Upravljanje mrežnim uslugama i resursima u federativnim mrežnim okruženjima

Doktorska disertacija

Jovana Vuleta-Radoičić

Elektrotehnički fakultet

Univerzitet u Beogradu

April 2016

Multi-domain service and resource management in federated network environments

Doctoral Dissertation

Jovana Vuleta-Radoičić

School of Electrical Engineering

University of Belgrade

April 2016

PODACI O MENTORU I ČLANOVIMA KOMISIJE

Mentor:
dr Zoran Jovanović, redovni profesor,
Univerzitet u Beogradu – Elektrotehnički fakultet

Članovi komisije:
dr Pavle Vuletić, docent,
Univerzitet u Beogradu – Elektrotehnički fakultet

dr Dušan Starčević, redovni profesor,
Univerzitet u Beogradu – Fakultet organizacionih nauka

dr Slavko Gajin, docent,
Univerzitet u Beogradu – Elektrotehnički fakultet

dr Žarko Stanisavljević, docent,
Univerzitet u Beogradu – Elektrotehnički fakultet

Mnogo je ljudi kojima dugujem zahvalnost što je ova teza posle dosta vremena uobličena. Posebno se zahvaljujem svom mentoru profesoru Zoranu Jovanoviću i docentu Pavlu Vuletiću koji su mi svojim velikim strpljenjem i podrškom pomogli da ova teza bude napisana. Zahvaljujem se svim kolegama, sadašnjim i bivšim, na podrški tokom proteklih godina i što uvek nađu vremena za svako moje stručno pitanje i dilemu.

I na kraju, zahvaljujem se svojoj porodici, a posebno roditeljima, na strpljenju koje su imali pomažući mi i omogućavajući mi da se posvetim radu.

Ova teza je delimično finansirana sredstvima GÉANT projekta finansiranog od strane Evropske komisije (grant agreement no. 238875).

Rezime

Upravljanje višedomenskim mrežnim uslugama je inherentno kompleksnije od upravljanja uslugom na nivou jednog servis provajdera. Usled ograničenog geografskog pokrivanja uslugama ili zbog nedostatka pojedinih usluga u portfolijima, pojedinačni pružaoci usluga su prinuđeni da se udruže kako bi ponudili komunikacione usluge potencijalnim korisnicima. Svaka situacija u kojoj je potrebno više pružalaca da bi se obezbedila određena usluga je situacija u kojoj postoji federacija pružalaca, model pružanja usluga i upravljanja njima je tzv. federativni servisni model, a upravljanje takvim okruženjem federativno upravljanje mrežama (eng. *federated network management*), koji se u literaturi često zove i višedomenski ili multi-domenski model upravljanja mrežama (eng. *multi-domain network management*). Postoji više modela federacija u zavisnosti od vrste odnosa između domena, a danas su najčešće tzv. labave federacije kod kojih učesnici u federaciji zadržavaju potpunu kontrolu nad resursima koje imaju u vlasništvu. Na ovaj način funkcioniše i današnji Internet, koji se sastoji od autonomnih sistema, a upravo ova autonomija domena predstavlja i najveću prepreku efikasnom pružanju usluga kada je potrebno da uslugu pruži više entiteta zajednički.

U tezi su analizani postojeći pristupi za automatsko upravljanje resursima i uslugama u udaljenim domenima koji učestvuju u pružanju multidomenskih mrežnih usluga. Postojeća rešenja predlažu veliki broj protokola, algoritama i jezika koji obećavaju da mogu da reše problem federativnog upravljanja uslugama na pojedinim nivoima, ali model organizacije federacija i njen uticaj na izvodljivost predloženih rešenja je zanemaren. Takođe, praksa pokazuje da i dalje nema dobrog rešenja kojim bi se omogućilo efikasno i proverljivo upravljanje uslugama koje zahtevaju učešće više od jednog administrativnog domena. Vremena uspostavljanja usluga su velika, a ozbiljnije garancije za kvalitet usluge ne postoje (najčešće je to samo dostupnost usluge).

Za uspostavljanje i održavanje instance višedomenske usluge neophodno je koordinisano i blagovremeno izvršiti potrebne operativne procedure u svim domenima koji učestvuju u pružanju instance usluge. Heterogenost mrežne infrastrukture, različiti procesi, procedure, prava pristupa i legislativa u različitim domenima su ključni razlozi koji čine upravljanje uslugama u federativnom okruženju veoma

složenim. Sa druge strane, potreba da instanca višedomenske usluge bude visoko dostupna i pouzdana, te stalna konkurencija na tržištu koja zahteva kratka vremena uspostavljanja usluga nameću uvođenje što je moguće većeg stepena automatizacije radi što efikasnijeg upravljanja uslugama, a samim tim i resursima koji su uključeni u pružanje usluga. U slučaju labavo spregnutih federacija, domeni zadržavaju potpunu kontrolu nad resursima koji učestvuju u višedomenskim uslugama što potpunu automatizaciju pružanja federativnih usluga čini u opštem slučaju nemogućom jer upravljanje resursima iz drugih domena predstavlja ugrožavanje autonomije pa automatska alokacija i promena konfiguracije resursa iz drugih domena koji učestvuju u pružanju usluge nije dozvoljena da bi se izbeglo dvostruko upravljanje nad resursima. Maksimum automatizacije koji se može postići u tom slučaju je automatska razmena poruka između domena uključenih u pružanje višedomenske usluge. U tezi je prikazan sistem koji omogućava automatizaciju razmene poruka između domena (*Federated Trouble Ticket System*) i opisana njegova realizacija u GÉANT-NREN federativnom mrežnom okruženju.

Ukoliko domeni pojedine fizičke ili virtuelne uređaje i/ili njihove funkcije stave u zajednički pul resursa kojim upravlja federacija, upravljanje pulom resursa postaje centralizovano i mogu se primeniti neka od predloženih proširenja arhitektura i modela upravljanja računarskim mrežama. Tada se adekvatno definisanim pravilima koja određuju ponašanje sistema može obezbediti da udaljeni domen može upravljati isključivo svojim resursima i resursima koji učestvuju u instancama usluga koje domen pruža.

U specijalnim slučajevima i za pojedine mrežne usluge moguće je realizovati sisteme koji omogućavaju domenima da definišu nivoe autorizacije za zahteve za resursima koji dolaze iz udaljenih domena čime se stvaraju uslovi za automatizaciju procesa uspostavljanja usluga ili njihovih kasnijih korekcija. U tezi je opisana platforma za uspostavljanje i pronalaženje adekvatne automatizovane politike za upravljanje resursima u udaljenim domenima sa generičkim modelom podataka. Naime, za specifikaciju pravila pristupa resursima je korišćen ABAC model (eng. *Attribute Based Access Control Model*) kontrole prava pristupa što omogućava specifikaciju modela podataka parametrima i u trenutku instanciranja polisa. *Framework* je implementiran u XACMLv3 jeziku za specifikaciju polisa i realizovan je sistem koji omogućava automatsko uspostavljanje i održavanje višedomenskih usluga u mrežama sa softverski definisanim načinom upravljanja *(SDN – Software Defined Networks)* u skladu sa definisanim ograničenjima domena.

Ključne reči: Multi-domain management, Servisi, Kontrola prava pristupa
Naučna oblast: Elektrotehnika
Uža naučna oblast: Računarska tehnika i informatika
UDK broj: 621.3

Abstract

Multidomain service management inherently involves more service operations and orchestration than service management within single domain. Usually every domain retains full control over its resources included in service instance provisioning, i. e. domains form loosely coupled federation in order to jointly provide multidomain service. Heterogenous network infrastructure as well as differences in operations procedures and workflows greatly aggravate federated service management, but domain reluctance to accept configuration or other changes originating from other domains makes full multidomain service automation management impossible as such changes are treated as direct threat to autonomy of the domain. In such environment most of automation domains can achieve is automated exchange of requests and statuses of requested processing, such as trouble ticket exchange through federated trouble ticket system, and human activities are still unavoidable in some steps. However, in specific situations and for specific types of services, such as fixed capacity multi-domain path configuration and activation, it is possible to define set of constraints and regulations that strictly and precisely control how remote domains can access and use resources involved in multidomain service operation. To ease and automate network infrastructure reservation in loosely coupled federations such as federation consisted of European National Research and Education Networks, we propose distributed context-aware policy-based access control framework which allows domains to individually and independently define policies concerning their resources and to use resources from others domains according to policies defined by the domain owner.

It can be argued that with the advance and improvement of various virtualization technologies it could be possible to make reservations of virtual parts of the infrastructure within the autonomous domains that would be used for federated services. From the service operations viewpoint, exposing and controlling virtual instances of resources in distant domains is not always the case of service operations in the federated service environment. A set of resources (physical or virtual) that are administered without any restriction by a single entity wherever these resources are physically placed constitute a case of single domain service operations. From the top level viewpoint such case certainly is the case of federated service environment,

as multiple domains create service instances and appropriate contracts and policies that regulate the use of virtual resource instances in remote domains have to be defined, but on a service operations level it is not, because all the resources are/can be controlled by a single entity.

Key words: Multi-domain management, Services, Access control
Scientific area: electrical and computer engineering
Scientific subarea: computer engineering
UDC number: 621.3

Sadržaj

Slike

Tabele

Glava 1

Uvod

Usled ograničenog geografskog pokrivanja uslugama ili zbog nedostatka pojedinih usluga u portfolijima, pojedinačni pružaoci usluga često nisu u stanju da samostalno ponude komunikacione usluge potencijalnim korisnicima. U tim situacijama pružaoci se udružuju i pružaju usluge kao konzorcijumi. Svaka situacija u kojoj je potrebno više pružalaca da bi se obezbedila određena usluga je situacija u kojoj postoji federacija pružalaca usluga. Model pružanja usluga i upravljanja njima je takozvani federativni servisni model, a upravljanje takvim okruženjem federativno upravljanje mrežama (eng. *federated network management*), koje se u literaturi često zove i višedomenski ili multi-domenski model upravljanja mrežama (eng. *multi-domain network management*). Postoji više modela federacija [37] u zavisnosti od vrste odnosa između domena, a danas su najčešće tzv. labave federacije kod kojih učesnici u federaciji zadržavaju potpunu kontrolu nad resursima koje imaju u vlasništvu. Na ovaj način funkcioniše i današnji Internet, koji se sastoji od autonomnih sistema, a upravo ova autonomija domena predstavlja i najveću prepreku efikasnom pružanju usluga kada je potrebno da uslugu pruži više entiteta zajednički.

Predmet istraživanja ovog rada je analiza postojećih rešenja, te dizajn i realizacija sistema za automatsko upravljanje resursima i uslugama u udaljenim domenima koji učestvuju u pružanju multi-domenskih mrežnih usluga u labavo spregnutim federacijama. Postojeća rešenja za rad višedomenskih usluga su se orijentisala na dva suprotna pristupa:

1. dizajn signalizacionih protokola niskog nivoa kojima se kreira kontrolna ravan za međudomensko upravljanje uslugama [34]. U ovom i sličnim radovima autori su zanemarivali administrativne probleme u pristupu resursima u udaljenim domenima i ograničenja koja oni nameću

2. specifikacija generalnih arhitektura i modela kojima se razlaže problem upravljanja uslugama u federativnom okruženju [37] [106] [14].

Iako se ovim radovima predlažu različiti protokoli, algoritmi i jezici koji obećavaju da mogu da reše problem federativnog upravljanja uslugama, model organizacije federacije i njen uticaj na izvodljivost predloženih rešenja je opet zanemaren. Takođe, praksa pokazuje da i dalje nema dobrog rešenja kojim bi se omogućilo efikasno i proverljivo upravljanje uslugama koje zahtevaju učešće više od jednog administrativnog domena. Vremena uspostavljanja usluga su velika jer se sve radi ručno i čovek operater mora da bude uključen, a ozbiljnije garancije za kvalitet usluge ne postoje (najčešće je to samo dostupnost usluge).

Upravljanje resursima uključenim u pružanje višedomenskih mrežnih usluga je mnogo kompleksnije od upravljanja resursima unutar jednog administrativnog domena. Za uspostavljanje i održavanje instance višedomenske usluge neophodno je koordinisano i blagovremeno izvršiti alokaciju, konfiguraciju i testiranje svih potrebnih resursa u svim domenima koji učestvuju u pružanju instance usluge. Heterogenost mrežne infrastrukture, različiti procesi, procedure, prava pristupa i legislativa u različitim domenima su ključni razlozi koji čine upravljanje uslugama u federativnom okruženju veoma složenim. Sa druge strane, potreba da instanca višedomenske usluge bude visoko dostupna i pouzdana, te stalna konkurencija na tržištu koja zahteva kratka vremena uspostavljanja usluga nameću uvođenje što je moguće većeg stepena automatizacije radi što efikasnijeg upravljanja uslugama, a samim tim i resursima koji su uključeni u pružanje usluga. U slučaju labavo spregnutih federacija, domeni zadržavaju potpunu kontrolu nad resursima koji učestvuju u višedomenskim uslugama što potpunu automatizaciju pružanja federativnih usluga čini u opštem slučaju nemogućom jer upravljanje resursima iz udaljenih domena predstavlja ugrožavanje autonomije domena, pa automatska alokacija i promena konfiguracije resursa iz drugih domena koji učestvuju u pružanju instance usluge nije dozvoljena da bi se izbeglo dvostruko upravljanje resursima. Međutim, u specijalnim slučajevima i za pojedine mrežne usluge moguće je realizovati sisteme koji bi omogućili domenima da definišu nivoe autorizacije za zahteve za resursima koji dolaze iz udaljenih domena čime bi se stvorili uslovi za automatizaciju procesa uspostavljanja usluga ili njihovih kasnijih korekcija.

Kompromisno rešenje, koje bi pomirilo kontradiktorne zahteve za automatizacijom upravljanja multi-domen uslugama sa autonomijom domena, bi se eventualno moglo postići virtualizacijom mrežnih resursa i njihovih funkcija (eng. *NFV - Network Functions Virtualization*), ako bi domeni apstrahovali pojedine uređaje i/ili njihove funkcije i alocirali ih u zajednički pul resursa kojim bi upravljala federacija. U tom slučaju bi se adekvatno definisanim polisama (pravilima koja

određuju ponašanje sistema) moglo obezbediti da udaljeni domen može upravljati isključivo svojim resursima i resursima koji učestvuju u instancama usluga koje domen pruža. Drugi način je dizajn i realizicaja sistema koji bi omogućio svakom domenu koji učestvuje u federaciji da definiše pravila i polise pristupa resursima kojima bi se postavila kontrola nad resursima od strane matičnih domena, a ujedno i omogućilo prihvatanje onih zahteva iz udaljenih domena koji su unutar limita definisanih postavljenim ograničenjima.

Policy based management paradigma se decenijama koristi za upravljanje mrežama i distribuiranim sistemima [36]. Razvijeno je više framework-a za specifikaciju polisa (XACML [94], Ponder [29] [122] [133], Rei [121], [58]) koji su inherentno kompleksni i uključuju mnogo različitih procedura za kontrolu prava pristupa i upravljanje na različitim nivoima, na primer, pristup mrežnim uređajima i njihovo konfigurisanje, pristup OSS (*Operations Support System*) i BSS (*Business Support System*) komponentama i definisanje politike njihovog rada i slično. Pravila koja opisuju ponašanje ovakvih sistema tipično održava mnogo ljudi sa različitim zaduženjima i različitom ekspertizom. Primena koncepta *policy continuum* [111] omogućava razvrstavanje polisa na različite nivoe apstrakcije jer su autori polisa često upućeni samo u detalje potrebne za održavanje polisa na svom nivou. Policy-based management sistem treba da obezbedi da se ciljevi definisani na višem nivou apstrakcije što efikasnije ispravno preslikaju u polise na nižim nivoima (policy refinement). Na primer, pravilo: „osoblje za korisničku podršku može uspostaviti video konferenciju sa bilo kojim korisnikom koji nije iz računovodstva dok inženjeri za razvoj softvera mogu uspostaviti video konferanciju samo sa korisnicima iz računovodstva i finansija“ je na nižim nivoima potrebno translirati u konfiguracije mrežnih resursa (na primer: dozvoliti uspostavljanje TCI i UDP komunikacije po portovima 5060 i 5061 između mreža 10.0.1.0/24 i 10.0.5.0/24). Takođe je veoma bitno što ranije i što pouzdanije otkrivanje konfliktnih polisa jer ugrožavaju stabilnost sistema. Bez obzira na to da li je primenjena ACL (*access control list*) [102], RBAC (*role based access control*) [35], MAC (*mandatory access control*) [17], [35], DAC (*discretionary access control*) [35] [17], ABAC (*attribute based access control*) [102] ili RaDaC (*risk adaptive access control*) [59] kontrola prava pristupa, na nekom od nivoa *policy continuum*-a semantika polisa postaje vezana za konkretan model podataka sistema u kom se sistem koristi. Jedan od često korišćenih načina za postizanje semantičke interoperabilnosti različitih modela podataka je korišćenje ontologija [113] [112] [34].

U prvom poglavlju ovog rada su date osnove uvodne napomene i struktura teksta. Drugo poglavlje uvodi definiciju federacije i klasifikaciju federativnih mrežnih okruženja i prikazuje dve različito organizovane federacije u evropskoj akademskoj sredini.

Treće poglavlje objašnjava mrežne usluge i resurse u federativnim mrežnim okruženjima. Opisani su pristupi upravljanju višedomenskim uslugama i data je analiza poslovnih procesa u federativnim mrežnim okruženjima. Na osnovu dekompozicije procesa pružanja višedomenske usluge identifikovane su ključne poruke za upravljanje višedomenskim uslugama. Od načina realizacije ovih poruka zavisi stepen automatizacije pružanja višedomenskih mrežnih usluga. U ovom poglavlju je uočen ključni konflikt: ukoliko domeni zadrže potpunu kontrolu nad svojim resursima, u opštem slučaju potpuna automatizacija operativnih aktivnosti pri upravljanju višedomenskim uslugama nije moguća jer se komande iz udaljenih domena koje teže da upravljaju resursima u datom domenu smatraju ugrožavanjem autonomije domena.

U četvrtom poglavlju je prikazan originalni sistem za automatizaciju međudomenske interakcije korišćenjem *Trouble ticket sistema*, koji predstavlja jedno od mogućih rešenja, a ujedno i najviši mogući stepen automatizacije višedomenskih mrežnih usluga iako uključuje i intervenciju ljudskih operatera. Opisana je arhitektura sistema, identifikovane potrebne funkcionalnosti *Trouble ticket sistema*, definisan model podataka federativnog tiketa i razmotreni mogući načini komunikacije komponenata unutar domena i interkonekcije *Trouble ticket sistema* u *Federativni Trouble ticket sistem (FTTS)*. Na kraju poglavlja su dati implementacioni detalji i primer.

U petom poglavlju je opisan koncept decentralizovanog distribuiranog sistema koji omogućava potpunu automatizaciju uspostavljanja višedomenske usluge, bez učešća ljudi u lancu, u specijalnim slučajevima kada je semantika usluge jasna i precizno definisana. Na osnovu analize procesa uspostavljanja višedomenske putanje, razmene sposobnosti domena da učestvuju u instanci višedomenske usluge i kontrole prava pristupa resursima koje domen može ustupiti federaciji (federativnim resursima) identifikovane su ključne operativne aktivnosti potrebne za automatizaciju uspostavljanja višedomenske usluge. Oglašene sposobnosti domena se koriste u procesu dizajna instance višedomenske usluge. Definisan je minimalni model podataka potreban da bi se opisali višedomenska usluga, federativni resursi i skup ograničenja koje domen postavlja prema drugim učesnicima federacije. Detektovani su osnovni tehnološki preduslovi za realizaciju ovakvog sistema koji su opisani u narednim poglavljima.

Neophodan preduslov automatizacije uspostavljanja višedomenske usluge je sistem za kontrolu prava pristupa koji omogućava granularnu specifikaciju pravila, ograničenja i uslova pod kojim resurs može biti korišćen za uspostavljanje višedomenske mrežne usluge. Takav sistem spada u realizaciju *Policy-Based Management* paradigme, pa je u šestom poglavlju data komparativna analiza postojećih *Policy-Based Management sistema*. Takođe, u istom poglavlju je dat pregled osnovnih

koncepata i modela koji se koriste u sistemima za kontrolu prava pristupa i, na osnovu komparativne analize, zaključeno je da je najopštiji *Attribute Based Access Control (ABAC) Model* i da se svi ostali modeli mogu implementirati odgovarajućim izborom semantike atributa. Dalje je pokazano da je *Policy Based Access Control Model* specijalan slučaj ABAC modela u kom je specificirana semantika korišćenih atributa.

U šestom poglavlju je dodatno predložena originalna *Environmental Context-Aware Policy-Based Access Control Framework (ECAPBACF)* platforma za kontrolu prava pristupa federativnim resursima. Prema dosadašnjem saznanju ECAPBACF je jedini *framework* za kontrolu prava pristupa dizajniran za upravljanje federativnim resursima koji koristi opšti model podataka za specifikaciju polisa, a pored statički definisanih ograničenja uzima u obzir i kontekst operacije, tj. stanje okruženja. ECAPBACF omogućava dobijanje autorizacije za sve raspoložive federativne resurse jednim upitom. Uz to, ECAPBACF se može implementirati na proizvoljnom jeziku za opis polisa.

