguće je da je samo sloj podataka višeorganizacijski, a da postoji instanca aplikacije po korisniku, ukoliko to ima smisla.

Infrastruktura obuhvata sledeća tri modula, a važno je podržati što više modula iz razloga generičnosti primene rešenja:

1) Tradicionalna infrastruktura – infrastruktura sačinjena od serverskih fizičkih računara, virtuelizacionog sloja, virtuelnih mašina, sistema skladištenja podataka, ali ne nužno *cloud* okruženja opisana u 3.1. U principu, ova okruženja veoma podsećaju na IkS ponudu *cloud* usluga iz ugla softverskog servisa koji treba da se izvršava u okviru virtuelnih mašina.

2) *Cloud* IkS – Opisana u 3.1. Potrebno je obratiti pažnju na ograničenja koja IkS uvode, prevashodno iz ugla dostupnosti virtuelnih mašina i njihovog životnog ciklusa koji je postojan i unapred definisan. S druge strane, u tradicionalnim infrastrukturama, onaj koji je odgovoran za njihovo održavanje može fleksibilnije da sprovodi procese njihovog održavanja. Iz ovoga zaključujemo da ukoliko se napravi rešenje koje bi podržalo IkS, bez oslanjanja na neke od pomoćnih servisa u *cloud* okruženju, takvo rešenje bi trebalo da zadovoljava fleksibilnije tradicionalne infrastrukture.

3) *Cloud* PkS – Toliko su specifične da osim generalnog opisa u 3.1., dat je pregled iz ugla višeorganizacijkih arhitektura modernih PkS. Rešenja bazirana na PkS su validna, i kao takva ne smeju se izostaviti iz istraživanja, ali je važno napomenuti da dovodi u zavisnost od date komercijalne platforme, što obično dovodi do zaključavanja rešenja na određenog pružaoca *cloud* usluga.

Bilo bi veoma ograničavajuće analizirati višeorganizacijsku arhitekturu na samo jednoj od pomenutih arhitektura, a fokus ovog istraživanja je da predloži rešenje koje bi obuhvatalo sve tri.

Administracija obuhvata naredna tri modula, koja svaka platforma mora imati kako bi bilo omogućeno njeno korišćenje:

1) Upravljanje organizacijama – Dodavanje i uklanjanje organizacija je osnovna funkcionalnost platforme. Dodatno, sva konfiguracija za jednu organizaciju se mora obavljati kroz administrativni panel same platforme. Npr. konfigurisati odluku da li određena organizacija ima zasebnu bazu podataka kako bi se ispoštovali bezbednosni zahtevi određene organizacije, ili deljenu bazu podataka koja je efikasnija iz ugla troška. U okviru predstavljenog rešenja će biti prikazano kako je realizovan modul za administraciju, iako sam modul predstavlja tehničko rešenje koje može biti izvedeno na više načina.

2) Merenje – Potrošnja resursa mora biti merena, što može biti izazovno u višeorganizacijskom režimu rada. Sam modul za merenje može predstavljati novi, ili budući pravac istraživanja jer mogu biti poduzete drugačije metodologije za merenje, ili barem za pretpostavku koliko resursa troši neka organizacija. Sam izbor merenja, tj. šta se meri i na koji način jako zavisi od poslovnog modela, tj. njegove finansijske sekcije gde je definisano na koji način se servis naplaćuje.

3) Podsistem naplate – usko povezan sa sistemom za merenje potrošnje resursa, direktno je određen ugovorom, i njegovom sekcijom o servisnim nivoima (engl. *SLA – service level agreement*). Vrlo interesantna oblast istraživanja može biti razumevanje korisničkih navika i predlaganje drugačijih modela plaćanja, opisanih i u 3.3. Podsistem naplate takođe bi mogao da predstavlja odvojeno istraživanje, i nije u fokusu ovog istraživanja. Takođe, razlike u implementaciji podsistema naplate ne utiču na opštost rešenja predstavljenog ovim istraživanjem.

Bezbednost informacija je jako usko povezana sa pojmom višeorganizacijske arhitekture, jer je osnovni problem strah korisnika usluga da njihovi podaci budu izmešani sa podacima drugih organizacija. Sama pomisao na oslanjanje na logičku izolaciju uliva nepoverenje kod krajnjih korisnika, a nekada su i regulative te koje eliminišu upotrebu podele softverskih servisa. Međutim, sloj bezbednosti se ne odnosi i na ostale uobičajene stvari koje svaki softver mora da pruži:

1) Upravljanje korisnicima – može, a ne mora pripadati podsistemu za administraciju organizacija. Potrebno je omogućiti rad sa servisom identiteta na način da se mogu dodavati i brisati korisnici.

2) Upravljanje ulogama – sistemi se obično oslanjaju na autorizaciju baziranu na ulogama (engl. *RBAC – role based access control)* i proširuju tvrdnjama (engl. *claims based*) u *cloud* okruženju. Neophodno je omogućiti definiciju uloga kao i definisanje permisija za date uloge.

3) Međuorganizacijska zaštita – najkompleksniji problem koji se otvara uvođenjem višeorganizacijskih arhitektura. Obuhvata sledeće aspekte, gde bi svaki od njih mogao biti poseban pravac daljeg istraživanja, ali nije u direktnom fokusu ovog istraživanja:

a) Tehnički problemi jedne organizacije mogu da oštete drugu – arhitektura servisa koji je spreman da radi sa više organizacija mora da ima podršku zabrane potrošnje svih resursa od strane jedne organizacije. Npr. ukoliko bi deset organizacija delile jednu virtuelnu mašinu, potrebno je zaštititi zauzimanje svih računarskih resursa od strane jedne organizacije, što može

biti izazovno za implementiranje. Da bi se ovaj problem prevazišao, obično je potrebno imati servise na četvrtom nivou zrelosti višeorganizacijskih arhitektura opisanih u 3.5.2. Na taj način, svaka mašina može uslužiti zahtev bilo koje organizacije, što dozvoljava da uz dobar modul za monitoring, problem može da se reši namernim isključivanjem procesa na problematičnoj virtuelnoj mašini.

b) Arhiva i povratak sistema – potrebno je rešiti problem osvežavanja podataka u vremenu. Ukoliko je zahtev da je moguće vratiti sistem u određeno vreme u prošlosti, to i nije moguće uraditi na tradicionalan način jer sada sloj za perzistiranje čuva podatke od više organizacija. Arhitektura taj zahtev mora da podrži na drugi način. Povratak sistema u vremenu u slučaju otkaza sistema može doći u slučaju da je došlo do otkaza sistema, što podrazumeva da sistem nije radio ni za jednu organizaciju. Da bi tvrdnja da je sistem dostupan za sve ili nikoga bila tačna, još jednom je potrebno osloniti se na najzreliji pristup višeorganizacijskih arhitektura opisanih u 3.5.2.

c) Logička izolacija korišćenih resursa – višeorganizacijska podela može krenuti od centra podataka, fizičkih servera, sistema stalne memorije, mreže na infrastrukturnom nivou. Da li su jezgra procesora fiksirana po organizaciji ili nisu. U poslednje vreme se sve više usvajaju *cloud* infrastrukture kao zadovoljavajuće pod uslovom da su fizička jezgra procesora posvećena virtuelnim mašinama koje se ne menjaju. Druga strana su kontejneri, koji na nivou operativnog sistema vrše izolaciju resursa između različitih organizacija. Uopšteno, smatra se da je izolacija virtuelnih mašina jača od izolacije koju pružaju kontejneri, iz razloga što u prvom slučaju čitava memorija operativnog sistema na virtuelnoj mašini je posvećena jednoj organizaciji. Kada je reč o višeorganizacijskim arhitekturama, one podrazumevaju izolacije na nivou procesa u operativnom sistemu, što je direktna uloga konteksta organizacije i bezbednosnih mehanizama oko njega da se taj kontekst ne kompromituje. Iz tog razloga, kontekst organizacije predstavlja jezgro platforme za višeorganizacijsku transformaciju i fokus ovog istraživanja.

d) Zloupotreba konteksta– zbog svoje važnosti i direktne povezanosti sa fokusom ovog istraživanja, a to je kontekst organizacije, pristup sistemu je izdvojen kao važan i poseban modul, naveden pod 5).

4) Revizije – zbog svoje bezbednosne osetljivosti, od izuzetne je važnosti mogućnost revizije. Revizije mogu biti interne i u redovnim vremenskim intervalima, ili kao pomoćni alat za potrebe forenzike u slučaju da je na neki način jedna organizacija uspela namerno ili nenarmerno da našteti drugoj organizaciji – npr. izgladnjivanjem resursa.

5) Pristup sistemu – Odnosi se na mehanizme obezbeđivanja da jedna organizacija slučajno ili namerno zadobije kontekst druge organizacije. Ovakva greška bi dovela do najvećeg prestupa zastupljenog u ovakvim arhitekturama, i stoga će na ovom bezbednosnom aspektu biti stavljen akcenat u predloženom rešenju opisanom u 5.6 i 5.8. Suština je obraditi model pretnje iz aspekta ulaznih tačaka u sistem, koji neće biti fokus sam po sebi u ovom radu.

Bezbednosti aspekti pobrojani u 1), i 2) su toliko česti da sve naprednije *cloud* usluge sadrže i namenske SkS koji nudi višeorganizacijska rešenja kao servis identiteta. Kako postoje

standardni pristupi za implementaciju ovakvih rešenja, nisu od interesa za dublje istraživanje opcija, s tim da je u predloženom rešenju uzeta za primer sopstvena implementacija opisana u 5.8.

Servisni moduli su usko vezani za SkS poslovni model, i uvode potrebu za jasnim razgraničavanjem između različitih organizacija u smislu praćenja njihovog korišćenja servisa.

1) Izveštavanje – modul koji se bavi kontinualnim izveštavanjem, i upoređuje se u odnosu na definisani model naplate, koji pripada administraciji.

2) Kvalitet servisa – provera, po organizaciji, da li se dostavljani servisi dostavljaju onako kako je ugovoreno, tačnije, da li korišćenje oslikava vrednosti predstavljene *SLA* ugovorom. Takođe, implementacija praćenja kvaliteta dostavljenog servisa isto zavisi od modela naplate, i samim tim je vezano za podsistem naplate.

Integracije platforme obuhvataju potrebne preduslove da bi se višeorganizacijska platforma mogla sa što manje invazija upotrebiti u postojećim softverskim rešenjima, što obuhvata sledeće dve komponente:

1) Biblioteke – biblioteke za integraciju treba da budu napisane tako da se mogu uključiti u postojeća softverska rešenja i uz što manje izmene izvornog koda upotrebiti tako da se koriste odgovarajući servisi koji su deo rešenja. U predloženom rešenju, predstavljenom u 5.6, biblioteke su implementirane za jednu platformu, i bazirane su na aspektno orijentisanom programiranju kako bi se dekoracijom postojećih metoda dodala odgovarajuća funkcionalnost. Na taj način, izmena postojećih interfejsa se svodi na minimum.

2) *API* za servise – u predloženom rešenju postoji niz servisa koji omogućavaju orkestraciju konteksta organizacije na nivou čitavog rešenja koje je potrebno transformisati na višeorganizacijsko. S obzirom na primenjivanu SOA arhitekturu, neophodno je imati jasno definisane interfejse i metode koje se mogu koristiti kako bi se manupulisalo određenim servisima ciljnog rešenja. Na osnovu API specifikacije, kreiraju se biblioteke koje mogu pojednostaviti krajnju implementaciju.

Prilagodljivost je jedna od najvažnijih i često izučavanih oblasti povezanih sa višeorganizacijskim modelom SkS. Moglo bi se reći da uz jezgro višeorganizacijske platforme, koja mora da postoji, i predstavlja temelj za migraciju rešenja, najvažnija karakteristika za evaluaciju uspešnosti migracije jeste potreba za prilagodljivošću krajnjeg rešenja.

Potreba za prilagodljivošću zavisi od sledećih faktora: u kojoj meri je postojeće rešenje standardizovano, od poslovnog modela organizacije koja razvija softversko rešenje, i zakonskih regulativa ukoliko su primenljive na postojeće rešenje. Ukoliko je rešenje standardizovano, znači da je stepen prilagodljivosti mali, a isto važi i za rešenja koje moraju da odgovaraju određenim regulativama. Ovde je potrebno istaći da delovi rešenja mogu biti standardizovani ili uređeni regulatorno, što ostavlja prostor da se višeorganizacijska arhitektura u vertikalno distribuiranom sistemu sprovede parcijalno, pri čemu bi se delovi za prilagodljivost izdvojili u posebne servisne strukture koje bi bile orkestrirane od strane modula konteksta organizacije, koji će biti opisan u odeljku 5.6.

Ukoliko postoje delovi rešenja koje je potrebno prilagoditi, što je obično izazvano različitim klijentskim zahtevima na još uvek nedefinisanom tržištu sa poslovnim modelom

organizacije koja razvija dato rešenje usmerenim ka osvajanju što većeg dela tržišta, potrebno je omogućiti prilagodljivost. Čitava oblast prilagodljivosti višeorganizacijskih rešenja nije fokus ovog istraživanja jer je izučena u dovoljnoj meri da se može kreirati jezgro višeorganizacijske platforme bez propuštanja bitnih detalja, a sama oblast je veoma velika i zasebna po fokusu. Drugim rečima, oblast jezgra višeorganizacijske platforme i oblast prilagodljivosti ne utiču jedna na drugu. Dodatno, kombinacijom ove dve oblasti se može doći do ciljnog rešenja transformacije, pri čemu se ostale predstavljene oblasti mahom mogu implementirati uobičajenim inženjerskim praksama, tj. dublje istraživanje pristupa rešenja ne utiče direktno na odluku da li migrirati postojeće rešenje. U oblasti prilagodljivosti višeorganizacijskog SkS, izdvajaju se sledeće oblasti:

1) Mrežni model specifičnosti – predstavljen u [125] podrazumeva izdvajanje funkcionalnosti koje treba da se prilagode od onih koje mogu biti deljene. Na taj način se pokušava doći do što sitnijih komponenti gde se sve one koje ne mogu biti deljenje postavljaju u mrežnom logičkom modelu čija je svrha da omogući efektivnu raspodelu komponenti za prilagodljivost. Formalni model opisuje način za alokaciju resursa tako da potencijalno deljene komponente mogu biti iskorišćene prema četvrtom modelu zrelosti višeorganizacijskih rešenja, opisanom u 3.5.2. Za ostale komponente koje zahtevaju prilagodljivost, razrađuje se formalna metodologija koja se bavi optimizacijom korišćenja resursa za takve module. Prilagodljivost se zasniva na dinamičkoj kompoziciji servisa od raznih modula. Postoje algoritmi koji su razvijani tako da izoluju standardne od prilagodljivih delova, neki su predstavljeni u [126], što dovodi do prvobitnog iznetog zaključka, a to je da je potrebno razdvojiti deo softverskog rešenja koji je zajednički za sve organizacije, a u posebne softverske servise izdvojiti deo namenjen prilagodljivošću. Ovaj pristup ponovo implicira mikroservisnu arhitekturu kao preferabilnu za višeorganizacijske servise.

2) Konfigurabilnost – ukoliko je softver dovoljno konfigurabilan da zadovolji zahteve različitih klijenata, to može biti potencijalno rešenje i u višeorganizacijskom okruženju. Naravno, ovakva rešenja iziskuju viši stepen projektovanja softvera, što podrazumeva generalno veću ekspertizu i potencijalno negativno utiče na performanse sistema.

3) Kompatibilnost nadogradnje – s obzirom da se prilagodljivost može zasnivati na mrežnom modelu specifičnosti, može lako doći do toga da se delovi rešenja nezavisno nadograđuju. Iz tih razloga, jako je bitno da se tokom nadogradnje sačuvaju originalni interfejsi servisa ili da se svi zavisni servisi zajedno nadograde u isto vreme kada se i određena komponenta nadograđuje. U slučaju mnogo organizacija koje koriste servise, praćenje nadogradnje modula iz razloga obezbeđivanja kompatibilnosti može izazivati kompleksan problem, koji ne sme biti izostavljen pri projektovanju višeorganizacijskih SkS.

Za uspešnu migraciju postojećih softverskih rešenja na višeorganizacijski SkS je najvažnije sagledati prvo transformaciju na nivou poslovnog modela, opisanu u poglavlju 3.3. Na tehničkom nivou, odluka će zavisiti u velikoj meri od sledeće dve oblasti: bezbednosti informacija i prilagodljivosti SkS. Ukoliko je pak odluka za transformaciju pozitivna, najvažnije osigurati postojanje konteksta organizacije, tačnije jezgra višeorganizacijske platforme. U tom smislu, glavni fokus ovog istraživanja je na kontekstu višeorganizacijskog SkS, ali i na izučavanju primene konteksta u slučaju PkS. Razlog za akcentovanje PkS dolazi praćenjem industrije, što je opisano u 3.1, a sve kao posledica primenjivanja modela akademsko industrijskog transfera tehnologije opisanog u 1.3.

Naime, sve vodeće PkS usluge su okrenute kreiranju višeorganizacijskih SkS, pri čemu rešavaju veliki deo pratećih stvari, kao što su sledeći moduli: administracija, bezbednost i infrastruktura. Iz navedenih razloga, moderne PkS mogu predstavljati idealan izbor pri migraciji postojećih SkS. Glavna mana takvog pristupa je što se obično dolazi u situaciju da se krajnje rešenje može koristiti samo na određenoj platformi. Odatle i nastaje zahtev da jezgro višeorganizacijske platforme bude nezavisno od modula infrastrukture.

U narednim odeljcima je prikazano detaljno istraživanje u oba pravca – višeorganizacijski model baziran na modernom PkS, kao i nezavisno jezgro koje obezbeđuje kontekst organizacije za omogućavanje transformacije.

5.5. Višeorganizacijski model zasnovan na PkS

U ovom odeljku je prikazana implementacija višeorganizacijskog SkS koja za zadatak ima da iskoristi PkS u svrhu kreiranja skalabilnog sloja perzistencije podataka u scenariju interneta stvari. Ideja jeste istražiti višeorganizacijsku osobinu koju omogućavaju moderne PkS. Samo rešenje se može razdvojiti u dve celine, koje su i predstavljene odvojenim eksperimentima u poglavlju 6:

1) Modelovanje skalabilnog perzistencionog sloja za internet stvari
2) Telemetrija i akvizicija podataka

Primena višeorganizacijskog modela je u slučaju ovog rešenja u domenu pametnih elektroenergetskih mreža, ali ciljni rezultati omogućuju analizu i razumevanje primene PkS funkcija vezane za višeorganizacijski model i u tom smislu dobijeni rezultati ne gube na opštosti.

5.5.1. Opis problema

Osnovni problem koji se ovde pokušava rešiti je funkcija upravljanja opterećenjem, opisana u 4.5. Razlog je to što predstavlja buduće zahteve koji će biti deo NDMS sistema opisanog u 4.4 ili njegova prirodna ekstenzija, dok s druge strane takođe predstavlja kandidat za *cloud* platformu zbog svojih zahteva koji uključuju velik broj komunikacionih kanala i potrebu za skalabilnošću. Osim navedenog, iz ugla krajnjih potrošača, sistem je višeorganizacijski tako da je funkcija upravljanja opterećenjem idealan scenario da se izuči i predstavi analiza modernog PkS čija se višeorganizacijska osobina postiže particionisanjem podataka prema organizaciji, što je opisano u 3.6.1.

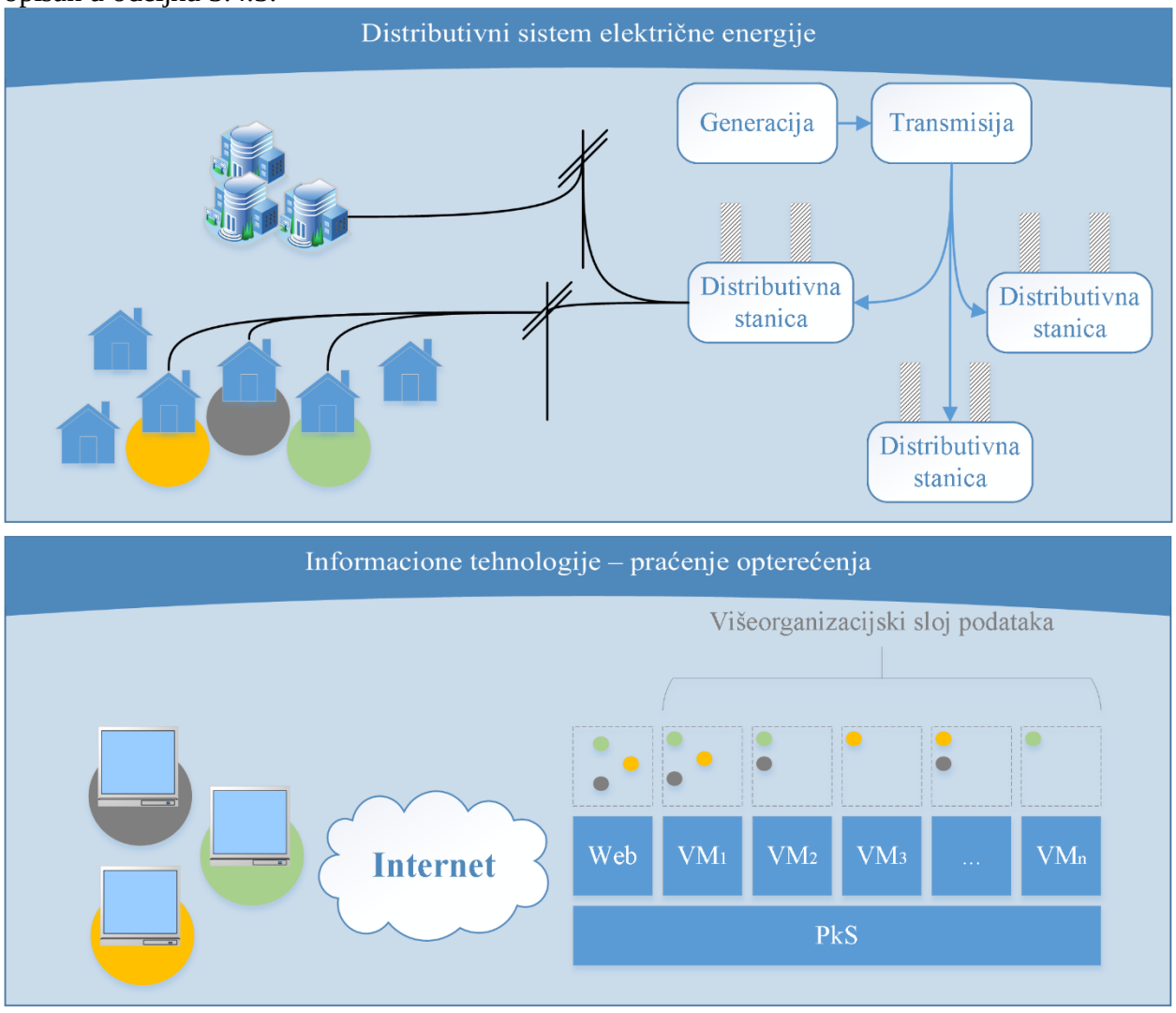
Slika 28 prikazuje vezu organizacija, u ovom slučaju domaćinstva, sa krajnjim sistemom iz dva pogleda koji se upotpunjuju – energetičarski i IT. Iz energetičarskog ugla, svako domaćinstvo predstavlja krajnjeg potrošača sistema koji napajanje dobija iz određene distributivne stanice. Topologija mreže u funkciji opterećenja dobija na značaju u sprezi sa NDMS sistemom. Dakle, iz NDMS ugla, svako domaćinstvo predstavlja krajnju tačku ispostave električne energije, tj. potrošača. Na slici su krugovima različitih boja predstavljena različita domaćinstva da bi se lakše napravila paralela sa IT sistemom.

Iz IT pogleda, svako domaćinstvo ima pametno brojilo koje je na neki način povezano sa *cloud* okruženjem gde se nalazi servis funkcije opterećenja. Naravno, između mogu postojati različiti koncentratori podataka, ali postavka infrastrukture za komunikaciju između pametnih brojila i *cloud* okruženja je irelevantna za višeorganizacijski dizajn *cloud* servisa. Osim pametnog brojila i eventualno ostalih pametnih uređaja koji predstavljaju deo industrijskog interneta stvari, domaćinstva predstavljaju krajnje korisnike informacionog sistema koji bi trebao u realnom vremenu da pruži uvid u potrošnju električne energije, kao i izveštaja i statistike. Jasno je da je

ovaj scenario prirodno višeorganizacijski, jer svako domaćinstvo povremeno koristi usluge informacionog *web* servisa.

U delu slike sa informacionim tehnologijama, domaćinstva su naznačena kao korisnici koji putem interneta pristupaju *web* baziranom *cloud* servisu. Na desnoj strani je prikazan dizajn rešenja baziran na PkS, gde je za čitav sistem odvojen klaster od n virtuelnih mašina. U prezentacionom sloju postoji n *web* virtuelnih servera, pri čemu je *web* servis višeorganizacijski što je naznačeno kružićima u bojama koji odgovaraju organizacijama.

Koristeći predloženo rešenje koje je opisano u odeljku 5.6, implementacija višeorganizacijskog *web* servisa je trivijalna. VM_1 do VM_n predstavljaju virtuelne mašine u klasteru posvećene distribuiranom PkSv2 sistemu gde je aplikativni model u stvari model učesnika opisan u odeljku 3.4.3.



Slika 28 - Svaki krajnji korisnik je organizacija

Opisani problem sadrži dve celine, ili dva smera. Jedna celina (odeljak 5.5.2) je modelovanje skalabilne baze podataka bazirane na učesnicima, gde jedan učesnik po organizaciji pruža izolaciono svojstvo neophodno za višeorganizacijske sisteme. Dodatno, u ovom problemu postoji veliki broj organizacija koje neće imati prevelike zahteve za podacima i računarskim resursima, što čini podelu učesnik po domaćinstvu idealnu.

Druga celina (odeljak 5.5.3) se odnosi na smer prikupljanja podataka pametnih brojila i ostalih uređaja učesnika industrijskog interneta stvari. Ovaj smer će omogućiti samo funkciju

upravljanja opterećenjem, i time biti u mogućnosti da uspešno proširi NDMS sistem, kao i da pruži uvid u trenutno stanje situacije kad je u pitanju potrošnja električne energije.

5.5.2. Skalabilna PkS baza podataka za internet stvari

U slučaju relacionih baza podataka su poznati šabloni za podelu sloja za perzistenciju i oni su detaljno opisani u 3.6. Mnogi radovi pokušavaju da centralizuju podatke svih organizacija čak iako ne postoji korelacija između podataka različitih organizacija. Dodatno, rad nad podacima različitih organizacija u višeorganizacijskom modelu je podrazumevano isključen.

U opisanom problemu, servis je višeorganizacijski prema krajnjim korisnicima a predstavlja izvor važnih podataka na drugoj strani, elektrodistribuciji. Podaci treba da budu agregirani u realnom vremenu da bi se postigao važan ulazni parametar za ostale komponente pametnih elektroenergetskih mreža uključujući i NDMS sistem, a to je trenutno stanje interneta stvari. U ovom primeru, pametni uređaji, tj. pametna brojila sa svojim mogućnostima će biti podrazumevana pod stvarima u konceptu industrijskog interneta stvari. Samo rešenje je lako proširivo, bez naglaska na rešavanju energetičarskih problema. Fokus opisanog scenarija potrebe za *cloud* baziranim višeorganizacijskim dizajnom u domenu pametnih elektroenergetskih mreža je na rešavanju problema opterećenosti elektroenergetske mreže kroz virtuelne elektrane. Koncept virtuelnih elektrana je predstavljen u odeljku 4.2.

Da bi se sprečio otkaz sistema usled preopterećenosti elektroenergetske mreže, postoje situacije gde se pribegava DR funkciji. Korišćenjem koncepta virtuelnih elektrana, postoje situacije gde će se kroz sistem pametnih elektroenergetskih mreža pokušati ušteda potrošnje električne energije tako što će se dobrovoljno isključiti veliki potrošaći za određene korisnike koju su zbog raznih finansijskih beneficija spremni na ovakvu vrstu interakcije sa elektrodistribucijom. Veliki potrošači, na primer, mogu biti sistemi za rashlađivanje prostorija u pametnim kućama, gde ne mora da dođe do njihovog potpunog isključenja, nego do prelaska u režim potrošnje manje količine električne energije. Detaljan opis DR funkcije se nalazi u 4.5.

Za razliku od drugih [127]–[130], uzet je potpuno drugi pristup rešavanju problema, gde je primarni fokus na krajnje potrošače, korisnike višeorganizacijskog *web* servisa koji omogućuje pregled u realnom vremenu tekuće potrošnje struje kao i generalne statistike koja može biti od interesa korisniku moderne elektrodistribucije. Ovo je aspekt finalnog rešenja čiji je cilj da pruži informaciju o potrošnji električne energije krajnjim potrošačima. S obzirom da je ovde neophodno da čitav sistem funkcioniše performantno, a nije fokus u povremenim analitikama nad velikom količinom podataka (engl. *big data analytics*), pokušaj je da se ovi zahtevi zadovolje spajanjem podataka i funkcionalnosti servisa kreirajući takozvane instance servisa sa stanjem. Praktično većina današnjih cloud PkS se baziraju na tome da pružaju okruženje za servise sa stanjem i servise bez stanja. Kako je za servise bez stanja stvar mnogo jednostavnija i pri tome ne uvek primenljiva, opštost eksperimenta treba da bude na rešenju baziranom na servisima sa stanjem.

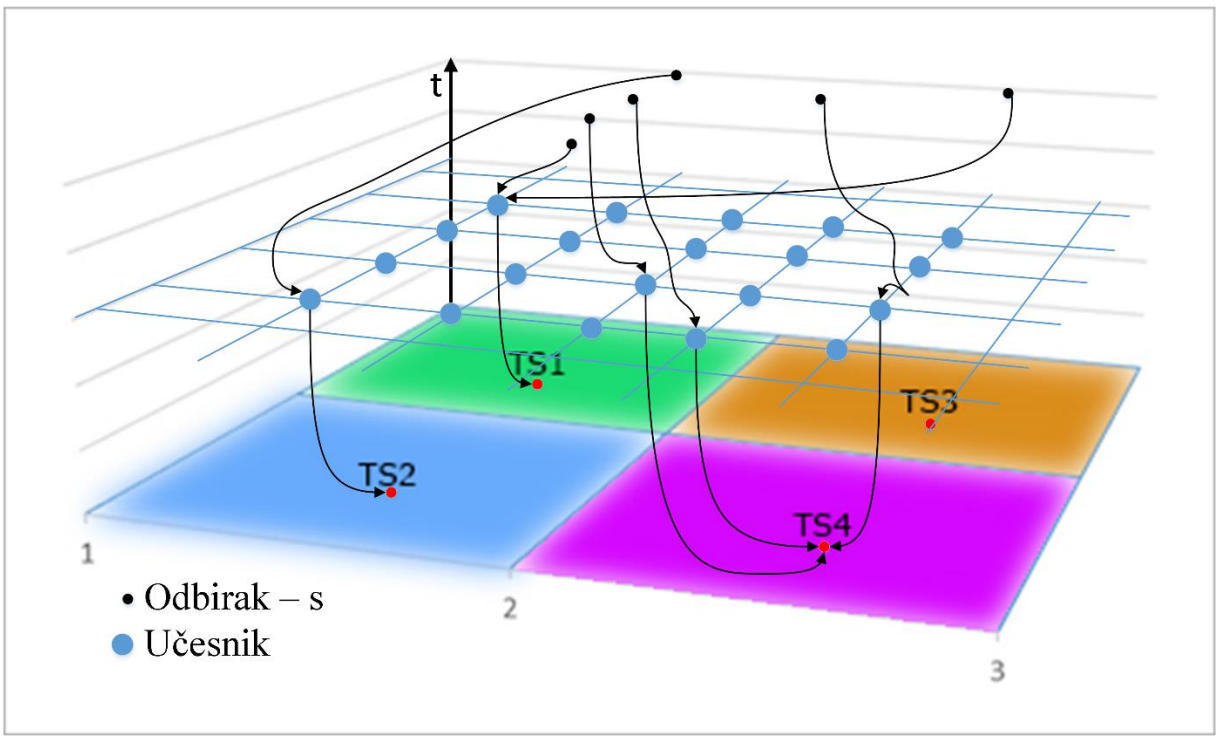
Dizajn baze podataka je baziran na modelu učesnika, detaljno opisan u 3.4.3. Učesnik može biti čitav računarski čvor, virtuelni proces ili nit sa privatnom memorijom [131]. Svaki učesnik se izvršava asinhrono, po svom dinamizmu umesto na sinhronizmu zasnovanom na globalnom časovniku. Ovo ukazuje na modelovanje distribuiranih sistema gde precizna sinhronizacija između komponenti nije izvodljiva [131]. U našem slučaju gde su pametni uređaji potpuno nezavisni

entiteti jedni od drugih, model baziran na učesnicima se čini prirodan. Korišćeno je PkS rešenje, vodeće u industriji, koje je bazirano na modelu učesnika – *Microsoft ServiceFabric* (MASF), a opisano je u poglavlju 3.4.1. Svaki učesnik je predstavljen kao programska nit u klasteru više računarskih čvorova, gde je čvor bilo fizička ili virtuelna mašina *cloud* okruženja. Model učesnika se zasniva na razmeni poruka bez deljenog promenljivog stanja u arhitekturama koje se baziraju na događajima [76].

Jedan učesnik je dodeljen jednom krajnjem korisniku i čuva podatke za datog korisnika. Jedan čvor u *cloud* okruženju može sadržati puno učesnika, ali u svakom slučaju relacija između učesnika ne postoji jer je svaki namenjen drugom krajnjem korisniku. Broj čvorova u sistemu diktira koliko instanci učesnika mogu biti sadržani, što čini ovakvo rešenje potpuno horizontalno skalabilnim.

Problem skalabilnosti baze podataka je široko izučavana oblast [127]–[130] sa fokusom na centralizovanje podataka i pružanjem svih generalnih funkcionalnosti vezanih za SQL i NoSQL baze podataka. Osim podataka koje koriste krajnji korisnici, za potrebe virtuelne elektrane, neophodni su agregirani podaci kako bi pružili značajan ulazni parametar aplikacijama pametnih elektroenergetskih mreža.

Slika 29 oslikava korelaciju između podataka različitih krajnjih korisnika (organizacija) koji stižu u sistem kao odbirci merenja u vremenu, a svaka organizacija je modelovana kao jedan učesnik u sistemu. Na dnu, prikazane su različite površine koje predstavljaju transformatore. Svaki pametni transformator šalje podatke servisima pametnih mreža koje mogu da kontrolišu opremu kao što su transformatori i drugi elementi koji sačinjavaju topologiju distributivne elektroenergetske mreže.



Slika 29 - Korelacija podataka nad organizacijama

Pametna kuća, ili jedan krajnji potrošač elektrodistribucije, predstavlja jednu organizaciju tj. korisnika u predloženom sistemu. Sve kuće koje se napajaju od istog transformatora se mapiraju na istu oblast. Agregirani podaci predstavljaju virtuelnu električnu energiju koja bi se mogla uštedeti na svakom od transformatora. Organizacije su prikazane na slici kao matrica učesnika u PkS sistemu. Y osa predstavlja vremensku jedinicu, tako da svaki odbirak *s* koji se nalazi u sistemu se definiše na sledeći način:

$$s = (t_{id}, \{p_1, p_2, p_3' \ldots\})$$

Gde važi sledeće: t_{id} je identifikator organizacije, p_i su vrednosti koje identifikuju vrednost merenja i tip uređaja, dok je p_j' vrednost koja identifikuje kontrolabilnog potrošača, tj. onog koji pripada virtuelnoj elektrani. Vrednosti treba da se agregiraju po pojedinačnoj oblasti koja se određuje pripadnošću transformatoru.

Ako bi aplikacije pametnih elektroenergetskih mreža imale znanje koliko struje može da se sačuva isključujući veće potrošače koji se napajaju sa određenih transformatora, mogla bi da unapredi svoj algoritam za dostizanje globalnog optimuma u jednom ekosistemu. Ovo je oblast od velikog interesa u domenu pametnih elektroenergetskih mreža, a ujedno i jedan od primera gde je potrebno agregirati podatke različitih organizacija. U ovom slučaju je potrebno brzo prikupiti podatke po regijama veoma brzo.

5.5.3. Telemetrija i višeorganizacijsko prikupljanje podataka

Slika 30 prikazuje dizajn akvizicionog sloja za internet stvari koji se oslanja na model baziran na učesnicima. Distribuirani učesnici (engl. *Distributed DB Actors*) čuvaju podatke, i predstavljaju prvi nivo apstrakcije dovoljan za prihvatanje i čuvanje trenutnog stanja stvari. Svaka instanca učesnika je vezana za jednog korisnika, tj. organizaciju. Postoji jedinstven identifikator za svakog učesnika.

Korak 1 prikazuje kako odbirak merenja biva sačuvan u adekvatnoj instanci učesnika. Ako je instanca učesnika već imala podatke, asinhrono se stari podaci prosleđuju apstrakcionom sloju za istorijske potrebe u koraku 2.

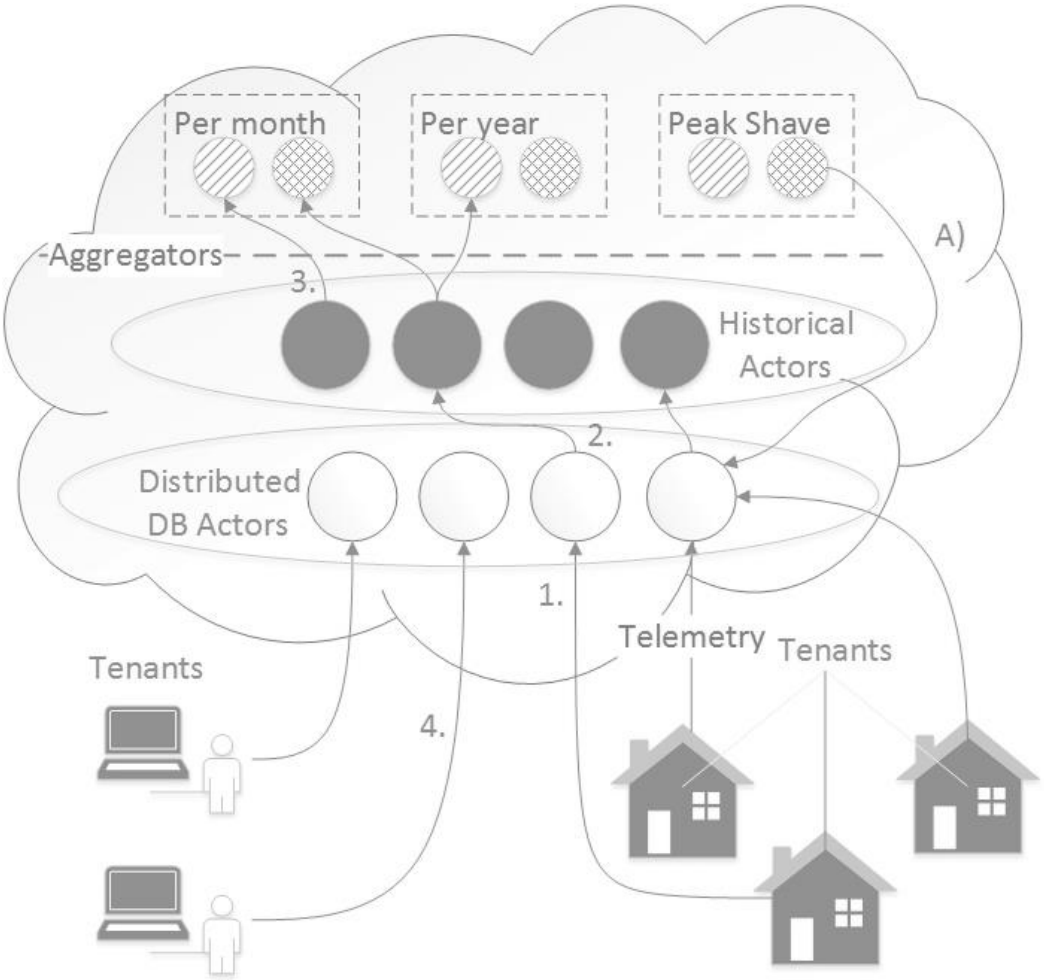
U koraku 4, organizacija može koristiti višeorganizacijski *web* servis kako bi nadgledala tekuću potrošnju električne energije u pametnoj kući. Jednom kada su podaci poslati učesnicima zaduženim za čuvanje istorije, mnoge funkcije mogu biti pozvane da bi agregirale istorijske podatke za višeorganizacijski *web* servis. Na primer, izveštaji mesečne ili godišnje potrošnje po uređaju se lako mogu dodati, kao što je ilustrovano korakom 3.

Razlozi za dizajniranjem rešenja u više slojeva su performantnost i skalabilnost. Imati trenutno stanje interneta stvari sa najbržim mogućim karakteristikama je cilj predloženog rešenja, posebno jer je ideja da se opštost ovog eksperimenta izvede i za potrebe ostalih aplikacija. Aplikacije pametnih mreža donose odluke bazirane na trenutnom stanju elektroenergetskog sistema kojim upravljaju, i za njih je najvažniji prvi sloj koji sadrži distribuirane učesnike.

Korak A) (Slika 30) predstavlja propitivanje trenutnog stanja čitavog sistema od strane jedne funkcije koja zahteva visok nivo performansi. Rezultati upita bi bili ulazni parametri za aplikacije pametnih elektroenergetskih mreža.

Instance učesnika se mogu identifikovati putem jedinstvenog identifikatora organizacije što čini čitavo rešenje horizontalno skalabilnim. Svaka nova organizacija ne može zauzeti više resursa od dela virtuelne mašine, tj. čvora u cloud sistemu, dok će se za svaku novu organizaciju kreirati nova instanca učesnika. Oslanjajući se na fleksibilnost *cloud* sistema, učesnici se mogu množiti u virtuelnoj beskonačnosti resursa jer se klasteri čvorova mogu širiti preko celog centra

podataka. Gubitak podataka je izbegnut jer se koriste po bar tri kopije za svaku instancu učesnika, kako je opisano u odeljku 3.4.3. Jedinstveno mesto otkaza praktično ne postoji.



Slika 30 - Slojevi za akviziciju i ekstenziju funkcionalnosti

Ako se izostavi integracija sa ostalim aplikacijama, rešenje svakako mora pružiti istorijske podatke kroz višeorganizacijske *web* servise. Da bi se minimizovao broj instanci učesnika i da bi se podaci logički centralizovali, kreiran je sloj istorijskih učesnika koji primaju stare odbirke merenja kada nova merenja pristignu. Bilo kakva funkcionalnost može biti dodata kroz nove učesnike koji će koristiti istorijske učesnike radi agregacije podataka.

Opisani proces čuvanja podataka u okviru učesnika je evaluiran eksperimentom opisanim u 6.4, dok je korak A, tačnije funkcionalnost preuzimanja kompletnog stanja sistema iz mnoštva nepovezanih učesnika potkrepljen eksperimentom opisanim u 6.5.

Način identifikacije organizacije i čuvanja konteksta organizacije u svakom zahtevu je opisan u narednim odeljcima ovog poglavlja, čija evaluacija je eksperiment opisan u 6.6.

5.6. Pregled uopštenog rešenja

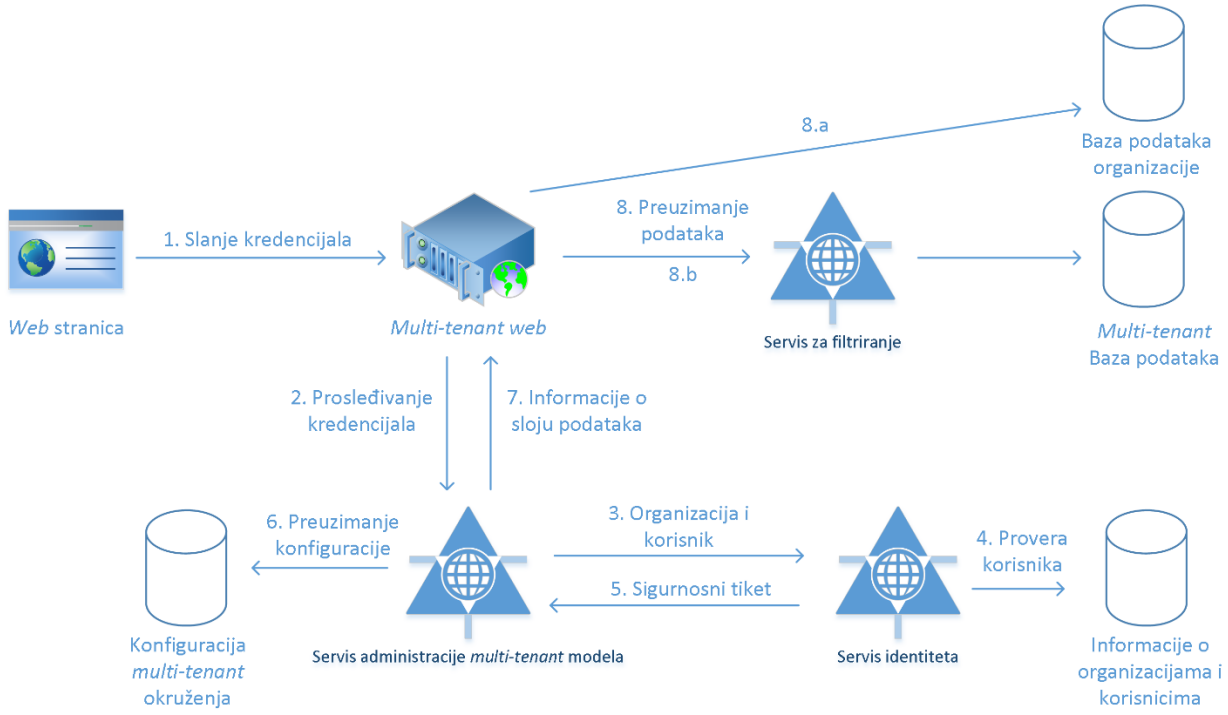
Oblast vezana za bezbednosni model višeorganizacijskih SkS aplikacija je relativno nova [27]. Postojeća rešenja su opisana u odeljku 5.3. Da bi se servis implementirao tako da bude visoko konfigurabilan, potrebno je visoko poznavanje domena problema. Teško je, ako je uopšte izvodljivo, napraviti visoko konfigurabilno opšte rešenje koje bi zadovoljilo pravljenje servisa sa različitim domenom problema. Stoga je važno posebno istaći da se rešenje koje predlaže ovaj rad

ne bavi problemom prilagodljivosti. Ipak, prilagodljivost je neosporno veoma važan faktor kod višeorganizacijskih SkS, kako je to i opisano u odeljku 5.4.

Slika 31 predstavlja dizajn krajnjeg rešenja na najvišem nivou apstrakcije. Za implementaciju rešenja, potrebno je odabrati tradicionalni servis i obezbediti infrastrukturu tako da servis postane višeorganizacijski.

Za primer transformacije, odabran je *MVC web* servis. Razlog je to što je *web* servis često predmet za višeorganizacijsku orijentisanost zbog svoje prirode. *Web* aplikacije su bez stanja i mogu da usluže velik broj korisnika u jedinici vremena. Za implementaciju bilo kog drugog tipa servisa bez stanja, postupci implementacije softverske podrške su isti.

Razvijena je i biblioteka za *MVC* softversku podršku koja bi omogućila upotrebu višeorganizacijske orijentisanosti. *Web* server u gornjem levom uglu (Slika 31) označava višeorganizacijsku *MVC web* aplikaciju. Ostali servisi na slici pripadaju okruženju koje je deo rešenja za implementaciju višeorganizacijskog modela.



Slika 31 - Dizajn rešenja za kontekst organizacije

Arhitektura rešenja je servisno orijentisana, jer postoje tri odvojena servisa koji učestvuju u pružanju funkcionalnosti ciljne platforme za migraciju postojećih rešenja, uz bezbednost višeorganizacijskog pristpua. Redosled međuservisne komunikacije je opisan brojevima (Slika 31). Na strani klijenta koji koristi višeorganizacijski *web* servis, nalazi se *web* pretraživač.

Prvi korak je upućivanje *HTTP* zahteva *web* serveru. Radi pojednostavljenja objašnjena, odmah će biti razrađen slučaj koji uključuje autorizaciju korisnika. Za pristup autorizovanim stranicama, korisnik mora biti prijavljen na sistem. Pretpostavka je da je upućen *HTTP* zahtev koji sadrži kredencijale za prijavu na sistem. Slanje kredencijala je obeleženo korakom jedan. Klijent koji se prijavljuje na sistem ne mora da zna (i ne treba da zna), da je *web* servis višeorganizacijski.

Čak je poželjno da ni na jedan način klijent ne može pretpostaviti da je uslužen od strane višeorganizacijskog servisa.

Korak dva jeste da se aplikacija obrati platformi, prosleđivajući potrebne parametre i kredencijale, kako bi dobila uputstva za pristup podacima. Ovo je razlika u odnosu na tradicionalne *MVC web* aplikacije, a detaljnije o razlikama u pristupu se nalazi u odeljku 5.7.

Korakom sedam je predstavljen odgovor bezbednosnog sloja platforme u kom se nalaze informacije na kojoj adresi se nalazi baza podataka, koji su joj pristupni podaci, ili na kojoj adresi se nalazi sloj za usluživanje podataka.

U koraku osam, višeorganizacijski *web* servis pristupa podacima i prikazuje ih klijentu u vidu *HTML* stranice. Korakom 8a je označen pristup zasebnih baza podataka, a 8b pristup potpuno deljenih baza podataka, opisanih u poglavlju 3.6.

Da bi se razradio bezbednosni sloj detaljnije, potrebno je vratiti se u korak tri, gde se kontaktira servis za administraciju višeorganizacijske platforme koja se obraća servisu identiteta, kako bi dobio odgovarajuću tiket koji predstavlja ulaznicu u čitav sistem. Više o izdavanju tiketa i servisu identiteta ima u odeljku 5.8. Servis za izdavanje identiteta, proverava u bazi podataka informacije o korisniku i organizaciju u koraku četiri. Preuzima tvrdnje vezane za korisnika, njih potpisuje i šalje kao povratnu vrednost u koraku pet.

U koraku šest, servis administracije višeorganizacijske platforme, proverava u kom kontekstu se organizacija izvršava. Preuzima potrebne informacije i prilaže dobijeni tiket kao odgovor u koraku sedam.

Sva tri servisa koja čine krajnje rešenje, zadovoljavaju četiri pravila vezana za servisno orijentisana okruženja, koja su opisana u odeljku 3.4.2. U radu je implementirano rešenje servisa identiteta, ali se usled svoje jasno definisane odgovornosti, može zameniti nekim od postojećih servisa identiteta. Tako bi se u krajnjem rešenju mogao koristiti *AD* (*Microsoft Active Directory*), ili neki od postojećih SkS identiteta koji se mogu pronaći u ponudi svih većih pružaoca IkS ili PkS usluga.

5.7. Odnos tradicionalne i troslojne višeorganizacijske MVC web aplikacije

Pri prelasku na višeorganizacijski model, svaki servis mora pretrpeti određene promene. Tako je i u slučaju *web* aplikacije. Oslanjajući se na implementirano rešenje servisno orijentisanog okruženja za podršku višeorganizacijskog modela, cilj je uz što manju izmenu postojećeg rešenja prilagoditi rešenje za rad sa više organizacija.

Pri implementaciji višeorganizacijskih *web* servisa, dolazi do deljenja resursa između različitih organizacija. Jednom kada se korisnik autorizuje, moguće je lako identifikovati kojoj organizaciji pripada. Na osnovu poznavanja korisnika, moguće je vršiti odabir odgovarajućih resursa. Problem je što u slučaju dok se korisnik nije prijavio na sistem, *web* aplikacija svakako mora biti u stanju da razlikuje različite organizacije. Dakle, u pitanju je javni deo *web* servisa.

Generalno, postoje dva pristupa za identifikaciju organizacije u okviru *web* aplikacije, koji se baziraju na razlikovanju *HTTP* zahteva:

- određivanje putem parametara *URL* adresa (engl. *URL routing*),
- određivanje putem *DNS* imena.

Određivanje putem parametara *URL* adresa se svodi na to da se u svakoj *URL* adresi nalazi parametar koji identifikuje organizaciju. Primer takve adrese bi bio: *localhost/tenantId=AA*, gde je AA identifikator organizacije. *MVC* softverska podrška podržava ulepšavanje adresa tako što je u stanju da ispisuje adrese u drugom formatu, koje su čitljivije prosečnom korisniku.

Određivanje putem *DNS* imena podrazumeva ekstraktovanje *host* polja iz *HTTP* zahteva. *Host* polje *HTTP* zahteva govori o tome preko kod *DNS* imena je korisnik uputio *HTTP* zahtev *web* serveru. Sada, na osnovu različitih domena različitih organizacija, moguće je identifikovati pravog korisnika.

Naizgled, i jedan i drugi pristup su slični, ali u suštini, mogu se koristiti u različitim situacijama. U rešenju koje predlaže ovaj rad, odabran je pristup određivanja putem *DNS* imena. Zamisao je da će *web* servisi biti namenjeni različitim organizacijama. Organizacije će imati različit izgled stranica iako će deliti istu instancu aplikacije, i namerno će biti postavljene na različitim domenima.

U slučaju kada se razvija višeorganizacijski servis koji će biti globalno korišćen, moguć je scenario u kom bi servis trebao biti dostupan uvek na istom domenu. U tom slučaju, pristup određivanja organizacije putem *DNS* imena ne bi bio primenljiv. S druge strane, određivanje putem parametara *URL* adresa je uvek primenljiv pristup, ali je mana ovog pristupa to što je uvek vidljiv parametar koji identifikuje organizaciju u zahtevu. Ovo jasno naznačava da je servis višeorganizacijski što ukazuje na moguće površine za napad. Iako od velike važnosti, sami bezbednosni aspekti su uočeni ali bezbednost ulaznih tačaka u sistem izlazi van okvira ove teze.

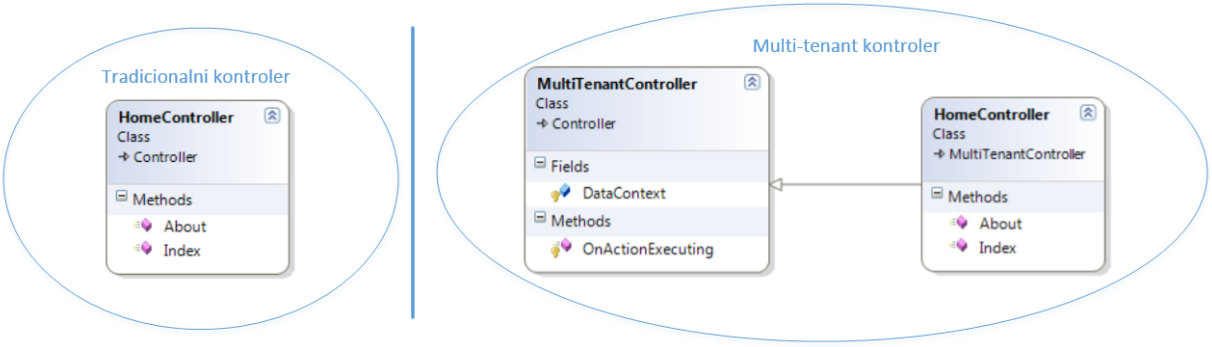
Da bi se razumele promene koje je potrebno uvesti, potrebno je dati kratak pregled tradicionalnih *web* aplikacija i potom objasniti kako biblioteke koje se mogu pridodati *MVC* softverskoj podršci, omogućavaju prelazak servisa na višeorganizacijski model. Biblioteke koje služe za integraciju *MVC* podršci su razvijani tako da koriste predloženo rešenje, i pripadaju rešenju integracije.

Generalno, u *MVC* pristupu, postoji klasa *Controller* koju nasleđuju sve klase koje implementiraju sloj kontrolera u *MVC* softverskoj arhitekturi. Kada pristigne novi *HTTP* zahtev, kreira se nova nit odgovarajućeg kontrolera, i izvršava se odgovarajuća akcija kontrolera.

HTTP zahtev se obrađuje u kontroler sloju, što implicira da će se višeorganizacijska orijentisanost rešavati upravo na ovom sloju. Klasa *Controller* je mesto na kom se *MVC* softverska podrška može proširiti kako bi se zadovoljilo višeorganizacijsko svojstvo. UML diagram (Slika 32) tradicionalne i višeorganizacijske *MVC* softverske podrške. Dodata je klasa *MultiTenantController* koju mora da nasledi svaka klasa od koje se očekuje da ima ulogu višeorganizacijski orijentisanog kontrolera. Dužnost klase *MultiTenantController* je da implementira čitavu logiku višeorganizacijskog pristupa. *MultiTenantController* klasa nasleđuje *Controller* klasu i redefiniše metodu *OnActionExecuting* koja se odnosi na bilo koju akciju.

Problem za implementaciju višeorganizacijskog kontrolera nastaje kada se uzme u obzir da jedan višeorganizacijski kontroler može imati akcije koje pripadaju javnom delu servisa, neke za koje je neophodna autorizacija, a neke su prelazne i služe za prijavljivanje korisnika na sistem. Od tipa akcije zavisi ishod i način ponašanja akcije.

Primera radi, akcije koje zahtevaju autorizaciju moraju prvo pribaviti tiket za pristup zaštićenim resursima, dok to nije slučaj za javne akcije. Ovo ukazuje na potrebu upotrebe dizajn šablona strategije (engl. *strategy pattern*).



Slika 32 - UML višeorganizacijske biblioteke

Da bi višeorganizacijski kontroler mogao da primenjuje odgovarajuće strategije (Slika 33), na neki način mora znati o kojoj akciji je reč. Pošto akcije sa različitim strategijama mogu pripadati istom višeorganizacijskom kontroleru, ne može se problem strategije rešiti na nivou višeorganizacijskog kontrolera. Višeorganizacijski kontroler je bilo koja klasa koja nasleđuje *MultiTenantController* klasu.

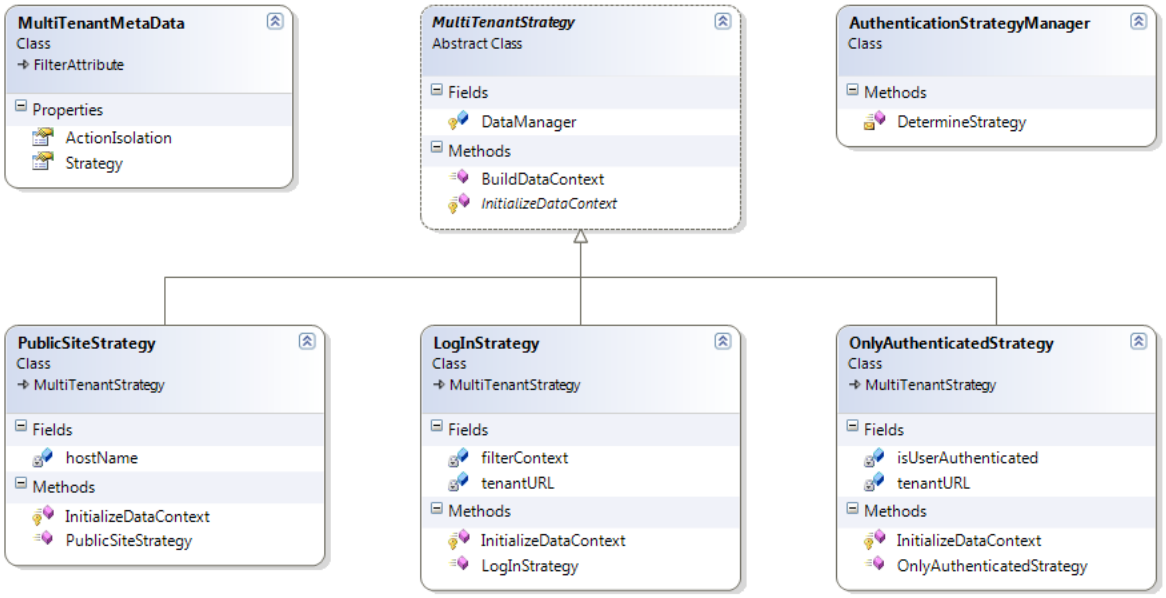
Pristup rešavanju problema vezanosti strategije za akciju, rešen je dekorisanjem akcija. Klasa *MultiTenantMetaData* predstavlja dekorater klasu (Slika 33). Polje *ActionIsolation* je tipa enumeracije i govori o tome da li je akcija javna, privatna ili akcija prijave korisnika na sistem. S obzirom da će se u zavisnosti od dekoratera akcije, izvršiti odgovarajuća strategija, može se smatrati da rešenje koristi aspektno orijentisani pristup, *AOP*.

Rezultat korišćenja *AOP* pristupa jeste da se svaka akcija mora dekorisati klasom *MultiTenantMetadata*. Ukoliko akcija nema dekorater, *MultiTenantController* će generisati izuzetak. S druge strane, ukoliko se nad akcijom pronađe dekorater, prosleđuje se klasi *AuthenticationStrategyManager*, koja se vidi na *UML* dijagramu (Slika 33).