ECAPBACF *framework* je formalno opisan korišćenjem FIA algebre. Za razliku od drugih sistema zasnovanih na *Fine-Grained Integration Algebri (FIA)* koji predlažu heuristike i optimizacije za poboljšanje performansi, ECAPBACF polise su dizajnirane tako da se implicitno izbegnu kompleksni algoritmi potrebni za određivanje integrisane polise. ECAPBACF za organizaciju pravila za kontrolu prava pristupa koristi *Attribute Based Access Control Model* što omogućava korišćenje meta-modela pri specifikaciji polisa. Pored prikaza matematičke zasnovanosti ECAPBACF *framework*-a korišćenjem FIA algebre, dat je i primer implementacije ECAPBACF *framework*-a korišćenjem *eXtensible Access Control Markup Language Version 3 (XACMLv3)* jezika za specifikaciju polisa i definisan meta-model XACML ECAPBACF polise. U istom poglavlju su prikazani i osnovni koncepti FIA algebre potrebni da bi se definisao ECAPBACF. Šesto poglavlje sadrži i analizu kompleksnosti algoritma za generisanje integrisane polise kojom je određena skraćena topologija, a time i topologija višedomenskog servisa, koja pokazuje da algoritam ima linearnu složenost, što ga čini primenljivim u realnim okruženjima.

Sedmo poglavlje donosi demonstraciju primene ECAPBACF platforme na primeru konfiguracije i aktivacije višedomenske putanje u mrežama sa softverski definisanim načinom upravljanja (*Software Defined Network (SDN)*) tako što je realizovan prototip sistema za uspostavljanje višedomenske putanje.

Osmo poglavlje je zaključak u kom su sumirani rezultati i prikazane smernice za dalje istraživanje koje bi se temeljilo na rezultatima ove doktorske disertacije.

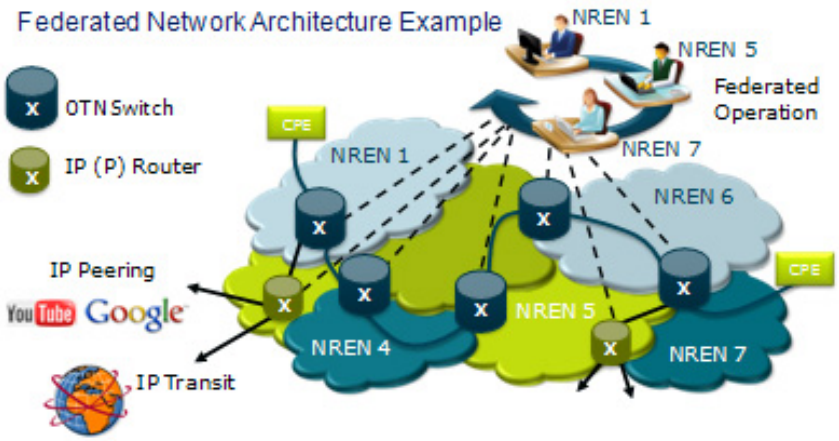
Glava 2

Federativna mrežna okruženja

Federacija je grupa heterogenih entiteta koji se udružuju radi ostvarivanja zajedničkih poslovnih ciljeva i interesa pri čemu svaki entitet zadržava određeni stepen lokalne autonomije pri ostvarivanju svojih poslovnih ciljeva [106] [39]. U literaturi postoji više definicija federativnog mrežnog okruženja [37]. Definicije variraju od "skup domena kojim upravlja ili jedan centralni entitet ili skup distribuiranih entiteta u kom svaki domen ima ograničena prava nad svojim lokalnim resursima i interesima" [106] do "skup nezavisnih entiteta ili mrežnih domena koji zadržavaju svu odgovornost u funkcionalnom i tehničkom smislu nad svojim upravljanjem" [37]. U prvom slučaju, članovi federacije ustupaju deo svog suvereniteta centralnom entitetu ili ograničavaju svoja prava u kontekstu saradnje u distribuiranom okruženju. Ova vrsta modela federacije se naziva čvrsto spregnuta federacija (eng. *tightly-coupled federation*), za razliku od labavo spregnute federacije (eng. *loosely-coupled federation*), u kojoj domeni zadržavaju potpunu nezavisnost i suverenost nad svojim resursima i aktivnostima.

2.1 GÉANT-NREN federativno okruženje

GÉANT i nacionalne obrazovno-naučno-istraživačke mreže (eng. *National Research and Education Networks (NREN)*) obrazuju naprednu komunikacionu infrastrukturu sa ciljem povezivanja svih istraživačkih i obrazovnih institucija u svim evropskim zemljama. Infrastruktura je organizovana u više slojeva i povezuje različite entitete (na primer, istraživačke i obrazovne kampuse i krajnje korisnike). Veza krajnjih korisnika iz dve institucije u različitim domenima je obezbeđena nizom nezavisnih računarsko-komunikacionih mreža počev od kampusa institucija krajnjih korisnika, preko NREN-ova koji se mogu sastojati i od regionalnih naučno-istraživačkih i obrazovnih mreža kojima upravljaju različiti subjekti, i najzad GÉANT mreže kao okosnice, što je prikazano na Slici 2.1.



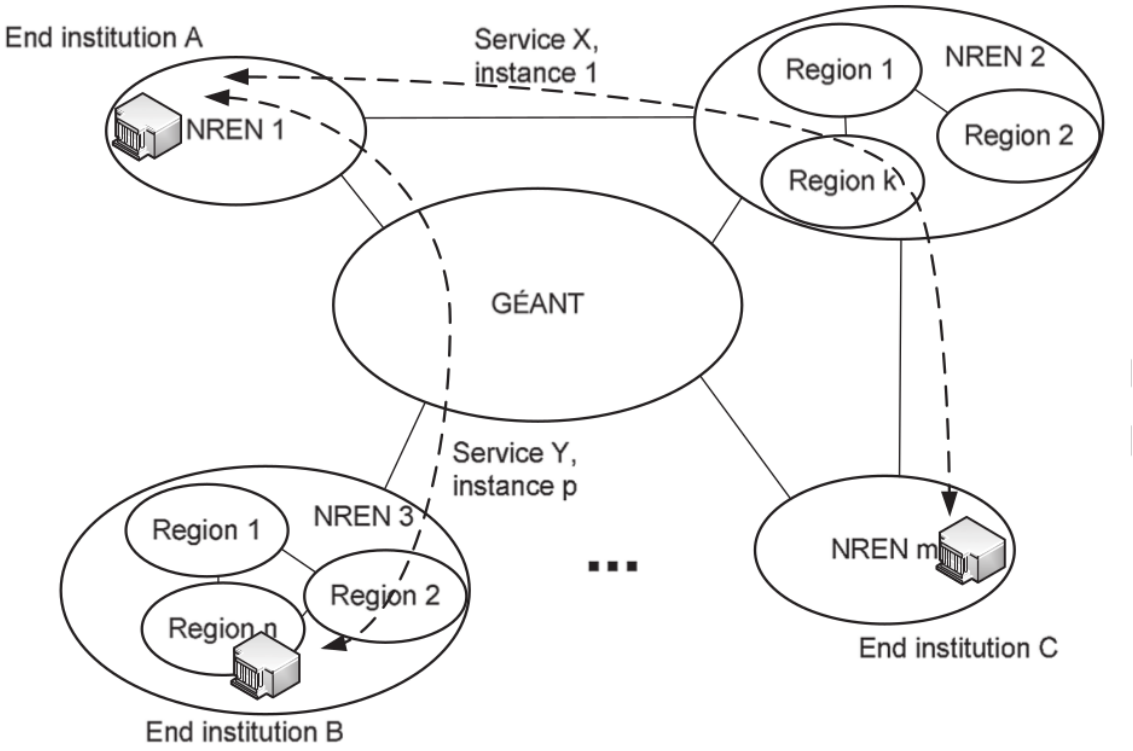
Slika 2.1 GÉANT-NREN federacija [1]

Jedan od glavnih ciljeva GÉANT-NREN okruženja je efikasno i kontinuirano pružanje i podrška (u operativno tehničkom smislu) naprednih mrežnih i drugih IT usluga za projekte, institucije i krajnje korisnike. Primeri servisa u GÉANT-NREN okruženju su iznajmljene veze (optičke talasne dužine ili kola na L2 nivou sa garantovanim propusnim opsegom) između istraživačkih institucija u različitim državama i mesta gde se organizuju eksperimenti (na primer CERN[2]). Moguće su i usluge sa više krajnjih tačaka (eng. *multipoint services*) kao što su *multipoint L3 VPN* zasnovan na MPLS VPN tehnologiji. Višedomenski servisi tipično prolaze kroz GÉANT mrežu, ali mogu biti i direktno između NREN-ova gde postoje direktne veze, kao što je prikazano na Slici 2.2.

Operativne aktivnosti vezane za višedomenske usluge u GÉANT-NREN okruženju tipično uključuju bar nacionalne NREN domene i GÉANT mrežu kao okosnicu. U GÉANT-NREN federaciji domeni udružuju svoje napore i resurse radi što efikasnijeg pružanja usluga krajnjim korisnicima. S obzirom na svrhu i ciljeve federacije, ove usluge se ne pružaju prema modelu kupac-dobavljač relacije između domena, što je slučaj kada komercijalni pružaoci usluga obezbeđuju pružanje (komercijalne) usluge. Kada komercijalni provajder pruža uslugu koja izlazi iz granica njegovog domena, projavder koji ima sklopljen ugovor sa krajnjim korisnikom je nosilac usluge koji angažuje druge (komercijalne) provajdere, tj. kupuje od njih uslugu da bi pružio višedomensku uslugu s kraja na kraj (eng. *end-to-end*). Za postizanje razumnog kvaliteta usluge *(eng. Quality of Service, QoS)* ugovaraju se *Service Level Agreement (SLA)* i *Operational Level Agreement (OLA)* parametri i uvek je samo jedan provajder odgovoran prema krajnjem korisniku koji ne mora ni biti svestan da je u pružanje usluge uključeno više provajdera. U GÉANT-NREN okruženju NREN-ovi

[1] http://geant3.archive.geant.net/Research/Future_Network_Research/Pages/FederatedNetworkArchitectures.aspx
[2] the European Organization for Nuclear Research http://home.cern

na dobrovoljnoj bazi udružuju svoje resurse i napore i ne postoji domen nosilac višedomenske usluge. Da bi pružanje višedomenskih usluga u okviru projekta GÉANT bilo što efikasnije postoji nekoliko koordinisanih napora sa ciljem da se osiguraju neometane operativne aktivnosti u koje spadaju: specifikacija arhitektura koje podržavaju upravljanje višedomenskim servisima [127], projektovanje različitih alata (OSS) za podršku operativnih aktivnosti pri pružanju višedomenskih servisa kao što su OSS alati za automatizaciju određivanja topologije, konfiguraciju i praćenje višedomenskih usluga ili dizajn infrastrukture potrebne za razmenu podataka OSS komponenti.



Slika 2.2 GEANT-NREN federativno mrežno okruženje

Iz navedenog opisa se može zaključiti da GÉANT-NREN okruženje predstavlja federativno mrežno okruženje. Trenutna organizacija međudomenskih odnosa u GÉANT-NREN federaciji je bliža labavo spregnutoj federaciji u kojoj su svi entiteti koji učestvuju u pružanju višedomenskih usluga autonomni u smislu da zadržavaju potpunu odgovornost nad svojim internim upravljanjem, resursima i korisnicima. Okruženja kao GÉANT-NREN su tipična u akademskom svetu, kao što su Internet2 u SAD, Canarie u Kanadi ili u manjim formami u federalnim državama čiji se NREN-ovi sastoje od regionalnih mreža (na primer, Renater u Francuskoj i DFN u Nemačkoj).

2.2 Grid federativno okruženje

Grid computing je skup povezanih računarskih resursa na različitim geografskim lokacijama širom Evrope. Svaki računar je deljen što pretvara mrežu *grid računara* u moćan virtuelni superkompjuter. Virtuelni superkompjuter se sastoji od labavo spregnutih računara koji zajednički obavljaju procesorski i memorijski zahtevne aktivnosti. *European Grid Initiative* je niz napora da se evropskim istraživačima obezbedi korišćenje grid računarskih resursa [70]. *European Grid* se sastoji od nacionalnih Grid inicijativa (eng. *National Grid Initiative (NGI)*). NGI su autonomni domeni koji zajedno sa EGI organizacijom (*EGI Organization (EGI.org)*) koordiniraju korišćenje zajedničkih grid računarskih resursa na evropskom nivou.

Iz aspekta upravljanja računarskim rersursima Grid federacija je labavo spregnuta federacija nezavisnih suverenih domena, dok je iz aspekta upravljanja mrežnom infrastrukturom Grid federacija slučaj labavo vezane federacije kojom upravlja jedan centralni entitet[3]. Naime, *EGI.eu* koordinira distribuirane mrežne operativne centre pružajući im:

- Višedomenske operativne aktivnosti (eng. *Federated operations*) kroz pojednostavljenje svakodnevnih operativnih aktivnosti u heterogenoj federativnoj infrastrukturi
- Koordinaciju operativnih aktivnosti (eng. *Operations coordination*) kroz sinhronizaciju operativnih aktivnosti kako bi se obezbedila neometana i neprekidna integracija računarskih servisa i smanjila fragmentacija

[3]EGI Operations http://www.egi.eu/infrastructure/operations/

Glava 3

Mrežne usluge i resursi u federativnim mrežnim okruženjima

3.1 Pristupi upravljanju višedomenskim uslugama

Postoji više standardizacionih tela koja se bave definisanjem i primenom standarda i primera dobre prakse u oblasti IT servisa i računarskih mreža (TMF, ITU-T, ITIL, ETSI, DMTF). Trenutne specifikacije i standardi u oblasti upravljanja mrežama obično modeluju mreže iz perspektive jednog (nevišedomenskog) komercijalnog provajdera i opisuju vezu različitih domena kao kupac-dobavljač odnos (koji je tipično kupi-prodaj odnos). U *TMF Business framework*-u se takav odnos modeluje kao *Business-to-Business (B2B) interakcija* [116] između pružalaca usluga. U ovom pristupu, veliki broj pružalaca usluga obrazuje lanac radi pružanja usluge krajnjem korisniku. Svaki odnos između pružalaca usluga je kupac-dobavljač odnos podržan *"Customer Interface Management Process"* i *"Supplier/Partner Management Process"* procesima. Kako ovi procesi nisu dovoljni da se u potpunosti podrže sve potrebne međudomenske relacije, posebno u labavo spregnutim federacijama, u literaturi se predlažu proširenja standardnih modela na nivou usluga i resursa [4] [8].

U literaturi [4] [8] [14] [39] se problem upravljanja federativnim mrežnim okruženjima rešava kroz dizajn arhitektura, specifikaciju *framework*-a i modela visokog nivoa koji imaju za cilj da razlože prostor problema federativnog pružanja usluga. Iako ovi radovi daju dobar pregled na visokom nivou i precizno razlažu problem upravljanja višedomenskim uslugama i pojedini predloženi modeli, tehnologije, jezici i algoritmi nude adekvatna tehnološka rešenja na različitim slojevima, ove analize zanemaruju uticaj modela federacije da primenljivost celokupnog procesa.

Dijametralno suprotan pristup upravljanju višedomenskim mrežnim okruženjima je specifikacija različitih signalizacionih protokola niskog nivoa koji stvaraju kontrolnu ravan za automatizaciju međudomenskih interakcija na operativnom

nivou [3] [132] [34]. Autori definišu protokole pod pretpostavkom da su gornji slojevi međudomenskih odnosa rešeni na strateškom i poslovnom nivou na način koji omogućava automatizaciju operacija. Sa sličnom pretpostavkom [130] predlaže načine za autonomno pružanje QoS garancija s kraja na kraj *point-to-point* linkova u višedomenskom okruženju. Iskustva u GÉANT–NREN federaciji pokazuju da takve pretpostavke ne treba uzeti zdravo za gotovo i da u labavo spregnutim federacijama postoji značajan otpor prema sistemima koji omogućavaju automatsko izvršavanje komandi na lokalnim resursima, a koje su inicirane iz drugih domena.

3.2 Upravljanje resursima u koji učestvuju u višedomenskim uslugama

Resursima koji učestvuju u pružanju federativnih mrežnih usluga se može upravljati na jedan od sledećih načina:

- Domen zadržava sva prava nad svojim resursima i brani pristup udaljenim domenima.
- Domen deo resursa ustupa federaciji i federacija centralizovano ili distribuirano upravlja njima.
- Domen zadržava kontrolu nad svojim resursima, ali definiše pravila pristupa koja omogućavaju udaljenim domenima da koriste resurse striktno prema definisanim ograničenjima.

Ako domen zadrži potpuni suverenitet nad svojim resursima, automatizacija pružanja federativnih usluga u opštem slučaju nije moguća. Udaljeno upravljanje resursima iz drugih domena se smatra ugrožavanjem autonomije domena [128], a automatizacija pružanja višedomenskih usluga podrazumeva da se delom infrastrukture domena može upravljati iz udaljenih domena.

Kompromisno rešenje kojim bi se mogla prevazići nevoljnost domena da njihovim resursima upravljaju drugi domeni je kreiranje pula zajedničkih resursa kojim upravlja federacija. U tom slučaju bi domeni mogli da pojedine fizičke ili vurtelne uređaje ili njihove funkcije ustupe federaciji. Adekvatno definisana pravila ponašanja sistema bi obezbedila da domen može upravljati tuđim resursima samo ako učestvuju u instancama višedomenskih usluga u čije je pružanje domen uključen. U ovom slučaju upravljanje uslugama i resursima je centralizovano i federacija treba da reši na organizacionom nivou mehanizme upravljanja. Drugi način postizanja kompromisa između autonomnosti domena i automatizacije upravljanja višedomenskim servisima i resursima potrebnim za njihovo pružanje je dizajn i realizacija

sistema koji omogućava svakom domenu koji učestvuje u federaciji da definiše pravila pristupa za sopstvene resurse predviđene za pružanje višedomenskih usluga. Na ovaj način može da se ostvari decentralizovano upravljanje u kojem ni jedan od domena nema veći značaj od drugih i gde zahtev za uslugom ili obrada usluga može da se vrši iz bilo kog domena. U nastavku teksta će se za ove resurse koristi termin *federativni resursi*. Pravilima bi se uspostavila kontrola nad federativnim resursima na lokalnom nivou i omogućilo prihvatanje onih zahteva iz udaljenih domena koji su unutar limita definisanih postavljenim ograničenjima.

3.2.1 Analiza postojećih rešenja

Definisanje pravila koja propisuju upravljanje federativnim resursima iz zajedničkog pula resursa se može postići delegacijom prava pristupa resursima. [16] predlaže apstraktni sloj delegacije (eng. *abstract delegation layer*) koji omogućava kontrolisanu delegaciju prava pristupa drugim domenima i ima za cilj da sakrije raznorodnost sistema za kontrolu prava pristupa u višedomenskom okruženju. [23] proširuje *eXtensible Access Control Markup Language Version 2 (XACMLv2)* komponente i model toka i obrade podataka kako bi se omogućila dinamička delegacija prava pristupa u kojoj lanac delegacije prava nije unapred poznat. Iako konceptualno mogu rešiti problem upravljanja deljenim pulom resursa, predložena rešenja ne omogućavaju finu granularnost kontrole prava pristupa federativnim resursima koja je potrebna za upravljanje višedomenskim mrežnim uslugama.

[63] uvodi meta-model za opisivanje resursa na nivou pojedinačnog pružaoca usluge proširen apstrakcijama potrebnim za modelovanje kontrole prava pristupa. Sličan pristup imaju i proširenja objedinjenog jezika za modelovanje (eng. *Unified Modeling Language (UML)*)[1] UMLsec [56] i SecureUML [72] koja integrišu podatke vezane za kontrolu prava pristupa u UML specifikacije. Predloženi koncepti omogućavaju specifikaciju pravila kontrole prava pristupa na nivou pojedinačnog domena i mogu se primeniti za centralizovanu kontrolu prava pristupa resursima koji pripadaju zajedničkom pulu resursa.

Rešenja zasnovana na *break-glass* principu [15] [88] [103] pretpostavljaju da svi resursi pripadaju istom domenu i formalno definišu uslove i korake potrebne da se odobri korišćenje resursa koji nije dostupan pod normalnim okolnostima. [88] daje formalan opis modela za kontrolu prava pristupa i polisa za delegaciju prava pristupa. Predloženi koncepti se mogu primeniti za centralizovanu kontrolu prava pristupa resursima koji pripadaju zajedničkom pulu resursa.

Pojedini autori [69] su prepoznali da je za automatizaciju upravljanja uslugama u federativnom okruženju neophodan mehanizam za razmenu konteksta federa-

[1] http://www.uml.org

tivne usluge između domena. Predloženi *framework* koristi *OWL*[2] ontologije za modelovanje dogovorenih sposobnosti na visokom nivou i ne bavi se semantičkom interoperabilnošću domena i mapiranjem modela podataka koji se koriste u različitim domenima.

Neka od opisanih rešenja mogu podržati automatizovano upravljanje federativnim resursima samo u slučaju da domeni ustupe deo svoje infrastrukture federaciji. Ukoliko domeni zadrže potpuni suverenitet, automatsko upravljanje resursima nije moguće jer upravljanje resursima iz udaljenih domena narušava autonomiju domena vlasnika resursa. Opisana rešenja se ne mogu primeniti ni kada domeni zadrže kontrolu nad resursima, ali definišu pravila pristupa kako i kada će prihvatati komande iz udaljenih domena jer nisu dovoljno ekspresivna i ne mogu obezbediti dovoljno granularnu kontrolu prava pristupa.

3.3 Analiza poslovnih procesa u federativnim mrežnim okruženjima

Kao što je već pomenuto u Glavi 2, operativne aktivnosti tokom pružanja multi-domenskih usluga u labavo spregnutim federacijama, kao što je GÉANT– NREN federativno okruženje, se razlikuju od onih koje se obično sreću u komercijalnim multi-provajder okruženjima. Kada se više komercijalnih provajdera udružuje radi pružanja usluge, tipično je provajder koji ima ugovor sa korisnikom odgovoran za pružanje usluge. Ukoliko provajder ne može samostalno obezbediti potrebnu uslugu, sklapa ugovore sa drugim komercijalnim provajderima kako bi usluga mogla biti pružena. Interakcije između provajdera se ostvaruju prema kupac-dobavljač partnerskim procesima [117].

U federativnom mrežnom okruženju, kao što je GÉANT–NREN federacija, pri pružanju usluga postoji opšta saglasnost između domena o skupu usluga koje će federacija da obezbedi, resursima koje će domeni koji učestvuju alocirati za ovu svrhu i ciljanim nivoima kvaliteta višedomenskih usluga. Domeni udružuju svoje resurse i napore u cilju pružanja usluga koje se mogu tražiti i inicirati iz proizvoljnog domena i ni za jednu instancu usluge ne postoji ni jedan domen koji je više odgovoran od drugih domena za obezbeđivanje usluge s kraja na kraj. Stoga, ne postoji jasan dobavljač-kupac odnos između domena.

Međudomenska interakcija u GÉANT–NREN federaciji uglavnom podrazumeva razmenu tehničkih informacija potrebnih za pružanje instance servisa, bez pratećih ugovora. Pružanje pojedinih tipova usluga iziskuje dodatne troškove (na primer, nove mrežne kartice i interfejse) i u ovim slučajevima postoje troškovi vezani za

[2] *OWL Web Ontology Language (OWL)* http://www.w3.org/TR/owl2-overview/

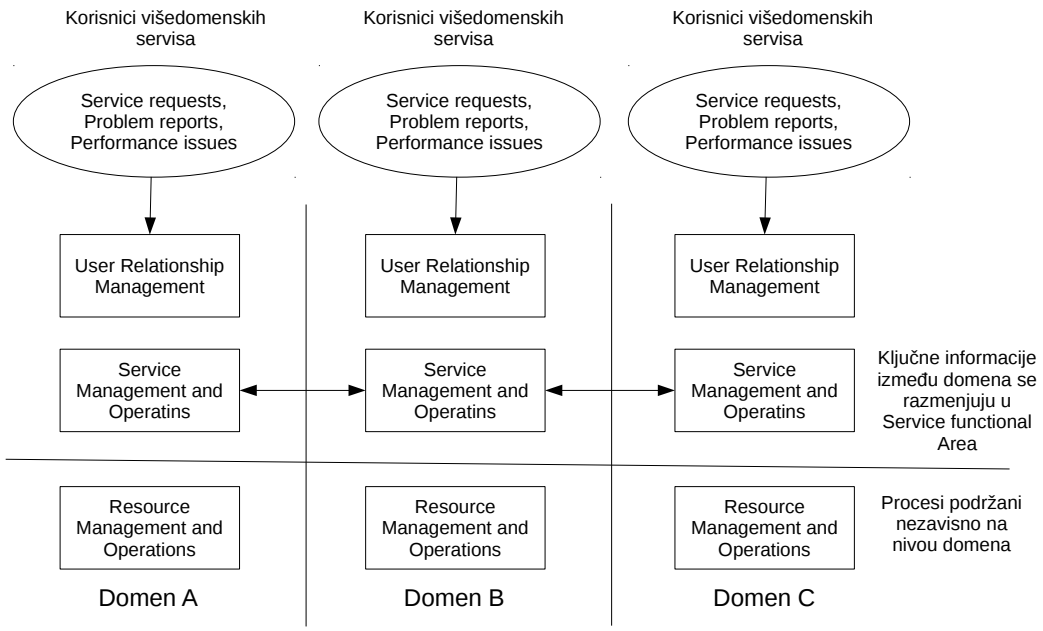
pružanje usluga koji se naplaćuju od korisnika usluga i/ili domena, ali ipak, odnos između domena se ne može tretirati kao dobavljač-kupac partnerski odnos.

Neke od trenutno dostupnih višedomenskih usluga u GÉANT–NREN okruženju su *dedicated lambda*, *bandwidth-on-demand* i *multi-point VPN servis* u kojima se opseg talasnih dužina, putanja sa rezervisanim propusnim opsegom ili *multi protocol label switching (MPLS)* VPN uspostavljaju između krajnjih institucija u različitim NREN-ovima uz učešće nekoliko domena. Priroda interakcije između domena u ovim višedomenskim uslugama labavo spregnute federacije se može posmatrati i analizirati kroz karakteristične poslovne procese kao što je pružanje multidomenske usluge ili rešavanje problema tokom pružanja federativne usluge.

Tehnološki nezavisan proces pružanja višedomenske usluge u labavo spregnutom fedearativnom okruženju u BPMN[3] 2.0 notaciji je prikazan na Slici 3.2. Deo ovog procesa se očigledno koristi za traženje nove instance višedomenskog servisa, ali pojedini njegovi delovi imaju ključnu ulogu u mnogim *end-to-end* aktivnostima, kao što je "zahtev za-promenom" ili "rešavanje problema", jer ove korektivne aktivnosti često zahtevaju promenu konfiguracije instance višedomenske usluge i/ili resursa. Detaljniji opis ovog i drugih federativnih procesa iz GÉANT–NREN okruženja (na primer, upravljanje problemima federativnih usluga i upravljanje kvalitetom federativnih usluga u različitim scenarijima) su opisani u [107].

[3] *Business Process Model and Notation* http://www.bpmn.org

3.4 Dekompozicija procesa pružanja višedomenske usluge



Slika 3.1 Federated service area

Za dizajn instance federativne usluge neophodno je da svi članovi federacije imaju konzistentne relevantne informacije potrebne za pružanje usluge. Stoga domeni moraju imati sinhronizovane kataloge usluga, inventare instanci servisa i informacije o sposobnostima drugih domena, kao i topologiji federacije. Svi ovi podaci se tipično nalaze u *Federated Service Inventory (FSI)* repozitorijumu. FSI može biti realizovan kao centralizovana ili distribuirana baza podataka.

Razmena informacija potrebnih za popunjavanje FSI se obavlja kroz interakciju domena u okviru *service functional area processes*[4] ili alata koje podržavaju ove procese (*Operations Support System (OSS)*). Domeni oglašavaju svoje sposobnosti i podatke u vezi servisa drugim domenima i tako proširuju svoju *service area*[5] u jedinstvenu *federated service area*[6] kao što je prikazano na Slici 3.1. Razmena podataka između *service functional area*[7] procesa otprilike odgovara federaciji u *Service monitoring and configuration* nivou [55].

[4]TMF business framework terminologija
[5]TMF business framework terminologija
[6]TMF business framework terminologija
[7]TMF business framework terminologija

Iako je FSI važan za pružanje višedomenskih usluga, posebno u federacijama sa velikim portfolijima usluga, njegov dizajn i opis je van glavnog fokusa i okvira ove teze. [80] opisuje *federal relationship management system* čije se funkcionalnosti u velikoj meri podudaraju sa potrebnim FSI funkcionalnostima, pa je u nastavku analize pretpostavljeno da centralizovan ili distribuiran FSI postoji i da sadrži sve informacije potrebne za dizajn višedomenskog servisa.

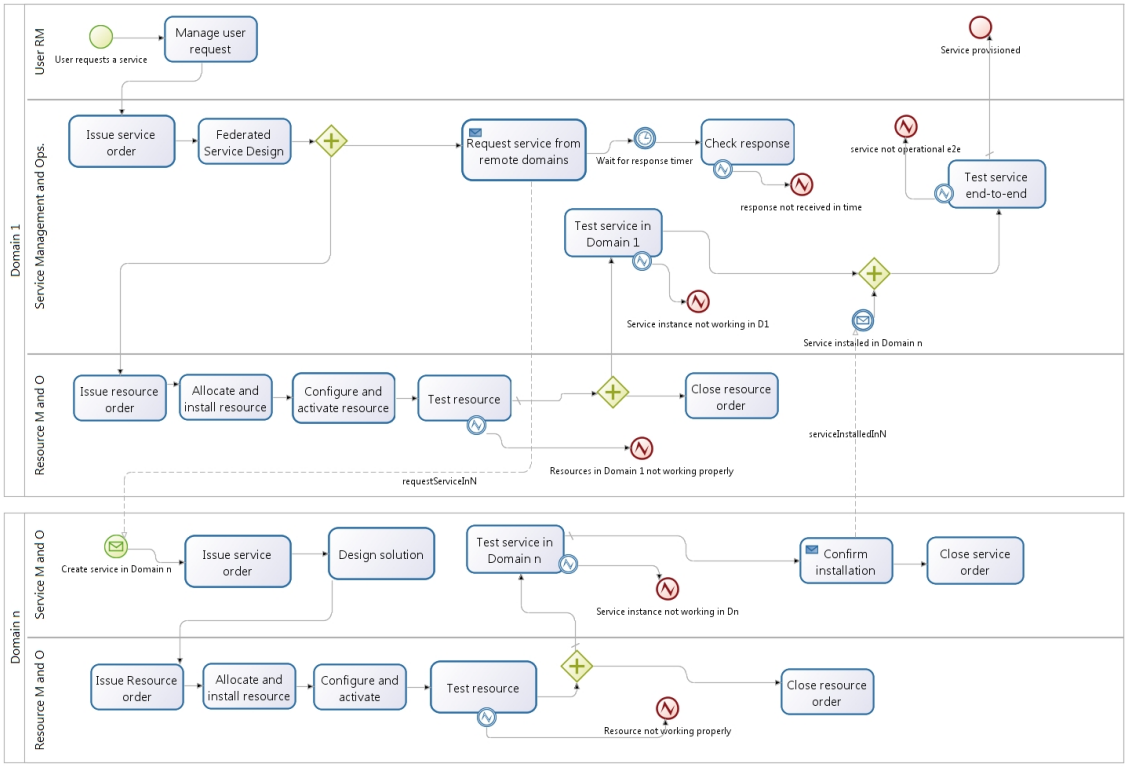
Da bi bitne aktivnosti i relevantni podaci u procesu pružanja federativne usluge bili naglašeni, dijagram toka procesa (Slika 3.2) ne sadrži validaciju korisničkih zahteva i provere da li je traženi federativni servis moguć. Takođe, radi lakše čitljivosti, aktivnosti koje se tiču obrade grešaka i eventualne povratne petlje uzrokovane greškama su namerno izostavljene i na njihovim mestima je naznačen samo generički *Error event*.

Proces pružanja federativne usluge tipično počinje zahtevom koji krajnji korisnik podnosi matičnom NREN-u. Validacija i autorizacija korisničkog zahteva se obavljaju u okviru *User Interface Management procesa*. Aktivnosti *User Interface Management procesa* su nezavisne u svim NREN-ovima zbog jezičkih barijera i različitih ugovornih obaveza koje NREN-ovi imaju prema svojim korisnicima.

Nakon validacije i autorizacije, korisnički zahtev se prosleđuje *Service Management and Operations procesu* da bi se na osnovu zahteva krajnjeg korisnika kreirao jedan ili više naloga za pružanje usluge koji se zatim prosleđuju *Federated Service Design (FSD) procesu* matičnog domena. FSD proces je proširenje *TMF design solution process*-a za federativno mrežno okruženje. FSD proces dizajnira višedomenski servis, što, u osnovi, znači da korišćenjem podataka o povezanosti domena iz FSI repozitorijuma i oglašenih sposobnosti domena da učestvuju u pružanju konkretne instance usluge, određuje međudomensku topologiju servisa, tj. skup domena koji će biti uključeni u pružanje instance usluge. Nakon dizajna instance federativne usluge i identifikacije domena koji će učestvovati u njenom pružanju, zahtev za višedomenskom uslugom (*requestServiceInN message* poruka) se šalje domenima izabranim da učestvuju u pružanju instance višedomenske usluge kako bi nastavili nezavisnu obradu zahteva.

RequestServiceInN message pored opisa dizajnirane usluge sadrži i specifikaciju usluge s kraja na kraj i veze između domena. Svaki domen na osnovu zahteva i opisa federativne usluge nezavisno određuje kako će nova višedomenska usluga koristiti lokalne resurse domena. Nakon dizajna dela višedomenske usluge unutar svojih granica, domen izdaje lokalne naloge za alokaciju, instalaciju, konfiguraciju i testiranje potrebnih resursa u potpunosti zadržavajući svoju autonomiju. Nakon zatvaranja lokalnih naloga, domen obaveštava domen koji je tražio instancu usluge o statusu višedomenske usluge unutar domena (*serviceInstalledInN message* poruka).

Pošto svi domeni obave rezervaciju i konfiguraciju svojih resursa, domen inicijator zatvara zahtev za višedomenskim servisom i ažurira FSI.



Slika 3.2 Proces pružanja višedomenske usluge

Važno je napomenuti da procesi upravljanja kvalitetom federativnih usluga, kao i procesi rešavanja problema koji se tiču višedomenskih usluga, umesto FSD komponente koriste FSI repozitorijum. U ovim slučajevima se domeni koji učestvuju u pružanju instance višedomenske usluge nalaze u FSI.

Federated Service Interaction procesi obuhvataju razmenu podataka između domena potrebnu za uspostavljanje, funkcionisanje i održavanje mrežnih servisa pruženih krajnjim korisnicima u višedomenskom okruženju u kom postoji veza između domena. Ovi procesi omogućavaju proširenje *Service Management and Operations* procesa za višedomensko okruženje.

Federated Service Interaction procesi mogu biti grupisani u četiri glavna podprocesa:

- *Federated service information exchange*
- *Enable federated service configuration and activation*
- *Enable federated service problem management*
- *Enable federated service quality management*

Neko od odgovornosti *Federated Service Interaction* procesa su:

- Razmena sposobnosti domena koje se tiču federativnih servisa, kao i upravljanje i administracija podataka u FSI među kojima su:
  - Održavanje konzistentnog kataloga servisa svih višedomenskih servisa u svim domenima
  - Omogućavanje razmene podataka o dostupnim tipovima višedomenskih servisa (tipovima servisa za koje su domeni alocirali resurse)
  - Omogućavanje razmene podataka o aktivnim insntancama federativnih usluga, kao i evidencija rezultata testova, performansi i svih ostalih podataka koji se tiču servisa
  - Traženje informacije o dostupnosti servisa u udaljenom domenu
  - Izveštavanje o dostupnosti servisa
- Određivanje višedomenske topologije servisa (domena koji učestvuju u pružanju servisa) i veza između domena koje će servis koristiti
- Traženje konfiguracije i aktivacije instance servisa u udaljenom domenu
- Izveštavanje o iskorišćenosti infrastrukture od strane višedomenskih servisa
- Ažuriranje informacija o sposobnostima domena
- Traženje informacije o kvalitetu servisa u udaljenom domenu
- Odgovaranje na zahtev za podacima o kvalitetu servisa u udaljenom domenu
- Traženje i obrada informacije o problemima u vezi servisa u udaljenom domenu
- Odgovaranje na zahtev za podacima u vezi problema servisa iz udaljenog domena
- Omogućavanje posredovanja u korišćenju servisa, instruiranje (davanje uputstava) i izveštavanje
- Razmena ostalih podataka o servisima

Povezivanje podataka potrebnih za upravljanje kvalitetom i rešavanje problema iz različitih domena, monitoring federativnih servisa s kraja na kraj, analiza trendova i izveštavanje o performansama servisa nisu procesi koji striktno pripadadaju *Federated Service Interaction* procesima, ali su neophodni za obavljanje aktivnosti na

operativnom i tehničkom upravljanju servisom. Za ove procese se na nivou GÉANT federacije koristi *multi-domain monitoring tool Perfsonar* [10].

Prethodna analiza pokazuje da je razmena podataka među domenima koji učustvuju u instanci servisa ključna razlika između pružanja usluge u federativnom okruženju i pružanja usluge na nivou jednog servis provajdera. Razmena podataka između domena je neouphodna u različitim fazama procesa pružanja višedomenske usluge i dešava se između različitih procesa koji pripadaju *User Relationship Management* i *Service Management and Operations* procesima. Informacije se razmenjuju kroz niz poruka, kao što su *RequestServiceInN message* ili *serviceInstalledInN message*, kao što je prikazano na slici 3.2. Ove dve poruke nisu jedine poruke koje se razmenjuju tokom pružanja federativne usluge, ali su ključne za upravljanje višedomenskim uslugama jer predstavljaju zahtev i potvrdu obrade zahteva. To može biti komunikacija između admnistratora u mrežnim operativnim centrima (eng. *Network Operations Center (NOC)*) kao što su telefonski pozivi ili e-mail ili deo signalizacionih protokola OSS komponenti koje automatski kontrolišu usluge u domenima. Način realizacije ovih poruka je bitan za automatizaciju pružanja federativnih servisa.

3.5 Jedinstvo komande i automatizacija višedomenskih usluga

Prethodna analiza pokazuje da je razmena poruka između domena ključna aktivnost koje omogućava pružanje višedomenskih usluga, bez obzira da li su u pitanju signalizacione poruke niskog nivoa, poruke u vezi problema ili performansi višedomenskog servisa. Stoga je poželjno da se međudomenska razmena poruka obavlja direktno između komponenti koje mogu automatski konfigurisati potrebne resurse kako bi se ponašanje celog sistema prilagodilo detektovanim promenama okruženja.

Međutim, automatizovane operativne aktivnosti vezane za višedomensku uslugu, kao što su upravljanje konfiguracijom ili performansama, zahtevaju izvršavanje komandi u udaljenim domenima što direktno ugrožava autonomiju domena. Konkretno, mrežni operativni centri domena u GÉANT-NREN okruženju su se opirali implementaciji rešenja za upravljanje mrežama koja su omogućavala udaljenim domenima da ostvare bilo kakvu kontrolu nad lokalnim resursima domena. Takva rešenja su narušavala regularne operativne procedure jer su omogućavala da dva nezavisna entiteta mogu kontrolisati isti mrežni resurs čime se dolazi do situacije da postoje dvostruki lanci komande (eng. *double chain of command*). Jedinstvo komande (eng. *unity of command*), tj. samostalno upravljanje mrežnim i drugim resursima je obavezno u svim domenima u labavo spregnutim federacijama, što

znači da je primanje, prihvatanje i izvršavanje zahteva iz udaljenih domena koji traže promenu konfiguracije mrežnih uređaja neprihvatljivo. Ako domeni zadržavaju potpunu kontrolu nad svojim resursima koji učestvuju u federativnim uslugama, izvestan nivo uplitanja operatera je neophodan u svim poslovnim procesima upravljanja federativnom uslugom, što automatizaciju procesa upravljanja višedomenskim servisima u labavo spregnutim federacijama čini nemogućom. Sa druge strane, razmena informacija o sposobnostima domena, dostupnim resursima, problemima ili indikatorima performansi nikada nije tretirana kao pretnja autonomiji domena koji učestvuju u pružanju federativne usluge, jer mrežni operativni centri mogu filtrirati te podatke u skladu sa svojom sigurnosnom politikom. Stoga nema nikakvih prepreka za automatizovanu detekciju uzroka problema u udaljenim domenima, dok je promena konfiguracije iz udaljenih domena neprihvatljiva.

Glava 4

Automatizacija pružanja usluge u labavo spregnutoj federaciji u opštem slučaju višedomenske usluge

Automatizacija razmene identifikovanih ključnih poruka između domena koji u labavo spregnutoj federaciji zadržavaju potpunu autonomiju se može postići korišćenjem *Trouble Ticket Sistema (TTS)* proširenih tako da podrže federativnu razmenu poruka. Administratori se oslanjaju na TTS kako bi se evidentirao i pratio tok rešavanja problema, grešaka i drugih događaja u vezi pružanja višedomenskih usluga. Kada u pružanju instance servisa učestvuje više domena i kada je neprihvatljivo da OSS sistemi izvršavaju komande inicirane u udaljenim domenima, najčešća praksa je korišćenje e-mailova i telefonskih poziva. Cena ovog izbora je nemogućnost efikasnog praćenja rešavanja problema u udaljenom domenu jer e-mail nema inherentne mogućnosti TT sistema, kao što su praćenje, eskalacija ili isticanje tiketa.

Mogućnost orkestracije kreiranja tiketa u TTS-ima domena koji učestvuju u pružanju instance višedomenske usluge nedvosmisleno poboljšava i unapređuje upravljanje višedomenskim uslugama, jer omogućava svim domenima da imaju sve potrebne informacije, a da pri tome u potpunosti zadrže svoju autonomiju. Sličnu ideju je uspešno primenio multinacionalni komercijalni telekom operater za podršku menadžmenta identiteta između različitih nacionalnih ogranaka [2] [81] kroz integraciju TTS-a korišćenjem standardizovanog OSS/J API-a. Pored toga, mogućnost razmene (eng.) *trouble ticket*-a unutar federacije (koalicije) je, zajedno sa zahtevom za instanciranjem usluge, označena kao ključna interakcija u kontrolnoj ravni u sistemima za upravljanje odbranom [115].