AuthenticationStrategyManager klasa služi za upravljanje odabirom strategije. U metodi *DetermineStrategy*, nalazi se logika kojom se na osnovu dekoratera određuje prava strategija izvršavanja. Povratna vrednost metode je tipa *MultiTenantStrategy*, koja je apstraktna klasa. Sadrži apstraktnu metodu *InitializeDataContext* koju će svaka klasa naslednica morati da implementira. Konkretne strategije su one koje nasleđuju klasu *MultiTenantStrategy*. Svaka od njih mora implementirati metodu inicijalizacije konteksta podataka, ali svaka je slobodna da učini to na proizvoljan način.

Prava strategija se postavlja tako što se kreira instanca objekta neke od konkretnih klasa koje nasleđuju apstraktnu klasu *MultiTenantStrategy*. Kada se prava strategija postavi, strategija se koristi da izgradi kontekst podataka. Svaka strategija generiše kontekst podataka koji nasleđuje

klasu *DataServiceContext*. Cilj je da svaka strategija, u zavisnosti od konfiguracije organizacije, i u zavisnosti od akcije, kreira kontekst podataka sa različitim svojstvima. Činjenica da se u svakom slučaju kreiranja konteksta podataka generiše objekat iste klase, omogućava potreban polimorfizam kako bi se postigla željena transparentnost. U slučaju bilo koje strategije, klijentska strana pristupanja udaljenim podacima ostaje ista.



Slika 33 - UML prikaz strategija

Usled korišćenja strategija i upravljanje odabirom strategija iz višeorganizacijskog kontrolera, izmene u izvornom kodu aplikacije su svedene na minimum. Čitava izmena u samom kodu postojećih *MVC web* aplikacija, svodi se na dekorisanje akcija i na korišćenje klase *MultiTenantController* umesto *Controller*. Dekorisanje akcija se odvija u zavisnosti u kom delu prezentacije se one logički nalaze.

Tradicionalni *web* servisi su n-slojne arhitekture, a u najvećem broju slučajeva je n jednako tri. Troslojna arhitektura podrazumeva sloj prezentacije, poslovne logike i sloj podataka. Arhitektura se u tom smislu ne menja puno, jer logički slojevi i dalje ostaju isti. Menja se način autentifikacije, koji se kompletno izmešta u servisno okruženje razvijano posebno radi implementacije izdvojenog bezbednosnog modela. Cilj servisno orijentisanog višeorganizacijskog okruženja je da spreči mogućnost greške u višeorganizacijskom *web* servisu. Ovo znači da se više u konfiguraciji ne može čuvati adresa do baze podataka, jer bi tako postojao rizik slučajne zamene baze podataka među različitim organizacijama. Putanja do servisa koji uslužuje podatke se dobija od servisnog okruženja zajedno sa bezbednosnim tiketom koji će kasnije omogućiti preuzimanje podataka.

Uspostava veze ka servisu za usluživanje podataka je implementirana u samom višeorganizacijskom kontroleru koji se oslanja na šablon strategije i *AOP* kako bi pružio odgovarajući kontekst podataka. Autorizacija korisnika i informacije o bazi podataka se dobijaju od servisnog okruženja čija je implementacija predmet ovog rada.

5.8. RBAC kontrola pristupa bazirana na tvrdnjama

Tvrdnja (engl. *claim*) je izjava koju subjekat vezuje za sebe ili za neki drugi subjekt. Izjava između ostalog, može biti vezana za ime, identitet, ključ, grupu, privilegiju ili mogućnost. Tvrdnju izdaje odgovorno telo, i više ili jedna tvrdnja se pakuju u bezbednosne žetone koje izdaje izdavač. Izdavač se često skraćeno naziva *STS* (engl. *Security Token Service*) [132].

Kontrola pristupa bazirana na tvrdnjama (engl. *claims-based*) predstavlja celokupan koncept vezan za način autorizacije korisnika. Najčešći pristup realizacije autorizacije kod tradicionalnih *ASP.NET web* aplikacija je koristeći *WebForms* autentifikaciju. U okviru *WebForms* autentifikacije je zastupljen *RBAC* (engl. *Role Based Access Control*) pristup, tako da su pojmovi uloga, permisija i resurs već poznati u implementaciji prilikom korišćenja *ASP.NET WebForms* autentifikacionog sloja.

WebForms autentifikacija je često korišćen način implementacije autentifikacije u *ASP.NET web* servisima. U slučaju korišćenja *WebForms* autentifikacije, sama logika autorizacije korisnika se implementira u okviru *web* servisa. *Web* servis direktno pristupa bazi podataka u kojoj se nalaze korisnici i njihove uloge. Iz ovog razloga, proističe više stavki zbog kojih se u implementaciji ovog rešenja odustalo od *WebForms* autentifikacije.

Prva stavka je želja da se izvrši razdvajanje odgovornosti po servisima. Upravo razdvajanje odgovornosti je u najvećoj meri doprinelo odabiru kontrole pristupa bazirane na tvrdnjama. Višeslojna *web* aplikacija ne bi trebalo da bude u stanju da pristupi bilo kakvoj bazi podataka koja sadrži podatke više organizacija. Na ovaj način je rizik od greške u samoj aplikaciji izbačen.

Druga stavka je mogućnost finije granulacije samog *RBAC* pristupa u slučaju kontrole pristupa bazirane na tvrdnjama. Treća stavka jeste mogućnost iskorišćavanja postojeće baze podataka organizacija o svojim članovima. U slučaju *WebForms* autentifikacije, pristup bi bio da se podaci prekopiraju iz postojeće baze podataka i na taj način bi dolazilo do toga da se moraju administrirati dve baze podataka.

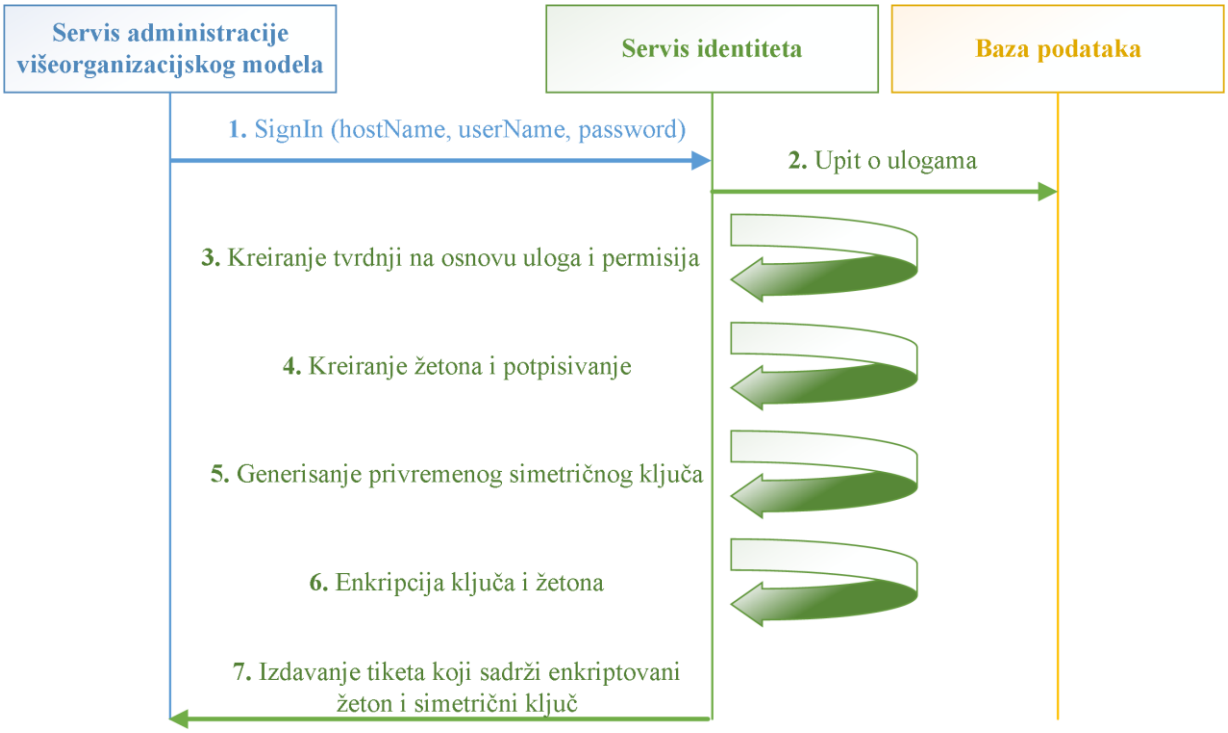
U slučaju izmeštanja autentifikacione logike u zaseban servis, moguće je adaptirati postojeća rešenja i praviti upite direktno nad postojećim bazama podataka organizacija. Bitno je da se utvrdi identitet korisnika koji želi da pristupi privatnoj sekciji sistema, a odgovornost servisa za pružanje identiteta je upravo to.

U implementiranom rešenju, zahtev za prijavu korisnika na sistem stiže iz servisa za konfiguraciju višeorganizacijskog modela, koji je nazvan *MultiTenantAdministrationPortal*. Potom postoji sekvenca zahteva da bi se dobio bezbednosni tiket (Slika 34). Servis identiteta koji je opisivan u ovom poglavlju ima za ulogu proveru identiteta korisnika i izdavača bezbednosnog tiketa, *STS*.

U prvom koraku, servis identiteta dobija organizaciju, korisničko ime i šifru. Na osnovu organizacije, servis vrši odabir konkretne baze podataka ili ukoliko je baza podataka o korisnicima višeorganizacijska, primenjuje odgovarajući filter u upitu.

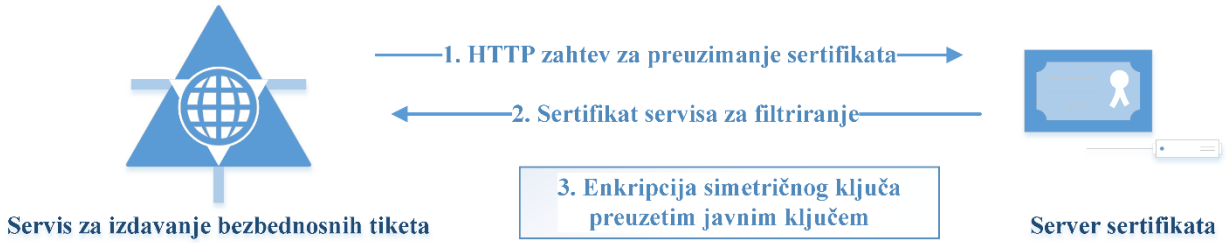
Na osnovu korisničkog imena i šifre, servis u drugom koraku saznaje uloge i permisije korisnika u skladu sa *RBAC* modelom. U trećem koraku servis kreira instance klase *Claim* koja predstavlja tvrdnju.

U četvrtom koraku, kreira se žeton predstavljen uređenim parom (*String opisDozvole*, *Enum tipDozvole*) i žeton se potpisuje privatnim ključem servisa za identifikaciju zarad neporecivosti. Tip dozvole može biti dozvola na nivou permisije ili uloge.



Slika 34 - Zahtev za izdavanje bezbednosnog tiketa

U koracima pet i šest se odvija enkripcija za namenjeni servis. Za enkripciju i neporecivost poruka se koriste *X509v3* sertifikati. Sertifikati između ostalih informacija, sadrže javni ključ koji se koristi u postupcima asimetrične kriptografije. Servis identiteta dobija sertifikat prijemne strane putem servera koji sadrži sertifikate (Slika 35).



Slika 35 - Enkripcija bezbednosnog tiketa namenjena prijemnoj strani

Proces enkripcije se oslanja na obe vrste kriptografije, simetričnu i asimetričnu. Simetrična kriptografija se odlikuje time što se isti ključ koristi i za kriptovanje i za dekriptovanje. Kod asimetrične kriptografije, kreira se par ključeva: privatni i javni. Podaci se kriptuju privatnim ključem, dok je onda moguće jedino javnim ključem dekriptovati poruku. Obično se ovaj postupak primenjuje na potpisivanje poruke i na ovaj način se postiže neporecivost poruka. Simetrična kriptografija je mnogo brža od asimetrične zbog složenosti algoritma, i ovo je razlog što se u implementaciji koristi i asimetrična i simetrična kriptografija.

Proces izdavanja tiketa podrazumeva više kriptografskih procesa. Nakon kreiranja bezbednosnog žetona, prvo se žeton potpiše koristeći privatni ključ izdavača žetona, *STS* servisa.

Potpisom je izdavač žetona neporecivo nedvosmislen. Nakon potpisivanja žetona, kreira se privremeni simetrični ključ. Novokreirani simetrični ključ se koristi za enkripciju žetona, što obezbeđuje nečitljivost sadržaja sve dok prijemna strana ne sazna vrednost ključa. Izdavač žetona uvek namenjuje žeton za prijemnu stranu, ali je problem to što može doći do čitanja poruke u toku međuslojne komunikacije.

Ovaj problem je rešen tako što se se privremeni simetrični ključ enkriptuje javnim ključem ciljnog servisa. U implementaciji ovog rešenja, ciljni servis je filtrirajući servis koji je opisan u odeljku 5.9. Na ovaj način, dekripciju privremenog simetričnog ključa može izvršiti samo servis za filtraciju podataka, jer jedino ovaj servis ima pristup svom privatnom ključu.

Na opisani način korišćenja kriptografije bazirane na sertifikatima, postignuta je mogućnost za implementaciju kontrole pristupa bazirane na tvrdnjama. U pristupu baziranom na tvrdnjama, neophodno je da postoji izdavačko telo od poverenja. Da bi se poverenje uspostavilo, i da bi servis bio siguran da su tvrdnje izdate od strane validnog *STS*, neophodno je da se obezbedi bezbednost u komunikacionom kanalu. Bezbednosni tiket prolazi u svom životnom ciklusu kroz više servisa, i neophodno je koristiti kriptografiju kako bi se sadržaj zaštitio od čitanja za posredne servise. Što je još važnije, tvrdnje se moraju zaštititi od modifikacije. Potpisivanjem i enkripcijom su navedeni zahtevi rešeni.

Tvrdnje pomažu pri faktorizaciji logike autentifikacije izvan logike aplikacije. Usled odabira kontrole pristupa bazirane na tvrdnjama, došlo je do razdvajanja odgovornosti kao što je to i zamišljeno u servisno orijentisanom pristupu. Došlo je i do mogućnosti izmene implementacije autentifikacionog mehanizma jednostavnom zamenom servisa koji je u ulozi izdavača bezbednosnih tiketa. Konačno, logika autentifikacije korisnika je izmeštena. Pojednostavljen je sam izvorni kod aplikacije jer koristi kontrolu pristupa baziranu na tvrdnjama.

5.9. Implementacija filtrirajućeg servisa

Filtrirajući servis pripada sloju podataka, i primenjuje se u slučaju potpuno deljene baze podataka (odeljak 3.6.5). U implementaciji rešenja koje predlaže ovaj rad, odabran je pristup potpuno deljene baze podataka. Rešenje je pravljeno tako da bude lako proširivo, i da podržava sva tri šablona za višeorganizacijskog sloja podataka. Trenutno rešenje ne podržava samo pristup deljenih baza sa zasebnim šemama. Izabrani pristup je potpuno deljena baza podataka jer je sa bezbednosne strane najizazovniji problem, a i omogućava najveći stepen iskorišćenosti višeorganizacijske baze podataka.

U slučaju implementacije pristupa deljenih baza sa zasebnom šemom, potrebno je implementirati zaseban servis. Odgovornost novog servisa bi bila kreiranje nove šeme i novog korisnika baze podataka čija bi podrazumevana šema bila ona koja je kreirana u prethodnom koraku. Nakon uspešne inicijalizacije baze podataka, servis bi još morao na neki način pravilno da upiše višeorganizacijske meta podatke. Ovakav servis je predmet za budući rad i njegovo dodavanje će biti lako omogućeno jer je u servisu za konfigurisanje višeorganizacijskog modela ovakav vid proširivosti već predviđen.

Kada se podaci više organizacija čuvaju u okviru jedne baze podataka, svakoj tabeli se dodaje kolona diskriminacije. Stoga, neophodno je primeniti filtrirajuću logiku. U *SQL* upitu bi filtrirajući upit sadržao identifikator organizacije u *where* klauzuli:

```
Select * from ImeTabele where TenantID = IdentifikatorOrganizacije
```

Na ovaj način, logika filtriranja bi mogla da se implementira u okviru pogleda u bazi podataka (engl. *View*) koristeći zasebne korisničke naloge po organizaciji. Identifikator korisničnog naloga u okviru baze podataka bi imao ulogu *IdentifikatorOrganizacije* filtera. U ovakvom pristupu, postoji problem jer na aplikativnom nivou može da se pozove upit nad tabelom, a ne nad pogledom što bi dovelo do preuzimanja podataka svih organizacija. Stoga, ovakav pristup ipak nije primeren.

Usled ograničenja sistema za upravljanje bazama podataka vezanog za filtriranje podataka, višeorganizacijsko filtriranje se mora rešiti na aplikativnom nivou. Kako ova logika ne bi bila implementirana u svakom višeorganizacijskom servisu zasebno, odgovornost servisa za filtriranje jeste usluživanje korektnih podataka. Strogo gledano, ulazni parametri u servis su *SQL* upit i dobijeni bezbednosni tiket, a odgovor su traženi podaci.

Uvođenjem servisa za usluživanje podataka, arhitektura celokupnog rešenja dobija notu resursno orijentisane arhitekture, ili kraće *ROA* (engl. *Resource Oriented Architecture*) arhitektura. Za implementaciju ROA, korišćena je tehnologija *WCF Data Services*.

WCF Data Services pružaju pristup podacima, koji su reprezentovani kao objekti modela entiteta, putem *HTTP* protokola. Podaci se adresiraju koristeći *REST* (engl. *Representational State Transfer*) *URI*, a sama tehnologija je i zamišljena da obezbedi implementaciju *REST* servisa. Kako je u *ROA* arhitekturama i zastupljeno, resursima se pristupa putem *HTTP GET* metoda navodeći *URI*, dok servis vraća traženi resurs. Servis može biti konfigurisan tako da podatke vraća u čistom *XML* formatu, *JSON* formatu ili kao *XML* uz *RDF*.

Uz *WCF Data Services*, postoji biblioteka za korišćenje ovih servisa, samo je potrebno uključiti odgovarajući prostor imena u klasi koja treba da koristi servis podataka. *LINQ* upiti, koji su proširenje *.NET* okruženja, se automatski transliraju na *REST* upite, tako da je transparentnost korišćenja sloja podataka postignuta na ovaj način. Stabilnost *WCF Data Services* tehnologije je razlog zašto je ona odabrana u implementaciji ovog rešenja. S druge strane, usled svoje interoperabilnosti, postoji smanjenje efikasnosti jer se skupovi podataka moraju serijalizovati u *XML*. Ovo će uticati na performanse jer osim vremena potrebnog za serijalizaciju, izlazni *XML* je memorijski zahtevniji od izvornog binarnog formata.

Zamena *WCF Data Services* tehnologije bi omogućilo ubrzanje performansi, jer bi bilo moguće izbaciti nepotrebnu serijalizaciju i deserijalizaciju. Isto tako, *WCF Data Services* sadrži ozbiljna ograničenja, kao što je nemogućnost pisanja *join* upita sa klijentske strane. Sve ovo utiče na potrebu za *ROA* pristupom, gde bi se repozitorijum logika iz šablona repozitorijuma morala implementirati u okviru servisa za filtriranje podataka. Zbog ovakvih principa, servis filtriranja podataka postaje skoro deo konkretnog rešenja problema i kao takav gubi svoju generičnost. Zamenom *WCF Data Services* tehnologije, mogla bi se dobiti potpuna transparentnost što i jeste bio cilj. Gotovo je izvesno, da bi prvi postupak za dalje razvijanje rešenja koje preporučuje ovaj rad, bilo da se sama tehnologija izmeni ili prilagodi tako da podržava potpunu funkcionalnost.

Pre svakog filtriranja podataka, neophodno je proveriti autorizaciju samog zahteva (Slika 36). Kao i kod višeorganizacijskog *web* servisa koji je opisan u odeljku 5.7. Sloj za filtriranje podataka mora razlikovati da li je zahtev od autorizovanog korisnika ili ne. Cilj čitavog rešenja jeste transparentnost korišćenja višeorganizacijskih servisa. U transparentnost spada pisanje upita kao da ne postoji višeorganizacijska baza podataka.

Kada bi se filtriranje podataka po organizaciji vršilo u samom upitu u okviru višeorganizacijskog servisa koji koristi okruženje, ovo bi dovelo do toga da postoji rizik da dođe do katastrofalne greške. U slučaju izostavljanja filtera po organizaciji ili greške u samom filteru, došlo bi do učitavanja pogrešnih podataka. Zbog toga je cilj rešenja da podaci budu usluženi samo za određenu organizaciju, ali bez dodavanja filtera u okviru upita koji se šalju samom sloju podataka.

U implementaciji filtrirajućeg servisa, u *header* sekciji *HTTP* zahteva se mora naći bezbednosni tiket ili identifikator organizacije. U slučaju da ne postoje odgovarajuća polja, servis će generisati izuzetak koji signalizira narušavanje bezbednosti. Ukoliko je reč o javnim podacima, dovoljno je da se pronađe samo identifikator organizacije. U suprotnom, neophodan je bezbednosni tiket.

Slika 36 - Servis filtriranja podataka

Filtriranje je bazirano na presretačima poruke zahteva (engl. *interceptors*). *HTTP* zahtev sa upitom biva presreten i prvo se traže odgovarajuća polja u *header* sekciji. Implementacija se razlikuje u slučaju autorizovanog korisnika i neautorizovanog.

Pošto je slučaj autorizovanog korisnika kompleksniji, uzeće se za primer da se u okviru *header* sekcije nalazi bezbednosni tiket. Bezbednosni tiket je prvo potrebno raspakovati, postupak obrnut od postupka potpisivanja i enkriptovanja tiketa koji je opisan u odeljku 5.8. Prvo se vrši dekripcija privremenog simetričnog ključa, što servis čini svojim privatnim ključem. Ovo je ključno za bezbednost čitavog sistema. Privatni ključ mora ostati samo u posedu servisa za filtriranje podataka, u suprotnom, čitav sistem postaje ranjiv na zlonamerne napade. Nakon dešifrovanja privremenog simetričnog ključa, isti se koristi za dekripciju bezbednosnog žetona. Potom se vrši validacija potpisa tako što se preuzme sertifikat servisa izdavača bezbednosnih tiketa. Svi sertifikati su javno dostupni. Ukoliko se u bilo kom koraku čitanja bezbednosnog žetona dogodi greška, servis će smatrati žeton oštećenim i generisati izuzetak.

Nakon dešifrovanja bezbednosnog žetona, servis za filtriranje preuzima tvrdnje koje su zapisane u okviru žetona i najvažnije, identifikator organizacije. Identifikator organizacije koristi

za filtriranje podataka. Tvrdnje koje su od značaja su permisije i uloge. Koristeći uloge koje korisnik poseduje, preuzimaju se iz baze podataka permisije vezane za odgovarajuće uloge.

Moć kontrole pristupa bazirane na tvrdnjama, jeste pojačanje osnovnog *RBAC* modela, a bazira se na tome što uz tvrdnju uloge, korisniku se može pridodati i proizvoljna permisija. Na ovaj način postoji finija granulacija upravljanja permisijama, tako što je moguće određenom korisniku, osim uloge dodeliti i određenu permisiju.

Na kraju, kada se preuzmu sve permisije vezane za korisnika, vrši se provera da li korisnik sa permisijama koje poseduje može da preuzme podatke koje zahteva. Ukoliko ima odgovarajuće permisije, podaci se preuzimaju iz baze podataka. Potom, podaci se serijalizuju u *XML* format i to je ujedno odgovor filtrirajućeg servisa.

Servis za filtriranje podataka omogućava pristup potpuno deljenih baza podataka i u slučaju kada baze podataka nemaju opciju podele podataka u odnosu na vrednost nekog od atributa u tabeli. Osim toga, servis za filtriranje podataka ima veoma važnu ulogu u bezbednosti čitavog sistema jer predstavlja ulaznu tačku za preuzimanje podataka.

S obzirom na proveru autorizacije i osiguranje podele podataka po organizacijama, servis za filtriranje podataka je mesto gde se rizik od greške mora eleminisati. Prostor za optimizaciju performansi na ovom nivou postoji, jer se može promeniti način provere korisnika, izbaciti serijalizacija a i moguće je promeniti način presretanja poruka u slučaju višestrukih spajanja tabela.

5.10. Servis konfiguracije višeorganizacijskog modela

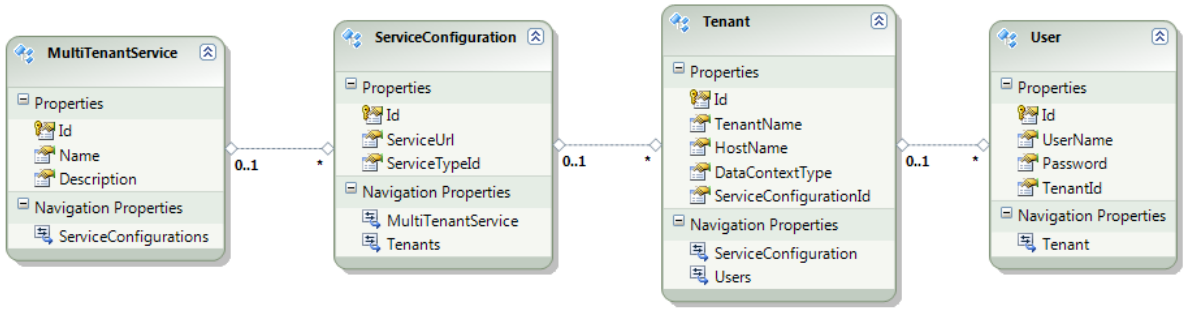
Cilj servisa konfiguracije višeorganizacijskog modela je da pruži logički centralizovano mesto na kom se mogu konfigurisati višeorganizacijski servisi, i to po konkretnim organizacijama koje su pretplaćene na višeorganizacijski SkS. Da bi ovo bilo moguće, višeorganizacijski servis mora se obraćati servisu za konfiguraciju višeorganizacijskog modela. Na ovaj način, pri svakom zahtevu koji uključuje preuzimanje podataka, višeorganizacijski servis uspostavlja komunikaciju sa servisom za konfiguraciju višeorganizacijskog modela.

S obzirom na stalnu komunikaciju sa servisom konfiguracije višeorganizacijskog modela, može se posmatrati da je višeorganizacijski model konfigurabilan tako da su posledice vidljive u realnom vremenu. Tako, ukoliko administrator promeni lokaciju servisa za filtriranje podataka, već u narednom zahtevu će saobraćaj biti preusmeren na drugu adresu. Ovaj komoditet u upravljanju višeorganizacijskim servisima se plaća povećanom komunikacijom u servisno orijentisanom okruženju.

Model baze podataka koja bi sadržala najosnovnije informacije potrebne za konfiguraciju različitih višeorganizacijskih servisa je jednostavan (Slika 37). Može se uočiti da je ovakav model orijentisan ka konfiguraciji višeorganizacijskog servisa.

Organizacija je predstavljena tabelom *Tenant*. Jedna organizacija može imati više korisnika. Prikazana polja korisnika su korisničko ime i šifra, ali ostali podaci o korisnicima se dobavljaju od servisa za identitet. Organizacija sadrži polja vezana za identifikaciju različitih organizacija, i ima polje *DataContextType* koja ukazuje na pristup realizacije višeorganizacijske baze podataka. Organizacija sadrži više servisnih konfiguracija jer organizacija generalno koristi

više višeorganizacijski orijentisanih servisa. Servisne konfiguracije se odnose na određene višeorganizacijske servise, koji imaju svoj naziv i opis.



Slika 37 - UML model baze metapodataka višeorganizacijskog servisa

Jedno ograničenje koje ovakav model nameće, jeste da se u okviru jedne organizacije ne mogu menjati načini implementacije višeorganizacijske baze podataka po korisniku. Ovo bi bilo moguće ako bi se polje *DataContextType* premestilo u tabelu korisnika. Međutim, namerno je implementacija ostavljena tako da po organizaciji može da se bira kontekst podataka, jer bi u suprotnom došlo do veoma razgranatog rešenja i poteškoće bi nastale u fazi održavanja višeorganizacijskog okruženja. Kada bi se rešenje proširilo za servis koji implementira pristup deljenih baza podataka sa različitom šemom, bilo bi dovoljno dodati po jednu torku u tabele *ServiceConfiguration* i *MultiTenantService*. *DataContextType* bi bilo potrebno postaviti na odgovarajuću konstantu i na taj način bi rešenje bilo prošireno.

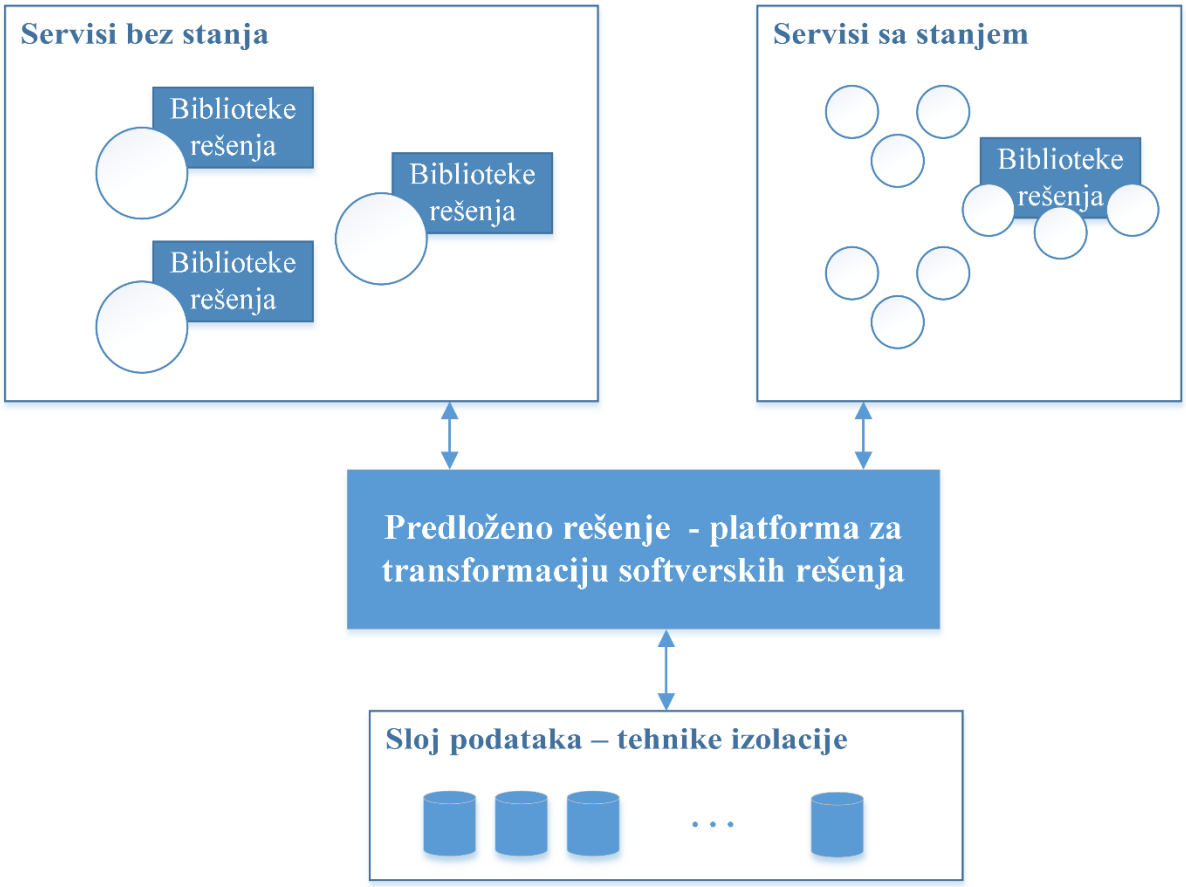
Servis za konfigurisanje višeorganizacijskog modela je realizovan s jedne strane kao *web* servis, a s druge strane kao *WCF* servis. Upis u bazu podataka se vrši isključivo kroz interfejs *web* aplikacije. Za menjanje konfiguracije i dodavanje nove, potrebno je da administrator višeorganizacijskog okruženja bude autorizovan korisnik sistema.