Federativno GRID okruženje (Poglavlje 2.2) je predložilo realizaciju međusobno spregnutih TTS-a na evropskom nivou. Predlog sadrži normalizovan model podataka

normalized network trouble ticket-a i dijagram promene stanja *trouble tiketa* [134]. Međutim, predloženo rešenje odslikava organizaciju GRID federativnog okruženja koje se značajno razlikuje od GÉANT-NREN federacije. Naime, kao i u GRID federaciji, i u predloženom povezivanju TTS-a postoji centralni entitet koji prikuplja i prenosi *trouble tikete* ostalim TTS-ima. Osim toga, zbog istovremenog pokretanja GRID operativnih centara pod okriljem programa Evropske unije, bilo je lakše uvesti zajedničke smernice i alate. Realnost GÉANT–NREN zajednice je drugačija jer su svi entiteti već imali sopstvene alate i operativne procedure pre nego što se pojavila ideja o federativnim operativnim procedurama. Iako je GRID rešenje razmatrano kao kandidat za implementaciju, ono se ne može primeniti na labavo spregnuto federativno okruženje GÉANT–NREN zajednice.

4.1 Sistem za automatizaciju međudomenske interakcije korišćenjem *Trouble Tiket Sistema*

U narednim poglavljima će biti opisan sistem koji omogućava efikasno i pouzdano upravljanje višedomenskim uslugama i istovremeno omogućava mrežnim operativnim centrima da ekskluzivno upravljaju lokalnim resursima da bi se održala autonomija domena. To se postiže uvođenjem potpuno decentralizovanog Federativnog TTS-a (FTTS), za razliku od centralizovog TTS-a predloženog u [134]. Isto tako, za razliku od Ascom pristupa [2] koji zahteva čvrstu integraciju i unifikaciju komunikacionih interfejsa između domena, FTTS rešenje se bolje uklapa u heterogenu prirodu domena koji učestvuju u GÉANT-NREN okruženju. Opisani FTTS nije ograničen samo na osnovne funkcionalnosti TTS-a (upravljanje problemima i otkazima) ili upravljanje identitetima (eng. *Identity management*) [2] već se može primeniti na mnogo širi skup operativnih procedura u vezi upravljanja višedomenskim uslugama. Pored podrške pružanju višedomenskih usluga, FTTS podržava i upravljanje višedomenskim problemima i performansama u kojima su potrebne korektivne akcije.

4.2 Glavne funkcionalnosti *Trouble Tiket Sistema*

Tiket je skup podataka koje se tipično čuvaju kao atributi (ili polja) tiketa, u vezi teme koja se prati. TTS sistem obično vodi evidenciju o svemu što se dešava sa tiketom. Iako svaki TTS ima skup unapred definisanih (ugrađenih) atributa tiketa koji su potrebni za praćenje većine tema, skup atributa tiketa je često prilagodljiv. Korisnički definisani atributi tiketa su poznati kao (eng.) *custom fields*. *Custom fields* omogućavaju potpunu kontrolu modela podataka tiketa.

Tokom praćenja teme, tiket menja stanja prema mašini stanja koja određuje njegov životni ciklus, tj. radni proces institucije u kojoj se tiket koristi. TTS sistemi obično imaju mogućnost da korisnik definiše uslove koji kontrolišu da li je prelazak u sledeće stanje dozvoljen. Takođe postoji i mogućnost specifikacije funkcija, tzv. (eng.) *post-funkcija* koje će biti izvršene kada prelazak tiketa u sledeće stanje bude završen.

Skup atributa tiketa, obaveznih i opcionih, mašina stanja koja opisuje životni ciklus tiketa, kao i mogući podtiketi (eng. *subtickets*) određuju tip tiketa. U zavisnosti od interne organizacije i svrhe TTS sistema, različiti tipovi tiketa su obično posebno prilagođeni praćenju različitih tema. Različiti tipovi tiketa obično imaju različite nivoe bezbednosti.

4.3 Arhitektura federativnog TTS sistema

Nezavisni TTS sistemi domena članova federacije kooperativno formiraju FTTS kroz zajednički distribuirani proces rada (eng. *workflow*). Poslovni proces FTTS-a se može koncizno opisati koristeći primer procesa rada prikazan u pseudokodu na Slici 4.1. Opisani proces rada u potpunosti podržava procese opisane u Poglavlju 3.4. Predloženi poslovni proces je TT_STATUS dijagram promene stanja [134] prilagođen labavo spregnutoj federaciji autonomnih domena. Da bi se prijavio problem ili tražila nova instanca višedomenskog servisa neophodno je kreirati tiket u TTS-u matičnog domena. U zavisnosti od procesa, pristupa se različitim OSS sistemima da bi se dobili podaci neophodni za kreiranje tiketa u drugim domenima. Ako se FTTS koristi za podršku procesa pružanja višedomenske usluge, koristi se FSD komponenta da bi višedomenska usluga bila dizajnirana (*design the service*) i određeni domeni koji će biti uključeni u pružanje instance federativnog servisa. Ako je u pitanju praćenje rešavanja problema ili performansi, FTTS pristupa FSI repozitorijumu da bi dobio potrebne podatke, tj. skup domena koji učestvuju u pružanju višedomenske usluge. Nakon toga, TTS inicijator procesa će proslediti relevantne podatke o višedomenskom servisu TTS-ima drugih domena i tiket će preći u stanje *State:Inactive.Waiting_for_other* u kom će čekati odgovor TTS-a iz ostalih domena. TTS-i u udaljenim domenima kreiraju tikete na osnovu dobijenih podataka i domeni, u zavisnosti od vrste događaja, postupaju u skladu sa svojim operativnim procedurama. Čim udaljeni domen obezbedi pružanje dela instance federativne usluge, odnosno, završi rezoluciju problema, u granicama svoje nadležnosti tiket prelazi u stanje *Closed.Resolved* ili *Closed.Nonexistent_problem* i odgovarajuća poruka se šalje TTS-u domena koji je inicirao kreiranje tiketa. Pošto udaljeni domeni nisu nužno upućeni u sve detalje potrebne za proveru *end-to-end* statusa instance

federativnog servisa, domen inicijator proverava funkcionisanje servisa s kraja na kraj i zatvara tiket. Ukoliko je potrebno, domen kreira novi tiket sa ažuriranim podacima i opisani proces se ponavlja.

Slika 4.1 FTTS - proces rada

Opisani radni proces ne treba smatrati ni najboljim ni preporučenim, već samo smernicom koja se može dodatno prilagoditi specifičnim potrebama federativnog okruženja ili koristi za podršku procesima pružanja i upravljanja federativnim uslugama.

4.4 FTTS - projektne odluke

Interno istraživanje u okviru GÉANT-NREN federacije [45] je pokazalo da 29 ispitanih NREN i regionalnih mreža koristi 11 različitih TTS sistema i sličnih alata, od kojih su neki realizovani samo za te potrebe (eng. *custom-made*). S obzirom na prirodu odnosa domena GÉANT-NREN federacije, rešenja bazirana samo na konfiguraciji i prilagođavanju (eng. *customization*) postojećih rešenja su bolje prihvaćena nego dizajn, razvoj, implementacija i kasnije održavanje novih softverskih alata ili dodataka za razne sisteme. Takođe, domeni članovi federacije nerado pristaju na prelazak na nove sisteme i alate jer to može izazvati dodatne troškove migracije i obuke. Stoga se opisano rešenje bazira na postojećim mogućnostima TTS-a koji su već instalirani i koriste se u NREN-ovima. FTTS je zasnovan na preporukama i standardima (RFC, TMF i OOS/J inicijativa) kako bi u što je moguće manjoj meri remetio uspostavljene procedure mrežnih operativnih centara.

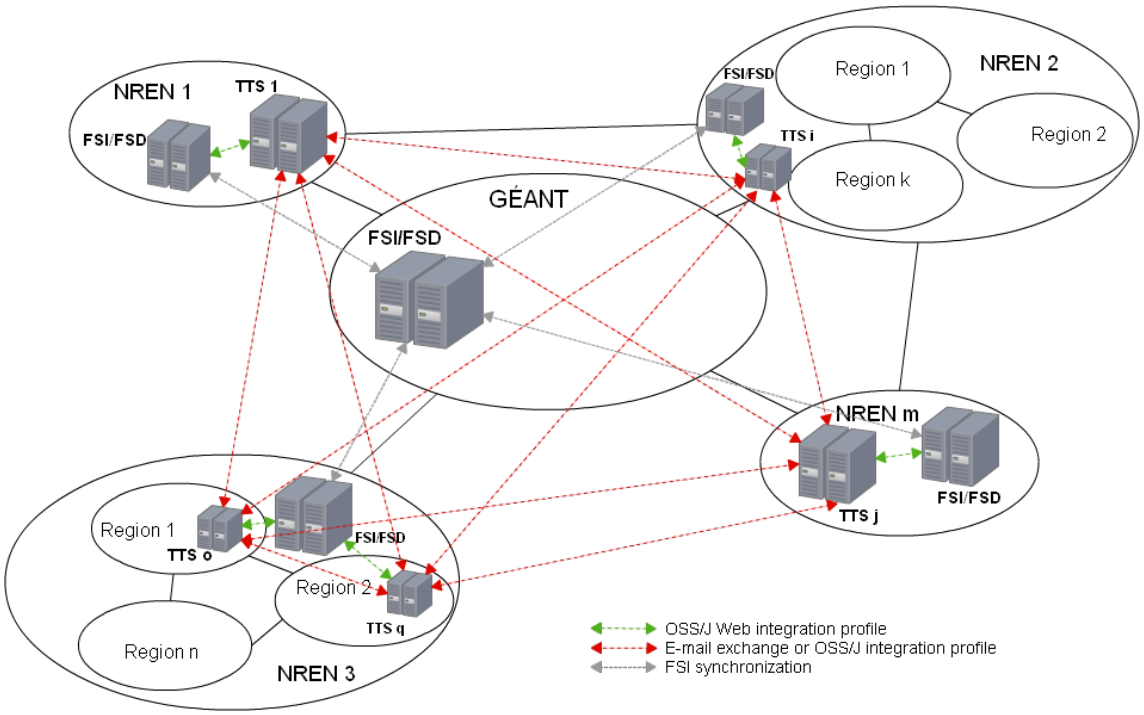
Da bi mogao da učestvuje u federativnom TTS-u, TTS domena mora imati sledeće funkcionalnosti koje postoje u većini *open-source* i komercijalnih TTS sistema:

- Različiti tipovi tiketa - mogućnost korišćenja više različitih tipova tiketa i definisanje novih tipova tiketa
- Promenljiv model podataka tiketa - *custom fields* se mogu koristiti za promenu modela podataka tiketa
- Podesiv životni ciklus (*eng. workflow*) tiketa sa mogućnošću promene stanja tiketa i post-funkcija, pri čemu različiti tipovi tiketa mogu imati različite životne cikluse
- *Post function* može slati *e-mail*-ove
- Mogućnost eksterne *e-mail* komunikacije, pri čemi prijem *e-mail*-a može inicirati kreiranje tiketa i promenu stanja tiketa i
- Mogućnost korišćenja korisnički definisanih predložaka (eng. *template*) u *e-mail* komunikaciji.

Realizacija FTTS prikazanog na Slici 4.2 obuhvata sledeće faze koje će biti opisane u narednim glavama:

- Dizajn modela podataka TT tiketa;
- Specifikacija komunikacije između TTS sistema domena;
- Definicija životnog ciklusa TT tiketa i

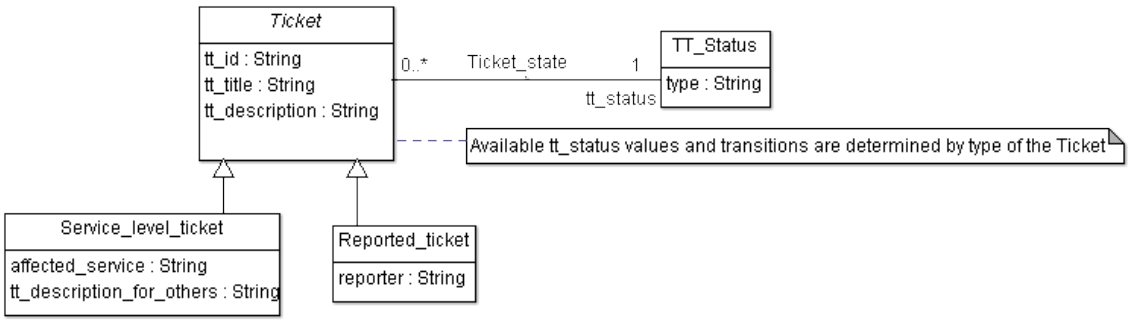
- implementacija i izbor komponenata.



Slika 4.2 Federativni trouble ticket sistem

4.5 Model podataka federativnog TT

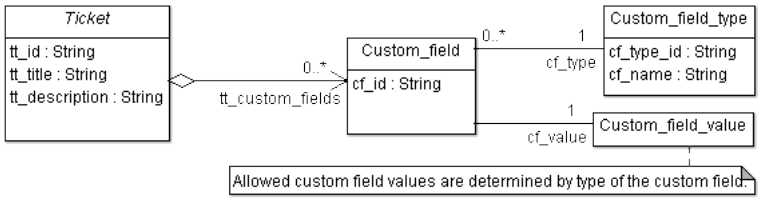
Pri modelovanju federatvinog TT prikazanog na Slici 4.3 korišćen je minimalan podskup atributa normalizovanog TT modela podataka [134] kao zajednički imenilac TT-a u TTS sistemima domena u GÉANT-NREN federativnom okruženju. Svaki tiket ima atribute *tt_id* koji je jedinstveni identifikator tiketa u matičnom TTS-u i *affected_service* koji je jedinstveni identifikator instance servisa u FSI repozitorijumu.



Slika 4.3 Federativni tiket

Opisani model tiketa se dodavanjem adekvatno konfigurisanih *custom-field* atributa (Slika 4.4) može lako proširiti do normalizovanog TT modela. Na isti

način se može formirati tiket proizvojnog sadržaja kako bi se podržale potrebne procedure mrežnog operativnog centra. Atribut *ticket_type* koji se koristi za tipizaciju TT-a može biti dodat na isti način.



Slika 4.4 Federativni tiket - custom fields

4.6 Komunikacija između TTS-a domena

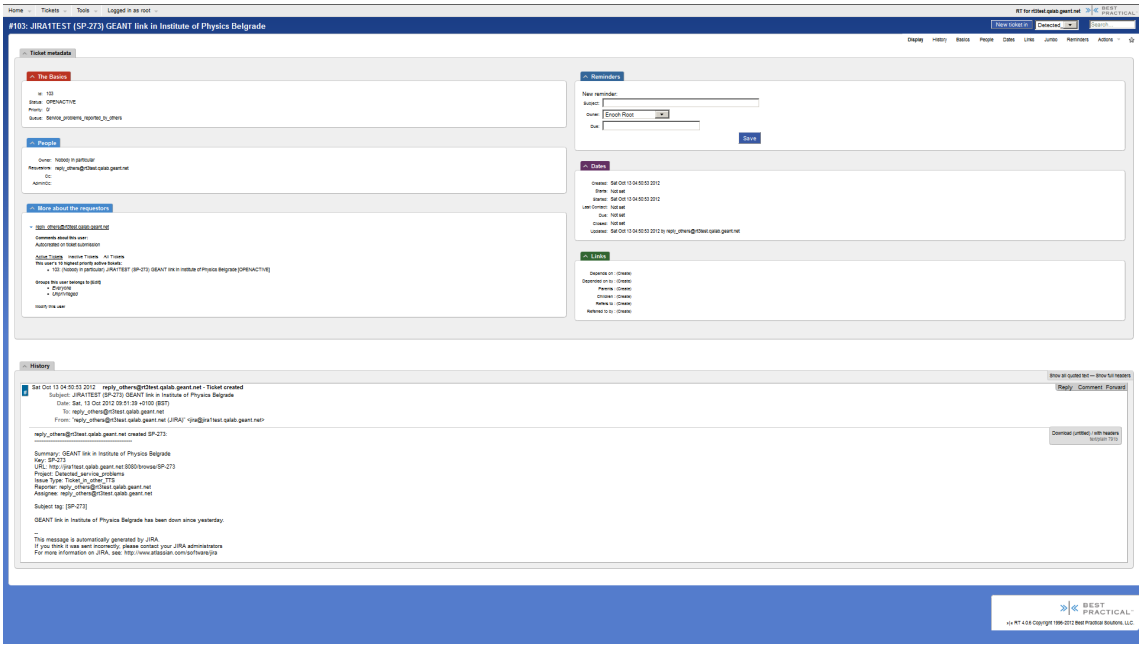
Inter-konekcija TTS-a u različitim domenima može biti realizovana standardizovanim komunikacionim protokolima kao što je *OSS/J API integration profile* [118], korišćenjem (softverskih) arhitektura za podršku komunikacije nezavisnih (softverskih) servisa (eng. *Enterprise Service Bus*-a) ili *e-mail* komunikacijom. Prva dva pristupa su preporučeni primeri dobre prakse za upravljanje mrežom jer garantuju interoperabilnost baziranu na standardima. Međutim, pošto ni jedan NREN nema TTS koji podržava bilo koji od navedenih standardnih načina povezivanja, izbor standardizovanog metoda bi značio razvoj i kasnije održavanje različitih softverskih dodataka za razne TTS-e.

Istraživanje je pokazalo da TTS u skoro svakom domenu ima interfejs, na primer, REST API (eng. *Representational State Transfer*), *XML-RPC* i *JSON-RPC*, koji drugi sistemi mogu koristiti za pristup podacima o tiketima. Na žalost, različiti TTS-i imaju različite modele podataka tiketa, tako da ovi interfejsi obezbeđuju samo sintaksnu, ali ne i semantičku interoperabilnost. Za semantičku interoperabilnost je neophodno obezbediti mapiranje atributa tiketa različtih TTS-a, kao i razvoj dodatnih softverskih komponenata radi obezbeđivanja automatskog mapiranja modela podataka različitih TTS-a.

Sa druge strane, skoro svaki TTS podržava slanje *e-mail* poruka i kreiranje tiketa na osnovu sadržaja *e-mail*-a u formatu koji liči na *JSON*. Pošto se format i sadržaj ovih *e-mail*-ova može kontrolisati korisnički definisanim predlošcima, razmena poruka između TTS-a se može obezbediti adekvatnom konfiguracijom TTS-a. Takođe, slanje *e-mail*-ova na odgovorajuće adrese, kao i promena stanja tiketa mogu biti izvršeni u okviru post-funkcija što znači da *e-mail* bazirana inter-konekcija TTS-a ne iziskuje razvoj novog softvera.

4.7 Komunikacija između FSI/FSD i TTS

Glavni zadatak FSI/FSD komponente je da odredi domene koji učestvuju u instanci višedomenske usluge. Komunikacija između FSI/FSD i TTS-a se odvija prema klijent-server modelu. Po prijemu adekvatno formatirane *e-mail* poruke, FSI/FSD_WS_client komponenta poziva Web servis FSI ili FSD komponente kako bi dobila listu domena uključenih u pružanje instance servisa i na osnovu rezultata upita određuje *e-mail* adrese TTS-a domena. Konačno, FSI/FSD_WS_client šalje e-mail sa listom adresa TTS-u domena koji je inicirao zahtev. *E-mail* mora sadržati *tt_id* tiketa čije je kreiranje iniciralo komunikaciju da bi TTS mogao da upari primljeni *e-mail* sa tiketom.

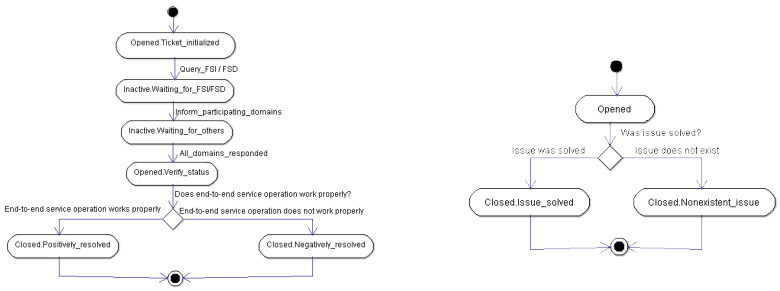


Slika 4.5 FTTS - razmena podataka

4.8 Životni ciklus federativnog TT

Kao što je već navedeno, propagacija detalja i statusa instance višedomenske usluge se ostvaruje automatskim kreiranjem i rezolucijom TT-a u TTS-ima NREN-ova koji su uključeni u instancu servisa. To se postiže prilagođenim životnim ciklusima tipova tiketa TTS-a. Svaki tiket prolazi kroz različita stanja prateći tok rezolucije koji mu određuje životni ciklus. Prelazak iz stanja u stanje može biti iniciran eksplicitnom akcijom korisnika ili događajem kao što su prijem *e-mail*-a, promenom stanja tiketa roditelja ili tiketa deteta, istek određenog vremenskog perioda, ... Prelazak u novo stanje može biti bezuslovan ili kontrolisan nekim uslovom. Takođe, iz jednog stanja može biti moguća tranzicija u više različitih stanja, u zavisnosti od rezultata izračunavanja uslova.

Dijagram promene stanja koji opisuje životni ciklus tiketa u domenu u kom je proces iniciran je prikazan na Slici 4.6a, dok je životni ciklus tiketa u ostalim domenima prikazan na Slici 4.6b. Oba životna ciklusa odgovaraju opisu FTTS-a na visokom nivou koji je prikazan na Slici 4.1. Automatsko kreiranje i rezolucija tiketa, kao i promene stanja u skladu sa navedenim životnim ciklusom tiketa mogu biti izazvani prijemom *e-mail*-a, a automatsko slanje *e -mail*-a se može izvršiti u okviru post-funkcija ili se TTS može konfigurisati tako da šalje *e-mail*-ove na određene adrese pri promeni stanja tiketa odgovorajućeg tipa.