5.11. Integracija servisa za podršku više organizacija

Predloženo rešenje je za ciljni softver dostupno u vidu dostupnih interfejsa (API) pošto je samo rešenje SOA bazirano i biblioteka koje na pravi način koriste dati API. Ideja je da se na ciljnom okruženju uz što manje modifikacija obezbedi ulazni kontekst u sistem. Kontekst organizacije predstavlja informaciju o kojoj organizaciji je reč u svakom zahtevu, a ona se dobija prilikom prijave na sistem što znači da rešenje obuhvata i upravljanje identitetom i pravima pristupa, tj. integrisano je kako je opisano u odeljku 5.8.

Kako se rešenje postavlja uz ciljni softver, sama platforma nije ograničena na vid infrastrukture koja se koristi, od javnih *cloud* okruženja do privatnih infrastruktura.

Slika 38 prikazuje poziciju predloženog rešenja pri dizajnu migracije. Svi postojeći servisi u postojećem softveru se mogu integrisati sa predloženom platformom putem biblioteka, koje se baziraju na tome da se interfejsi prošire dekoracijama nad postojećim metodama u vidu AOP paradigme. Dekoracije će uticati na integrisanje sa servisima predloženog rešenja i na taj način pružiti jasnu informaciju o organizaciji za svaki pojedinačni zahtev. Osim postojanja konteksta, biblioteke su takođe u stanju koristeći API rešenja da preuzimaju i snimaju podatke u sloju za čuvanje podataka.



Slika 38 - Integracija predloženog rešenja

Kod servisa bez stanja, kao što je i sam *web* servis koji je dat kao primer u opisu rešenja, proširenje je trivijalno i servisi se mogu automatski deliti između više organizacija. S obzirom da je znanje o kontekstu implementirano u okviru rešenja i pratećih biblioteka, ciljni softver uopšte ne mora da se modifikuje u smislu da vrši upravljanje kontekstom organizacije. Ukoliko ima potrebe da se unutar poslovne logike razlikuju organizacije (slučaj prilagodljivosti), servisi unutar svoje poslovne logike mogu biti modifikovani tako da promene odgovaraju pravim zahtevima.

Kod servisa sa stanjem je situacija naravno komplikovanija, gde postoje dva pravca. Mogu se integrisati na isti način, uključivanjem biblioteke, ali tada je potrebno izmeniti internu logiku servisa da može da vodi računa o različitim stanjima različitih organizacija. Ovde platforma pomaže jer rizik od greške je smanjen, ali i dalje ne rešava glavne probleme, a to je podela stanja i izolacija. Predloženo rešenje je efikasnije u slučaju odluke da se mikroservisi (ili servisi) sa stanjem instanciraju po organizaciji, a da sama platforma vodi računa o tome da se na osnovu konteksta organizacije adresira prava instanca servisa. U praksi je ovaj način verovatniji ishod, jer servis sa stanjem predstavlja praktično spajanje podataka i funkcionalnosti, što znači da predstavlja idealnu izolacionu jedinicu iz ugla stanja.

Treća stvar koja stiže kroz biblioteke (koje pozivaju API rešenja) jeste informacija o lokaciji interfejsa za rad sa podacima, ili lokacija baze podataka u slučaju rada sa potpuno odvojenim bazama podataka. Uloga platforme u slučaju odvojenih baza podataka je analogna sa ulogom pri radu kod odvojenih servisa sa stanjem.

Sumarno, integracija sa rešenjem je jednostavna i sastoji se od uključivanja biblioteka uz relativne modifikacije ciljnog koda. Tabela 7 daje pregled potrebnih modifikacija u zavisnosti od toga koji nivo višeorganizacijskog svojstva se cilja. Kolona „Posvećeni“ označava to da li je postojeći deo softvera zamišljen da bude instanciran po organizaciji, ili da se njegova poslovna logika izmeni tako da podržava višeorganizacijsko svojstvo.

Tabela 7 - Mogućnosti integracije

Deo ciljnog softvera	Posvećeni?	Modifikacije
Servisi bez stanja	Da	Zbog lakše mogućnosti podele, praktično nema smisla ne primenjivati višeorganizacijsko svojstvo u potpunosti na primeru servisa bez stanja.
Servisi bez stanja	Ne	Uključivanjem biblioteke, servis u potpunosti postaje višeorganizacijski i u mogućnosti je da procesira zahteve različitih organizacija na bezbedan način.
Servisi sa stanjem	Da	Nisu potrebne dodatne modifikacije jer će ciljno rešenje rutirati zahteve pravim servisima. Postojeći softver učestvuje u višeorganizacijskom sistemu bez modifikacija.
Servisi sa stanjem	Ne	Uključivanjem biblioteke, na bezbedan način dobija se kontekst organizacije u svakom zahtevu. U zavisnosti od svoje poslovne logike, postojeći servis mora pretrpeti izmene tako da u svojoj poslovnoj logici efikasno deli stanje i izoluje podatke različitih organizacija. Pri tome, mora se zadovoljiti mogućnost skalabilnosti. Uzimajući u obzir ograničenja, servisi sa stanjem će se interno deliti u retkim situacijama, tj. kada je interno stanje dovoljno malo i performanse nisu od kritičnog značaja.
Baze podataka	Da	Nisu potrebne dodatne modifikacije jer će ciljno rešenje rutirati zahteve pravim servisima. Postojeća baza podataka učestvuje u višeorganizacijskom sistemu bez modifikacija.
Baze podataka	Ne	Baze podataka moraju biti izmenjene prateći tehnike izolacije opisane u 3.6.2. Za pristup potpuno deljenih baza podataka, rešenje sadrži filtrirajući servis koji se može iskoristiti kroz date biblioteke.

Svaki softverski sistem se sastoji od niza komponenti. Ako se fokusiramo na SOA, može se zaključiti da je sistem sačinjen od skupa servisa. Prednost je što se svaki od servisa može preneti na višeorganizacijsko svojstvo ili ne.

U sumi, rešenje može biti mešovito po pogledu aplikativne višeorganizacijske podele ciljnog softvera. S druge strane, gustina servisa čak i koji se interno ne dele na višeorganizacijsko

svojstvo dovodi do boljeg iskorišćenja infrastrukture na kojoj se nalaze, a to je omogućeno predloženim rešenjem. Gustina je omogućena bezbednom logičkom izolacijom, jer je moguće različite instance istog servisa namenjenog različitim organizacijama imati postavljene u okviru istog klastera virtuelnih mašina, posebno u slučaju korišćenja PkS v2 kao osnove nad kojim se izvršava ciljni softver.

6. Testiranje i rezultati

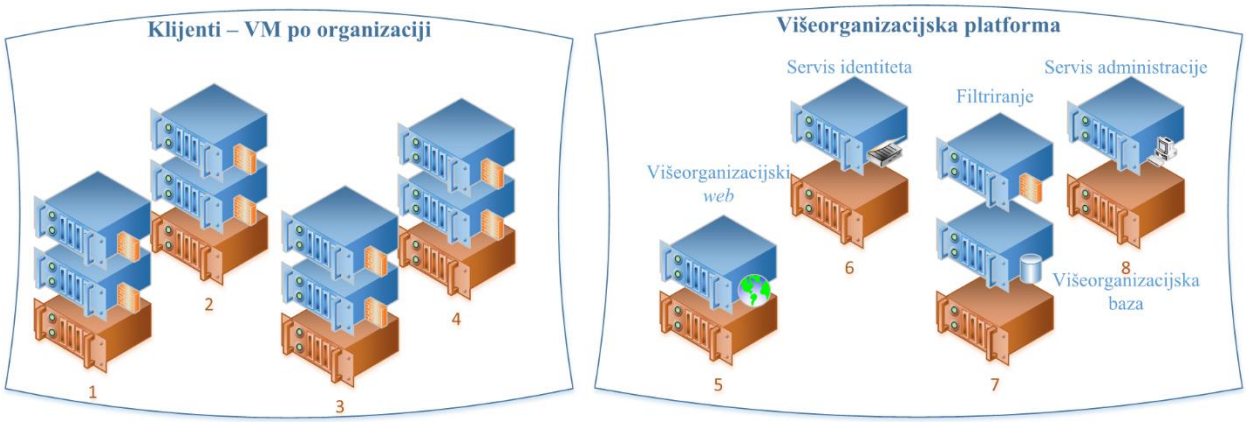
Od postavka hipoteze do postavke eksperimenata koji će moći opovrgnuti podhipoteze identifikovana su dve grupe eksperimenata sa različitim testnim scenarijima i različitim ciljevima. Naredne dve celine su sledeće:

1) PkSv2 nativno višeorganizacijsko svojstvo za aplikativni model baziran na mikroservisima i aplikativnom modelu učesnika – Moguće je postići horizonatlnu skalabilnost oslanjajući se na višeorganizacijsko PkS rešenje bez negativnih uticaja na celokupnu performantnost rešenja
    a. Eksperiment I – akvizicija podataka i korišćenje višeorganizacijske skalabilne baze podataka bazirane na modelu učesnika
    b. Eksperiment II – propitivanje učesnika
2) Višeorganizacijska platforma – Moguće je iskoristiti platformu za transformaciju na multi-tenant model bez suštinskih arhitekturalnih promena postojećeg rešenja uz minimalne negativne uticaje na celokupnu performantnost rešenja
    a. Eksperiment III – slabljenje performansi nakon uvođenja platforme za transformaciju postojećih rešenja na višeorganizacijski model.
    b. Eksperiment IV – kašnjenja uvedena kriptografijom u komunikaciji

Svaki od eksperimenata je organizovan u skladu sa predstavljenom metodologijom u odeljku 1.3, tako da postoje odeljci koji opisuju kontekst i opseg eksperimenta, izvršavanje, prezentaciju rezultata i analizu uz interpretaciju dobijenih rezultata. U narednom odeljku je dat opis testnog okruženja.

6.3. Opis testnog okruženja

Rešenje je implementirano i testirano u privatnom *cloud* okruženju koristeći IkS model. Tabela 8 sadrži specifikaciju fizičkih mašina, koje su prikazane kao serveri na dnu u braon boji (Slika 39). *Blade* serverski računari su bili raspoređeni u okviru jedne šasije koja je povezana na jedan sistem stalne memorije za skladištenje podataka. Mrežna propusnost između fizičkih servera je 1 Gbps.



Slika 39 - Testno okruženje

Čitava konfiguracija infrastrukture je konfigurisana bez ijedne tačke otkaza tako što na svakom nivou postoji dovoljan nivo redundancije: duple mrežne kartice, dupla optička konekcija ka sistemu stalne memorije, konfiguracija diskova u *RAID 5* skupovima diskova od četiri, što podrazumeva da se podaci neće izgubiti u slučaju otkaza jednog od četiri diska.

Svi fizički serveri su imali postavljen *Microsoft Hyper-V 3.0* virtuelizacioni sloj, dok je operativni sistem *Windows server 2012 R2*. Virtuelne mašine iz oba sloja koja uključuju klijentski sloj i višeorganizacijsku platformu su takođe koristile *Windows Server 2012 R2* operativni sistem. Na virtuelnoj mašini koja je izvršavala višeorganizacijski *web* servis jer korišćen *Microsoft IIS* web server, dok je *Microsoft SQL Server 2012* korišćen kao server višeorganizacijske baze podataka.

Tabela 8 - Opis fizičkih komponenti testnog okruženja

Komponenta	**Opis (*engl.*)**
Serverski računari: 1, 2, 3, 4, 5, 6, 7, 8	*Model: HP BL 660c Gen.8* *Processor: 4 x E5-4620v2 (2.6GHz/8core/20MB95W)* *Memory: 32 x 16GB (PC3 -149000R)* *Network: 2 x HP Ethernet 10Gb 2P 560M Adapter* *Network: 2 x HP Flex Fabric 10Gb 2 port 554FLB* *Storage: 2 x 146GB 6G SAS 15k*
Šasija	*HP Blc 7000* *4 x HP 6120XG Blade Switch* *2 x HP Blc Flex Fabric 10GB/24port* *Flex fabric with 2 x FC 8Gb and 2 x RJ45 SFP modules* *2 x HP 32A PDU*
Optički Kanal	*2 x HP 8/24 Base 16 port enabled switch* *HP Premier Flex LC/LC 2m cables* *HP 8Gb SW SFP pack*
Stalna memorija	HP 3PAR StoreServ 7200 2-N (Dual Controller) 3PAR M6710 300GB 6G SAS15k Hard disk

Testovi su izvršavani u okruženju koje se sastojalo od trinaest virtuelnih mašina koje su bile instalirane na osam *blade* fizičkih servera (Slika 39). Fizički serveri 1, 2, 3 i 4 su izvršavali po dve virtuelne mašine i pripadali logičkoj celini klijentskih računara, pri čemu je svaka virtuelna mašina predstavljala različitu organizaciju. Na fizičkim mašinama 5, 6, 7 i 8 je postavljena višeorganizacijska platforma, pri čemu su virtuelne mašine s namerom postavljene na odvojene fizičke servere da bi se što vernije simuliralo realno očekivano distribuirano *cloud* okruženje. Svaka komponenta predloženog rešenja je odvojena i izvršavana na posebnoj virtuelnoj mašini.

Reprezentativna PkSv2 koja je korišćena za je Microsoft Service Fabric. Aplikativni model programiranja koji je korišćen je model učesnika, opisan u 3.4.3. Učesnici su bili raspoređeni na pet simulacionih čvorova koristeći MASF PkS v2, u okviru jednog fizičkog računara od ukupno 2 fizička jezgra i 4 fizičke niti na frekvenciji od 2.3 GHz.

Višeorganizacijske baze podataka su bile popunjene sa 1.000.000 testnih torki, u okviru deset različitih ravnomerno raspodeljenih organizacija.

6.4. Eksperiment I – Akvizicija podataka

Eksperiment I se odnosi na akviziciju podataka u scenariju predstavljenom u odeljku 5.5, u sekciji C. U narednim odeljcima je predstavljen eksperiment.

6.4.1. Kontekst i opseg

Svrha testova je bila da pokaže da li se može koristiti distribuirana baza podataka bazirana na učesnicima, jer omogućava prirodnu višeorganizacijsku podelu gde svaki učesnik može voditi računa o podacima različitih klijenata, ili uređaja. Da bi se smatralo da baza može da se koristi, trebalo bi da zadovoljiti propusnu moć za definisani opseg eksperimenta.

Opseg eksperimenta se svodi na kreiranje simuliranih organizacija tako da odgovara predviđenom uzorku. Da bi se postiglo rešenje koje zadovoljava količine podataka koje su zahtevane u industrijskim distributivnim rešenjima koristiće se referentne veličine malih projekata za NDMS industrijske sisteme što podrazumeva elektrodistributivne mreže sa do ~250.000,00 krajnjih potrošača, gde referentna distributivna šema sadrži ~66.000,00 mernih signala. Dakle, ideja je proveriti u kojim uslovima MASF klaster baziran na učesnicima procesira 66.000,00 mernih signala.

Internet stvari su za potrebe eksperimenata bile simulirane u okviru distribuiranog klastera od deset virtuelnih mašina. Stvari, tj. uređaji u domaćinstvima, je bilo dvadeset po krajnjem potrošaču a bile su simulirane računarskim procesima koje su slale podatke direktno učesnicima. Svaka organizacija je slala podatke u četiri različite konekcije.

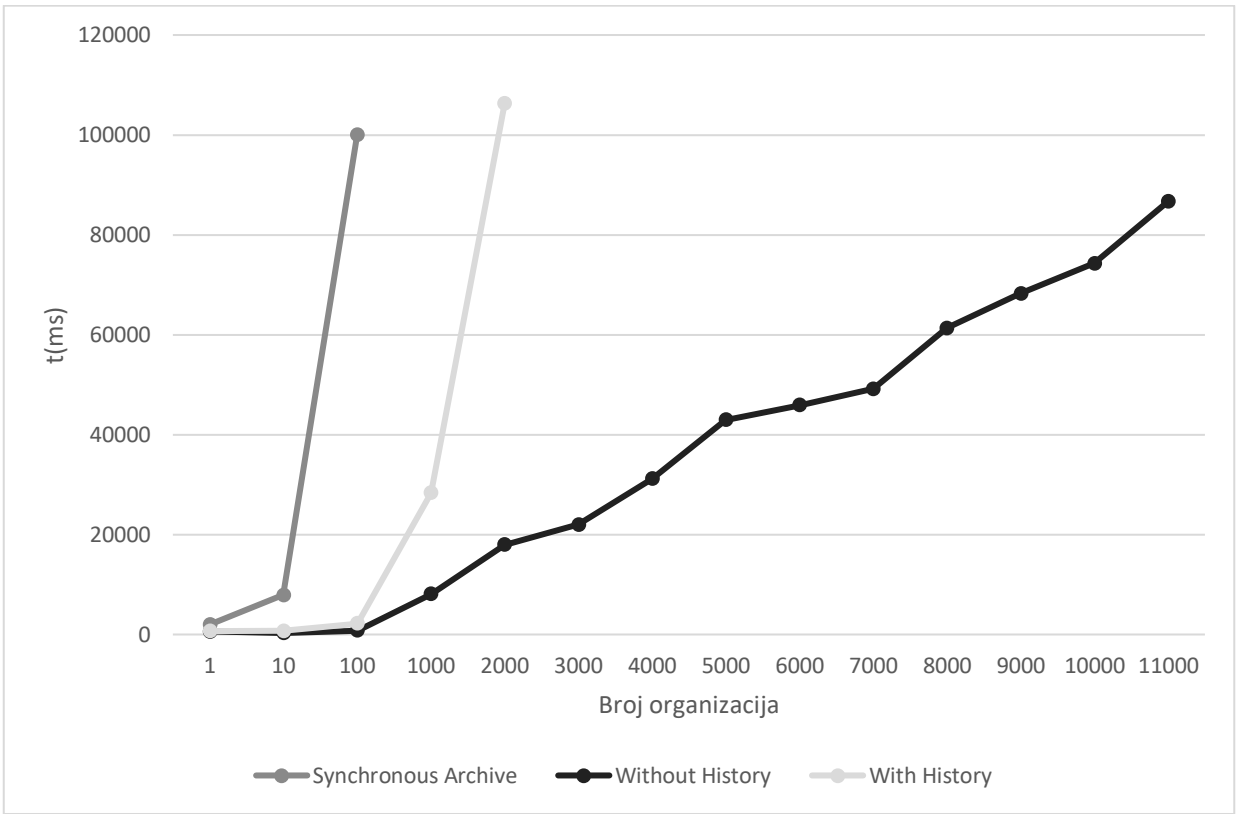
6.4.2. Izvršavanje

Eksperiment je kontinualno izvršavan u trajanju od pet dana. Dobijeni rezultati predstavljaju merenja sprovedena deset puta u okviru kontinualnog eksperimenta. S obzirom da je bilo tri tipa merenja navedenih u sekciji prezentacija rezultata, za svaki tip su vršena merenja dva puta dnevno.

6.4.3. Prezentacija rezultata

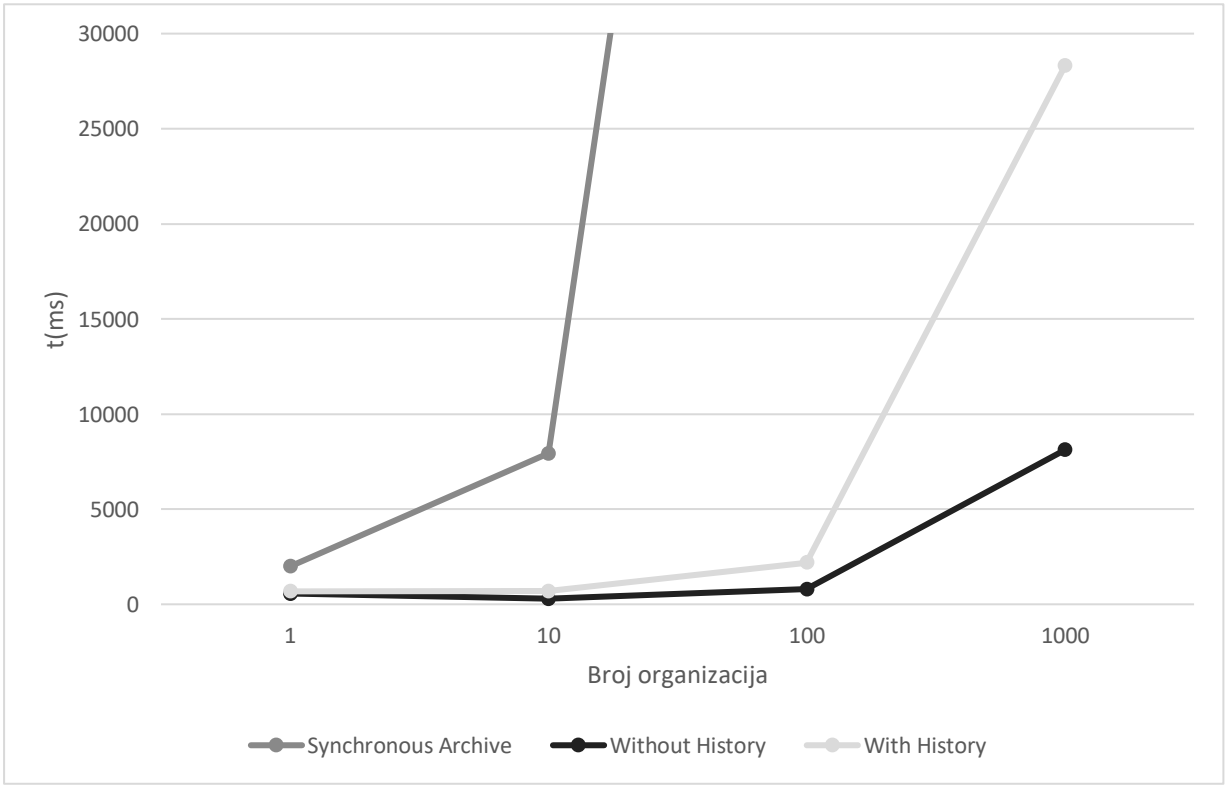
Slika 40 prikazuje grafik koji predstavlja jeddno popunjavanje baze podataka tako što prima podatke od simuliranih stvari, tj. podatke od organizacija. Inicijalno, baza podataka je bila potpuno prazna.

X osa prikazuje broj simuliranih organizacija, dok Y osa prikazuje vreme izvršavanja u milisekundama. „*Without History*“ kriva se odnosi za merenja koja nisu uključivala slanje podataka istorijskim učesnicima, dakle odnosi se na scenario bez čuvanja ikakve istorije. „*With History*“ kriva je za merenja u kojima su učesnici nakon primanja mernih odbiraka slali istorijske podatke istorijskim učesnicima kako bi u prvom sloju bile čuvane samo aktuelne merne vrednosti. „*Synchronous Archive*“ kriva je za merenja gde je slanje istorijskih podataka slato u sinhronom maniru.



Slika 40 - Potpuna akvizicija podataka

Slika 41 predstavlja samo uveličani pogled prethodnog grafa (Slika 40) da bi se mogla sagledati vremena izvršavanja za manji broj organizacija, do 1000, gde se počinje i primećivati eksponencijalni rast merenih vrednosti.



Slika 41 - Akvizicija podataka do 1000 organizacija

6.3.3. Analiza i interpretacija

Sinhrono arhiviranje je u potpunosti ukazalo na veoma nezadovoljavajuće performanse što je posledica prirode modela baziranog na učesnicima. To je razlog zašto se testiranje nije ni vršilo za slučaje gde postoji više od sto organizacija. Zaključak je da bi primena sinhronog arhiviranja dovela do potpuno nezadovoljavajućeg rešenja.

Možda bi bolje rešenje bilo koristiti pristup agregacije podataka kao što je to obrađeno u eksperimentu broj dva. Istorijski učesnici bi propitivali prvi sloj učesnika na predefinisanim vremenskim intervalima. Ovo zavisi od postavljenih ciljnih performansi krajnjeg rešenja kao i od potrebne tačnosti u vremenu istorisjkih servisa, ali je ovakvo rešenje u principu nezadovoljavajuće u opštem slučaju kakav je postavljen.

Telemetrija uspeva da ispod minuta učita dvostruko više u distribuiranu bazu podataka na postavljeni cilj od 66.000 mernih signala. Na primeru 3.000 organizacija (domaćinstava) simuliranih sa po 20 uređaja daje cifru od 60.000 mernih signala, što se učita u asinhronom distribuiranom rešenju za ispod 20 sekundi. Uz konstataciju da je eksperiment izvršen na samo jednom računaru sa posvećene dve fizičke niti što je jako ograničavajuće za gustinu učesnika, može se pretpostaviti skalabilnost rešenja u slučaju distribuiranog MASF klastera.

U slučaju ulančavanja učesnika (kriva „*With History*“) usled gustine učesnika dolazilo je često do njihove aktivacije i deaktivacije što je dalo vidljivo lošiji odziv. U slučaju distribuiranja rešenja, moglo bi se postići zadovoljavajuće rešenje iako ovi rezultati jasno pokazuju da učesnici nisu zamišljeni kao procesiranje toka podataka u brzom realnom vremenu.

Zaključak analize jasno ukazuje na to da se model učesnika treba koristiti u sledećim situacijama:

1) Ulančavanje učesnika u dubini više od jedan može izazvati nepredvidive odzive iz razloga što je asinhronizam zastupljen
2) Imamo višeorganizacijski problem koji uključuje veliki broj izolovanih organizacija sa relativno malim internim stanjem. Ukoliko se na nivou logike jednog učesnika može sprovesti procesiranje informacija od važnosti, model učesnika je u redu, ali ukoliko se očekuju performanse od lanca učesnika, model učesnika verovatno neće dati odgovarajuće performanse u razumnoj ekonomiji.
3) Skalabilnost višeorganizacijske baze podataka je idealna uz model učesnika, ali na račun asinhronizma, tj. gubitka kontrole za procesiranje toka podataka u realnom vremenu.

Na osnovu zaključaka iz eksperimenta I, uveden je eksperiment II za evaluaciju PkSv2 platforme.

6.5. Eksperiment II – Agregacija podataka

Eksperiment II se odnosi na prikupljanje istorije i proširivanje funkcionalnosti testirane u eksperimentu I. Sam scenario je detaljnije opisan u odeljku 5.5, u sekciji C, pod korakom A. Funkcionalnost preuzimanja kompletnog stanja sistema iz mnoštva nepovezanih učesnika je fokus eksperimenta II. U narednim odeljcima je predstavljen eksperiment.

6.5.1. Kontekst i opseg

Svrha testova je bila da pokaže da li se može koristiti distribuirana baza podataka bazirana na učesnicima u smislu propitivanja većeg broja učesnika u cilju formiranja slike o ukupnom

stanju. Da bi se smatralo da distribuirana baza zadovoljava potrebe, trebalo bi izmeriti odnos potrebnog vremena u sprezi sa brojem učesnika (organizacija) koje učestvuju u eksperimentu. Naravno, broj organizacija je ponovo određen opsegom eksperimenta. Planirano je da se eksperiment izvede tako što će se pratiti dve veličine prikazane u prezentaciji rezultata. Jedno je sama agregacija podataka nad skupom učesnika dok je u drugoj ideja da se nakon prikupljenog ukupnog stanja generiše komanda iz učesnika upućena ka uređajima domaćinstava.

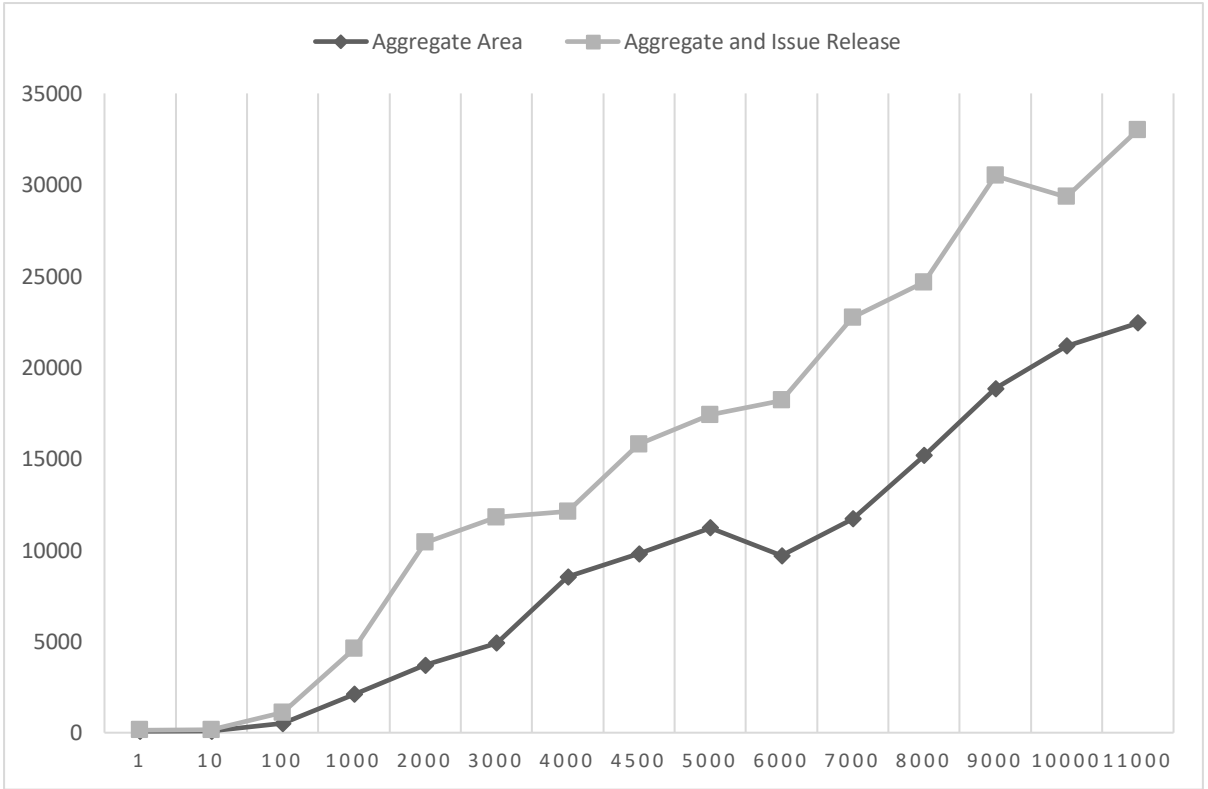
Opseg eksperimenta se svodi na kreiranje simuliranih organizacija tako da odgovara predviđenom uzorku i identičan je opsegu eksperimenta I. Razlika je u cilju koji se želi postići pobudnim uzorcima. Dok je u eksperimentu I ideja bila da se proveri u kojim uslovima MASF klaster baziran na učesnicima procesira 66.000,00 mernih signala, u ovom eksperimentu je komunikacija dvosmerna u smislu da učesnici mogu i da pošalju komande ka stvarima.