(a) Inicirajući domen (b) Udaljeni domen

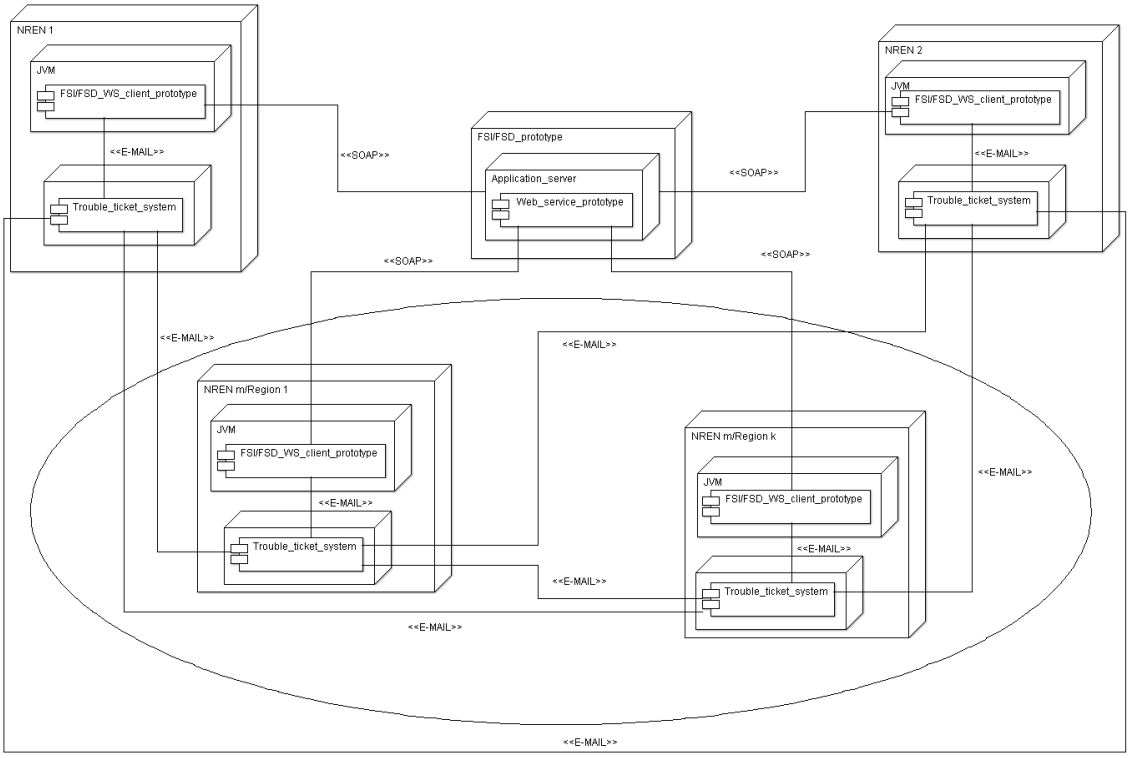
Slika 4.6 Životni ciklusi tiketa

Kao što je opisano u Poglavlju 4.3, da bi se zatražila nova instanca višedomenske usluge ili da bi se prijavio problem u vezi potojeće instance usluge, tiket odgovarajućeg tipa mora biti kreiran u TTS-a inicirajućeg domena. Kreiranje (federativnog) tiketa će aktivirati slanje *e-mail*-a generisanog prema odgovarajućem predlošku. Predložak koji se koristi za kreiranje *e-mail*-a zavisi od vrste tiketa. Na primer, predložak *"Query inventory"* se koristi za automatsko generisanje *e-mail*-a koji inicira poziv Web servisa, dok se predložak *"Forward ticket"* koristi za *e-mail*-ove koji će kreirati tikete u TTS-ima ostalih domena uključenih u pružanje usluge. Svaka *e-mail*

poruka mora sadržati jedinstveni identifikator tiketa (*ticket_id* na Slici 4.5) kako bi bila uparena sa odgovarajućim inicijalnim tiketom.

4.9 Izbor softverskih platformi za federativni TTS

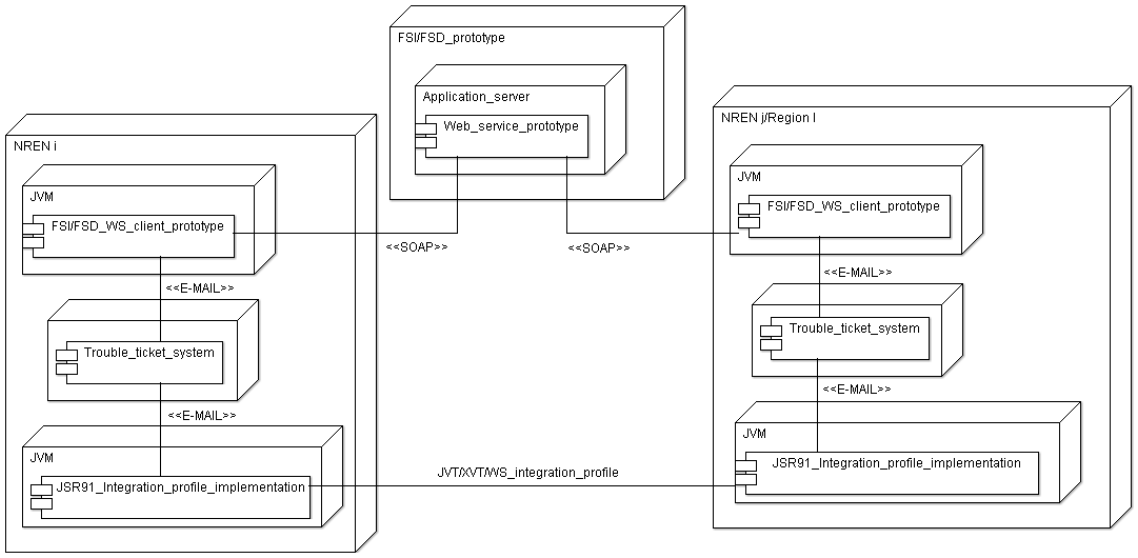
Potpuno funkcionalan FTTS je prikazan na četvrtom simpozijumu GEANT projekta. Budući razvoj FTTS zavisi od budućeg portfolija federativnih usluga u okviru GEANT - NREN okruženja i može uključivati produžetak evaluacije i razvoja modela FTTS tiketa u skladu sa zahtevima korisnika, kao i implementaciju novih životnih ciklusa federativnih tiketa.



Slika 4.7 FTTS - softverske komponente

Opisani FTTS je realizovan korišćenjem nekoliko instanci JIRA 5.1.5 i Request Tracker 4.0.5 TTS-a i FSI/FSD prototipa. Glavne komponente sistema, tj. instance komponenata prikazanih na Slici 4.2, su prikazane na Slici 4.7. JIRA i Request Tracker su na Slici 4.7 prikazani kao *Troube_ticket_system* i odabrani su jer većina NREN-ova koristi neki ova dva TTS-a [45]. FSI/FSD komponente (*FSI/FSD_prototype*) su implementirani kao instance *OSS common API Web integration profile [75].* Konkretan Web servis je opisan WSDL-om OSSJ-common-v1-5.wsdl. FSI/FSD_prototype se izvršava na aplikativnom serveru JBoss 5.1.0, dok je FSI/FSD_WS_client komponenta realizovana kao samostalna Java aplikacija.

Razmena podataka između TTS-a u različitim domenima može bazirana na *e-mail* komunikaciji ili Web servisima. Kada se za inter-TTS komunikaciju koristi *e-mail* (na primer, «E_MAIL» veza između NRENi i NREN J/Region k TTS-a na Slici 4.7, inicirajući TTS kreira tikete u drugim domenima koji učestvuju u instanci višedomenskog servisa slanjem *e-mail*-ova na adrese dobijene od FSI/FSD. U pitanju su adekvatno formatirane *e-mail* poruke koje se šalju na adrese TTS-a u udaljenim domenima u kojima prijem ovih *e-mail*-ova automatski kreira tiket odgovarajućeg tipa. Tiket se dalje analizira i obrađuje u okviru standardnih procedura NOC-a domena i na kraju će automatski kreirani tiket biti zatvoren. Zatvaranje tiketa aktivira automatski odgovor na *e-mail* koji je kreirao tiket. Kada svi domeni koji učestvuju i instanci višedomenske usluge zatvore svoje tikete, tj. svi domeni koji učestvuju su odgovorili inicirajućem domenu, tiket u inicirajućem domenu menja stanje da bi se proverio *end-to-end* status višedomenske usluge i tiket zatvorio na odgovarajući način.



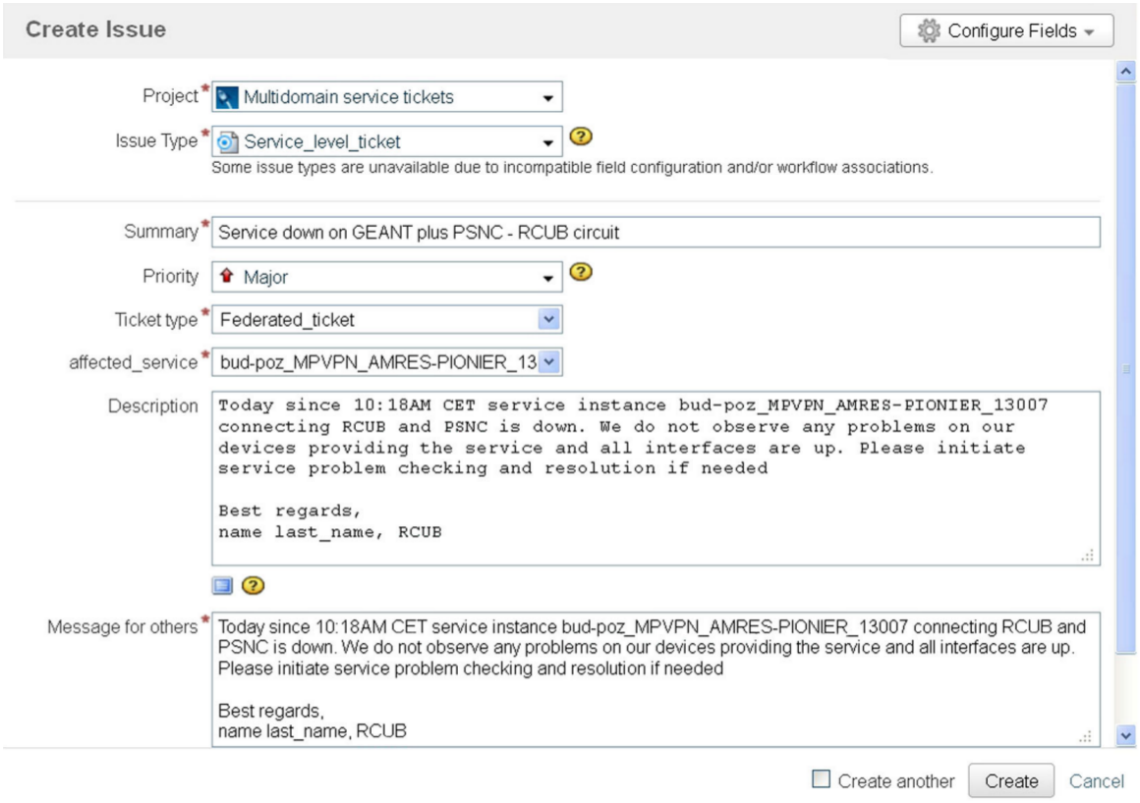
Slika 4.8 FTTS - softverske komponente (WS_integration_profile)

U slučaju korišćenja Web servisa za razmenu podataka između TTS-a (na primer, WS_integration_profile veza između NRENi i NRENj/Region Java Virtuelnih Mašina), neophodno je proširenje opisanog procesa komunikacije. Naime, neophodna je dodatna samostalna Java aplikacija (označena kao JVM na Slici 4.8) kao interfejs između TTS-a u matičnom domenu i Web servisa udaljenih domena. Aplikacija implementira OSS/J TT API Web integration profile kao što je specificirano u JVT-TroubleTicketSession.wsdl i odgovarajućim XML schema fajlovima (JSR 91) i pri prijemu adekvatno formatiranog *e-mail*-a poziva odgovarajući Web servis (JSR 91) u udaljenom domenu, a rezultate poziva Web servisa prosleđuje TTS-u u obliku *e-mail*-a. U ovom slučaju, nije potrebna dodatna TTS konfiguracija u odnosu na

konfiguraciju TTS-a u prethodnom slučaju, ali je potreban razvoj medijatora koji enkapsulira *e-mail* komunikaciju sa TTS-om.

4.10 FTTS - Primer korišćenja

U ovom poglavlju je opisan scenario koji prikazuje kako se korišćenjem FTTS-a i proširivanjem tipova tiketa koje domeni koriste može obezbediti podrška ključnih aktivnosti u rešavanju problema instance GÉANT Plus servisa[1]. Višedomenska usluga GEANTPlus omogućava uspostavljanje *point-to-point* putanja kapaciteta od 100Mbps do 10Gbps sa krajnjim tačkama u različitim domenima za potrebe projekata i istraživača u različitim domenima. Lokalni NREN-ovi i GÉANT su odgovorni za obezbeđivanje ovih putanja.



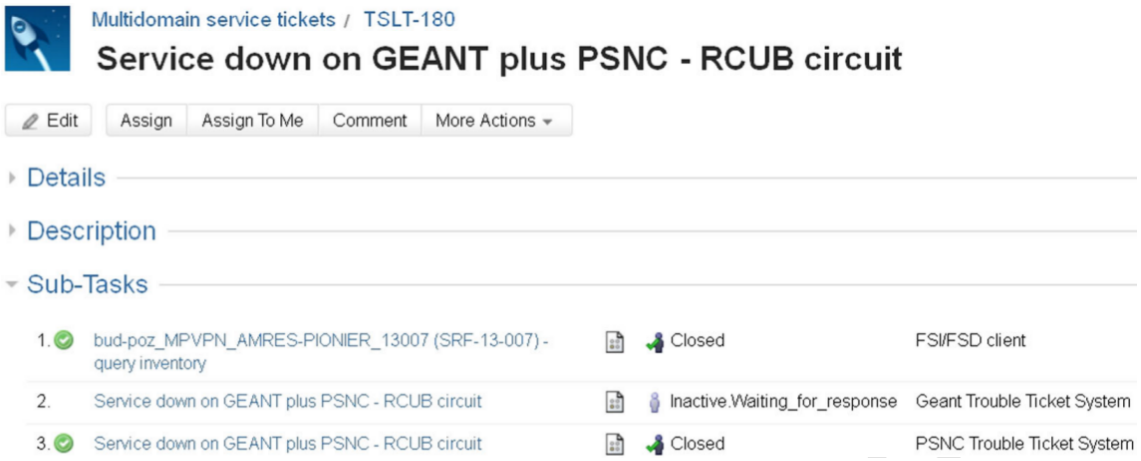
Slika 4.9 Kreiranje federativnog tiketa

GEANTPlus putanja pod imenom *bud-poz_MPVPN_AMRES-PIONIER_13007* i jedinstvenim identifikatorom *SRF-13-007* je uspostavljena između Računarskog centra Univerziteta u Beogradu, koji je u AMRES-u (NREN u Srbiji) i Poznan Super Computer Centre koji je u PIONIER-u (NREN u Poljskoj). Instanca usluge obuhvata tri domena: dva NREN-a i GÉANT između njih. Svaki domen koji detektuje

[1]http://www.geant.net/Services/ConnectivityServices/Pages/GEANTPlus.aspx

problem u vezi instance servisa treba da obavesti ostale domene koji učestvuju u obezbeđivanju putanje i da pokrene procedure za otkrivanje i rešavanje problema. Bez upotrebe FTTS-a, sledeće aktivnosti se izvršavaju "ručno": administratori traže skup domena koji učestvuju u instanci servisa i odgovarajuće kontakt podatke, obaveštavaju udaljene domene o otkrivenom problemu i traže odgovarajuće akcije, a zatim prate i organizuju dobijene odgovore i statuse rezolucije iz ostalih domena. Celokupna komunikacija između domena se tipično odvija putem *e-mail*-a ili telefona. FTTS automatizuje ove aktivnosti automatskim upitima FSI repozitorijuma i automatskim obaveštavanjem ostalih domena koji učestvuju u instanci servisa neposredno po kreiranju TT tiketa u domenu koji je prvi detektovao problem. FTTS takođe automatizuje evidenciju odgovora i statusa rezolucije u udaljenim domenima.

Na Slici 4.9 je prikazano kreiranje federatvinog TT-a koje nije komplikovanije od kreiranje tiketa za potrebe lokalnog domena. Jedine dodatne informacije koje treba uneti su *affected_service* koji se koristi kao jedinstveni identifikator instance servisa u FSI repozitorijumu i *ticket_type* kako bi se obezbedio potreban životni ciklus tiketa.



Slika 4.10 Praćenje životnog cikslusa federativnog tiketa

4.11 Zaključak

Jedan od glavnih zaključaka prikazane analize je da u labavo spregnutim federacijama postoji konflikt između autonomije domena i izvršavanja komandi iz udaljenih domena. Ukoliko domeni zadrže potpunu kontrolu nad svojim resursima, u opštem slučaju potpuna automatizacija operativnih aktivnosti pri upravljanju višedomenskim uslugama nije moguća jer se komande iz udaljenih domena koje teže da upravljaju resursima u datom domenu smatraju ugrožavanjem autonomije domena. Da bi se omogućio maksimalan stepen efikasnosti operativnih aktivnosti pri

upravljanju višedomenskim uslugama dizajniran je i realizovan federativni TTS koji omogućava automatizaciju međudomenske interakcije. FTTS eliminiše operatera iz aktivnosti koje se tiču interakcije između domena, pri čemu je ljudska aktivnost i dalje neizbežna u ključnim operativnim aktivnostima (Poglavlje 3).

Rigorozna evaluacija radnog procesa koji podrazumeva učešće operatera u pojedinim aktivnostima nije moguća [33] zbog nepostojanja metrika koji bi mogle ispravno opisati ljudsku efikasnost (na primer, uzeti u obzir i mentalnu opterećenost, umor i udobnost fizičkog okruženja). FTTS koji ima sve funkcionalnosti TTS sistema i omogućava automatsku razmenu ključnih poruka između domena nesuljivo čini proces pružanja višedomenske usluge efikasnijim i otpornijim na greške.

Glava 5

Automatizacija konfiguracije i aktivacije višedomenske usluge

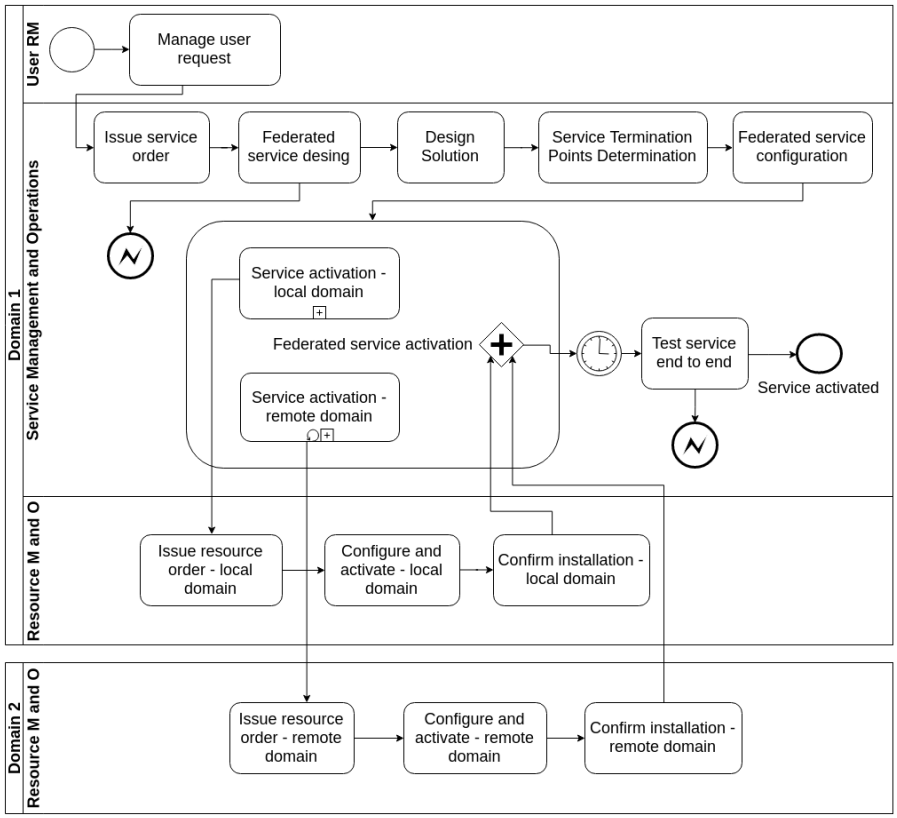
Višedomenske mrežne usluge nisu moguće bez spremnosti svih domena koji učestvuju u pružanju servisa da drugim domenima dozvole korišćenje dela svoje infrastrukture. Napredak tehnologija virtualizacije omogućava domenima da jednostavnije ponude svoje resurse federaciji domena u kojoj žele da učestvuju. Na taj način se u specijalnim slučajevima i za pojedine tipove višedomenskih mrežnih usluga može stvoriti okruženje u kome je moguće automatizovati upravljanje višedomenskim uslugama, pri čemu kontrola nad većinom infrastrukture domena i ključnim poslovnim procesima ostaje pod ingerencijom domena. U nastavku poglavlja će biti identifikovani ključni preduslovi koji moraju biti ispunjeni da bi u specijalnim slučajevima, kada je semantika usluge dobro definisana i dogovorena između domena, bilo moguće automatizovati proces uspostavljanja višedomenske mrežne usluge. U ovom poglavlju će biti opisan i decentralizovani distribuirani sistem koji omogućava domenima da formulišu i sprovedu ograničenja u vezi korišćenja svojih resursa, da oglase svoje sposobnosti drugim domenima, a da prihvate i izvrše samo zahteve iz udaljenih domena koji su u skladu sa postavljenim ograničenjima.