6.5.2. Izvršavanje

S obzirom da se eksperiment II izvršavao sukcesivno nakon izvršavanja ciklusa eksperimenta I, izvršavanje je takođe trajalo pet dana. Dobijeni rezultati predstavljaju merenja sprovedena deset puta u okviru kontinualnog eksperimenta. S obzirom da je bilo dva tipa merenja navedenih u sekciji prezentacija rezultata, za svaki tip su vršena merenja dva puta dnevno.

6.5.3. Prezentacija rezultata

Slika 42 prikazuje grafik koji oslikava kako su dodatni učesnici u sistemu prikupljali podatke od prvog sloja učesnika kako bi agregirali podatke u korisne informacije za ostatak sistema.



Slika 42 - Prikupljanje podataka

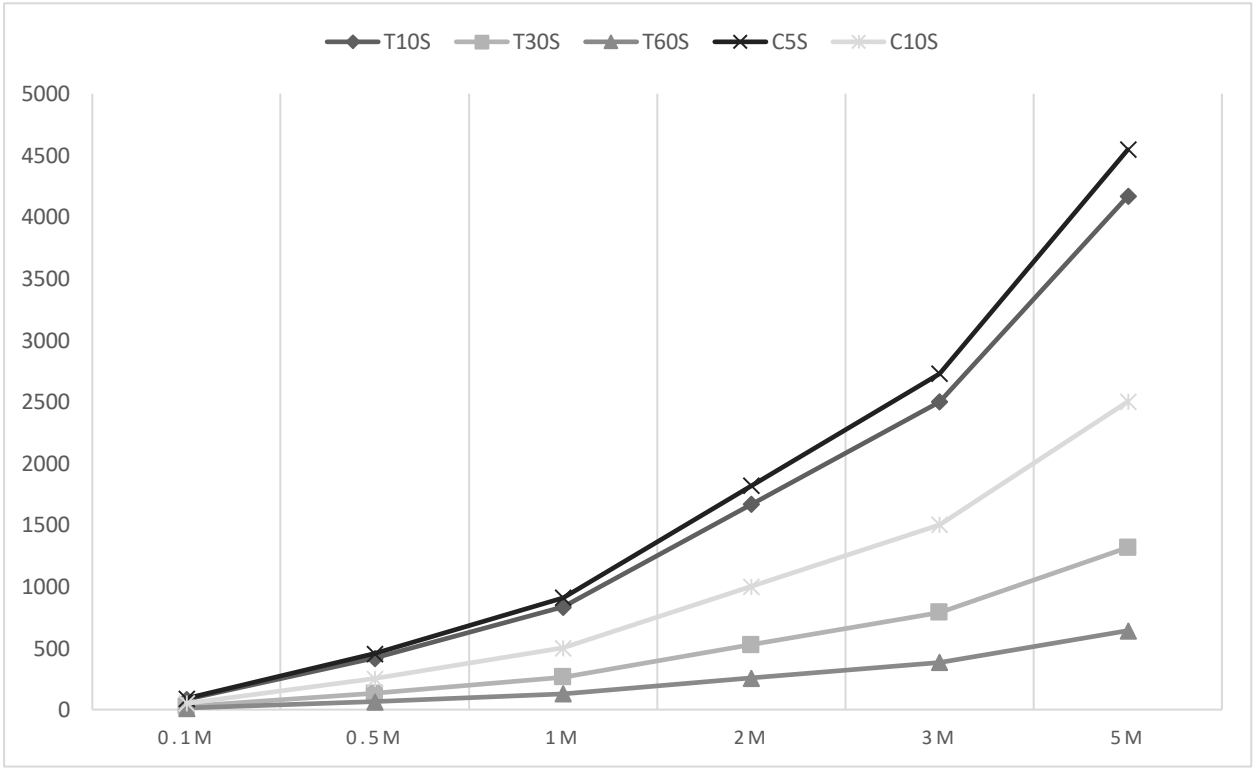
X osa prikazuje broj organizacija, gde je jedna organizacija (konkretno u eksperimentu domaćinstvo) modelovana jednom instancom učesnika prvog sloja rešenja. Y osa prikazuje potrebno vreme u milisekundama. Kriva „*Aggregate Area*“ prikazuje proces propitivanja svih organizacija za količinu potrošnje koju je moguće kontrolisati u programu virtuelne elektrane, kao i agregiranje podataka po transformatorskoj regiji. Kriva „*Aggregate Area and Issue Release*“ prikazuje proces agregiranja svih podataka i izdavanje komande svim učesnicima u jednoj transformatorskoj regiji.

6.5.4. Analiza i interpretacija

Performanse ove krive su od najvećeg interesa za analizu jer da bi se napravila ušteda u energiji, sistem mora biti u stanju da reaguje u kratkim vremenskim odbircima. U suprotnom, samo rešenje nema potpunu korist osim za poznavanje situacije na terenu.

Agregacija stanja uspeva da agregira stanje dvostruko više u odnosu na postavljeni cilj od 66.000 mernih signala za 10 sekundi. Na primeru 3.000 organizacija (domaćinstava) simuliranih sa po 20 uređaja daje cifru od 60.000 mernih signala, agregacija se linearno brže izvrši za 5 sekundi. Iako je eksperiment vršem u strogo ograničenim resursima, linearnost rezultata je prisutna, što potvrđuje horizontalnu skalabilnost koja se tvrdi u aplikativnom modelu učesnika.

U slučaju merenja scenarija gde učesnici dodatno i generišu komande za uređaje, merena vremena se udvostručavaju. Ova pojava je na liniji sa zaključkom iz eksperimenta I, da je ulančavanje učesnika neefikasno u MASF okruženju.



Slika 43 - Odnos troška i performansi

Radi jasnije analize, uvedena je funkcija analize troška (Slika 43). X osa prikazuje broj organizacija, dok bi se broj signala za različite veličine distributivnih kompanija dobio ako se broj organizacija pomnoži sa 20. Y osa pokazuje estimirani trošak koji je izražen u broju virtuelnih mašina potrebnih da podrže bazu distribuiranih učesnika. T10S, T30S i T60S krive predstavljaju estimaciju za prikupljanje jednog kompletnog stanja telemetrije u 10, 30 i 60 sekundi, respektivno.

C5S i C10S krive prestavljaju estimaciju za izdavanje komandi iz agregacionih učesnika za 5 i 10 sekundi, respektivno. Horizontalna skalabilnost čitavog rešenja omogućava da se napravi ovakva funkcija estimacije troška.

Na osnovu eksperimenta se može zaključiti da u slučaju učesnika, poželjno je raditi sa učesnicima koji ne zahtevaju značajnu interakciju sa eksternim komponenatama, uključujući i propitivanje stanja većeg skupa učesnika.

6.6. Eksperiment III – Penal ogranizacionog konteksta

Eksperiment III se odnosi na testiranje rada predloženog rešenja platforme za migraciju postojećih softverskih rešenja na višeorganizacijska, a koja je predstavljena od odeljka 5.6 do 5.11.

6.6.1. Kontekst i opseg

U rešenju su inkorporirani šabloni i pristupi za implementaciju bezbednosnog rešenja. Pod bezbednošću se u najvećoj meri podrazumeva zaštita od preuzimanja podataka pogrešne organizacije. Rešavajući probleme bezbednosti, arhitektura konačnog rešenja je zasnovana na eksternalizaciji bezbednosnog modela i predložena arhitektura se sastoji od više servisa čijom sinergijom se postiže bezbednost višeorganizacijskog modela. Sve ove promene su dovele do distribuiranog rešenja, i povećale broj poruka koje se razmenjuju u komunikaciji, što posledično utiče na performanse sistema.

Stres testiranje ovakvog rešenja je teško izvodljivo. Potrebna je velika količina resursa da bi se rešenje istestiralo onako kako se očekuje u realnom okruženju. Stoga, akcenat testiranja je na poređenju monolitnog tradicionalnog rešenja sa rešenjem koje se oslanja na predloženo rešenje u ovom radu.

Ako se merenja izvrše tako da jedna organizacija vrši pristup *web* aplikaciji u slučaju oba rešenja, može se uporediti koliko se gubi na performansama prelaskom na višeorganizacijski model. Na osnovu ovih merenja, može se odrediti da li *SLA* biva narušen. Kasnije, bilo bi pogodno istestirati svaki od servisa zasebno da bi se dobila optimizacija rešenja na više nivoa.

Test obuhvata merenje rezultata i za slučaj tradicionalne aplikacije, i za slučaj razvijenog višeorganizacijskog rešenja. Višeorganizacijsko rešenje je testirano i pod opterećenjem istovremenog višestrukog pristupanja *web* aplikaciji.

Čitav scenario je krajnje jednostavan, i u slučaju tradicionalnog i u slučaju višeorganizacijskog servisa. Servis treba da odgovori na zahtev tako što će poslati odgovarajuću *web* stranicu kao odgovor. Da bi se stranica izgenerisala, aplikacija mora da preuzme jednu torku od tri atributa iz tabele u bazi podataka. U oba slučaja, radi se o istoj tabeli i tabela je popunjena sa 1.000.000 testnih torki. U okviru ovog eksperimenta, testiran je javni deo *web* aplikacije što znači da se autorizacija nije ni vršila.

Pri pokretanju scenarija sa više organizacija, klijenti koji pripadaju različitim organizacijama (razlikuju se po identifikatoru organizacije koji se prosleđuje tokom kreiranja HTTP zahteva) simultano zahtevaju stranicu, pri čemu su testni klijenti bili postavljeni na različite virtuelne mašine koje pripadaju različitim testnim organizacijama (opis raspodele je dat u odeljku 6.3).

6.6.2. Izvršavanje

Pretpostavka je da servis za filtriranje podataka oduzima najviše vremena. Prvo, zbog presretanja poruka i provere permisija korisnika. Drugo, zbog serijalizacije podataka i njihovog slanja. Od najvećeg interesa je uticaj čitavog okruženja na ukupan *SLA*. Iz tog razloga, vršiće se vrsta testiranja po principu crne kutije, gde će se na kraju uporediti koliko performanse opadaju prelaskom na višeorganizacijski model aplikacija. Potom, izvršiće se testovi opadanja performansi pri simultanom pristupu više organizacija.

Sva merenja su sprovodena po 10.000 puta u okviru jednog ciklusa testiranja. Ciklus testiranja podrazumeva izvršavanje jednog scenarija u predefinisanom broju generisanja uzoraka za taj ekperiment. Testni klijent je krajnje jednostavan i prilagođen testiranju po principu crne kutije. Realizovan je tako da zahteva *web* stranicu i meri vreme potrebno za preuzimanje *web* stranice. U slučaju višeorganizacijskog modela merenja, u testu sa više organizacija od jedne, u isto vreme su pokrenuti klijenti različitih organizacija. Na ovaj način je izvršena simulacija istovremenog pristupanja instanci višeorganizacijskog servisa.

Što se tiče višeorganizacijskih osobina i razmatranja njihovih penala na performanse, koristićen je broj organizacija koji je dovoljan da ostvari uštede od 30% ili više u postavkama kompleksnih poslovnih softverskih rešenja. Na osnovu rezultata objašnjenih u odeljku 7.4.5, na samo šest organizacija se uštede od 30% prevazilaze.

6.6.3. Prezentacija rezultata

Dobijena merenja su prikazana tabelarno (Tabela 9). Prva kolona, model, govori o tome da li se radi o tradicionalnom ili višeorganizacijskom modelu servisa. Druga kolona, br. organizacija, prikazuje broj organizacija koji je učestvovao u ciklusu merenja. Treća kolona, organizacija, prikazuje redni broj organizacije pri testiranju rezultata. Poslednje tri kolone su vremena potrebna za preuzimanje *web* stranice, minimalno, prosečno i maksimalno. Izraženo vreme je u milisekundama.

Tabela 9 - Vremena izvršavanja javnog dela servisa

Model	Br. organizacija	Organizacija	min [ms]	avg [ms]	max [ms]
Tradicionalni	1	1	3	4.3	94
Višeorganizacijski	1	1	18	20.22	320
	2	1	18	22.94	87
		2	18	22.94	73
	4	1	18	26.78	193
		2	18	27.17	267
		3	17	27.09	265
		4	18	26.96	236
	6	1	17	31.15	90
		2	17	31.33	112
		3	18	30.97	80
		4	17	31.31	122
		5	17	31.45	87
		6	17	31.52	89

Minimalna vremena su u slučaju višeorganizacijskog modela ujednačena za proizvoljan broj organizacija. Ovaj rezultat je očekivan, jer je logično da će na uzorku od 10.000 ponavljanja, biti barem jedan slučaj u kom organizacija dobije instancu servisa samo za sebe. U tom slučaju, zahtev biva uslužen za minimalan vremenski period, koji je u svim slučajevima merenja bio oko 18ms. U slučaju maksimalnih vremenskih perioda, u okviru istih grupa testiranja, vremena su približno jednaka.

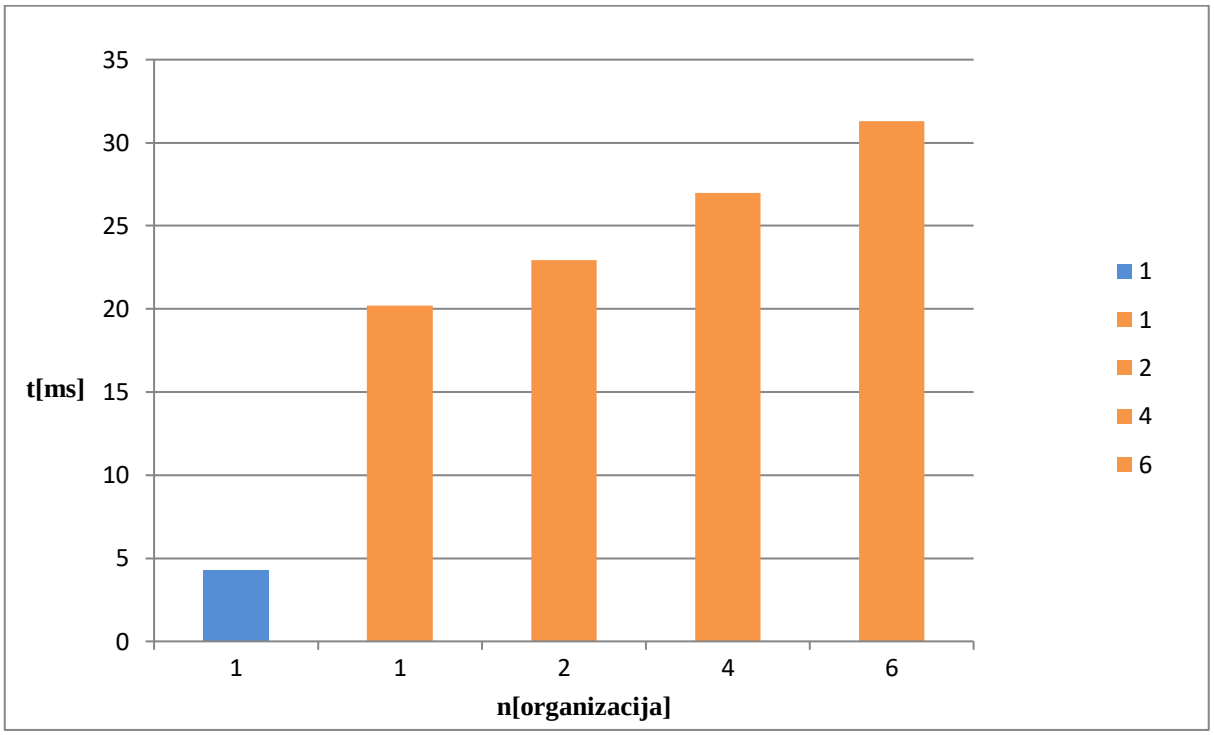
Za razliku od minimalnih vremena, razlika u vrednostima za maksimalna vremena se mogu objasniti povremenim paralelnim pristupima od strane različitih organizacija. Stoga, maksimalna vremena nisu identična kao u slučaju minimalnih, ali su i dalje u očekivanom rasponu vrednosti ukoliko se porede međusobno.

Prosečna vremena u okviru istih grupa merenja su približno ista, što govori o ravnomernoj podeljenosti instance među različitim organizacijama. Činjenica da su prosečna vremena bliska po vrednosti, dovela je do agregiranog prikaza (Tabela 10).

Tabela 10 - Agregirani pregled merenja

Model	Br. organizacija	avg [ms]
Tradicionalni	1	4.3
Multi-tenant	1	20.22
	2	22.94
	4	27
	6	31.29

Cilj agregiranog prikaza je da pruži jasniji uvid na uticaj performansi koji je izazvan prelaskom na višeorganizacijski model i povećavanjem broja organizacija koje dele jednu istu instancu aplikacije. Vizuelizacija agregacija je predstavljena grafikom (Slika 44).



Slika 44 - Pad performansi povećanjem broja organizacija

Na grafiku je tradicionalni model obeležen plavom bojom, dok je narandžastom bojom prikazan višeorganizacijski model. Na horizontalnoj osi se nalazi broj organizacija koje koriste aplikaciju. Na vertikalnoj osi se nalazi vreme izvršavanja izraženo u milisekundama.

6.6.4. Analiza i interpretacija

Grafik (Slika 44) ukazuje na pad performansi prelaskom na višeorganizacijski model. Najveći pad performansi se oseća upravo u prelasku sa tradicionalnog modela na višeorganizacijski model. Ako se uporede tradicionalni i višeorganizacijski model, potrebno je približno pet puta više vremena za preuzimanje iste stranice. Petostruko usporenje može da dovede do zbunjujuće krajnje poruke, jer je usporenje suštinski u redu veličine 15 ms po zahtevu, što bi

važilo i za kompleksnije scenarije koji mogu sami po sebi duže trajati. Tada ne bi važila izjava da višeorganizacijski model usporava rešenje petostruko.

Razlog za pad performansi leži u servisno orijentisanoj arhitekturi rešenja i povećanoj razmeni poruka između servisa. Najveći pad je izazvan servisom za filtriranje podataka jer se uvodi dodatno presretanje poruke, njeno čitanje i odluka da li se permisije proveravaju u slučaju da bezbenosni žeton sadrži identifikator organizacije. Dodatno, trenutna implementacija filtera podrazumeva filtriranje i serijalizaciju podataka kroz XML što dodatno usporava čitav proces. Optimizacijom rada filtrirajućeg servisa, višeorganizacijski model bi sigurno bio brži, tako da se ovi rezultati mogu uzeti kao najgori scenario.

U slučaju više organizacija, na grafiku se vidi da je pad performansi oštriji od linearnog opadanja. Treba se uzeti u obzir da *web* server nije bio skaliran na više instanci pri merenju višeorganizacijskog modela, tako da je broj zahteva rastom broja organizacija linearno bio veći što je dovodilo do većeg opterećena *web* servisa. Pretpostavka je da postoji prag zasićenja, nakon kojeg bi performanse počele naglo da opadaju, ali da to nije ni blizu testiranih šest organizacija.

Zbog mogućnosti usporenja filtrirajućeg servisa pri proveri permisija vezanih za sam zahtev, sledeći eksperiment je planiran tako da pruži odgovor na ovo pitanje.

Hipoteza 3 je delimično pokrivena ovim eksperimentom, gde je pad performansi egzaktno predstavljen, a naredni eksperiment je upotpunjuje i u svojoj analizi sadrži predloge za optimizaciju ovih rezultata.

6.7. Eksperiment IV – Penal kriptografije

Eksperiment IV se odnosi na deo platforme koja čini predloženo rešenje. Model baziran na tvrdnjama se osigurava potpisivanjem i enkriptovanjem bezbednosnih žetona što je opisano u odeljku 5.8.

6.7.1. Kontekst i opseg

Potrebno je uporediti vreme prilikom korišćenja javnog dela *web* aplikacije i vreme koje uključuje prijavu korisnika na sistem koristeći razvijani način autorizacije korisnika. Rezultati će dati uvid u vreme koje se potroši na čitavu kriptografiju koja se koristi u modelu autorizacije bazirane na tvrdnjama, kao i na vreme koje se potroši pri proveri permisija korisnika.

6.7.2. Izvršavanje

Test ima za cilj da pokaže koliko se vremena troši na kriptografske procese i na kontrolu pristupa zasnovanu na tvrdnjama. Dok se prethodni eksperiment odnosi na javni deo *web* aplikacije, i kao takav ne sadrži deo vremena koji se gubi na autentifikacionu logiku, ovaj test se bavi čitavim autentifikacionim procesom, što uključuje izdavanje bezbednosnog žetona i njihovu enkripciju.

Čitav proces autentifikacije se odvija samo u slučaju prijave korisnika na sistem, tako da se izvršavanje testa sastojalo od merenja uzastopnih autorizacija na sistem. Opseg izvršavanja je zadat na isti način kao i u prethodnom eksperimentu, tj. 10.000 merenja u jednom ciklusu testiranja. Testni klijent i metodologija testiranja je ista kao u prethodnom eksperimentu.

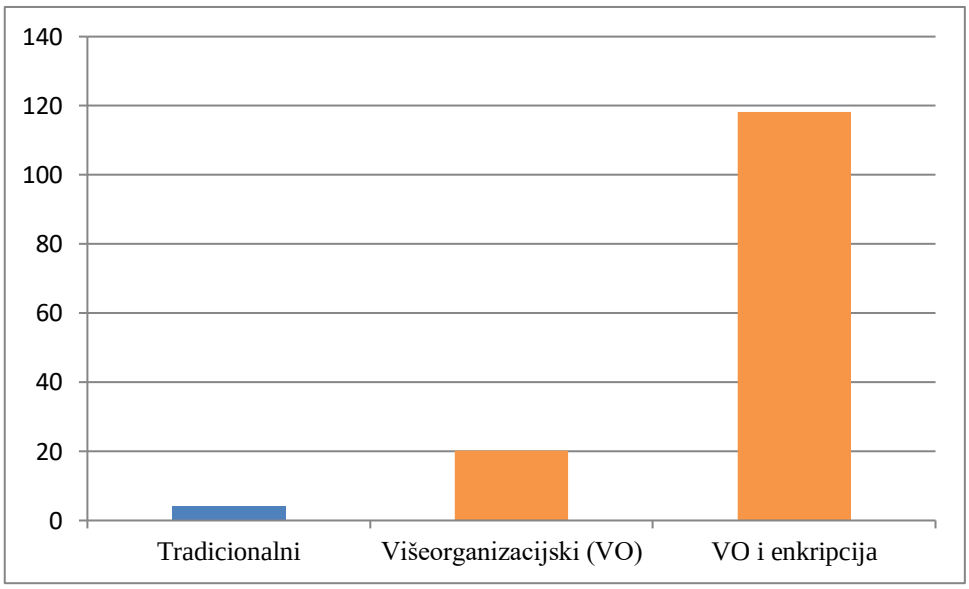
6.7.3. Prezentacija rezultata

Tabela 11 sadrži merenja, a njena vizuelizacija je predstavljena grafički (Slika 45).

Tabela 11 - Višeorganizacijski model sa i bez autorizacije

Enkripcija	Organizacija	min [ms]	avg [ms]	max [ms]
Da	1	106	118.19	1457
Ne	1	18	20.22	320

Narandžastom bojom je predstavljen višeorganizacijski model, dok je plavom prikazan tradicionalni model koji ne koristi autentifikaciju. U sredini je prikazano rešenje koje ne koristi autentifikaciju, s desne strane rešenje koje koristi autentifikaciju.



Slika 45 - Uticaj kriptografije na pad performansi

5.4.5. Analiza i interpretacija

Grafik ukazuje na pad performansi kada se u čitavom rešenju koristi autentifikacija i kontrola pristupa zasnovana na tvrdnjama. U tradicionalnom modelu, autentifikaciona logika je implementirana u okviru same aplikacije i provera identiteta se svodi samo na dodatni upit na bazu podataka. Stoga, sama autentifikacija u monolitnom rešenju ne bi dovela do značajnog pada performansi.

Pad performansi je značajan, šest puta više vremena je potrebno. Povećanje vremena se objašnjava višestrukom enkripcijom, preuzimanjem sertifikata putem *HTTP* protokola, povećanjem poruke usled kreiranja bezbednosnog žetona kao i proverom permisija korisnika. Sama autentifikaciona logika i njen pad performansi nije toliko od interesa s obzirom da se prijavljivanje korisnika na sistem radi retko. Ono što je zabrinjavajući deo rezultata jeste vreme koje se gubi zbog provere u permisija i dekripcija bezbednosnog tiketa u filtrirajućem servisu.

Analiza eksperimenta III i eksperimenta IV ukazuju na to da u sistemima kojima su performanse i SLA od vitalnog značaja, a takav je NDMS sistem, predloženo rešenje mora da se optimizuje. Optimizacije su moguće jer bi ovakvo rešenje, s namerom i postavljeno tako da se

svaki servis izvršava na posebnom čvoru *cloud* okruženja, predstavlja rešenje sa najgorim rezultatima mogućim.

Optimizacija bi obuhvatala distribuiranje žetona tvrdnji, pa i samog servisa administracije višeorganizacijskog modela (odeljak 5.10) na sve ciljne servise platforme. Predloženo rešenje platforme bi moglo fizički da se približi ciljnim servisima koji je koriste tako što bi sami servisi predstavljali niz procesa koji dele stanje, pri čemu bi održavali svoje replike i birali lider proces putem algoritma kao što je RAFT [133].

Na ovaj način, administracija servisa konfiguracije višeorganizacijskog modela bi se vršila putem *web* interfejsa koji bi kontaktirao liderski proces grupe servisa. Pošto bi se tvrdnje i servis konfiguracije višeorganizacijskog modela nalazili na istim virtuelnim mašinama kao i servisi koji ih koriste, njihova međusobna komunikacija bi bila neuporedivo brža. Ovaj vid optimizacije, praktično keširanje i približavanje podataka funkcijama koje ih koriste je moguć u scenarijima kao što je ovaj, a to je slučaj jako čestih čitanja, a retkih upisa (tvrdnje, kao i konfiguracija nisu često promenljive veličine).

Potreba za optimizacijom obuhvata prostor za budući rad, jer bi na osnovu implemeniranih optimizacija, sadašnja pretpostavka je da bi se višeorganizacijski model vrlo približio tradicionalnom po pitanju performansi. S druge strane, predložene optimizacije bi dovele do tesnije integracije platforme sa ciljnim okruženjem, i novim izazovom ako bi se sama platforma trebala ponuditi kao višeorganizacijska (što ne mora da bude zahtev).

Dakle, predloženo rešenje je u skladu sa hipotezom 3, izazvalo značajan pad performansi (5 puta po zahtevu), ali za potrebe sistema sa kritičnim odzivom kao što je NDMS moguće su optimizacije rešenja koje se baziraju na već naučenim praksama iz distribuiranog programiranja.

7. Vizija višeorganizacijske transformacije NDMS sistema

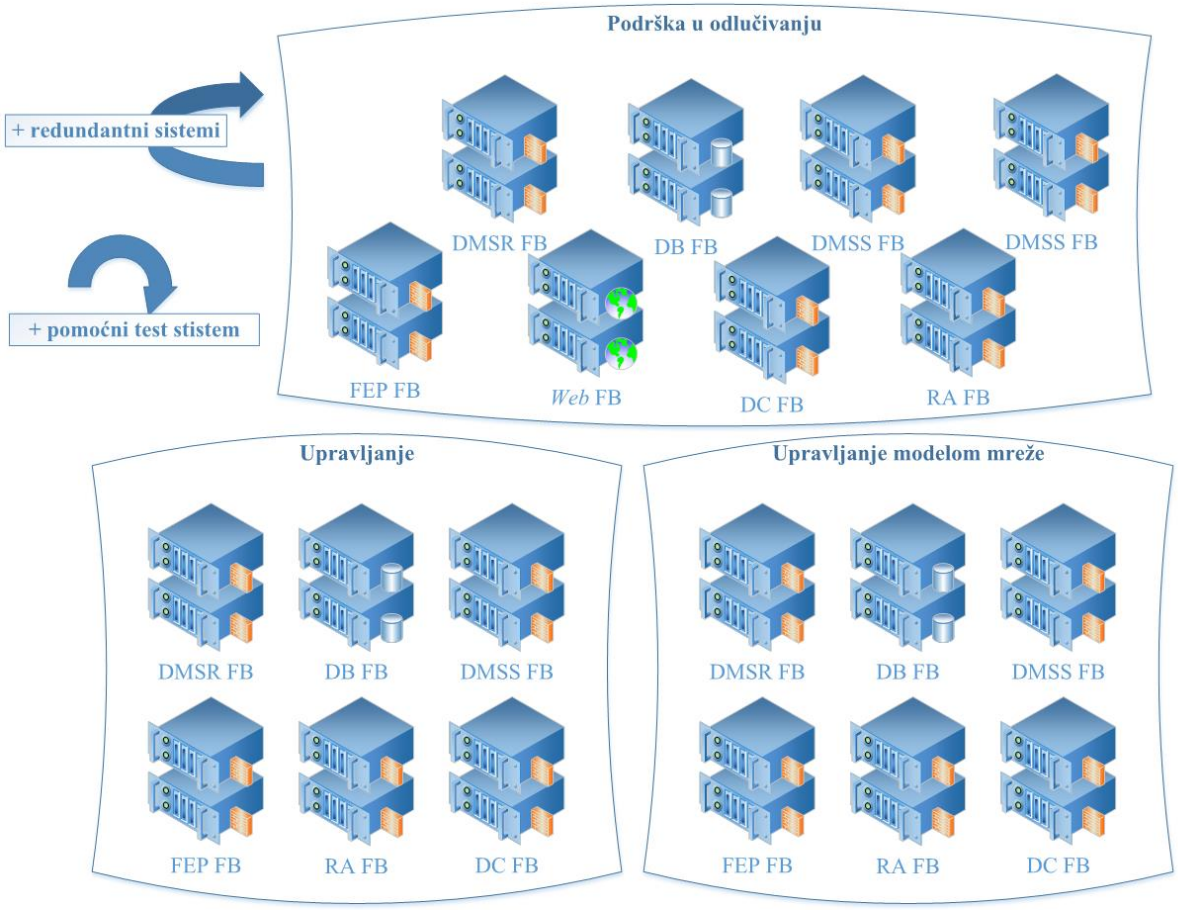
Da bi se predstavila migracija jednog sistema koji predstavlja srce pametnih elektroenergetskih mreža, biće predstavljeno trenutno NDMS rešenje u odeljku 7.1. NDMS je predstavljen na visokom nivou iz ugla IT arhitekture i funkcionalnih blokova od kojih je ona sačinjena.

U odeljku 7.2 je prikazano gde je najveći trošak resursa jednog NDMS sistema kako bi se videlo šta je ciljna grupa resursa koji se trebaju uštedeti prelaskom na višeorganizacijski model. Takođe, jako je važno navesti preduslove za uspešnu migraciju jer prelazak na višeorganizacijski model može uključivati više promena, a sve je prikazano u odeljku 7.3.

Odeljak 7.4 pojašnjava benefite višeorganizacijskog NDMS sistema, u skladu sa definisanim hipotezama a u cilju motivacije da se vizija migracije ispuni.

7.1. Postojeći NDMS

U ovom radu je korišćen primer NDMS sistema koji je detaljno opisan u radovima [37], i [83], dok je postojeći NDMS obrađen samo iz ugla IT arhitekture (Slika 46).



Slika 46 - Virtuelizovani NDMS [95]

Uz objašnjenje funkcionalnih blokova, dat je i pogled za dalju analizu potencijalne transformacije. U tradicionalnom slučaju, NDMS sistem se obično postavlja u dva centra podataka kako bi se obezbedio kontinuitet u poslovanju, dok se u jednom centru podataka obično konfiguriše u tri izolovana podsistema kako bi se postigla veća bezbednost čitavog sistema.

Svaki od NDMS podsistema je namenjen različitom poslovnom kontekstu. Iz tog razloga, u okviru poslovnih procesa elektrodistributivnih kompanija postoje različiti opisi poslova koji imaju prava pristupa nekom od podsistema. Zbog separacije dužnosti, gotovo je pravilo da opisi poslova koje koriste različite sisteme nikada ne bivaju pripisani jednoj osobi. Tri podsistema koji čine NDMS su:

1. upravljački – kritični sistem namenjen upravljanju fizičkom elektroenergetskom mrežom,
2. podsistem odlučivanja – namenjen podršci planiranja mreže, zaštite, analizama, izveštajima i za pristup sistemu korporativnim korisnicima,
3. podsistem za upravljanje modelom mreže – sistem za upravljanje promenama i sprovođenje promena nad modelom elektroenergetske mreže.

Osim tri podsistema koji čine NDMS, postoje još redundantni sistemi zbog kontinuiteta poslovanja kao i pomoćni sistemi koji predstavljaju odgovarajuće replike namenjene isprobavanju novih funkcionalnosti i generalnom testiranju. Svi sistemi su izgrađeni iz funkcionalnih blokova (FB), a u nastavku su navedeni svi funkcionalni blokovi koji se mogu pronaći u NDMS sistemu:

Web - predstavlja funkcionalni blok sa unapred postavljenim *web* serverom i klijentskim NDMS *web* aplikacijama. Ovo skalabilno rešenje, sastoji se iz tri korisničke aplikacije:

1. Osnovna klijentska aplikacija omogućuje korporativnim korisnicima pristup stanju mreže radi građenja svesti o situaciji. Postoji podrška za većinu aplikativnih paketa koji su podrazumevani u NDMS sistemu.
2. Podrška pozivnom centru omogućuje operaterima u pozivnom centru da obave prijem i obradu incidenata prijavljenih od strane korisnika kao što su na primer nestanak struje ili pad kvaliteta isporuke električne energije.
3. Podrška rukovanju inženjerskim posadama na terenu koja se sastoji od *web* servisa prilagođenog za tablet i mobilne uređaje.

Postojanje *web* klijenta je preduslov za uspešan SkS iz razloga što se nativne aplikacije ne mogu održavati bez pristupa klijentskim sistemima što otežava preuzimanje odgovornosti za SkS pružaoca usluga. U svakom slučaju, *web* aplikacija je primer koji je u ovom radu korišćen kao idealan kandidat za višeorganizacijsku arhitekturu tako da je *web* klijent prvi kandidat za predstojeću migraciju.

DMSR (eng. *DMS Real Time*) – predstavlja server distributivnog menadžment sistema za rad u realnom vremenu. Uobičajeno se nalazi u sva tri podsistema pri čemu postoji replikacija podataka između instanci kako bi stanje servisa bilo poravnato. DMSR predstavlja srž NDMS sistema i praktično sadrži sve funkcionalnosti tj. poslovnu logiku koja čini NDMS. Sačinjen je od više servisa projektovanih SOA principima, i većina su servisi sa stanjem. Problem visokih performansi i servisa sa stanjima dovodi do verovatne odluke da sami servisi DMSR servera budu posvećeni po organizaciji. Verovatno je odluka da se ovi servisi migriraju na pravu višeorganizacijsku arhitekturu poslednji korak u savremenom *cloud* baziranom NDMS SkS.

DMSS (eng. *DMS Simulation*) - po softverskoj strukturi je veoma sličan DMSR ali obuhvata skup aplikativnih paketa za rad u autonomnom režimu sa povremenim poravnanjem. Zbog svoje prirode, u koraku kada se DMSR treba migrirati na višeorganizacijsku arhitekturu, DMSS bi bio odličan kandidat zbog svoje sličnosti jer bi poslužio kao pilot projekat za DMSR.

FEP (eng. *Front End Processor*) sadrži protokole za komunikaciju sa poljem, odgovoran je za prikupljanje podataka u realnom vremenu od RTU i Intelligent Electronic Device (IED) uređaja, neposredno izdavanje upravljačkih komandi uređajima, konverziju analogno/digitalnih veličina u inženjerske merne jedinice, osnovne provere i prosleđivanje obrađenih veličina DMS funkcionalnom bloku. Postoje razni protokoli i konfiguracije komunikacije između FEP servisa i uređaja, od kojih se izdvajaju dva koncepta komunikacije: 1) Periodično prikupljanje podataka prema unapred definisanom vremenskom intervalu; 2) Samoinicijativno prosleđivanje značajnih promena od strane uređaja. Pored osnovnih analognih i digitalnih vrednosti, mogu se prikupljati i prateći podaci.

DC funkcionalni blok sadrži domenski kontroler koji je namenjen bezbednosti u smislu pružanja odgovore na zahteve za autentifikacijom. Migracijom na višeorganizacijske sisteme, DC postaje idealan kandidat za uštede, a posebno ukoliko se usvoji pristup bezbednosti bazirane na tvrdnjama kako je to opisano u odeljku 5.8.

FS (eng. *Field Simulator*) je odgovoran za simulaciju uređaja u polju i njihove pobude koje se koriste kao ulazi za DMS i SCADA funkcionalne blokove za svrhu testiranja (u inicijalnim fazama implementacije), treninga, obuke operatera, itd. FS komponenta se koristi povremeno od strane različitih organizacija, tako da sam prelazak na virtualizaciju ostvaruje dovoljne mogućnosti za uštedom. S druge strane, ako se uzme velik broj projekata u obzir, zbog troškova održavanja i upravljanja FS instancama, višeorganizacijsko svojstvo bi verovatno bila isplativa opcija i u ovom slučaju. Svakako, ovo je komponenta sa najmanjim prioritetom za migraciju (osim u slučaju da se napredne simulacije prodaju kao zaseban servis u SkS pristupu, ali analiza novih poslovnih modela je van okvira ove teze).

RA (eng. *Remote Access*) - predstavlja servis za stvaranje komunikacionog kanala između DMS aplikativnog servera i spoljašnjih korisnika koji se nalaze van lokalne mreže (ne pripadaju istom domenu). Njegova posebna prednost je što izlaže samo jedan komunikacioni port što smanjuje površinu potencijalnog napada i olakšava kontrolu bezbednosti sistema. RA je servis bez stanja ili u strogom slučaju uzeto, sa lakim stanjem, tako da bi trebalo da se migrira na najzreliji višeorganizacijski model.

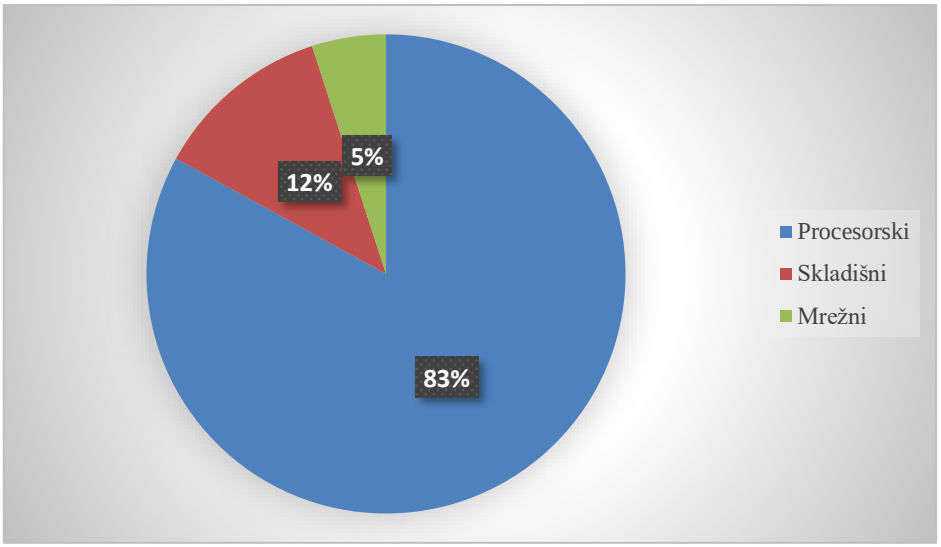
DB (eng. *Database Module*) - sastoji se od servera relacionih baza podataka odgovornih za čuvanje dinamičkih podataka sa polja, istorijskih promena mreže i arhiviranje podatatka različitih funkcionalnih blokova. DB je sigurno kandidat, već u prvom koraku za migraciju na višeorganizacijski model. Tehnike izolacije podataka u relacionim bazama su i bile predmet ovog rada.

7.2. Cloud ekonomija za NDMS

U slučaju virtuelizovanog NDMS sistema, *cloud* elasticitet je teško iskoristiv jer je ovo servis čiji funkcionalni blokovi su potrebni non-stop, sa zahtevima za dostupnost servisa od „četiri

devetke“ i više, tj. jednako ili više od 99.99%. U postojećoj arhitekturi, funkcionalni blokovi se skaliraju prema najvećim očekivanim potrebama, pa još sa dodatnom rezervom u pogledu potrebnih resursa. Vertikalno skaliranje gde bi se dinamički smanjivale količine RAM i CPU resursa nije opcija jer javna cloud okruženja i danas nemaju podršku za takav vid skaliranja – skaliranje je bazirano na horizontalnom skaliranju.

S obzrom na stalnu dostupnost virtuelnih mašina, i zahtevima za čuvanje podataka u periodu od pet godina, urađena je analiza troškova stavljanja postojećeg NDMS sistema na javna *cloud* okruženja. Slika 47 prikazuje odnos potrebnih resursa, gde je sasvim jasno da se na polju procesorskih resursa mogu ostvariti najveće uštede. Mrežni protok je veoma mali u odnosu na ukupnu cenu i u svakom slučaju je neophodan. Skladišni prostor se ne može smanjiti višeorganizacijskim svojstvom, dok se procesorski kapaciteti svakako uštedeti. Pozitivno je što je najveći deo troškova na raspolaganju za uštede. Bitno je napomenuti da bi višeorganizacijsko svojstvo nad bazom podataka rezultovalo uštedom nad procesorskim resursima jer obuhvataju virtuelne mašine sa *SQL* server instancama pri čemu su takve virtuelne mašine zbog potrebnih licenci skuplje. Sama količina podataka koja se čuva se ne smanjuje višeorganizacijskim svojstvom, a sam kapacitet i performanse skladišnog sloja se računaju pod skladišnim resursima.

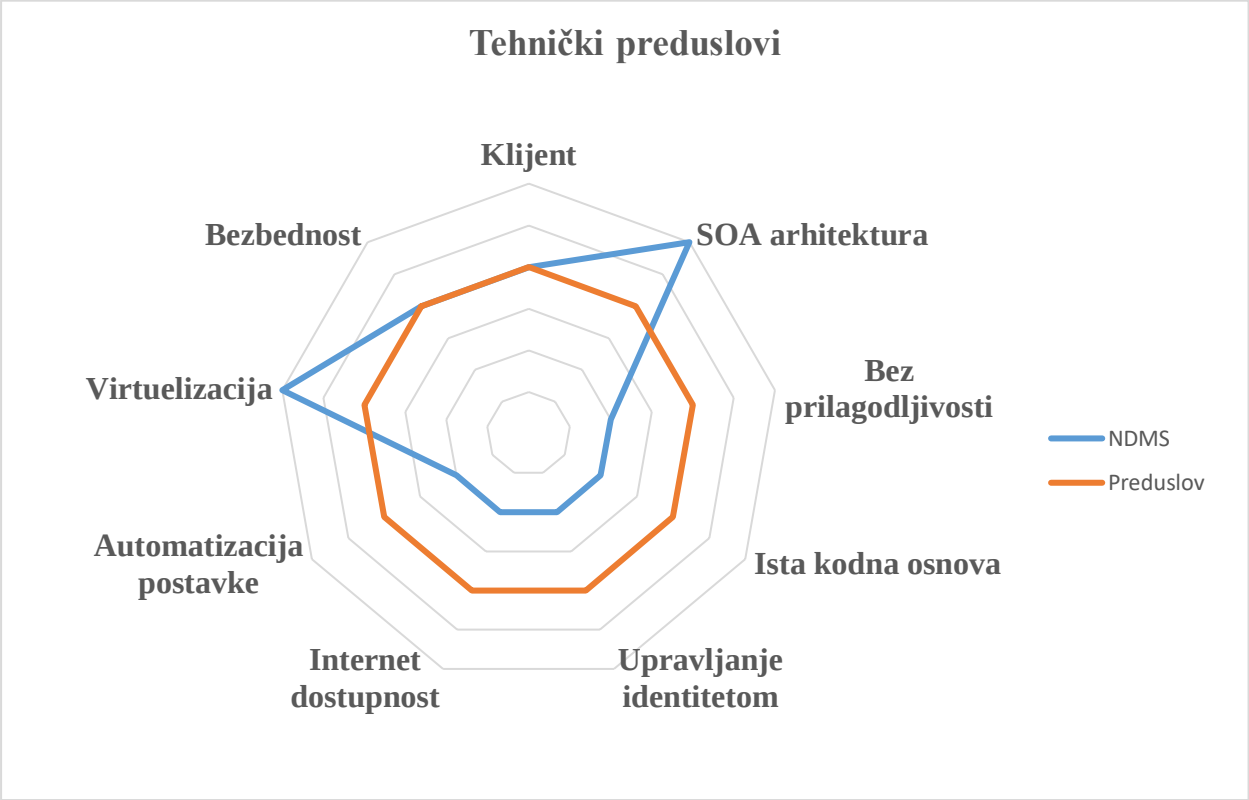


Slika 47 - Raspodela cloud resursa u NDMS-u

Analiza je sprovedena za tri referentna NDMS industrijska projekta – mali, srednji i veliki. Cene su računate za *Microsoft Azure* i *AWS* javna *cloud* okruženja. Iako razlike u odnosima nisu velike, u analizi je prikazana aritmetička sredina ovih šest slučajeva.

7.3. Tranformacija na višeorganizacijski SkS

Slika 48 prikazuje radar grafik spremnosti tekućeg NDMS rešenja na implementaciju višeorganizacijske osobine. Prevashodno koraku migracije rešenje, neophodno je sagledati trenutno rešenje. Za viziju višeorganizacijskog NDMS sistema, nakon analize potrebnih tehničkih preduslova, uzeće se za pretpostavku da su svi uslovi ispunjeni kako bi se u odeljku 7.4 prikazali benefiti primene višeorganizacijskog svojstva.



Slika 48 - Radar tehnološke transformacije

Tehnički preduslovi neophodni za migraciju su sledeći:

1) Klijent – odnosi na to da li je u pitanju debeli ili tanki klijent, idealno *web* klijent. Postojeći NDMS sistem ima *web* klijent, ali namenjen samo čitanju podataka, dok je deblji klijent nativna aplikacija što nije u redu ni za SkS model. Ipak, zbog postojanja *web* klijenta, ova kategorija je ocenjena kao prolazna, tj. da je preduslov ispunjen.
2) Bezbednost proizvoda – odnosi se na *cloud* bezbednost, gde je npr. kriptografija podataka tokom mirovanja neophodna jer je javno *cloud* okruženje po definiciji višeorganizacijsko. Dodatne bezbednosne kontrole neophodne zbog višeorganizacijskog pristupa bi sasvim sigurno usporilo bilo koje ciljno rešenje, a NDMS sistemu je brz odziv od ključnog značaja.
3) Virtuelizacija – postojeći NDMS je virtualizovan, kako je opisano u radu [37] gde se svi funkcionalni blokovi izvršavaju u virtuelizovanom okruženju, tako da je ovaj preduslov zadovoljen.
4) SOA Arhitektura – NDMS je servisno orijentisane arhitekture, tako da je i ovo svojstvo zadovoljeno.
5) Automatizacija postavke – podrška za automatsko kreiranje servisa, okruženja, njihovo automatsko spuštanje i dodavanje čvorova u logički klaster distribuiranih servisa.
6) Bez prilagodljivosti – NDMS je rešenje koje svoj poslovni model bazira na prilagođavanju rešenja zahtevima klijenata. Samo tržište još uvek nije jasno definisano, pa samim tim proizvod nema standardizovanu verziju rešenja. Ovo je jako teško za podršku iz ugla višeorganizacijskog sistema, ali srećne okolnosti su da bi to u trenutnom vremenu otpisalo DMSR FB za migraciju, dok se ostali funkcionalni blokovi i dalje mogu razmatrati za migraciju.

7) Internet dostupnost – zbog svoje kritičnosti koja može uticati na destabilizaciju nacionalno važne infrastrukture, tj. pametnih elektroenergetskih mreža, NDMS nije rešenje koje može biti dostupno putem interneta. To je u koliziji sa SkS modelom, gde je zamisao da svaki pojedinačni servis bude javno dostupan. Servisi međusobno bi se mogli uvezati virtuelnim mrežama, ali je problem šta raditi u slučaju komunikacije između klijentskih zgrada i *cloud* centara podataka. Uvođenje MPLS ili VPN putem interneta tehnologija komplikuju i poskupljuju rešenje, dodatno pogoršavajući performanse i latencije koja je već ugrožena zbog postojeće fizičke udaljenosti. Zaključak je da čak iako krajnje tačke tehnoloških servisa nisu javno dostupne, moraju istovremeno biti dostupne svim organizacijama koje ih koriste.
8) Ista kodna osnova – osobina se više odnosi na preduslov za postojanje ekonomski održivog SkS modela, ali i olakšanu migraciju na višeorganizacijski model. Na taj način, lakše bi bilo adresirati i implementirane prilagodljivosti iz razloga što bi svaka instanca, čak i da sadrži prilagodljivosti, imala većinu iste kodne osnove.
9) Upravljanje identitetom – u postojećem NDMS rešenju, bezbednost je zasnovana na DC funkcionalnom bloku što znači da su same *Windows Server* virtuelne mašine pripadnici jednog domena. U tom slučaju, bezbednosni model je vezan i za infrastrukturu, koja bi trebala da bude deljena među više organizacija. To se može prevazići prelaskom na bezbednost baziranu na tvrdnjama, jer bi se celokupan model prebacio samo na aplikativni nivo.

Za potrebe narednog odeljka, smatraće se da je radar NDMS sistema zadovoljio sve preduslove, kako bi bile izneta moguća vizija višeorganizacijskog NDMS sistema baziranog na predloženom rešenju.

7.4. Višeorganizacijski NDMS

Spremni NDMS za SkS je moguće migrirati na višeorganizacijski po funkcionalnim blokovima posebno jer platforma za migraciju omogućava putem servisa administracije da se određeni zahtevi rutiraju na određeni način. Na taj način, postojeća rešenja koja bi bila migrirana u *cloud* okruženje, mogu da se migriraju deo po deo.

Tabela 12 - Faze migracije na višeorganizacijski NDMS

FB/Faze	1.	2.	3.	4.
Web	IV			
DMSR			I	
DMSS			III	
DB	III/IV			
FEP		II		
DC	IV			
RA		IV		
FS				IV

Migracija bi se mogla odigrati u četiri faze koje bi obuhvatile promene nad različitim funkcionalnim blokovima (Tabela 12). U ćelijama tabele su prikazani modeli višeorganizacijske zrelosti opisane u odeljku 3.5.2.

S obzirom da tabela daje globalni prikaz migracije, u narednim odeljcima su obrađene sve četiri faze, sa naglaskom na benefite migracije, kao i način na koji bi se migracija mogla izvesti uz predloženo rešenje. Usled rezultata višeorganizacijske platforme prikazanih u odeklju 6.6.3, računica u analizi višeorganizacijskog NDMS sistema će se odnositi samo na šest organizacija u višeorganizacijskom klasteru. U pokaznim računicama će se uzeti pretpostavke da se gleda samo glavni sistem, bez redundantnog koji se nalazi na drugom centru podataka, i da postoje tri standardna podsistema NDMS sistema i pomoćni za testiranje.

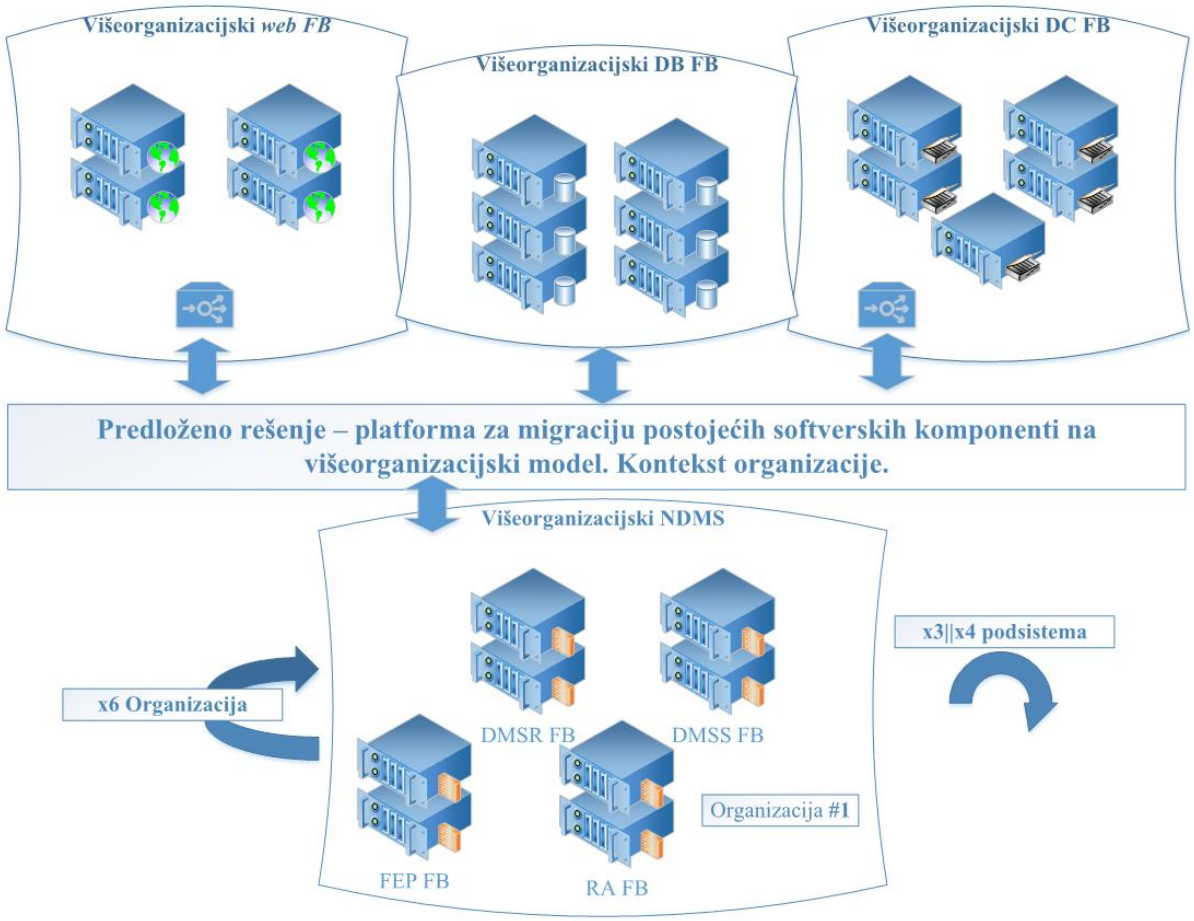
7.4.1. Faza I

U sklopu prve faze, bilo bi adekvatno migrirati najpre DC, a potom *web* FB, a na kraju i DB FB. Naravno, svaki FB bi se mogao migrirati i u paraleli, jer po prirodu nemaju zavisnost koja bi zahtevala sekvencijalnost.

DC FB je kandidat za migraciju na IV model zrelosti gde bi se u okviru implementiranih domen kontrolera na svakoj instanci virtuelne mašine nalazili podaci svih organizacija. To je moguće zahvaljujući RBAC šemama koje ni za čitave organizacije ne zahtevaju velik skladišni prostor iz ugla modernih sistema. Danas, zbog bezbednosti baziranoj na tradicionalnim sistemima, svaki podsistem upravljanja sadrži dve virtuelne mašine namenjene DC FB. Uzimajući u obzir pretpostavke, po jednoj organizaciji ukoliko se ne bi vršile višeorganizacijske modifikacije, potrebno je 8 virtuelnih mašina (dve po podsistemu uključujući i pomoćni test sistem) da bi se DC FB implementirao na NDMS sistemu. U slučaju šest organizacija, potrebno je 48 virtuelnih mašina, dok bi se u višeorganizacijskom slučaju mogle implementirati sa samo 3 virtuelne mašine. Da se dodaju u klaster još dve mašine radi boljeg odziva, višeorganizacijska migracija bi rezultovala u korišćenju 5 virtuelnih mašina umesto 48, samo za DC funkcionalni blok.

Bitno je napomenuti u DC FB slučaju, da bi migracija zahtevala drugačiji bezbednosni model, gde bi virtuelne mašine ostalih funkcionalnih blokova prestale biti deo određenih domena, nego bi pripadale domenu različitih klastera koji bi bili organizovani po višeorganizacijskim funkcionalnim blokovima. Na taj način, NDMS sistem bi se pretvorio u orkestraciju višeorganizacijskih servisa organizovanih po navedenim FB (Slika 49). Na slici je prikazan samo primer sistema za podršku odlučivanju, a može se uporediti sa slikom tradicionalnog NDMS sistema (Slika 46).

Što se tiče DC FB, servisi identiteta već postoje u svim komercijalno dostupnim *cloud* okruženjima. Mogla bi se izvršiti jednostavnija integracija, ali problem može nastati u tome što su svi ti servisi javno dostupni putem interneta. Ukoliko postoje ograničenja da bude pristup omogućen javnim servisima zbog regulativa u elektroenergetskm mrežama ili eksplicitnim zahtevima klijenata, morao bi se implementirati višeorganizacijski DC koji bi bio dostupan putem intraneta. Višeorganizacijski DC FB bi bio jednostavan za implementaciju uz predloženo rešenje ovog doktorata, a pitanje je vremena kada će se regulative i zahtevi klijenata prilagoditi potpuno drugačijoj prirodi bezbednosti *cloud* okruženja u odnosu na tradicionalna rešenja.



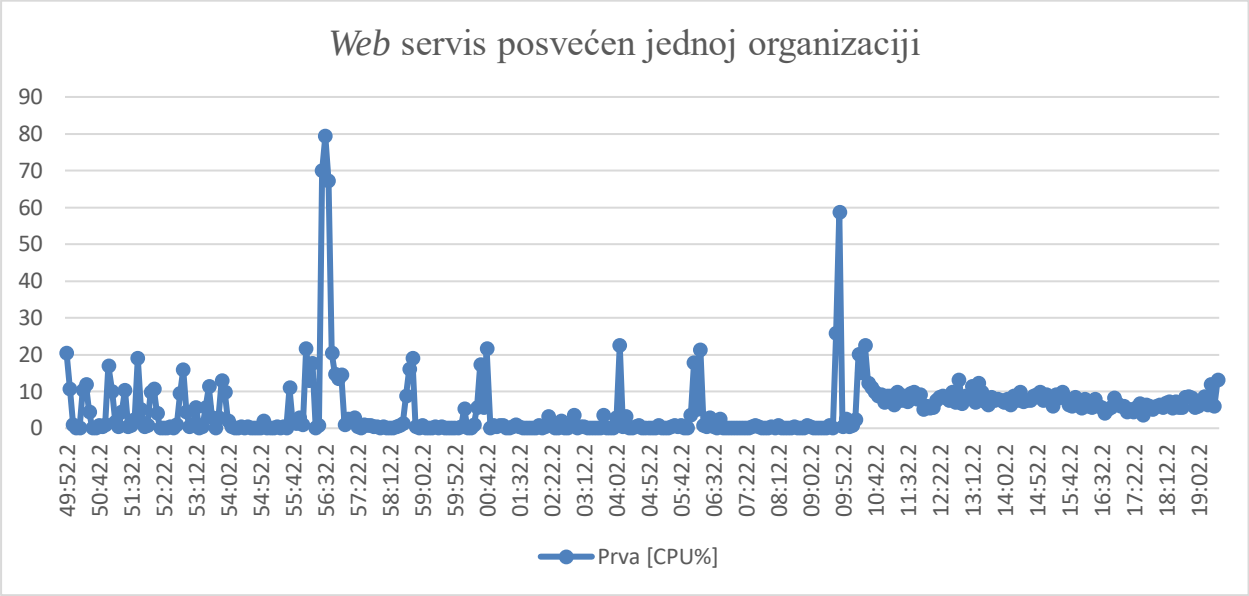
Slika 49 – NDMS u višeorganizacijskom kontekstu

Web FB migracija je i bila predmet ovog rešenja, opisanog u odeljcima od 5.6 do 5.11. U slučaju rešenja posvećenog jednoj organizaciji, jedan *web* server mora da se postavi u kombinaciji od dva *web* servera zbog visoke dostupnosti, što znači da za 6 organizacija postoji 12 *web* servera (po dve u podsistemu za podršku odlučivanja ne računajući pomoćni sistem).