5.1 Dekomopozicija procesa konfiguracije i aktivacije instance višedomenske usluge

Kao što je već navedeno u Poglavlju 3.1, ukoliko domeni zadrže potpuni suverenitet nad svojim resursima, automatizacija, u opštem slučaju, nije moguća [128] jer postoji konflikt između autonomije domena i izvršavanja komandi iz udaljenih domena. Dekompozicijom i analizom procesa pružanja višedomenske usluge (Poglavlje 3.4) identifikovane su ključne poruke za upravljanje višedomenskim uslugama od čije

realizacije zavisi stepen automatizacije. Domeni, u opštem slučaju, ne dozvoljavaju izvršavanje komandi iz udaljenih domena jer se to kosi sa njihovom autonomijom i u Poglavlju 3.4 je pokazano da je ljudska aktivnost neophodna tokom razmene identifikovanih ključnih poruka, tj. za uspostavljanje instance usluge unutar granica domena u okviru procesa *Service Configuration Management* i *Service Activation Management*.

U slučajevima kada svi domeni koriste istu semantiku višedomenske usluge i kada mogu postaviti, oglasiti i primeniti granularna pravila, ograničenja i uslove (polise) pod kojima resurs može biti korišćen za pružanje višedomenske usluge, moguće je u potpunosti automatizovati uspostavljanje višedomenske mrežne usluge, to jest konfiguraciju i aktivaciju višedomenske usluge. Na Slici 5.1 je prikazan proces pružanja višedomenske usluge u opisanom specijalnom slučaju. Da bi se naglasile bitne aktivnosti, izostavljene su aktivnosti potrebne za obradu grešaka i na njihovim mestima je prikazan samo generički *Error event*.



Slika 5.1 Dekompozicija procesa konfiguracije i aktivacije instance višedomenske usluge

Instancu višedomenske usluge tipično traži krajnji korisnik, najčešće podnošenjem naloga za uspostavljanje usluge u svom matičnom domenu. Za dizajn instance višedomenske usluge *Federated Service Design (FSD)* procesu su potrebne ažurne

sposobnosti svih domena da učestvuju u pružanju instance višedomenske usluge. Sposobnosti domena FSD može utvrditi na dva načina:

- upitom svakog domena neposredno pre početka dizajna instance usluge
- upitom centralizovanog ili distribuiranog *Federated Service Inventory (FSI)* repozitorijuma koji sadrži oglašene sposobnosti domena

Bez obzira da li FSD dolazi do sposobnosti domena korišćenjem repozitorijuma ili tokom interakcije sa ostalim domenima, domeni moraju imati mogućnost da definišu, oglase i razmene svoje sposobnosti, tj. polise korišćenja resursa ponuđenih federaciji. FSD na osnovu dobijenih sposobnosti domena i parametara naloga za uspostavljanje servisa određuje topologiju višedomenskog servisa, konkretno skup domena i njihovih resursa koji će učestvovati u pružanju instance višedomenske mrežne usluge. Za razliku od opšteg slučaja opisanog u Poglavlju 3, *Design solution (DS)* proces u ovom specijalnom slučaju može dizajnirati instancu višedomenske usluge s kraja na kraj, a ne samo uputiti *requestServiceInN* zahteve domenima koji će učestvovati i time im delegirati dizajn instance servisa unutar njihovih granica. Da bi *Service Configuration Management* i *Service Activation Management* aplikacije mogle da upravljaju resursima u udaljenim domenima, pored specifikacije i oglašavanja polisa, domeni moraju podržati i automatizovano prihvatanje komandi iz udaljenih domena u skladu sa oglašenim polisama.

5.2 Sistem za automatizaciju uspostavljanja višedomenske usluge

Dekompozicija procesa konfiguracije i aktivacije instance višedomenske usluge (Poglavlje 5.1) pokazuje da za automatizaciju operativnih aktivnosti vezanih za pružanje višedomenske usluge u labavo spregnutom federativnom okruženju moraju biti ispunjena dva ključna preduslova:

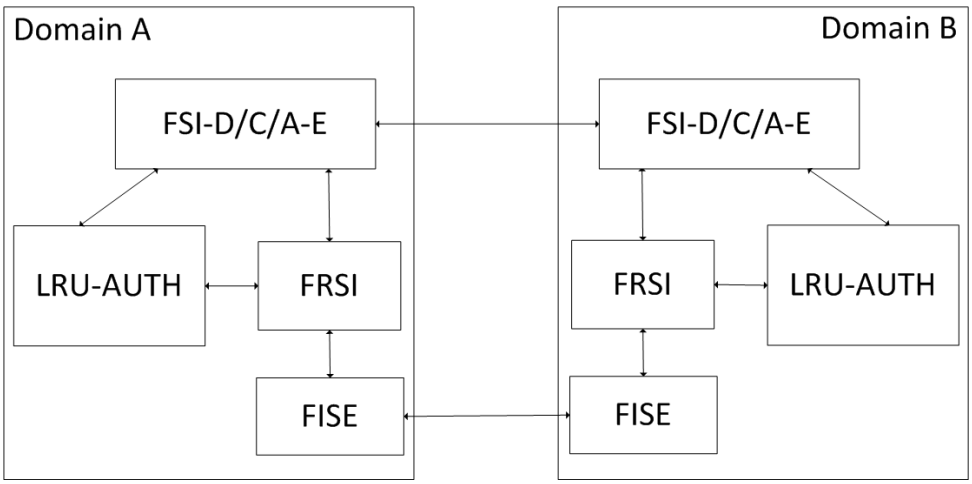
- Domeni moraju imati mogućnost da oglase i razmene svoje sposobnosti sa ostalim domenima. Ovo se može postići definisanjem protokola za razmenu sposobnosti domena koji omogućava domenima da objave skup svojih sposobnosti koje mogu ponuditi federaciji i skup ograničenja pod kojim će realizovati zahteve iz drugih domena.
- Domeni moraju imati mogućnost da definišu i primene pravila i ograničenja pod kojim drugi domeni mogu koristiti njihove resurse. Ovo se može obezbediti sistemom za kontrolu prava pristupa koji omogućava specifikaciju granularnih pravila autorizacije korišćenja resursa domena.

Ukoliko su ova dva preduslova ispunjena, u specijalnim slučajevima kada je semantika usluge jasna i precizno definisana, moguće je realizovati sistem koji omogućava automatizaciju uspostavljanje višedomenske usluge, bez učešća ljudi u lancu.

5.2.1 Arhitektura sistema za automatsko uspostavljanje višedomenskih usluga

Decentralizovani distribuirani sistem za kontrolu prava pristupa prikazan na Slici 5.2 omogućava svakom domenu labavo spregnute federacije (Slika 5.3) da samostalno i nezavisno od drugih domena definiše ograničenja u pogledu svojih resursa i da primi, prihvati i izvrši zahteve iz drugih domena strogo prema propisanim pravilima. Pored statički definisanih ograničenja, sistem uzima u obzir i uslove koje nameće nepredvidljiv kontekst operacije kao što su već rezervisani resursi. Sistem takođe omogućava razmenu sposobnosti domena, usaglašavanje podataka o topologiji federacije i koordinaciju operacija koje se tiču pružanja višedomenskih usluga. Svaki domen ima bar jednu instancu sistema.

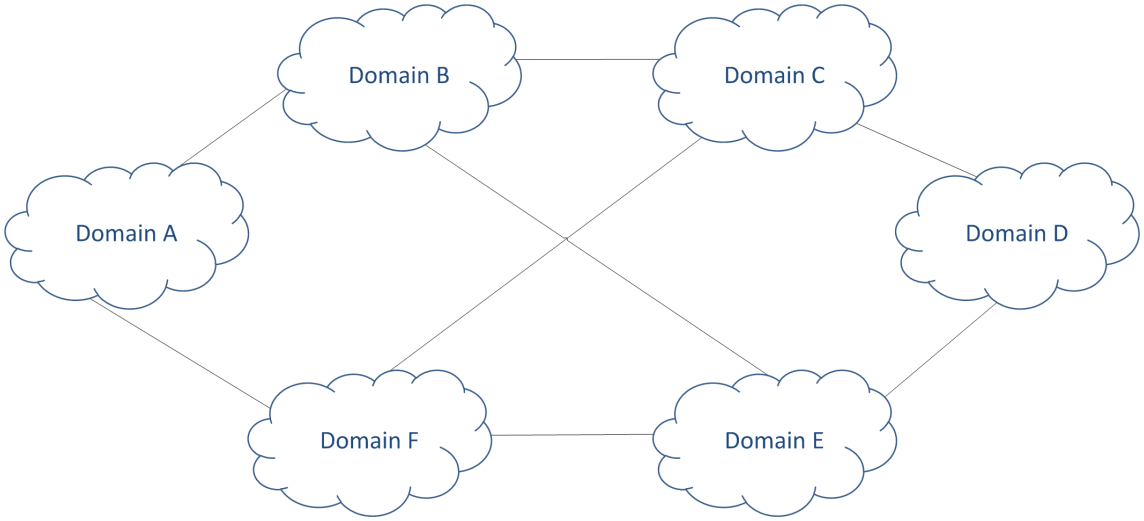
Ovakav koncept rada kod koga se formira topologija servisa na osnovu skupa ograničenja (*Constraint Based Routing*) je već realizovan u okviru *MPLS Traffic Engineering* tehnologije na nivou protokola rutiranja između mrežnih elemenata unutar jednog domena [61]. Predloženi sistem generalizuje ovaj koncept i realizuje ga na nivou servisa, čime se omogućava uz definiciju odgovarajuće semantike, kreiranje proizvoljnih servisa u višedomenskom okruženju, a ne samo virtuelnih putanja putem MPLS enkapsulacije. Nedavno je napravljeno proširenje koncepta rutiranja sa ograničenjima (BGP-LS [47]) kojim se omogućava iznošenje informacija o stanju linkova izvan domena čime se stvaraju preduslovi za stvaranje višedomenskih putanja, ali i ovo rešenje se ograničava samo na stvaranje virtuelnih kola, a ne i drugih kompleksnijih mrežnih usluga.



Slika 5.2 Arhitektura sistema za upravljanje udaljenim resursima

Glavne komponente sistema su: inventar resursa domena (*local resource inventory zaštićen podsistemom za autorizaciju - LRI-AUTH*) koji ostalim komponentama sistema ograničava pristup samo na resurse koji su ustupljeni federaciji, inventar višedomenskih usluga i resursa (*federated resource and service instance inventory - FRSI*), komponenta za dizajn, konfiguraciji i aktivaciju višedomenske usluge (*federated service instance design/configuration/activation element - FSI-D/C/A-E*) i komponenta koja oglašava sposobnosti matičnog i održava lokalnu bazu oglašenih sposobnosti ostalih domena (*federated inventory synchronization element - FISE*).

Domeni regularno razmenjuju podatke o svojim sposobnostima, što omogućava da svi domeni imaju konzistentne podatke o topologiji federacije. To znači da FRSI komponenta svakog domena pored statičkih ograničenja koje su domeni specificirali, sadrži i podatke o svim instancama međudomenskih usluga i rezervacijama resursa ustupljenih federaciji sa odgovarajućim nivoom detalja. Domeni koji učestvuju u pružanju instance usluge moraju imati sve podatke potrebne za pružanje usluge u okviru lokalnog domena, dok je ostalim domenima dovoljno da čuvaju samo generalne informacije o instanci višedomenske usluge, kao što su uključeni domeni, rezervisani resursi i vrsta usluge.



Slika 5.3 Labavo spregnuta federacija

Kada krajnji korisnik zatraži novu instancu višedomenske usluge, FSI-D/C/A-E radi upit lokalne FRSI komponente da bi dobio skraćenu topologiju međudomenskih veza na osnovu koje će dizajnirati instancu višedomenske usluge. FSI-D/C/A-E je takođe zadužen za koordinaciju domena koji će učestvovati u novoj instanci usluge tokom njene konfiguracije i aktivacije. Domeni koji učestvuju u pružanju novokreirane instance višedomenske usluge ažuriraju svoje sposobnosti tokom konfiguracije i aktivacije usluge. Pošto se sposobnosti domena redovno razmenjuju, nakon određenog vremena će svi domeni ažurirati podatke u svojim FRSI komponentama i mreža će ponovo konvergirati.

5.2.2 Federated resource and service instance inventory (FRSI)

Federated resource and service instance inventory (FRSI) vodi evidenciju o svim resursima ustupljenim federaciji (*federativnim resursima*) zajedno sa ograničenjima u pogledu njihovog korišćenja, uspostavljenim instancama višedomenskih usluga i federativnim resursima dodeljenim svakoj instanci višedomenske usluge. Za usluge koje domen pruža zajedno sa drugim domenima, FRSI takođe čuva relevantne podatke o konfiguraciji lokalnih resursa. FRSI svih domena koriste isti (zajednički) model podataka nametnut vrstama usluga koje federacija pruža i dele semantiku modela podataka.

FRSI je zadužen i za filtriranje federativnih resursa dostupnih novim instancama usluga na osnovu pravila postavljenih od strane vlasnika resursa i rezervacija resursa.

5.2.3 Local resource inventory (LRI-AUTH)

Local resource inventory zaštićen podsistemom za autorizaciju (LRI-AUTH) je repozitorijum lokalnih resursa domena. Ova komponenta filtrira resurse i pravila za njihovo korišćenje u okviru federacije lokalnoj FRSI komponenti i čuva sve detalje potrebne za konfiguraciju lokalnih resursa dodeljenih federaciji.

5.2.4 Federated inventory synchronization element (FISE)

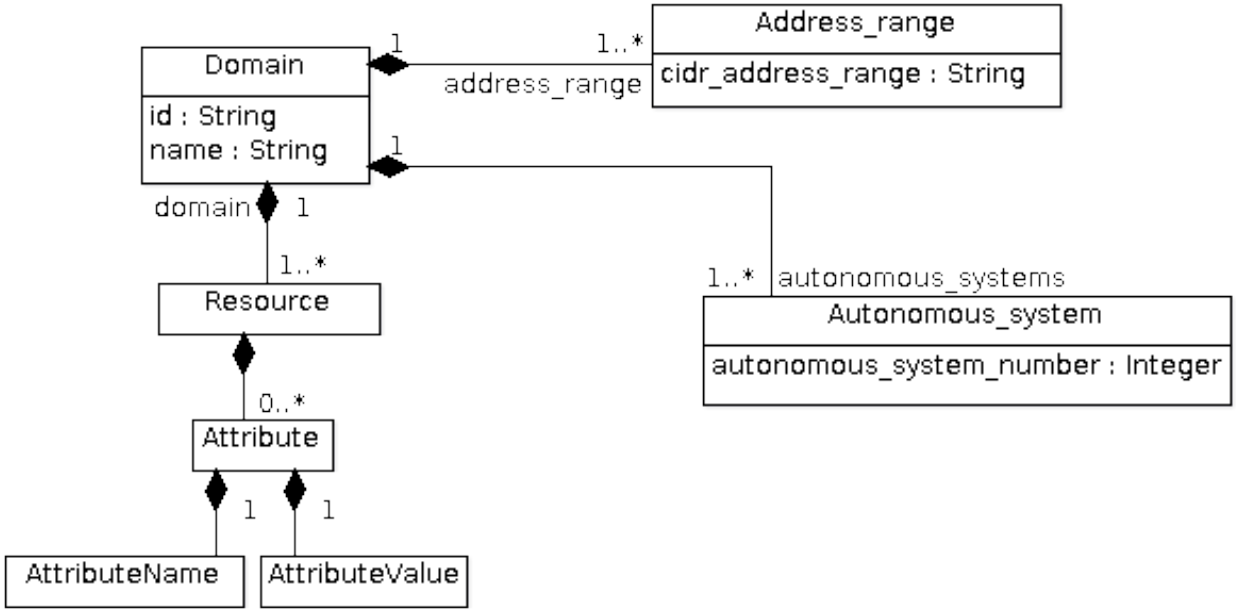
Federated inventory synchronization element (FISE) obezbeđuje *East/West* komunikaciju i sinhronizaciju podataka koji se čuvaju u lokalnom FRSI repozitorijumu. FISE periodično radi upit ostalih domena radi ažuriranja podataka u FRSI komponenti lokalnog domena.

5.2.5 Federated service instance design/configuration/activation element (FSI-D/C/A-E)

Krajnji korisnik koristi *Federated service instance design/configuration/activation element (FSI-D/C/A-E)* da bi tražio novu instancu multi-domen servisa. FSI-D/C/A-E zatim pristupa lokalnoj FRSI komponenti kako bi dobio listu svih federativnih resursa koji, shodno sposobnostima matičnih domena, aktuelnim rezervacijama federativnih resursa i parametrima iz naloga, mogu biti upotrebljeni za dizajn nove instance servisa. Na osnovu prikupljenih podataka i parametara FSI-D/C/A-E dizajnira novu instancu usluge. Ukoliko tražena instanca usluge može biti uspostavljena, FSI-D/C/A-E inicira njeno kreiranje slanjem konfiguracionih zahteva svim udaljenim domenima koji će učestvovati u pružanju nove usluge i izvršava potrebne promene konfiguracije resursa u lokalnom domenu. Udaljeni domeni konfigurišu svoje resurse prema primljenom konfiguracionom zahtevu kako bi omogućili pružanje usluge u okviru svojih granica, ažuriraju podatke u svojim FRSI komponentama i vraćaju notifikaciju o statusu izvršenja zahteva. Pošto primi potvrde iz svih domena određenih da učestvuju u novoj instanci usluge, FSI-D/C/A-E obaveštava korisnika da je instanca usluge dostupna. Domeni koji nisu bili uključeni u uspostavljanje nove instance višedomenske usluge će ažurirati podatke u FRSI komponentama tokom sledećeg ciklusa sinhronizacije i svi domeni će ponovo imati konzistentnu sliku topologije federacije. Ukoliko neki od domena pošalje negativnu notifikaciju ili promene konfiguracije u lokalnom domenu nisu moguće, FSI-D/C/A-E povlači zahtev, traži od ostalih domena da ponište eventualne promene konfiguracije i poništava promene konfiguracije u lokalnom domenu i promene podataka u FRSI i LRI-AUTH komponentama. Ostali domeni takođe vraćaju prethodnu konfiguraciju i

nakon izvesnog vremena mreža će ponovo konvergirati i FRSI komponente u svim domenima će ponovo imati podatke o istim entitetima.

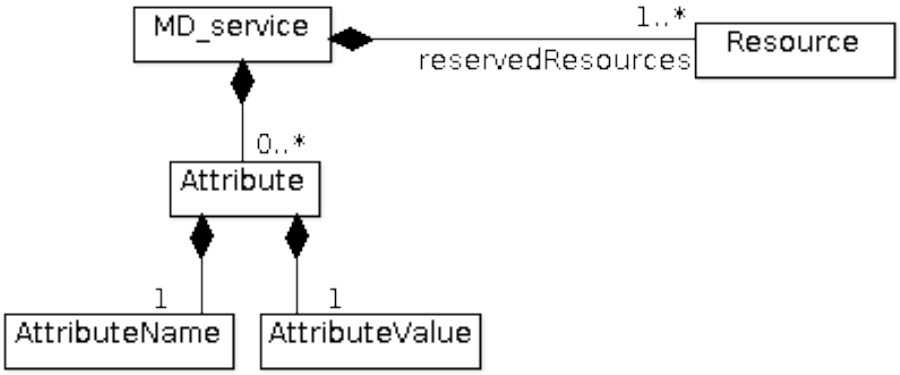
5.3 Minimalni model podataka višedomenske usluge, resursa i domena



Slika 5.4 Minimalni model podataka ključnih apstrakcija sistema

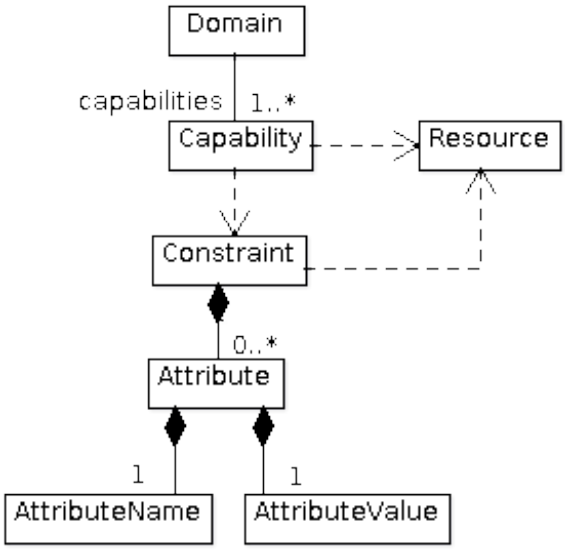
Ključne apstrakcije sistema domen i resurs ustupljen federaciji su modelovane kao *Domain* i *Resource*, respektivno (Slika 5.4).

Domain je određen jedinstvenin identifikatorom *id* i nazivom *name*. Svaki domen ima jedinstven adresni prostor (*Address_range*), autonomne sisteme (kompoziciju entiteta *Autonomous_system*) i resurse koji mogu biti korišćeni za pružanje višedomenskih usluga. Svaka instanca višedomenske usluge *MD_service* koristi resurse koje su Domain ponudili federaciji, što je prikazano na Slici 5.5.