U slučaju višeorganizacijskog *web* rešenja modela zrelosti IV, potrebno je najviše 7 virtuelnih mašina (6 organizacija sa *web* serverom i jedna mašina redundantne rezerve), ali i manje. Razlog zašto je potrebno manje mašina je zbog balansiranosti skaliranja rešenja u višeorganizacijskim arhitekturama.

Slika 50 pokazuje opterećenje *web* FB industrijskog NDMS sistema na realnom industrijskom projektu pri prosečnoj upotrebi. Odbirci procentualne opterećenosti CPU resursa su uzimani svakih pet sekundi, a na graficima je prikazan interval od pola sata. Ekstrapoliranjem rezultatata prostim sumiranjem u slučaju tri i šest organizacija sa promenutim vremenskim odrednicama upotrebe, benefiti višeorganizacijskog svojstva postaju vidljivi (Slika 51 i Slika 52).

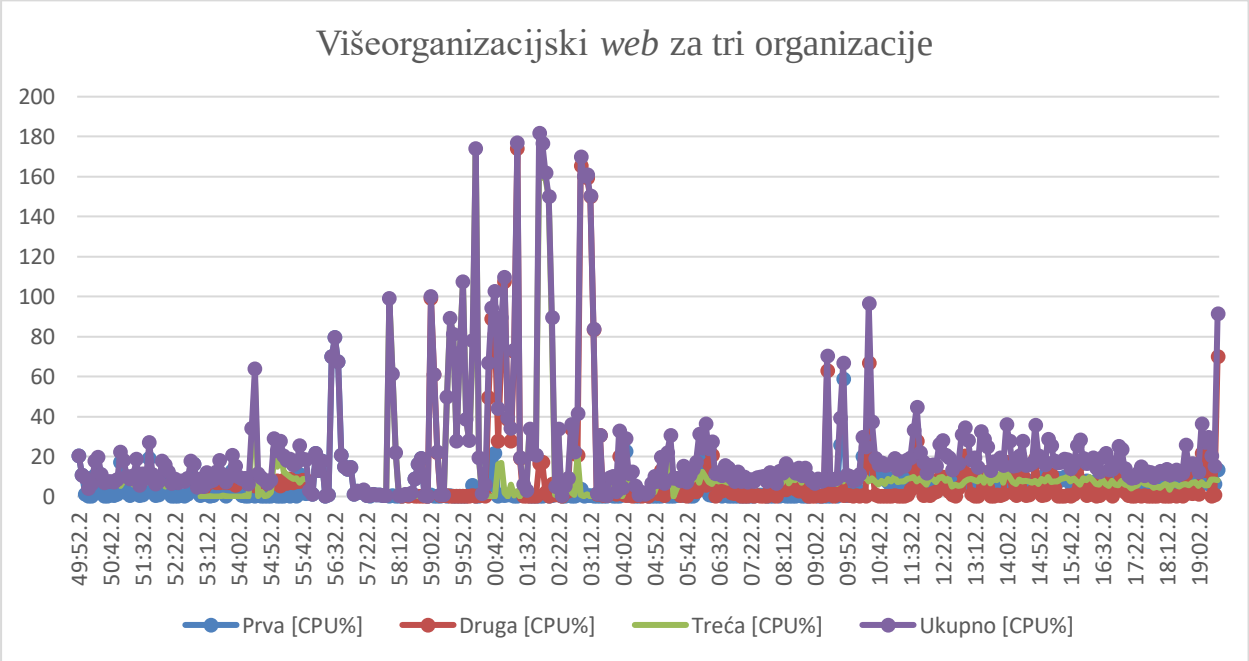
Kriva opterećenja se stabilizuje i omogućava prediktabilnije opterećenje, ali i veće iskorišćenje postojećih resursa. Računanjem ušteda, postaje evidentno da bi i četiri virtuelne mašine umesto dvanaest dovele do istog kvaliteta usluga.



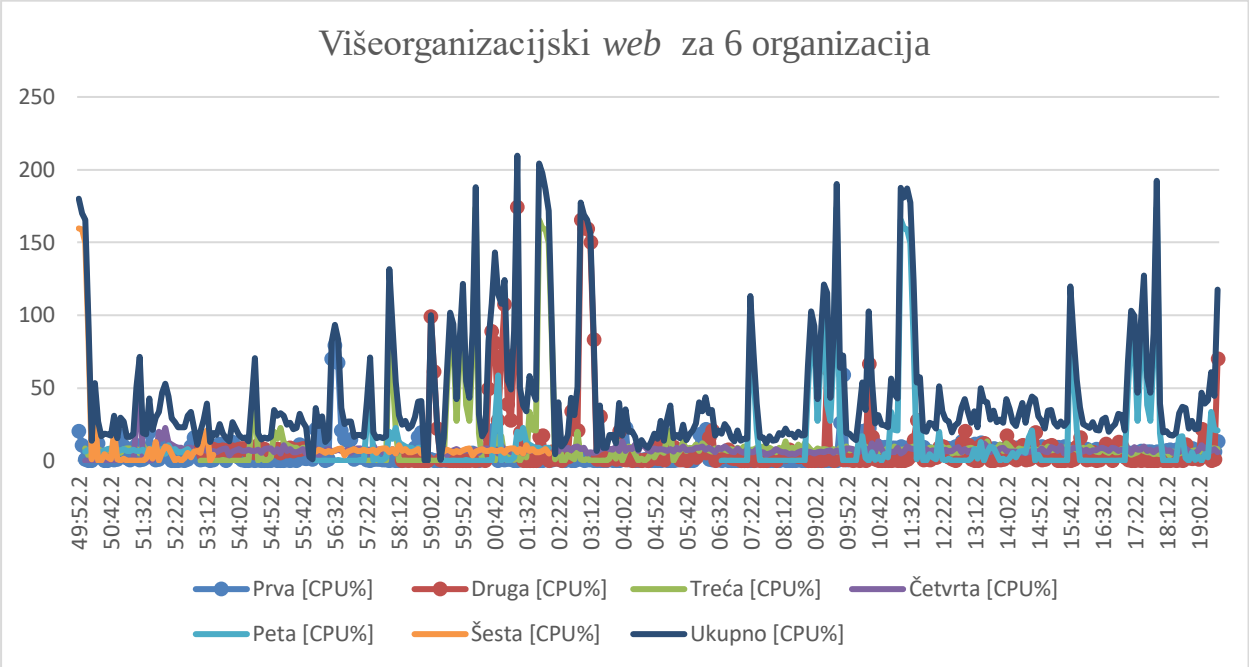
Slika 50 - *Web* FB opterećenje

Sagledavanjem merenja industrijski testiranog *web* FB, vidi se da u slučaju jedne organizacije (Slika 50), maksimalno opterećenje u merenom vremenu ide do 80%, dok se dodatnih 20% čuvaju kao rezerva pri skaliranju servera.

U slučaju tri organizacije (Slika 51), maksimalno opterećenje se penje do 180% što znači da bi dve virtuelne mašine bile već dovoljne, dok se u slučaju šest organizacija (Slika 52) maksimalno opterećenje penje na tik iznad 200%, što znači da bi verovatno i tu samo dve virtuelne mašine bile dovoljne. Naravno, rezerva treba i mora da postoji, tako da bi tri virtuelne mašine i više nego zadovoljile potrebe.



Slika 51 - Višeorganizacijski *web* za tri organizacije



Slika 52 - Višeorganizacijski *web* za šest organizacija

DB FB je složeniji za analizu, i iz tog razloga je u tabeli naveden kao III ili IV nivo zrelosti. Naime, ukoliko se relacione baze podataka migriraju jedan na jedan uz izolaciju podataka opisanu u odeljku 3.6.2, reč je III modelu zrelosti jer iako klaster SQL server mašina može da raste, raste broj omogućenih replika za čitanje. Dakle, može se izvršiti izolacija podataka, ali i dalje to ne znači da je podržan IV model zrelosti.

U principu, III model zrelosti verovantno ne može podržati šest organizacija, čak iako su dovoljno male, ali bi se moglo ići na vertikalnu desintregraciju u smislu da se različiti tipovi podataka čuvaju u različitim višeorganizacijskim bazama podataka modela zrelosti III. Sve u svemu, za baze podataka ozbiljnih zahteva, u slučaju NDMS sistema baziranih na podacima koji se čuvaju u vremenskim serijama, migracija na DB FB servis bi podrazumevala promenu paradigme, tj. korišćenje sistema za velike količine podataka (engl. big data) kako je to opisano u srodnim istraživačkim radovima [41] i [42]. Na oba načina, broj virtuelnih mašina bi se mogao znatno smanjiti.

U tradicionalnom slučaju, za šest organizacija bi bilo podrebno barem 30 virtuelnih mašina za servere relacionih baza podataka (upravljački i podsistem podrške odlučivanju po par mašina i jedna mašina u podsistemu za upravljanje modelom mreže. Od tih 30 virtuelnih mašina, 12 je tu samo iz razloga podrške visoke dostupnosti. U višeorganizacijskom slučaju, pogotovo u slučaju sistema za velike podatke, 6 virtuelnih mašina bi bilo dovoljno za podršku ovakvih zahteva.

Na kraju faze I, može se uporediti potreba samo za brojem virtuelnih mašina za DC, *web* i DB FB. U tradicionalnom slučaju, za šest organizacija je potrebno 48 + 12 + 30 = 90 VM, dok je u slučaju višeorganizacijskog rešenja potrebno 5 + 4 + 6 = 15 VM.

Na samo šest organizacija, i tri FB u prvoj fazi migracije, moguće je pružiti stabilnije servise sa 15 VM umesto 90 VM. Stabilnost se oslikava u IV modelu zrelosti, što je predstavljeno u narednom odeljku.

7.4.2. Visoka dostupnost

Trenutno objavljeni SLA u slučaju *Microsoft Azure* su 99.95% u slučaju posedovanja dve ili više mašina, dok je kod *AWS* SLA 99.99% na mesečnom nivou. Uzmimo radi pokazivanja visoke dostupnosti slučaj da će mašine biti dostupne mesečno 99.95% vremena, nevezano za komercijalne penale i uslove.

Pre analize dostupnosti, bitno je predstaviti računicu dostupnosti za paralelne čvorove. Dva čvora se smatraju paralelnim ukoliko se smatra da je njihova kombinacija otkazala onda kada su oba čvora otkazala. Kombinovani sistem je dostupan ako je jedan od paralelnih čvorova dostupan. Odavde proizilazi formula dostupnosti:

$D = 1 - \prod(1 - D_{VMi})^2$, *gde je:*

Dvmi – i-ta virtuelna mašina koja je učesnik kao paralelni čvor

Formula implicira da povećenjem paralelnih komponenti, dostupnost sistema postaje veća. Iako je već intuitivno jasno da veći broj paralelnih instanci u višeorganizacijskom modelu dovodi do veće sistemske dostupnosti, tabelarni prikaz će za svaki od funkcionalnih blokova jasno oslikati prednosti višeorganizacijskog modela (Tabela 13).

Tabela 13 - Visoka dostupnost višeorganizacijskih servisa

Komponenta	Dostupnost [%]	Period otkaza [min/godišnje]
VM	98%	10512
2 x VM (tradicionalni FB)	99.96%	210.24
3 x VM	99.9992%	4.2048
4 x VM (*Web* FB)	99.999984%	0.084096
5 x VM (DC FB)	99.99999968%	0.00168192
6 x VM (DB FB)	99.9999999936%	3.34E-05

U slučaju DC FB, tradicionalno bismo imali po dve DC mašine u tri podsistema. Svaki od podsistema tada godišnje ima ispade od 210 minuta, ako uzmemo pretpostavku da će mašine u *cloud* okruženju biti 98% dostupne. U ukupnoj dostupnosti sistema, gde je DC FB jedan od serijski vezanih zavisnosti, ovakva dostupnost je neprihvatljiva. Tada bismo morali ići na tri mašine po organizaciji, što bi čitav odnos uštede povećalo u korist višeorganizacijskog modela. U slučaju više od samo šest organizacija, jasno je da svi FB koji su u višeorganizacijskoj izvedbi, osim uštede donose i veću dostupnost, koja ide do granica da su mašine u sumi nedostupne manje od jednog sekunda godišnje.

Predlog rešenja omogućava paralelizam u računanju dostupnosti usled zrelosti višeorganizacijskog modela IV, što je u skladu sa hipotezom 3, definisanom u odeljku 3.7.3. U narednom odeljku je prikazan drugi deo benefita višeorganizacijskog modela koji je u skladu sa hipotezom 3.

7.4.3. Faza II

U sklopu faze II, predlog je migracija FEP i RA FB. FEP FB prikazuje još jednu vrlo važnu osobinu višeorganizacijskog modela koja dovodi do povećane dostupnosti, barem iz komercijalnog ugla SkS sistema.

Pitanje je da li će se tokom migracije na cloud okruženje FEP FB migrirati iz razloga zadržavanje kontrole nad poljem van *cloud* sistema. Opet, organizacije mogu imati fizičke SCADA master uređaje koji pružaju dovoljan nivo kontrole, gde su FEP FB u stvari više interfejsi ka NDMS sistemu i njegove ulazne tačke, nego mesto kontrole za korisnike NDMS sistema. U tom slučaju, vredi analizirati potencijalnu migraciju FEP FB na višeorganizacijski servis.

Iako bismo mogli reći da je FEP FB suštinski servis bez stanja, on nije pogodan za IV model višeorganizacijske zrelosti zbog uspostavljanja TCP konekcija između RTU i IED uređaja i FEP servisa. Naime, primarni zahtev FEP servisa jesu performanse i brzo procesiranje podataka. Trenutno zastupljeni protokoli su DNP3 i IEEE60870 standard od kog je u SCADA sistemima prisutan IEC 60870-5-101 protokol, nije za očekivati da će svaki put zahtev biti rutiran na drugu mašinu zbog TCP konekcija i prirode tih protokola. Protokoli su detaljno opisani uz svoje impakte u zavisnosti od upotrebe u radovima [134] i [135]. Iz tih razloga, naznačeno je da je očekivano od FEP FB da podrži model zrelosti II, ali na sledeći način.

Pretpostavimo da postoji 10 organizacija koje dele FEP FB. Radi jednostavnijeg pojašnjenja, može se pretpostaviti da datih 10 organizacija sadrže sličan broj krajnjih signala koji će biti procesuirani od strane FEP FB. U tradicionalnom režimu, svaka od organizacija bi posedovala dve virtuelne mašine za FEP FB zbog visoke dostupnosti, ali ne i više mašina zbog optimizacije troška. Ukoliko glavna instanca FEP FB otkaže, replika postaje glavna, ali je taj proces za FEP FB dugotrajan jer ponovo treba uspostaviti veliki broj TCP konekcija do krajnjih uređaja, i protokoli treba da uspostave komunikaciju s njima. Krajnji korisnici NDMS sistema mogu primetiti problem u radu koji bi se oslikao kao kratkotrajni pad FEP FB.

Zbog problematičnog ponovnog uspostavljanja konekcija, u višeorganizacijskoj izvedbi, ne može se upotrebiti IV model zrelosti. Dakle, na 10 organizacija, bilo bi takođe neophodno da postoje parovi mašina zbog redundanse što bi dovelo do istih ukupnih 20 virtuelnih mašina. Međutim, način korišćenja višeorganizacijskog FEP FB, može dovesti do bolje dostupnosti servisa tako što će se jedna organizacija rasporediti na svih 20 raspoloživih čvorova. Na taj način, 10 organizacija bi po desetinu svojih konekcija vezala za jedan čvor. U tom slučaju, otkazom bilo koje od virtuelnih mašina, svaka organizacija bi iskusila problem na desitini svoje mreže, a ne na celokupnoj. Na ovaj način, omogućen višeorganizacijskim modelom, mogli bi se ponuditi SLA za FEP servis finije granulacije – npr. tvrdnja da će dostupnost nadzora mreže biti 100%, pri čemu se računa da je mreža dostupna ako je više od 50% signala uspešno pod nadzorom. Ovakav SLA se ne može ponuditi u izvedbi koja nije višeorganizacijska, uzimajući u obzir približno slična rešenja po količini potrebnih resursa. U smislu dostupnosti krajnjeg servisa koji se nudi organizacijama, rešenje koje pruža višeorganizacijsko svojstvo, u skladu sa hipotezom 3 (odeljak 3.7.3), pruža veću dostupnost ponuđenih servisa.

RA servis zbog svoje jednostavnosti lako može da implementira višeorganizacijsko svojstvo, oslanjajući se na rešenje ovog rada. Svi benefiti visoke dostupnosti i optimalnog iskorišćenja resursa su direktno primenljivi i na RA servis.

7.4.4. Faze III i IV

Faze III i IV se odnose na FB koji su van opsega ove disertacije. DMSR i DMSS FB su vrlo specifične arhitekture bez čijeg opisa se ne može smislenije analizirati potencijalna primena višeorganizacijskog svojstva. Opisi interne implementacije nisu dati u relevantnim istraživanjima na koja se ovo čitavo poglavlje oslanja.

Dovoljno je konstatovati da su DMSR i DMSS suštinski isti FB koji rade u drugačijim konekstima, i da su servisi sa stanjem gde osobina horizontalne skalabilnosti nije zadovoljena. S obzirom da su uštede višestruke implementacijom faze I i faze II, pitanje je da li je ekonomski isplativa i neophodna transformacija DMSR FB na višeorganizacijski. Dodatno, faze I i faze II se bave manje kritičnim servisima iz ugla performansi, tako da industrijskom implementacijom ovih faza bi bilo sigurno novih naučenih lekcija koje bi dale odgovor na izvodljivost migracije ovih FB. Stoga, prikazano je da (Tabela 12) DMSS može biti razmatran za III model zrelosti jer u ovom FB su performanse mnogo manje važne nego u DMSR koji je predložen za I model zrelosti.

FS FB posledično zavisi od DMSR i DMSS, ali može se kontatovati da bi imalo smisla razmatrati njegovu migraciju u slučaju više od 20 organizacija što FS FB blok čini manje interesantnim za analizu pre samog početka migracije.

7.4.5. Benefiti transformacije

Visoka dostupnost i ekonomičnost rešenja.

Povećanjem broja organizacija, visoka dostupnost ide na virtuelnih 100% dostupnosti, tj. FB postaju nedostupni manje od jedne sekunde godišnje, kako je pokazano u odeljku 7.4.2, a u skladu sa hipotezom 2 (odeljak 3.7.3).

Hipoteza 1 adresira tvrdnju ekonomičnosti rešenja gde je ideja da se prikaže egzaktno u procentima kolike bi bile uštede primenom višeorganizacijskog modela. Uzimajući rezultate u mogućnosti uštede virtuelnih mašina u odeljku 7.4.1 i ukrštanje tih rezultata sa početnim stanjem *cloud* ekonomije NDMS sistema prikazane u odeljku 7.2, dobijeni rezultati su da se višeorganizacijskim modelom može uštedeti **32%** od resursa NDMS sistema već na 6 organizacija.

8. Zaključak

Iako je SkS poslovni model, vrlo često se vezuje za pojavu sve zastupljenijih *cloud* okruženja koja svojom prirodom utiču na pojavu SkS modela. Takođe utiču na model razvijanja servisa. Osim dugo prisutne SOA, javljaju se i mikroservisi, pa i napredniji aplikativni modeli kao što je model učesnika.

U osnovi uspeha *cloud* sistema jeste ušteda resursa. Skalabilnost i elasticitet omogućavaju SkS modelu mogućnost odgovora na pojavu velikog broja klijentskih organizacija i samih klijenata. Deljenje razvijanih servisa je u duhu *cloud* okruženja koji je sam po sebi deljen, a podrazumeva se da je dizajn takvih servisa višeorganizacijski.

Istraživanje je bilo bazirano na izučavanju višeorganizacijskog modela prateći akademiju, ali i industriju kroz zahteve koji se postavljaju pred moderne *cloud* bazirane servise kao i ponudama vodećih *cloud* usluga. Kao ishod, predloženo je rešenje najveće prepreke za usvajanje višeorganizacijskog modela, a to je bezbednost s naglaskom na čuvanje privatnosti ispravnom izolacijom podataka. Osim mogućnosti razvoja višeorganizacijskih servisa, akcenat je bio na sagledavanju transformacije postojećih softverskih rešenja na višeorganizacijska.

Zbog sve prisutnijih *cloud* okruženja i aplikativnih modela koje su posledica inovacija *cloud* okruženja, u predloženom rešenju su eksperimentalno evaluirana i rešenja bazirana na modernim PkS v2 i modelu učesnika. Jednom migrirani softver u *cloud* okruženja definitivno može i treba da koristi prednosti novih paradigmi.

Pristup koji preporučuje ovaj rad je servisno orijentisana softverska podrška koja implementira pouzdani bezbednosni model pri pružanju konteksta organizacije. Na ovaj način, bezbednosni model je izmešten iz same aplikacije i privatnost podataka različitih organizacija postaje odgovornost softverske podrške. Svaki od servisa se može posebno optimizovati i mogući načini su opisani pri analizi i interpretaciji rezultata eksperimenata.

Kako je navedeno, istraživanje ne nastoji da pruži analizu celokupnog bezbednosnog aspekta novih rešenja. Dodatno, ne bavi se vrlo važnim aspektom prilagodljivosti servisa jer taj aspekt ne utiče direktno na problem konteksta organizacije a s druge strane je određen zahtevima, regulativama kao i domenom problema servisa.

Krajnje rešenje je testirano u okviru privatnog *cloud* okruženja. Na osnovu rezultata, može se zaključiti da je istraživanje u skladu sa hipotezama. Višeorganizacijski pristup narušava performanse, što u trenutnoj izvedbi predloženog rešenja uvodi kašnjenja po zahtevu reda veličine 15 – 20 ms. S druge strane, uštede od 32% na nivou resursa *cloud* okruženja za NDMS sistem, kao i za tri reda veća visoka dostupnost čine rešenje interesantnim za dalji razvoj i optimizaciju. Predlozi nekih od optimizacija su opisani u samom radu, a ubrzanje svakog od servisa pojedinačno bi sigurno dovelo do unapređenja konačnog rešenja.

Dalji rad obuhvata optimizaciju rešenja i omogućenje industrijski spremne platforme, dok sa akademske strane je potrebno istražiti bezbednosne apsekte pobrojane u odeljku ograničenja.

Ako se uzme u obzir da u okviru *cloud* okruženja uštede u korišćenim resursima dovode do direktne dobiti, višeorganizacijsko svojstvo, kao što i sve vodeće metodologije preporučuju, predstavlja neizostavnu komponentu za bilo koji SkS na sve oštrijem globalnom tržištu.

Literatura

[1] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*, vol. 9783642290. Springer, 2012.

[2] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin, "A model for technology transfer in practice," *IEEE Softw.*, vol. 23, no. 6, pp. 88–95, 2006.

[3] "Predictions 2018: Cloud Computing Accelerates Enterprise Transformation Everywhere." [Online]. Available: https://www.forrester.com/report/Predictions+2018+Cloud+Computing+Accelerates+Enterprise+Transformation+Everywhere/-/E-RES139611. [Accessed: 12-Nov-2017].

[4] W. Y. Chang, H. Abu-Amara, and J. F. Sanford, *Transforming enterprise cloud services*. Springer, 2010.

[5] M. Cancila, D. Toombs, A. D. Waite, and K. Elias, "Gartner: 2017 Planning Guide for Cloud Computing," 2016.

[6] B. James, *The enterprise cloud*. O'Reilly, 2015.

[7] B. Briggs and E. Kassner, *Enterprise Cloud Strategy*. Redmond: Microsoft Press, 2016.

[8] A. Osterwalder and Y. Pigneur, *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. John Wiley & Sons, 2010.

[9] A. Giessmann, P. Kyas, P. Tyrväinen, and K. Stanoevska, "Towards a better Understanding of the Dynamics of Platform as a Service Business Models," *Hawaii Int. Conf. Syst. Sci.*, pp. 965–974, 2014.

[10] M. Schief and P. Buxmann, "Business Models in the Software Industry," *45th Hawaii Int. Conf. Syst. Sci.*, pp. 3328–3337, 2012.

[11] J. Petrikina, P. Drews, I. Schirmer, and K. Zimmermann, "Integrating business models and enterprise architecture," *Proc. - IEEE Int. Enterp. Distrib. Object Comput. Work. EDOCW*, pp. 47–56, 2014.

[12] T. Li, T. He, and Y. Zhang, "Service-centric Business Model in Cloud Environment," *Serv. Comput. (SCC), 2015 IEEE*, pp. 530–537, 2015.

[13] M. Kralj, P. Kazmi, and A. Ruth, *An Overview of Cloud Adoption Framework*. Amazon Web Services, 2015.

[14] A. Ojala, "Software-as-a-service revenue models," *IT Prof.*, vol. 15, no. 3, pp. 54–59, 2013.

[15] I. Churakova and R. Mikhramova, "Software as a Service: Study and Analysis of SaaS Business Model and Innovation Ecosystems," FACULTEIT ECONOMIE EN BEDRIJFSKUNDE, 2010.

[16] S. Bibi, D. Katsaros, and P. Bozanis, "Business application acquisition: On-premise or SaaS-based solutions?," *IEEE Softw.*, vol. 29, no. 3, pp. 86–93, 2012.

[17] J. Alonso, L. Orue-Echevarria, M. Escalante, J. Gorroñogoitia, and D. Presenza, "Cloud modernization assessment framework: Analyzing the impact of a potential migration to Cloud," in *c2013 IEEE 7th International Symposium on the Maintenance and Evolution of*

Service-Oriented and Cloud-Based Systems, MESOCA 2013, 2013, pp. 64–73.

[18] L. O.-E. Arrieta, "From Software as a good to software as a service: Preparing the evolution of software products into the cloud," *Int. Work. Maint. Evol. Serv. Cloud-Based Syst. (MESOCA - ICSM)*, pp. 58–59, 2012.

[19] A. Menychtas *et al.*, "ARTIST methodology and framework: A novel approach for the migration of legacy software on the cloud," in *Proceedings - 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2013*, 2014, pp. 424–431.

[20] H. Bruneliere, J. Cabot, J. L. C. Izquierdo, L. O. E. Arrieta, O. Strauss, and M. Wimmer, "Software Modernization Revisited: Challenges and Prospects," *Computer (Long. Beach. Calif).*, vol. 48, no. 8, pp. 76–80, 2015.

[21] J. Alonso, L. Orue-Echevarria, and M. Escalante, "Cloud compliant applications: A reference framework to assess the maturity of software applications with respect to cloud," in *2015 IEEE 9th International Symposium on the Maintenance and Evolution of Service-Oriented Systems and Cloud-Based Environments, MESOCA 2015 - Proceedings*, 2015, pp. 41–45.

[22] H. Trivedi, "Cloud Adoption Model for Governments and Large Enterprises," *Mit*, no. May, p. 82, 2013.

[23] H. Gangwar, H. Date, and R. Ramaswamy, "Developing a Cloud-Computing Adoption Framework," *Glob. Bus. Rev.*, vol. 16, no. 4, pp. 632–651, 2015.

[24] S. Frey and W. Hasselbring, "An extensible architecture for detecting violations of a cloud environment's constraints during legacy software system migration," in *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, 2011, pp. 269–278.

[25] B. W. B. Wu *et al.*, "Legacy systems migration-a method and its tool-kit framework," *Proc. Jt. 4th Int. Comput. Sci. Conf. 4th Asia Pacific Softw. Eng. Conf.*, no. December, pp. 312–320, 1997.

[26] C. Pahl and H. Xiong, "Migration to PaaS clouds - Migration process and architectural concerns," in *c2013 IEEE 7th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems, MESOCA 2013*, 2013, pp. 86–91.

[27] M. Almorsy, J. Grundy, and A. S. Ibrahim, "TOSSMA: A tenant-oriented SaaS security management architecture," in *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*, 2012, pp. 981–988.

[28] J. Song, F. Han, Z. Yan, G. Liu, and Z. Zhu, "A SaaSify tool for converting traditional web-based applications to SaaS application," in *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, 2011, pp. 396–403.

[29] Z. H. Wang, C. J. Guo, B. Gao, W. Sun, Z. Zhang, and W. H. An, "A study and performance evaluation of the multi-tenant data tier design patterns for service oriented computing," in *IEEE International Conference on e-Business Engineering, ICEBE'08 - Workshops: AiR'08, EM2I'08, SOAIC'08, SOKM'08, BIMA'08, DKEEE'08*, 2008, pp. 94–101.

[30] X. Zhou, D. Zhan, L. Nie, F. Meng, and X. Xu, "Suitable database development framework for business component migration in SaaS multi-tenant model," *Proc. Int.*

Conf. Serv. Sci. ICSS, pp. 90–95, 2013.

[31] P. Aghera, S. Chaudhary, and V. Kumar, "An approach to build multi-tenant SaaS application with monitoring and SLA," in *Proceedings - International Conference on Communication Systems and Network Technologies, CSNT 2012*, 2012, pp. 658–661.

[32] M. Almorsy, J. Grundy, and A. S. Ibrahim, "SMURF: Supporting multi-tenancy using re-aspects framework," in *Proceedings - 2012 IEEE 17th International Conference on Engineering of Complex Computer Systems, ICECCS 2012*, 2012, pp. 361–370.

[33] J. Zheng and W. Du, "Toward easy migration of client-server applications to the cloud," *ICSOFT-EA 2014 - Proc. 9th Int. Conf. Softw. Eng. Appl.*, pp. 101–108, 2014.

[34] N. Dalčeković, S. Vukmirović, S. Stoja, and N. Milošević, "Enabling the IoT paradigm through multi-tenancy supported by scalable data acquisition layer," *Ann. des Telecommun. Telecommun.*, vol. 72, no. 1–2, pp. 71–78, 2016.

[35] H. Farhangi, "The path of the smart grid," *IEEE Power Energy Mag.*, 2010.

[36] A. P. Johnson, "The history of the smart grid evolution at Southern California Edison," in *Innovative Smart Grid Technologies Conference, ISGT 2010*, 2010.

[37] N. D. Popović, D. S. Popović, and I. Seskar, "A novel cloud-based advanced distribution management system solution," *IEEE Trans. Ind. Informatics*, 2018.

[38] H. Liu, "Big data drives cloud adoption in enterprise," *IEEE Internet Comput.*, vol. 17, no. 4, pp. 68–71, 2013.

[39] Y. Simmhan *et al.*, "Cloud-Based Software Platform for Big Data Analytics in Smart Grids," *Comput. Sci. Eng.*, vol. 15, no. 4, pp. 38–47, 2013.

[40] A. Meloni and L. Atzori, "A cloud-based and RESTful Internet of Things platform to foster Smart Grid technologies integration and re-usability," in *2016 IEEE International Conference on Communications Workshops, ICC 2016*, 2016.

[41] N. Dalcekovic, S. Vukmirovic, I. Kovacevic, and J. Stankovski, "Providing flexible software as a service for smart grid by relying on big data platforms," in *2017 International Symposium on Networks, Computers and Communications, ISNCC 2017*, 2017.

[42] I. Kovacevic, A. Erdeljan, S. Vukmirovic, N. Dalcekovic, and J. Stankovski, "Combining real-time processing streams to enable demand response in smart grids," in *2017 International Symposium on Networks, Computers and Communications (ISNCC)*, 2017, pp. 1–6.

[43] S. U. October, "Smart Grid Maturity Model Update | October 2010 About the Smart Grid," *Smart Grid Matur. Model*, 2010.

[44] R. B. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao, "NIST cloud computing reference architecture," in *Proceedings - 2011 IEEE World Congress on Services, SERVICES 2011*, 2011.

[45] P. Mell and T. Grance, "The NIST Final Version of NIST Cloud Computing Definition Published," *Nist Spec. Publ.*, 2011.

[46] S. S. Pritom and H. Lutfiyya, "Geography aware virtual machine migrations for distributed cloud data centers," in *2015 IEEE 4th International Conference on Cloud Networking, CloudNet 2015*, 2015.

[47] P. Mell, “What’s Special about Cloud Security?,” *IT Prof.*, 2012.

[48] V. Lloyd, “ITIL,” *Comput. Bull.*, 2003.

[49] V. Arraj, “ITIL ® : the basics,” *Axelos*, 2013.

[50] D. Cannon, “ITIL 2011 - Service Strategy,” *TSO for the Office of Government Commerce, London*. Stationery Office, 2011.

[51] Great Britain Cabinet Office, *ITIL Service Design*. Stationery Office, 2007.

[52] O. of G. Commerce, *ITIL Service Transition*. Stationery Office, 2011.

[53] O. of G. Commerce, *ITIL® Service Operation*. Stationery Office, 2008.

[54] The Cabinet Officer Inc., *ITIL ® Continual Service Improvement*. Stationery Office, 2011.

[55] H. Mintzberg, “The Strategy Concept I: Five Ps for Strategy,” *Calif. Manage. Rev.*, 1987.

[56] M. M. Ravanfar, “Analyzing Organizational Structure based on 7s model of McKinsey,” *Int. J. Acad. Res. Bus. Soc. Sci.*, 2015.

[57] R. L. Daft, J. Murphy, and H. Willmott, *Organization theory and design*. Cengage learning EMEA, 2010.

[58] R. Simons, “How new top managers use control systems as levers of strategic renewal,” *Strateg. Manag. J.*, 1994.

[59] J. R. Galbraith, “Organizing to deliver solutions,” *Organ. Dyn.*, 2002.

[60] A. Osterwalder, Y. Pigneur, G. Bernarda, and A. Smith, *Value proposition design: how to create products and services customers want: get started with ...* Wiley, 2014.

[61] M. E. Porter, “The five competitive forces that shape strategy,” *Harv. Bus. Rev.*, 2008.

[62] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger, “Multi-tenant databases for software as a service: schema-mapping techniques,” in *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2008.

[63] A. Azeez *et al.*, “Multi-tenant SOA middleware for cloud computing,” in *Proceedings - 2010 IEEE 3rd International Conference on Cloud Computing, CLOUD 2010*, 2010.

[64] C. Teixeira, J. S. Pinto, R. Azevedo, T. Batista, and A. Monteiro, “The building blocks of a PaaS,” *J. Netw. Syst. Manag.*, 2014.

[65] S. Walraven, E. Truyen, and W. Joosen, “Comparing PaaS offerings in light of SaaS development: A comparison of PaaS platforms based on a practical case study,” *Computing*, 2014.

[66] D. Bernstein, “Cloud Foundry Aims to Become the OpenStack of PaaS,” *IEEE Cloud Comput.*, 2014.

[67] C. Vecchiola and R. Buyya, “Aneka: a software platform for .NET-based cloud computing,” *High Speed Large Scale Sci. Comput.*, vol. 18, pp. 267–295, 2009.

[68] N. Chohan *et al.*, “AppScale: Scalable and open AppEngine application development and deployment,” in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, 2010.

[69] A. Corradi, L. Foschini, S. Fraternale, D. J. Arrojo, and M. Steinder, “Monitoring applications and services to improve the Cloud Foundry PaaS,” in *Proceedings - International Symposium on Computers and Communications*, 2014.

[70] A. Eivy, "Be Wary of the Economics of 'Serverless' Cloud Computing," *IEEE Cloud Comput.*, 2017.

[71] N. Pathak, *Pro WCF 4*. Apress, 2011.

[72] R. Mietzner, T. Unger, R. Titze, and F. Leymann, "Combining Different Multi-tenancy Patterns in Service-Oriented Applications," in *2009 IEEE International Enterprise Distributed Object Computing Conference*, 2009.

[73] O. Zimmermann, "Microservices tenets: Agile approach to service development and deployment," *Comput. Sci. - Res. Dev.*, 2017.

[74] P. A. Bernstein and S. Bykov, "Developing Cloud Services Using the Orleans Virtual Actor Model," *IEEE Internet Comput.*, 2016.

[75] S. Tilkov, "Vaughn Vernon on Reactive Programming with the Actor Model," *IEEE Softw.*, 2016.

[76] L. Kang, H. Zhan, and C. Donggang, "Expressing and Composing Actors for Deterministic and Scalable Programming in Cloud," 2015.

[77] Microsoft, "Reliable Actors Overview." [Online]. Available: https://docs.microsoft.com/en-us/azure/service-fabric/service-fabric-reliable-actors-introduction. [Accessed: 06-Jun-2018].

[78] C. P. Bezemer, A. Zaidman, B. Platzbeecker, T. Hurkmans, and A. Hart, "Enabling multi-tenancy: An industrial experience report," in *IEEE International Conference on Software Maintenance, ICSM*, 2010.

[79] Microsoft, "SaaS Maturity Models." [Online]. Available: http://www.theresearchpedia.com/research-articles/saas-maturity-model-by-microsoft. [Accessed: 15-Mar-2019].

[80] A. S. Tanenbaum and M. Van Steen, *Distributed Systems: Principles and Paradigms, 2/E*. Pearson Prentice Hall, 2007.

[81] S. Stoja, S. Vukmirović, N. Dalcekovic, D. Capko, and B. Jelacic, "ACCELERATING PERFORMANCE IN CRITICAL TOPOLOGY ANALYSIS OF DISTRIBUTION MANAGEMENT SYSTEM PROCESS BY SWITCHING FROM MONOLITHIC TO MICROSERVICES," *Rev. Roum.*, vol. 63, no. 3, pp. 338–343, 2018.

[82] E. Elmroth, P. Leitner, S. Schulte, and S. Venugopal, "Connecting Fog and Cloud Computing," *IEEE Cloud Comput.*, 2017.

[83] S. K. Abd, R. T. Salih, S. A. R. Al-Haddad, F. Hashim, A. B. H. Abdullah, and S. Yussof, "Cloud computing security risks with authorization access for secure Multi-Tenancy based on AAAS protocol," in *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 2016.

[84] A. Jasti, P. Shah, R. Nagaraj, and R. Pendse, "Security in multi-tenancy cloud," in *Proceedings - International Carnahan Conference on Security Technology*, 2010.

[85] W. J. Brown, V. Anderson, and Q. Tan, "Multitenancy - Security risks and countermeasures," in *Proceedings of the 2012 15th International Conference on Network-Based Information Systems, NBIS 2012*, 2012.

[86] V. Strezoski, *Osnovi elektroenergetike*. Novi Sad: Fakultet tehničkih nauka u Novom Sadu, 2014.

[87] A. Singhal and R. P. Saxena, "Software models for smart grid," in *2012 1st International Workshop on Software Engineering Challenges for the Smart Grid, SE-SmartGrids 2012 - Proceedings*, 2012.

[88] A. Mnatsakanyan and S. W. Kennedy, "A Novel Demand Response Model with an Application for a Virtual Power Plant," *IEEE Trans. Smart Grid*, vol. 6, no. 1, pp. 230–237, 2014.

[89] N. Etherden, V. Vyatkin, and M. H. J. Bollen, "Virtual Power Plant for Grid Services Using IEC 61850," *IEEE Trans. Ind. Informatics*, 2016.

[90] "Smart Grid Landscape," 2018. [Online]. Available: http://www.ecnmag.com/sites/ecnmag.com/files/legacyimages/ECN/Articles/2011/04/Smart Grid Diagram-web.jpg. [Accessed: 10-Sep-2018].

[91] Y. Guo, S. Feng, K. Li, W. Mo, Y. Liu, and Y. Wang, "Big data processing and analysis platform for condition monitoring of electric power system," *2016 UKACC 11th Int. Conf. Control*, 2016.

[92] B. Cheng, S. Longo, F. Cirillo, M. Bauer, and E. Kovacs, "Building a Big Data Platform for Smart Cities: Experience and Lessons from Santander," in *Proceedings - 2015 IEEE International Congress on Big Data, BigData Congress 2015*, 2015.

[93] M. Kim, J. Choi, and J. Yoon, "Development of the Big Data Management System on National Virtual Power Plant," in *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, 2015.

[94] J. Giri, "Proactive Management of the Future Grid," *IEEE Power Energy Technol. Syst. J.*, vol. 2, no. 2, pp. 43–52, 2015.

[95] N. D. Popović, "NAPREDNI DISTRIBUTIVNI MENADŽMENT SISTEM ZASNOVAN NA CLOUD INFRASTRUKTURI," Univerzitet u Novom Sadu, 2018.

[96] R. M. Oviedo, F. Ramos, S. Gormus, P. Kulkarni, and M. Sooriyabandara, "A comparison of centralized and distributed monitoring architectures in the smart grid," *IEEE Syst. J.*, 2013.

[97] S. F. Bush, S. Goel, and G. Simard, *IEEE Vision for Smart Grid Communications: 2030 and Beyond Roadmap*. 2013.

[98] M. A. Anuradha and A. Massoud, *IEEE Vision for Smart Grid Controls: 2030 and Beyond*. 2013.

[99] V. Krsman, "Specijalizovani algoritmi za detekciju, identifikaciju i estimaciju loših podataka u elektrodistributivnim mrežama," Univerzitet u Novom Sadu, 2017.

[100] H. L. Storey, "Implementing an integrated centralized model-based distribution management system," in *IEEE Power and Energy Society General Meeting*, 2011.

[101] P. Chakravarty and M. Gamini Wickramasekara, "A better GIS leads to a better DMS," in *2014 Clemson University Power Systems Conference, PSC 2014*, 2014.

[102] A. P. Sakis Meliopoulos, E. Polymeneas, Z. Tan, R. Huang, and D. Zhao, "Advanced distribution management system," *IEEE Trans. Smart Grid*, 2013.

[103] R. H. Katz *et al.*, "An information-centric energy infrastructure: The Berkeley view," *Sustain. Comput. Informatics Syst.*, 2011.

[104] Q. Zhou, W. Guan, and W. Sun, "Impact of demand response contracts on load

forecasting in a smart grid environment," in *IEEE Power and Energy Society General Meeting*, 2012.

[105] H. Aalami, G. R. Yousefi, and M. Parsa Moghadam, "Demand response model considering EDRP and TOU programs," in *Transmission and Distribution Exposition Conference: 2008 IEEE PES Powering Toward the Future, PIMS 2008*, 2008.

[106] A. Yousefi, H. A. Ebrahim, S. Mohsen, and P. Moghaddam, "Enhancement of spinning reserve capacity by means of optimal utilization of EDRP Program," in *Proceedings of the Fourth IASTED International Conference Power and Energy Systems (AsiaPES 2008)*, 2008.

[107] H. A. Aalami, M. P. Moghaddam, and G. R. Yousefi, "Demand response modeling considering Interruptible/Curtailable loads and capacity market programs," *Appl. Energy*, 2010.

[108] X. He, N. Keyaerts, I. Azevedo, L. Meeus, L. Hancher, and J. M. Glachant, "How to engage consumers in demand response: A contract perspective," *Util. Policy*, 2013.

[109] C. A. Ardagna, R. Asal, E. Damiani, T. Dimitrakos, N. El Ioini, and C. Pahl, "Certification-Based Cloud Adaptation," *IEEE Transactions on Services Computing*, 2018.

[110] M. Almorsy, J. Grundy, and A. S. Ibrahim, "Adaptable, model-driven security engineering for SaaS cloud-based applications," *Autom. Softw. Eng.*, 2014.

[111] S. Yin, A. Hameurlain, and F. Morvan, "SLA Definition for Multi-tenant DBMS and its Impact on Query Optimization," *IEEE Trans. Knowl. Data Eng.*, 2018.

[112] Y. Wang, Q. He, and Y. Yang, "QoS-Aware Service Recommendation for Multi-tenant SaaS on the Cloud," in *Proceedings - 2015 IEEE International Conference on Services Computing, SCC 2015*, 2015.

[113] W. T. Tsai, X. Sun, Q. Shao, and G. Qi, "Two-tier multi-tenancy scaling and load balancing," in *Proceedings - IEEE International Conference on E-Business Engineering, ICEBE 2010*, 2010.

[114] R. Vidhyalakshmi and V. Kumar, "Design comparison of traditional application and SaaS," in *2014 International Conference on Computing for Sustainable Global Development, INDIACom 2014*, 2014.

[115] N. Dalcekovic, "Arhitektura softvera za podršku multi-tenant model aplikacija u cloud okruženju," *Zb. Rad. Fak. Teh. Nauk.*, vol. 11, pp. 2110–2114, 2013.

[116] Q. Zuo, M. Xie, G. Qi, and H. Zhu, "Tenant-based access control model for multi-tenancy and sub-tenancy architecture in Software-as-a-Service," *Front. Comput. Sci.*, 2017.

[117] S. Walraven, W. De Borger, B. Vanbrabant, B. Lagaisse, D. Van Landuyt, and W. Joosen, "Adaptive Performance Isolation Middleware for Multi-tenant SaaS," in *Proceedings - 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing, UCC 2015*, 2015.

[118] S. Alqahtani, X. He, and R. Gamble, "Adapting compliance of security requirements in multi-tenant applications," in *Proceedings - 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems, FAS*W 2017*, 2017.

[119] L. Li, Y. Wang, Y. Ding, and Y. Zhang, "Multi-tenants Data Duplication Secure Storage in SaaS," in *Proceedings - 2015 9th International Conference on Innovative Mobile and*

Internet Services in Ubiquitous Computing, IMIS 2015, 2015.

[120] W. Lang, S. Shankar, J. M. Patel, and A. Kalhan, "Towards multi-tenant performance SLOs," *IEEE Trans. Knowl. Data Eng.*, 2014.

[121] X. Zhang, B. Shen, X. Tang, and W. Chen, "From isolated tenancy hosted application to multi-tenancy: Toward a systematic migration method for web application," in *Proceedings 2010 IEEE International Conference on Software Engineering and Service Sciences, ICSESS 2010*, 2010.

[122] A. Rafique, D. Van Landuyt, and W. Joosen, "PERSIST: Policy-Based Data Management Middleware for Multi-Tenant SaaS Leveraging Federated Cloud Storage," *J. Grid Comput.*, 2018.

[123] S. Jansen, G. J. Houben, and S. Brinkkemper, "Customization realization in multi-tenant web applications: Case studies from the library sector," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010.

[124] A. R. Ortega, M. Noguera, J. L. Garrido, K. Benghazi, and L. Chung, "Multi-Tenancy Multi-Target (MT 2): A SaaS Architecture for the Cloud," in *International Conference on Advanced Information Systems Engineering*, 2012, pp. 214–227.

[125] H. Moens, E. Truyen, S. Walraven, W. Joosen, B. Dhoedt, and F. De Turck, "Cost-Effective feature placement of customizable multi-tenant applications in the cloud," *J. Netw. Syst. Manag.*, 2014.

[126] L. Ramachandran, N. C. Narendra, and K. Ponnalagu, "Dynamic provisioning in multi-tenant service clouds," *Serv. Oriented Comput. Appl.*, 2012.

[127] H. Zhuang, K. Lu, C. Li, M. Sun, H. Chen, and X. Zhou, "Design of a more scalable database system," in *Proceedings - 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015*, 2015.

[128] F. Gessert, F. Bucklers, and N. Ritter, "Orestes: A scalable Database-as-a-Service architecture for low latency," in *Proceedings - International Conference on Data Engineering*, 2014.

[129] Z. Ding, X. Gao, J. Xu, and H. Wu, "IOT-StatisticDB: A general statistical database cluster mechanism for big data analysis in the internet of things," in *Proceedings - 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, GreenCom-iThings-CPSCom 2013*, 2013.

[130] L. Jiang *et al.*, "An IoT-Oriented Data Storage Framework in Cloud Computing Platform," *IEEE Trans. Ind. Informatics*, vol. 10, no. 2, pp. 1443–1451, 2014.

[131] G. Agha, "Actors programming for the mobile cloud," in *Proceedings - IEEE 13th International Symposium on Parallel and Distributed Computing, ISPDC 2014*, 2014.

[132] A. Singh and K. Chatterjee, "Identity management in cloud computing through claim-based solution," in *International Conference on Advanced Computing and Communication Technologies, ACCT*, 2015.

[133] L. Lamport *et al.*, "In Search of an Understandable Consensus Algorithm," *Proc. 2014 USENIX Annu. Tech. Conf.*, 2004.

[134] C. J. Smith, "THE IMPACT OF SCADA PROTOCOL CHOICE ON PIPELINE

MONITORING AND DECISION MAKING," in *2018 Petroleum and Chemical Industry Conference Europe (PCIC Europe)*, 2018, pp. 1–9.

[135] R. Kalapatapu, "SCADA Protocols and Communication Trends," *October*, 2004.

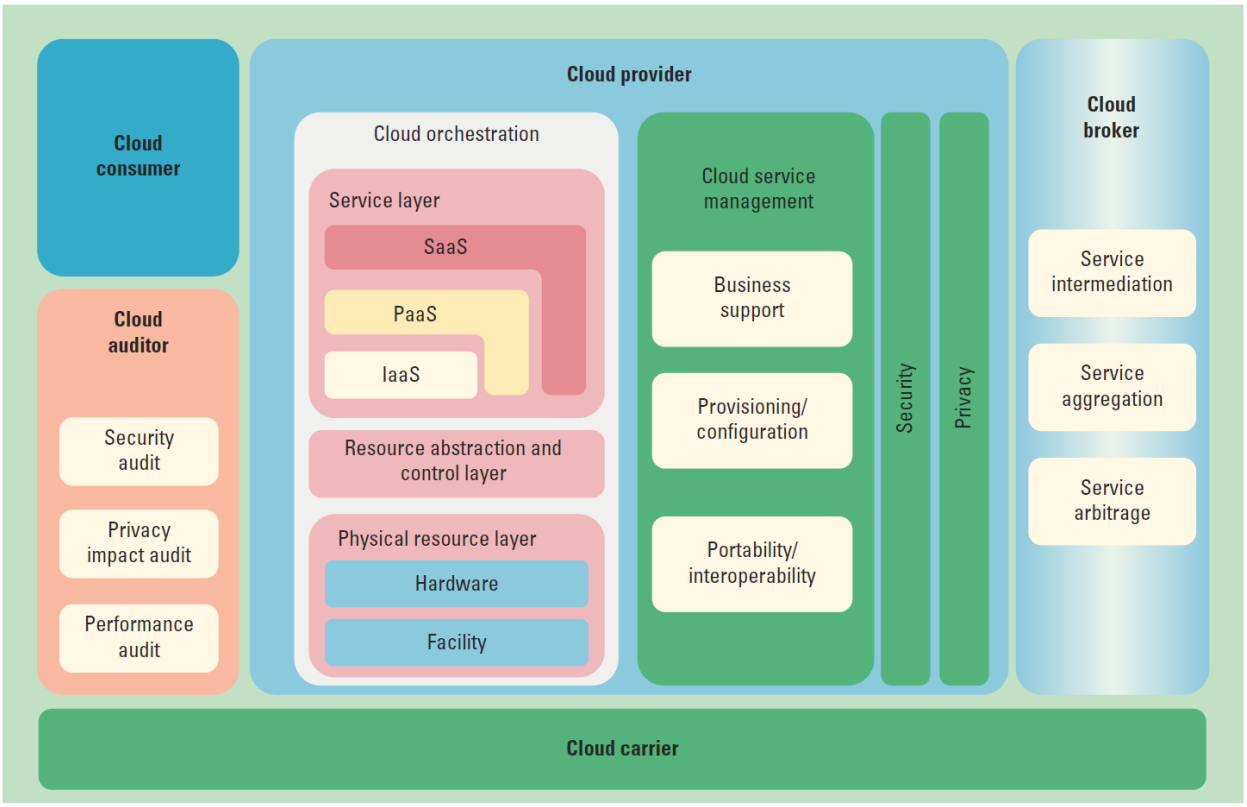
[136] L. Youseff, M. Butrico, and D. Da Silva, "Toward a unified ontology of cloud computing," in *Grid Computing Environments Workshop, GCE 2008*, 2008.

Dodatak A – *Cloud* terminologoija

Iako je termin „*Cloud computing*“ baziran na kolekciji mnogih starih i novih koncepata, u nekim poljima istraživanja kao što su *SOA*, distribuirano programiranje, *grid* računarstvo i virtuelizacija, ovaj termin je privukao mnogo pažnje u poslednjih par godina [136]. U ovom prilogu je predstavljena terminologija iz [136] koja razlikuje šest klasifikacionih elemenata:

- *Service type*
- *Resource deployment*
- *Hardware*
- *Runtime tuning*
- *Security*
- *Middleware*

Termini su namerno ostavljeni na engleskom iz razloga što je cilj poglavlja da izvrši mapiranje poznatih pojmova na one koji su korišćeni, kao i da prikaže njihovu poziciju na nivou celokupne referentne arhitekture *cloud* okruženja. Referentna arhitektura je data u [44] odakle je preuzeta i grafička reprezentacija (Slika 53) *cloud* okruženja.



Slika 53 - NIST Referentni *cloud* model [44]

Service type se odnosi na različite vrste tipova servisa. Servisna ponuda *cloud* okruženja je organizovana slojevito (slojevi su prikazani u bloku orkerstracija). Slojevi su detaljno objašnjeni u odeljku 3.1.3

Resource deployment se odnosi na proces instaliranja softverskih paketa na određene *host* mašine i njihovog izlaganja kao servis a u cilju pružanja usluge krajnjem korisniku. Ovde se javlja ključni pojam koji je vezan za pojam *Cloud* računarstva, a to je virtualizacija resursa. *VDS* (engl.

Virtual Dedicated Servers) jeste ključna tehnologija jer skriva sve detalje hardverske infrastrukture i omogućava taj nivo transparentnosti koji je specifičan za *cloud* okruženja. Na referentnoj arhitekturi je predstavljen kao *Resource abstraction and control layer* u okviru bloka *Cloud orchestration*.

HaaS (engl. *Hardware as a Service*) – Ovaj sloj predstavlja stubište *cloud* okruženja jer se na ovaj sloj oslanjaju svi ostali slojevi. Omogućava korišćenje hardverske infrastrukture bez ramišljanja o njegovom održavanju i bez potrebe za znanjem o pojedinostima samog hardvera. Kod ovog sloja se vodi računa samo o procesoru (engl. *Compute unit*) i memoriji. Kod procesora se posmatra arhitektura procesora i instrukcijski skup koji on podržava, kao i broj procesorskih jedinica. Kod memorije se posmatra da li se radi o permanentnoj memoriji ili radnoj memoriji kao i o kapacitetu određenih vrsta memorija. Na referentnoj arhitekturi ovaj sloj je predstavljen kao *Physical resource layer*.

Runtime tuning se odnosi na mogućnost dinamičkog prilagođavanja veličine zauzetih resursa. *Runtime tuning* se deli na:

- *Load balancing* – predstavlja tehniku raspoređivanja opterećenja ravnomerno na više resursa. Omogućava skalabilnost, pouzdanost i minimalno vreme odgovora na zadati ulaz. Može biti softverski ili hardverski.

- *Resize* – označava prilagođavanje količine resursa koja se pruža jednom *VDS* [136]. Moguće je manuelno podesiti količine resursa, dok postoji i mogućnost automatskog prilagođavanja količine resursa u zavisnosti od opterećenja virtuelne mašine.

- *Checkpointing* – Predstavlja sliku *VDS* u određenom vremenu pri čemu su svi podaci vezani za tekuću virtuelnu mašinu čuvaju za slučaj pada virtuelne mašine. Ova tehnika služi da omogući visoku toleranciju aplikacije koja se izvršava u *cloud* okruženju.

Runtime tuning nije prikazan na referentnoj arhitekturi iz razloga što se odnosi na detalje vezane za *Cloud orchestration*, i termini predstavljaju tehnike koje omogućavaju *cloud* okruženjima fleksibilnost. Deo akcija obuhvaćeni terminom se mogu podvesti pod *Resource abstraction and control layer*.

Security – bezbednosni aspekti u *cloud* okruženjima. Ovaj blok uz *Privacy* na referentnoj arhitekturi se sastoji od mnoštva servisa i alata namenjenih povećanoj bezbednosti upravo iz razloga što su javna *cloud* okruženja u stvari izložena problemima višeorganizacijske osobine.

Middleware – sloj koji omogućava značajno olakšanje zadatka postavljanja aplikacija i njihovo instaliranje i razvoj u *cloud* okruženjima. Omogućava komunikaciju sa slojem infrastrukture koristivši neki od interfejsa kao što su *SOAP, REST* itd. Ovaj sloj u stvari kreira PkS.

Termini koji se ne nalaze u [136], ali se pominju u okviru referentne arhitekture [44] su:

Cloud consumer – Organizacije koje koriste SkS.

Cloud auditor – telo koje proverava *cloud* servise u odnosu na potreban nivo bezbednosti, privatnosti, performansi itd. Ovaj isti princip bi trebao da bude usvojen radi provere bezbednosti višeorganizacijske platforme predložene ovim radom.

Cloud carrier – posrednik konektivnosti između korisnika i pružaoca servisa. Na ovaj način, korisnici servisa mogu zatražiti enkriptovane i privatne komunikacione linije, što je u slučaju pametnih elektroenergetskih mreža važno iz razloga što internet nije dopustljiva opcija.

Cloud broker – omogućava lakšu integraciju *cloud* servisa. Postoje tri dominantne aktivnosti. *Service intermediation* se događa kada *cloud broker* unapređuje servis tako što donosi dodatne vrednosti, ili na neki način poboljšava servis. *Service Aggregation* se događa kada *cloud*

broker kombinuje više *cloud* servisa da pruži novi servis sa dodatnom vrednošću. *Service Arbitrage* je slično agregaciji s tim da broj servisa nije fiksan.

Cloud service management – odnosi se na disciplinu pružanja servisa, što je razrađeno u okviru SkS, i generalno poslovnih modela za pružanje bilo čega kao servis. Sastoji se od tri komponente. *Business support* se odnosi na sve servise koji su neophodni da podrže razvoj poslovanja. *Provisioning and Configuration* se odnosi na sve aspekte kreiranja, konfiguracije, promene pa i merenja potrošnje resursa. *Portability and Interoperability* pruža podršku migraciji tehnologije sa tradicionalnih okruženja na *cloud*, kao i migraciji podataka između različitih okruženja.

Važno je primetiti da se *cloud service management* odnosi na potrebu za promenom organizacione strukture kompanije koja pruža proizvod kao servis, kao i na postojanje dodatnih discplina i resursa posvećenih servisima. Za organizaciju koja planira da pređe sa isporuke rešenja na SkS, potrebno je da se promene poslovni procesi i interna organizacija. Ovi aspekti su van opsega ove disertacije iz razloga što je fokus na tehničkom aspektu migracije, a ne na poslovnim, organizacionim i ekonomskim.

Biografija

Kandidat Nikola Dalčeković je rođen 1989. godine u Novom Sadu. Završio je Gimnaziju „Jovan Jovanović Zmaj“ u Novom Sadu, 2008. godine. Fakultet Tehničkih Nauka u Novom Sadu je upisao 2008. godine da bi diplomski rad iz oblasti praćenja performansi distribuiranih sistema u *cloud* okruženju odbranio 2012. godine na departmanu za Računarstvo i automatiku, Fakulteta tehničkih nauka u Novom Sadu, sa prosečnom ocenom 9.81. Iste godine je upisao master studije na Fakultetu Tehničkih Nauka u Novom Sadu. Diplomski-master rad iz oblasti *multi-tenant* model aplikacija u *cloud* okruženju odbranio je 2013. godine na departmanu za Računarstvo i automatiku, Fakulteta tehničkih nauka u Novom Sadu sa prosečnom ocenom 10.00.

Od 2013. godine je zaposlen u kompaniji „*Schneider Electric DMS NS*“, dok je u istoj kompaniji imao angažman kao stipendista duže od dve godine. Počeo je kao razvojni inženjer, potom kao poslovni analitičar da bi 2015. godine počeo da radi kao menadžer proizvoda za *SaaS*, gde se i danas nalazi.

Angažovan je i na Fakultetu tehničkih nauka. U zvanje asistenta-mastera je izabran 2014. godine. Trenutno drži vežbe iz predmeta *Cloud Computing* u elektroenergetskim sistemima sa nepunim radnim vremenom. Godine 2013. upisao je doktorske studije na studijskom programu Energetika, elektronika i telekomunikacije, na Fakultetu tehničkih nauka u Novom Sadu. Položio je sve ispite predviđene programom sa prosečnom ocenom 10.00. Koautor je dva naučna rada objavljena u međunarodnim časopisima, radu u nacionalnom časopisu kao i na četiri rada objavljenih u saopštenjima sa međunarodnih skupova. Trenutno je angažovan na spremanju rada za istaknuti međunarodni časopis.

Oženjen je i živi u Novom Sadu. Od stranih jezika govori engleski i španski, dok vlada i osnovama ruskog i kineskog.