Slika 5.5 Minimalni model podataka višedomenske usluge

Domain može specificirati ograničenja na nivou pojedinačnog federativnog resursa Resource ili grupe federativnih resursa. Sposobnosti koje Domain oglašava (*Capability*) zavise od resursa ponuđenih federaciji i ograničenja koje je Domain specificirao (*Constraint*), što je prikazano na Slici 5.6.



Slika 5.6 Minimalni model podataka sposobnosti domena

Za opisivanje apstrakcija Resource, Constraint i MD_service mogu biti potrebni različiti atributi čiji su nazivi, vrednosti i semantika određeni tipovima višedomenskih usluga koje federacija nudi. U modelu podataka je to predstavljeno tako što su te apstrakcije kompozicije atributa (*Attribute*) koji se sastoje od naziva atributa (*AttributeName*) i vrednosti atributa (*AttributeValue*). Formalnije, resurs koji može biti korišćen za pružanje višedomenskih usluga i instance višedomenskih usluga se mogu modelovati uređenim parovima koji se sastoje od atributa i vrednosti atributa: $\{\langle a, v_a \rangle \mid v_a \in V_a\}$. Svaki atribut a može imati jednu ili više vrednosti iz unapred definisanog skupa vrednosti V_a. Atributi i skupovi njihovih mogućih vrednosti

su određeni tipovima višedomenskih usluga. Svaka instanca višedomenske usluge koristi skup resursa što se može predstaviti preslikavanjem skupa instanci višedomenskih usluga u skup korišćenih resursa $f : MD\_service \rightarrow \{Resource_j^{Domain_i} \mid i = 1..N,\ j = 1..M(N)\}$, gde je $Resource^{Domain}$ resurs koji pripada domenu $Domain$. Pravila koja limitiraju korišćenje resursa u federativnom okruženju se, takođe, specificiraju u obliku ograničenja vrednosti koje mogu imati atributi koji opisuju resurse.

5.4 Mehanizam oglašavanja i razmene sposobnosti domena

Mehanizam koji domeni koriste za razmenu sposobnosti da učestvuju u višedomenskim uslugama zavisi od vrsta usluga koje federacija pruža. Domen može razmenjivati sposobnosti samo sa svojim susedima ili sa svim domenima federacije. Sposobnosti se mogu oglašavati u regularnim vremenskim intervalima i/ili pri promeni sposobnosti domena. Na primer, *Border Gateway Protocol (BGP)* [99] razmenjuje rute sa susedima na početku, a posle samo razliku kada se desi promena ruta.

Glava 6

Sistem za kontrolu prava pristupa federativnim resursima

Jedan od ključnih elemenata arhitekture - FRSI, opisan u Poglavlju 5.2.2, služi za filtriranje federativnih resursa dostupnih novim instancama usluga na osnovu pravila postavljenih od strane vlasnika resursa i rezervacija resursa. Pored filtriranja, FRSI mora da obezbedi kontrolu prava pristupa federativnim resursima kako bi se sprečilo neautorizovano korišćenje podataka, resursa ili servisa domena. Sistem za kontrolu prava pristupa je esencijalni deo svakog sistema i mehanizmi, njihovi modeli, platforme i tehnologije koje se koriste da bi se primenilo željeno ponašanje sistema se veoma razlikuju. U nastavku poglavlja će biti prikazana komparativna analiza postojećih koncepata kontrole prava pristupa, modela sistema za kontrolu prava pristupa i biće detaljno objašnjena *Policy-Based Management* paradigma koja je korišćena za realizaciju *Environmental Context-Aware Policy-Based Access Control Framework (ECAPBACF)* platforme za kontrolu prava pristupa federativnim resursima.

Environmental Context-Aware Policy-Based Access Control Framework (ECAPBACF) je originalna *policy-based management* platforma za kontrolu prava pristupa federativnim resursima. U literaturi postoje rešenja koja omogućavaju delegaciju prava pristupa [83] [16] [23] u višedomenskom okruženju, ali prema dosadašnjem saznanju, ECAPBACF je jedini *framework* za kontrolu prava pristupa dizajniran za upravljanje federativnim resursima i uslugama koji koristi opšti model federativnog resursa i usluge, a pored statički definisanih ograničenja uzima u obzir i kontekst operacije, tj. stanje okruženja. ECAPBACF za organizaciju pravila za kontrolu prava pristupa koristi *Attribute Based Access Control (ABAC) Model* što omogućava korišćenje meta-modela pri specifikaciji polisa. Za razliku od jezika i radnih okvira opisanih u Poglavlju 6.2, ECAPBACF omogućava dobijanje autorizacije (eksplicitne dozvole ili zabrane korišćenja) za sve federativne resurse jednim upitom,

čime je, implicitno određena i skraćena topologija međudomenskih veza. ECAPBACF je formalno opisan korišćenjem *Fine Grained Integration Algebre (FIA)* [97] i može se implementirati na proizvoljnom jeziku za opis polisa. Za razliku od drugih sistema zasnovanih na FIA algebri koji predlažu heuristike i optimizacije za poboljšanje performansi algoritma za određivanje integrisane polise koji inherentno ima eksponencijalnu složenost [101] [41] [42] [5] [6], ECAPBACF polise su dizajnirane tako da se implicitno izbegnu kompleksni algoritmi potrebni za određivanje integrisane polise i obezbedi linearna vremenska složenost algoritma za određivanje integrisane polise *O(n)*, što ga čini primenljivim u realnim okruženjima.

U Poglavlju 6.4.3 je prikazana implementacija ECAPBACF *framework*-a korišćenjem *eXtensible Access Control Markup Language Version 3.0 (XACMLv3)* [94] jezika za specifikaciju ABAC polisa, a u Poglavlju 7 je opisan ECAPBACF prototip za automatizaciju konfiguracije i aktivacije višedomenske putanje garantovanog kapaciteta. Pošto polise u prototipu koriste atribute određene tipom višodomenske usluge, prototip ima PBAC model organizacije prava pristupa, koji je specijalan slučaj ABAC modela u kom je specificirana semantika korišćenih atributa.

6.1 Komparativna analiza modela sistema za kontrolu prava pristupa

Svaki sistem za kontrolu prava pristupa mora, u većoj ili manjoj meri, obezbediti sledeća četiri aspekta sigurnosti:

- Identifikacija (eng. *Identification*) je dodela prava (privilegija) za izvršavanje akcija korisniku, na primer, otvaranje korisničkog naloga. Korisnik (eng. *user*) može biti osoba, računar, ruter, proces...
- Autentikacija (eng. *Authentication*) je provera da li korisnik ima pravo da koristi identitet zajedno sa pravima pridruženim identitetu.
- Autorizacija (eng. *Authorization*) je izražavanje odluke o pristupu kroz eksplicitnu dozvolu ili zabranu entititetu da izvrši akciju.
- Odluka o pristupu (eng. *Access decision*) je odlučivanje da li dozvoliti ili zabraniti entitetu da izvrši akciju.

6.1.1 Modeli kontrole prava pristupa

Logička organizacija pravila koja definišu ponašanje sistema za upravljanje kontrolom pristupa se naziva *model kontrole prava pristupa* (eng. *access control model (ACM)*).

Modeli kontrole prava pristupa se mogu klasifikovati prema različitim kriterijumima [102], [60], [17], [37], u zavisnosti od nivoa granularnosti sa kojim su podržani pojedini aspekti kontrole pristupa. ACM se može svrstati u različite klase u zavisnosti od konteksta klasifikacije. Na primer, svaki ACM se može implementirati korišćenjem *Attribute Based Access Control (ABAC)* modela odgovarajućim izborom atributa i dodelom adekvatnog značenja (semantike) atributima [66], [59], [102]. U ovom poglavlju će biti opisani samo modeli kontrole prava pristupa koji se pretežno koriste u upravljanju mrežama i distribiranim sistemima.

Naredni termini se u nastavku glave koriste sa sledećim značenjem:

- Subjekat (eng. *subject*) je aktivan entitet kome su pridružena prava (privilegije) da izvršava *akcije* nad *objektima*.
- Objekat (eng. *object*) je pasivan entitet, na primer resurs ili servis, nad kojim se izvršava *akcija*.
- Akcija (eng. *action*) je operacija *subjekta* nad *objektom*.
- Okruženje (eng. *environment*) je skup podataka o kontekstu (eng. *context*) akcije koji se mogu koristiti pri donošenju odluke o pristupu.

6.1.1.1 *Mandatory Access Control (MAC) model i Discretionary Access Control (DAC) model*

Mandatory Access Control (MAC) model i *Discretionary Access Control (DAC) model* spadaju u najstarije modele za kontrolu prava pristupa. U *Mandatory Access Control (MAC) modelu* su pravila vezana za akcije nad objektima bazirana na osetljivosti objekta i privilegija dodeljenih subjektu. Da bi se obezbedila hijerarhijska kontrola prava pristupa, svakom subjektu i svakom objektu se dodeljuje nivo bezbednosti (eng. *sercurity level*), a pravila na osnovu kojih se donosi odluka o pristupu propisuje centralni autoritet (eng. *central authority*) [35], [17].

U *Discretionary Access Control (DAC) modelu* svaki objekat pripada određenom subjektu ili grupi subjekata. DAC uvodi koncept posedovanja objekta (eng. *object ownership*) i subjekta koji ima disckreciono pravo da pristup objektu delegira drugim subjektima [17], [35].

6.1.1.2 *Access Control Lists (ACL)*

Access Control List (ACL) je najstariji model za kontrolu prava pristupa. To je DAC model [17] zasnovan na mapiranju subjekta u set akcija koje može izvršiti nad pojedinim objektima, a svakom objektu su pridružena mapiranja za sve subjekte.

Access control list je jedan od tri načina predstavljanja odnosa između subjekata, objekata i akcija. Taj način je određen matricom kontrole prava pristupa (eng. *access control matrix*) [77] Lampsonovog modela [68]. Access control liste je lako implementirati i koriste se u većini modernih operativnih sistema [102].

6.1.1.3 *Role Based Access Control (RBAC) Model*

Ključni koncept *Role Based Access Control (RBAC)* modela je *uloga* (eng. *role*) koja definiše odnos grupe subjekata i skupa dozvola za izvršavanje akcija. Svakom subjektu je dodeljen niz uloga koje određuju dozvole za izvšavanje akcija na objektima. Uloge obično odražavaju odnos između subjekta i organizacije u kojoj se koristi sistem za kontrolu prava pristupa. Da bi se izbegli previše "moćni" subjekti, subjektu treba dodeliti najmanje privilegije potrebne za određene aktivnosti (eng. *principle of least privilege*).

RBAC može se smatrati alternativom MAC i DAC modela [35]. RBAC olakšava upravljanje pravima pristupa na osnovu grupisanja subjekata prema ulogama, ali u velikim sistemima koji obuhvataju više odeljenja ili organizacija, usaglašavanje privilegija koje treba dodeliti ulogama može biti komplikovano [60] [102]. Postoji više proširenja i prilagođavanja RBAC modela kao što su *Organization Based Access Control (ORBAC) model* [35] i *Task-based Authorization Control (TBAC) model* [9] [114].

6.1.1.4 *Attribute Based Access Control (ABAC) Model*

U *Attribute Based Access Control (ABAC) modelu* odluka da li subjekat može izvršiti akciju nad objektom se zasniva na atributima subjekta, okruženja i objekta. Svaki atribut je karakteristika subjekta, objekta, okruženja ili akcije. Odluka o pristupu da li dozvoliti ili zabraniti akciju nad objektom se donosi poređenjem vrednosti atributa koji se zahteva sa vredonstima atributa definisanim u pravilima pristupa.

U poređenju sa ACL i RBAC modelima, najvažnija prednost ABAC modela je što subjekat ne mora biti unapred poznat, ali problem sa harmonizacijom i usaglašavanjem semantike atributa je ostao.

6.1.1.5 *Policy Bassed Access Control (PBAC) Model*

Policy-Bassed Access Control (PBAC) model razdvaja pravila koja definišu ponašanje sistema od funkcionalnosti sistema [11]. PBAC model se može smatrati standardizacijom atributa koji se koriste u *Attribute Based Access Control modelu* kako bi se ostvarili željeni ciljevi u upravljanju kontrolom pristupa [102]. Pravilo koje definiše ponašanje sistema se naziva polisa (eng. *policy*).

6.1.1.6 *Risk-aware access control (RAAC) model*

Osnovna ideja *Risk-aware access control (RAAC) modela* je uključivanje procene rizika u odluku o pristupu. Procena rizika mora da sadrži analizu rizika koji će nastati ako se akcija odobri, što uključuje mogućnost otkrivanja ili modifikacije podataka, odnosno ako se akcija zabrani, što može sprečiti ostvarenje poslovnih ciljeva. [25] predlaže implementaciju RAAC modela korišćenjem XACML jezika za specifikaciju polisa [94] koji [46] proširuje u *Risk-Aware Group Based Access Control (RA-GBAC).*

6.1.1.7 *Risk Adaptive Access Control (RAdAC) Model*

Risk Adaptive Access Control (RAdAC) Model proširuje navedene modele kontrole pristupa tako što uvodi koncept rizika i situacionih faktora u određivanju odluke o pristupu. Pored tradicionalnog oslanjanja na statičke atribute i pravila, RAdAC uključuje i faktore rizika i set heuristika u izražavanje merljivih metrika rizika. Nakon utvrđivanja nivoa rizika, primenjuje se polisa za odgovarajući nivo [102] [59]. Na primer, u zavisnosti od procenjenog nivoa rizika, autentikacija se radi na osnovu korisničkog imena i lozinke, odnosno digitalnog sertifikata.

6.1.1.8 *authoriZation Based Access Control (ZBAC) Model*

Svi navedeni modeli kontrole pristupa imaju zajedničku ranjivost kada se koriste u višedomenskom okruženju. Udaljeni domeni donose odluke o pristupu prema autentikaciji subjekta u njegovom domenu u vreme podnošenja zahteva u udaljenom domenu. Pored već navedenih problema organizacione i administrativne prirode oko standardizacije i sinhronizacije atributa i polisa, takav pristup krši (eng.) *principle of least privilege* i čini sistem ranjivim na (eng.) *confused deputy*[1] *attack* [50]. ZBAC prevazilazi ovaj problem koristeći autorizacije (donete odluke o eksplicitnoj dozvoli ili zabrani) kreirane pre nego što je subjekat zatražio akciju nad objektom. Naime, autentikacija subjekta u matičnom domenu rezultuje ovlašćenjima koja mogu biti predstavljena kao kriptovani kredencijali ili *SAML*[2] *assertions.* Ove autorizacije se mogu prosleđivati uz zahtev ili na neki drugi način. U svakom slučaju, odluka o pristupu je određena pre nego što je akcija tražena. Iako je ZBAC razvijen kao rešenje bezbednosnih problema za *SOA*[3] *framework*-e, primena ZBAC-a u postojećim sistemima i modelima ne zahteva velike promene [60].

[1] *Confused deputy* problem je prvi put opisan na primeru *mainframe timesharing* računara u kom kompajler ima veća ovlašćenja od korisnika. Korisnik je mogao promeniti datoteke kojima nema pristup ako pri pokretanju kompajlera kao izlaznu datoteku navede datoteku kojoj nema pristup. Primer *confused deputy* problema u Web okruženju je *Cross-site request forgery.*

[2] https://wiki.oasis-open.org/security/FrontPage

[3] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm

6.1.1.9 *Break-the-Glass (BtG) princip*

Break-the-Glass (BtG), odnosno *break-glass*, princip [32] [87] je pristup u kontroli prava pristupa koji definiše uslove, pravila i mehanizme kako i kada subjekat može dobiti veći nivo privilegija nego što ga ima pod normalnim okolnostima. BtG princip omogućava fleksibilnu kontrolu prava pristupa i *Break-the-Glass* tehnike se obično dodaju postojećim sistemima za kontrolu prava pristupa da bi se obezbedio hitan pristup (eng. *emergency access*) podacima u situacijama kada je to opravdano, na primeru u slučaju hitne medicinske pomoći kada je život pacijenta ugrožen. Primena BtG principa je najčešće bazirana na predefinisanim korisničkim nalozima koji se mogu koristiti u određenim situacijama. *Break-the-Glass Role-Based Access Control (BTG-RBAC)* [40] je predlog proširenja RBAC modela BtG principom.

6.1.2 Modeli kontrole prava pristupa - zaključak

Svaki od opisanih modela kontrole prava pristupa je pogodan za pojedine vrste sistema. Na primer, ACL model se još uvek koristi za kontrolu prava pristupa u operativnim sistemima, u organizacijama se tipično koristi neka varijanta RBAC modela ili neki od modela koji pri donošenju odluke o pristupu uzima u obzir i faktor rizika, dok je primena BtG principa česta u zdravstvenim sistemima. Uporedna analiza modela kontrole prava pristupa je pokazala da je ABAC model najopštiji model kontrole prava pristupa i da se adekvatnim izborom semantike atributa svi ostali modeli mogu implementirati korišćenjem ABAC modela.

Za logičku organizaciju pravila, uslova i ograničenja ECAPBACF platforme je korišćen ABAC model kontrole prava pristupa da bi se omogućila specifikacija polisa korišćenjem meta-modela federativnog resursa i višedomenske usluge koji su prikazani u Poglavlju 5.3 i time izbeglo sužavanje oblasti primene na pojedine tipove usluga. U realizovanom prototipu za automatsku konfiguraciju i aktivaciju višedomenske putanje garantovanog kapaciteta (Poglavlje 7) u kom su atributi određeni jer su poznati i tip i semantika višedomenske usluge koji federacija pruža, primenjen je PBAC model koji standardizuje i harmonizuje semantiku atributa ABAC modela.

6.2 *Policy-based management* paradigma

Policy-based management (PBM) paradigma se veoma dugo koristi za upravljanje mrežama jer pojednostavljuje upravljanje mrežom i olakšava automatizaciju [7] [19] [126]. S jedne strane, definicija i organizacija sigurnosnih polisa radi sprečavanja neovlašćenog pristupa podacima, resursima i uslugama je od ključnog značaja

za svaki sistem, dok, s druge strane, polise tretirane kao pravila koja određuju izbore u ponašanju sistema [110] obezbeđuju modus za upravljanje kompleksnim distribuiranim sistemima.

Formulacije polise variraju počev od najjednostavnijih *"IF condition THEN action"* preko kompleksnijih *"ON event THEN IF condition THEN action"* do polisa koje specificiraju *"subject-target-action-condition"* odnose [11], gde je *subject* "entitet ili skup entiteta od kojih potiče zahtev i koji su verifikovani kao ovlašćen/nije ovlašćen da obavi traženu akciju" [129] dok je *target* "entitet, ili skup entiteta, na koje se polisa odnosi" [129].

Prema RFC dokumentima koji definišu *policy-based management* terminologiju [129], [86], polisa je "definisan cilj, pravac ili metod delovanja koji vodi i određuje sadašnje i buduće odluke koje treba da budu izvršene u određenom kontekstu" [129], odnosno "skup pravila koji upravlja i kontroliše pristup mrežnim resursima" [86] i sadrži skup pravila koja povezuju niz akcija sa skupom uslova koje moraju biti zadovoljeni da bi akcije bile izvršene [129] [86]. Akcija, odnosno "akcija polise", je "definicija šta treba da se uradi da bi se sprovelo pravilo polise kada su ispunjeni uslovi pravila". Akcija može rezultovati aktivnostima nad *target*-om izvršenim od strane subjekta [129] [86]. Uslov, takođe nazivan i "uslov polise", "predstavlja neophodna stanja i/ili preduslove koji definišu da li akcija polise treba da bude izvršena" [129]. Uslovi koji su nametnuti nepredvidljivim kontekstom akcije i koji su nezavisni od resursa, akcija ili subjekata se nazivaju kontekst (eng. *context*) ili okruženje (eng. *environment*) [94] [12]. Značaj konteksta (stanja okruženja) i nivoa rizika je posebno prepoznat u pojedinim oblastima primene [77] [6] [122] što je rezultovalo razvojem tehnika za specifikaciju dinamičkih adaptivnih polisa, bilo kao novih *framework*-a [120] [125] [12], modela kontrole prava pristupa razvijenih oko koncepta rizika [25] [46] [78] [59] ili proširenja postojećih sistema [35] [114] [9] [60].

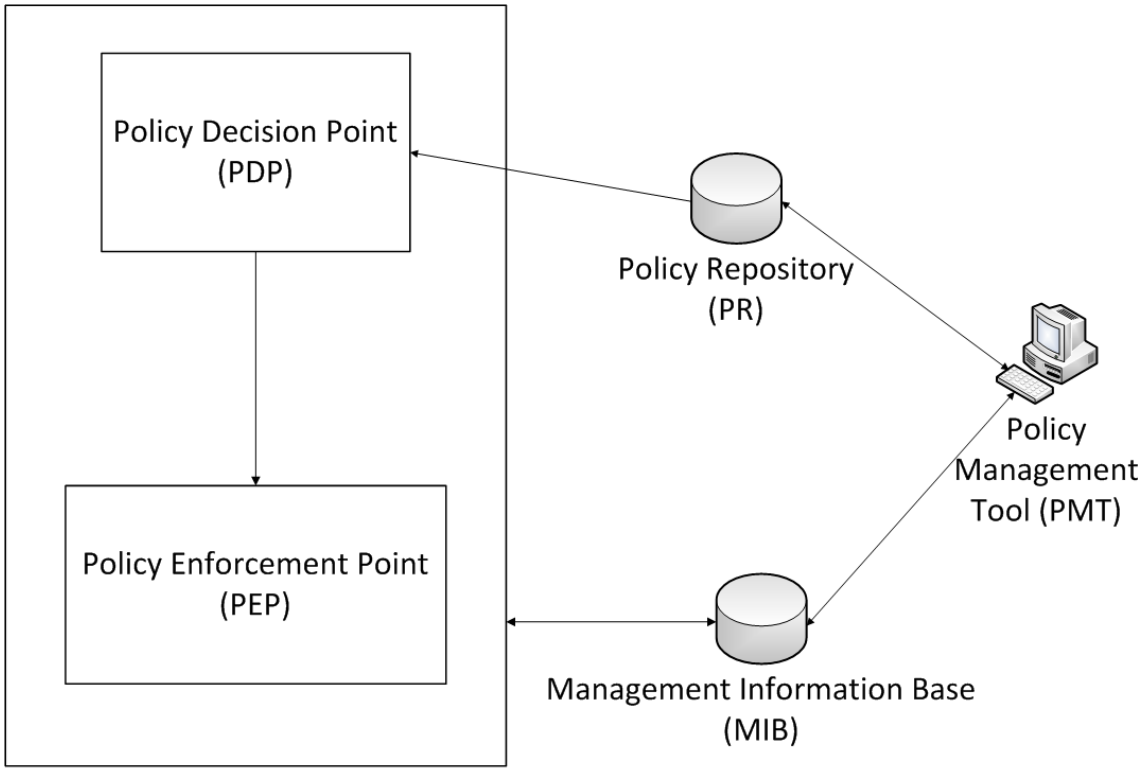
6.2.1 Arhitektura *policy-based management* sistema

Iako *policy-based management* sistemi specificiraju, organizuju i koriste polise na različite načine, većina je zasnovana na *policy-based management* arhitekturi razvijenoj od strane IETF Policy Framework (POLICY) Working Group i koja je prikazana na Slici 6.1 [7] [11].

Prema RFC3198 [129] glavne komponente sistema su:

- Policy Management Tool je komponenta koja se koristi da bi se definisale, ažurirale, testirale, analizirale i instalirale polise u komponenti *Policy Repository*;

- Policy Repository služi kao "skladište podataka koje čuva pravila polisa, njihove uslove i akcije i neophodne podatke o polisama";
- Policy Decision Point (PDP) komponenta je zadužena za donošenje odluka o pristupu, bilo za sebe ili za druge entitete koji su tražili odluku o pristupu [129];
- Policy Enforcement Point (PEP) je komponenta koja izvršava odluku o pristupu;
- Management Information Base (MIB) komponenta je opciona i koristi se za čuvanje podataka o resursima na koje se odnose polise; tipično se popunjava *discovery* servisom [7].



Slika 6.1 Tipična arhitektura policy-based management sistema

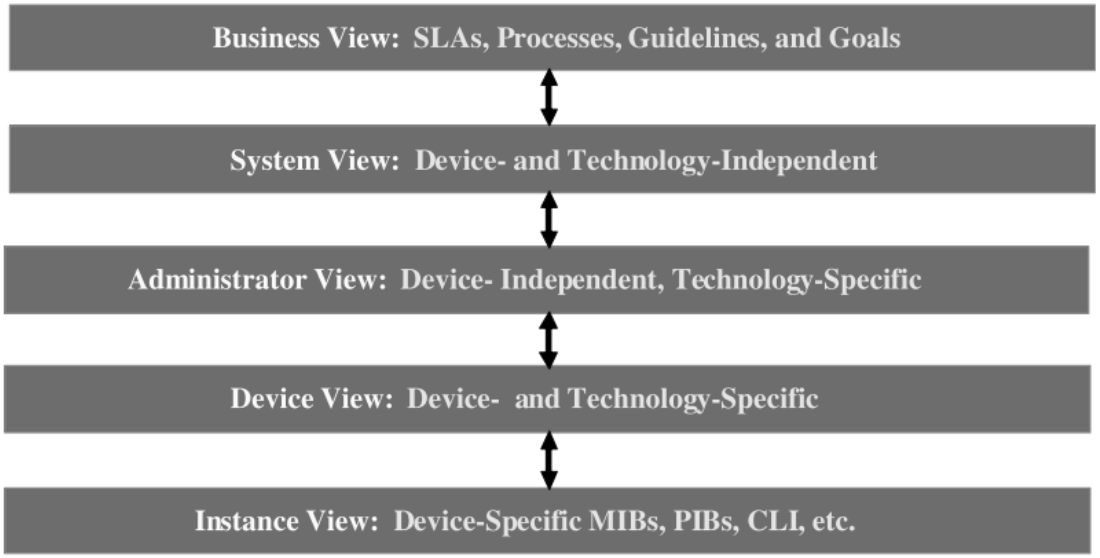
IETF Policy Framework sadrži Common Information Model (CIM) [7] sa ciljem da pruži zajedničku opštu definiciju resursa za koje se definišu pravila kontrole pristupa. CIM dozvoljava proširenja (eng. *vendor specific*)[4].

6.2.2 Kontinuum polisa (*Policy continuum*)

Složeni sistemi imaju mnogo polisa koje kreiraju i održavaju različite osobe. Pored organizovanja polisa u skupove polisa (eng. *Policy Set*) [94] ili hijerarhijski [29],

[4]http://www.dmtf.org/standards/cim

polisama se mogu dodeliti različiti novoi apstrakcije primenom koncepta *kontinuuma polisa* (eng. *policy continuum)* [30]. Kontinuum polisa se sastoji od konzistentnih pogleda vezanih u koherentnu celinu zajedničkim modela podataka koji takođe određuje odnose polisa na različitim nivoima. Svaki pogled je optimizovan za svoj nivo apstrakcije i koristi odgovarajuću terminologiju i jezike i *framework*-e za opisivanje polisa. Na primer, DEN-ng policy continuum ima pet nivoa kao što je prikazano na Slici 6.2.



Slika 6.2 DEN-ng policy continuum [30]

Na svakom nivou kontinuuma polisa polisa može postojati nezavisno ili biti izvedena (rafinisana (eng. *policy refinement*)) od polisa na gornjim nivoima. Polisa takođe može biti rafinisana ni u jednu, jednu ili više polisa na nižim slojevima. *Policy refiniment* mora zadržati ciljeve polisa definisanih na višim nivoima. Da bi se izbegao konflikt polisa, odnosi i zavisnosti između polisa na različitim nivoima kontinuuma se moraju uzeti u obzir prilikom ažuriranja polisa, jer modifikacija polisa na jednom nivou može zahtevati izmenu polisa i na drugim nivoima.

6.2.3 *Policy-based management* pristupi

U narednim poglavljima su navedeni pravci *policy-based management*-a koji se pretežno koriste za upravljanje mrežama i distribuiranim računarskim sistemima. Primenjeni mehanizmi, implementirani modeli kontrole prava pristupa, tehnologije i komponente koje se koriste da bi se primenila željena pravila veoma variraju i različiti radni okviri i jezici su se pojavljivali ili kao dopuna i ispravka nedostataka

prethodnih ili kao reorganizacija dobro poznatih koncepata kako bi se bolje prilagodili uvek promenljivim zahtevima korisnika [11], [12].

Postojeći radni okviri se mogu klasifikovati po različitim kriterijumima [28], [64], [12], [109] kao što su sistemi opšte namene [29], [122], [94], oni namenjeni pojedinim oblastima primene [31] [12] [60] [44] [85]) ili jezici zasnovani na logici prvog reda [54] sa nedvosmislenom semantikom i dobro definisanim formalizmima [24], jezici na bazi (eng.) *stratified* logike [53], jezici koji koriste (eng.) *Deontic* logiku [58], [57], [121], [64], jezici bazirani na pravilima [122], kao i *Web Ontology Language (OWL)*[5] pristupi [124], [123], [121], [64]. Isto tako, isti jezik ili *framework* se može klasifikovati u različite klase u zavisnosti od konteksta sistematizacije. Na primer, Ponder [122], [29] je poznat i kao *framework* opšte namene i kao jezik čije su polise zasnovane na pravilima.

6.2.3.1 *Common policy*

Common policy je *framework* autorizacionih polisa koji omogućava kontrolu prava pristupa podacima aplikacije. Polise su izražene u XML-u prema XML šemi definisanoj u RFC4745 [105]. Šema definiše "skup pravila" (*eng. rule set*) koji sadrže neuređen skup "pravila" (*eng. rule*), pri čemu "pravila" imaju "uslove" (*eng. conditions*), "akcije" (*eng. actions*) i "transformacije" (*eng. transformations*). Pretpostavlja se da će na nivou pojedinačne primene *common policy* biti proširen polisama specifičnim za oblast primene.

Common policy je predložen u okviru RFC 4745 u februaru 2007. godine [105] i trenutni status je *Proposed standard*[6] .

6.2.3.2 *Role Definition Language (RDL)*

Role Definition Language (RDL) je skup pravila koja određuju pod kojim uslovima (eng. *conditions*) klijentu (eng. *client*) može biti dodeljena uloga (eng. *role*). Uslovi za dobijanje uloge određuju pogodnost klijenta da dobije sertifikat sa ograničenjima i parametrima kredencijala. RDL omugućava specifikaciju konteksta, koji je opcion.

6.2.3.3 Ponder and Ponder2

Ponder je deklarativni jezik za specifikaciju polisa za upravljanje distribuiranim računarskim sistemima i bezbednosnih polisa distribuiranih sistema [29]. Isto ime se koristi i za alate za specifikaciju, analizu i primenu ovih polisa. Ponder podržava i pozitivne i negativne autorizacione polise, obligacione polise, uzdržavanje od

[5] https://www.w3.org/TR/owl2-overview/

[6] Official Internet Protocol Standards http://www.rfc-editor.org/search/standards.php#PS

primene polise, kao i pozitivnu i negativnu delegaciju polisa. Specifikacija konteksta polise je opciona. Polise mogu biti kompozitne i sadržati i meta-pravila, a objekti u Ponder-u mogu biti hijerarhijski organizovani [29] [95]. Ponder2 je zadržao koncepte iz Pondrera ali ih je značajno redizajnirao i reimplementirao. [122] [133]

Iako je pogodan za mnoge oblasti primene, Ponder je povučen i više se ne održava[7].

6.2.3.4 KAoS

KAoS je framework koji omogućava specifikaciju, analizu, detekciju i rezoluciju konflikata i primenu polisa. Polise mogu biti autorizacione ili obligacione i mogu se generisati na osnovu predložaka ili same sebe referencirati. Polise su koncepti u ontologijama izraženim u OWL2 (*Web Ontology Language*) i mogu se proširiti drugim konstruktima koa što su *role-value* mape KAoS se može primeniti u raznim oblastima. [124] [123] [13].

6.2.3.5 REI i REIN

REI je OWL-Lite baziran jezik za specifikaciju polisa koji omogućava i analizu i rezonovanje. Podržava pozitivne i negativne autorizacione polise, pozitivne i negativne obligacione polise i ukidanje obligacija [121] Polise se definišu kao dozvoljene i obavezne akcije na objektima. Odnosi između trdtradobjekata su se opisuju *role-value* parovima, a specifikacija konteksta je opciona. [58] [64]. Razvoj Rei je prekinut u maju 2015. godine[8] [12].

REIN (REI and N3[9]) je naslednik REI. REIN je distribuirani *framework* za specifikaciju polisa i rezonovanje. Svaka REIN polisa može imati sopstvenu ontologiju ili jezik. REIN omogućava korišćenje N3 pravila za povezivanje polisa i resursa [58] [64].

6.2.3.6 PERMIS

PERMIS[10] (PrivilEge and Role Management Infrastructure Standards) je autorizaciona infrastruktura zasnovana na polisama koja implementira hijerarhijski RBAC model kontrole prava pristupa [21] i opcionu specifikaciju konteksta. Uloge korisnika se čuvaju u atributima X.509 sertifikata [20] [22]. Podržani tipovi polisa su *"Credential Issuing"* polise, autorizacione polise, *"Delegation"* polise i *"Credential Validation"*

[7] http://www-dse.doc.ic.ac.uk/Research/policies/ponder.shtml
[8] REI http://rei.umbc.edu
[9] https://www.w3.org/DesignIssues/Notation3.html
[10] http://www.openpermis.info

polise [21]. Polise se specificiraju u XML-u i mogu biti enkapsulirane u digitalno potpisane atribute X.509 sertifikata.[11] PERMIS ima XACML i SAML interfejse.

6.2.3.7 *eXtensible Access Control Markup Language (XACML)*

eXtensible Access Control Markup Language (XACML) je deklaritivni jezik za specifikaciju autorizacionih i obligacionih polisa i model obrade (eng. *processing model*) koji opisuje kako odrediti odluku o pristupu na osnovu pravila (eng. *rule*) definisanih u polisama. XACML polise mogu sadržati više pravila koja se kombinuju u skladu sa izabranim algoritmom (eng. *combining algorithm*). XACML polise se pišu u XML-u i specifikacija konteksta je opciona [94]. XACML je OASIS standard.

6.2.3.8 FRENETIC i PYRETIC

FRENETIC i PYRETIC Frenetic su visoki SDN programski jezici [44] dizajnirani za specifikaciju i održavanje *packet-forwarding* polisa i monitoring mrežnog saobraćaja [43]. FRENETIC je implementiran i izvršava se u Frenetic-OCaml okruženju[12], a PYRETIC u Python-u[13]. Frenetic, Pyretic i projekti koji ih koriste se negde referenciraju kao *Frenetic familija jezika* (eng. *Frenetic language family*). Oba jezika se aktivno razvijaju.

6.2.3.9 *WS-Security Policy Language*

Web Services Security (WS-Security) [31] je proširenje SOAP protokola [82] da bi se obezbedila sigurnost Web servisa. *Web Services Security Policy Language (WS-SecurityPolicy)* specifikacija [31] omogućava primenu različitih *security* modela u Web servisima.

6.2.3.10 DataLog

Datalog [48] [64] je deklarativni programski jezik čija je sintaksa podskup *Prolog* [26] sintakse. Datalog je nastao kao jezik za pisanje upita (eng. *query language*) u relacionim bazama podataka [18] , a uz odgovarajuća proširenja se može koristiti za specifikaciju autorizacija [53], kao jezik za upravljanje poverenjem (eng. *Trust Management*) [71], u *cloud* okruženju [48] ili za upravljanje mrežama [73] [74].

[11] http://sec.cs.kent.ac.uk/permis/documents/concept.shtml
[12] https://github.com/frenetic-lang/frenetic
[13] http://www.frenetic-lang.org/pyretic/

6.2.4 *Policy-based management pristupi* - zaključak

Sistem za automatsku konfiguraciju i aktivaciju višedomenskih usluga treba da pomiri dva na izgled isključujuća zahteva:

- model podataka federativnog resursa (Poglavlje 5.3) nije poznat u napred jer je određen vrstama višedomenskih usluga koje federacija pruža i
- domeni znaju i dobro razumeju semantiku višedomenske usluge.

Sistemi koji koriste ontologije za opis osnovnih koncepata mogu obezbediti specifikaciju polisa nad meta-modelom federativnog resursa, ali pošto je semantika usluge koju federacija pruža dobro poznata i precizno određena, korišćenje ontologija za predstavljanje znanja i rezonovanja je nepotrebna aktivnost.

Od opisanih PBM pristupa koji ne koriste ontologije, Ponder2 [29], DataLog [48] i XACMLv3 [94] su veoma izražajni i podržavaju polise korišćenjem meta-modela podataka, dok ostali pristupi, kao što su WS-Security Policy Language [31] i Frenetic familija jezika [43] [44], imaju usku oblast primene. Pošto se Ponder2 više ne održava, a DataLog ne omogućava specifikaciju obligacionih polisa, za ECAPBACF implementaciju je izabran OASIS standard XACMLv3.

6.3 *Environmental Federated Policy-Based Access Control Framework (ECAPBACF)* platforma

U ovom poglavlju je opisana *environmental context-aware policy-based access control platforma ECAPBACF* koja omogućava granularnu kontrolu pristupa deljenim federativnim resursima. Pravila i ograničenja koja kontrolišu korišćenje federativnih resursa su opisana atributima što omogućava korišćenje meta-modela federativnih resursa u polisama. U nastavku poglavlja će ECAPBACF platforma biti opisan korišćenjem *Fine-grained integration algebre (FIA)* [97] [98], a zatim će biti prikazana ECAPBACF implementacija korišćenjem XACMLv3 jezika za specifikaciju polisa. FIA algebra je odabrana zato što na jednostavan način omogućava opisivanje polisa sa tri ili manje ishoda, a podržava i autorizacione i obligacione polise. Za razliku od jezika za opisivanje polisa zasnovanih na logici prvog reda (predikatskoj logici) [49] [53] [51] čije korišćenje traži poznavanje predikatske logike i često korišćenje algoritama eksponencijalne složenosti ili od jezika koji tipično koriste *Deontic logic* da bi organizovali koncepte i znanje u ontologije [124] [123] [13] [121] [58] [64] i posebno izvedenih logika za formalizaciju kontkretnog jezika za specifikaciju polisa [96], FIA je jedina razvijena za opis polisa koje imaju tri stanja i ne zavisi od jezika za opis polisa koji se koristi za implementaciju. [27] predlaže opšti sveobuhvatni

model kontrole prava pristupa koji je suviše opširan i glomazan za formalan opis ECAPBACF *framework*-a.

6.3.1 *Fine-grained integration algebra* - koncepti

U ovom odeljku su, jasnoće radi, opisani osnovni koncepti *fine-grained integration algebre (FIA)* [97] [98] potrebni za opisivanje ECAPBACF *framework*-a. *Fine-grained integration algebra* je algebra koja se koristi za izražavanje polisa sa tri moguća ishoda. Sintaksa FIA algebra se bavi jezikom, a semantika značenjem FIA polisa. Semantika FIA je sintaksno nezavisna jer se pri definiciji operacija nad polisama koristi meta-model polise. Sintaksa FIA algebre je određena jezikom u kom su izražene polise, na primer XACMLv2 [97] [97] i Ponder2 [133].

6.3.1.1 *Fine-grained integration algebra* - osnovne definicije i operatori

FIA algebra je određena uređenom n-torkom:

$$\langle \Sigma, P_Y, P_N, +, \&, \neg, \Pi_{dc} \rangle \tag{6.1}$$

u kojoj su Σ rečnik (vokabular) algebre, P_Y polisa konstanta koja dozvoljava sve, P_N polisa konstanta koja brani sve, sabiranje $+$ (eng. *addition*) i presek $\&$ (eng. *intersection*) dva binarna operatora, negacija $\neg$ (eng. *negation*) i projekcija[14] Π_{dc} (eng. *projection*) dva unarna operatora [97]. Rečnik algebre Σ je skup A svih *naziva atributa* $a \in A$ i skupova njihovih vrednosti $\{v \in V_a \mid a \in A\}$[15]:

$$\Sigma = \{\langle a, V_a \rangle \mid a \in A\} \tag{6.2}$$

U FIA terminologiji, zahtev za pristupom r je skup uređenih parova $\langle a, v \rangle$, gde je a *naziv atributa*, a $v \in V_a$ *vrednost atributa*:

$$r \equiv \{\langle a_i, v_i \rangle \mid i = 1..m\} \tag{6.3}$$

dok je FIA polisa P funkcija koja mapira skup svih zahteva R_Σ u skup svih mogućih efekata E:

$$P : R_\Sigma \to E \quad E = \{Y,\ N,\ N/A\} \tag{6.4}$$

[14]Da bi se izbegle nejasnoće, koristi se termin "projekcija" (eng. *projection*) umesto "projekcija domena" (eng. *domain projection*) iz [97] [98]

[15]Da bi se izbegle nejasnoće, koristi se termin "skup mogućih vrednosti za sve atribute a" sa oznakom V_a umesto originalnog "domen atributa" (eng. *attribute domain*) sa oznakom $dom(a)$ iz [97] [98]

gde efekat Y znači "Dozvoli pristup", efekat N "Zabrani pristup" i N/A da polisa nije primenljiva za zahtev. Ovakva formulacija polise čini semantiku polise nezavisnom od sintakse, tj. od jezika u kom je polisa izražena. Alternativno, FIA polisa se može prikazati i kao uređena trojka:

$$P \equiv \langle\ R^P_Y,\ R^P_N,\ R^P_{N/A}\rangle \tag{6.5a}$$

$$R_\Sigma = R^P_Y \cup R^P_N \cup R^P_{N/A} \quad R^P_Y \cap R^P_N = \emptyset \quad R^P_Y \cap R^P_{N/A} = \emptyset \quad R^P_N \cap R^P_{N/A} = \emptyset \tag{6.5b}$$

pri čemu je R^P_Y skup svih zahteva koje polisa preslikava u efekat Y, R^P_N svih zahteva koje polisa mapira u efekat N, a $R^P_{N/A}$ označava zahteve za koje se polisa ne može primeniti. Polise konstante P_Y i P_N se tada mogu zapisati:

$$P_Y \equiv \langle R_\Sigma,\ \emptyset,\ \emptyset\rangle \tag{6.6a}$$

$$P_N \equiv \langle \emptyset,\ R_\Sigma,\ \emptyset\rangle \tag{6.6b}$$

Pošto je komplement, odnosno dopuna, skupa zahteva R u odnosu na skup svih zahteva R_Σ, skup svih elemenata skupa R_Σ koji nisu elementi skupa R:

$$(R)^{\complement} \equiv \{r \notin R,\ r \in R_\Sigma\ \} \tag{6.7}$$

očigledno važi:

$$R^P_Y = (R^P_{N/A})^{\complement} \cap (R^P_N)^{\complement} \tag{6.8a}$$

$$(R^P_Y)^{\complement} = R^P_N \cup R^P_{N/A} \tag{6.8b}$$

$$R^P_N = (R^P_Y)^{\complement} \cap (R^P_{N/A})^{\complement} \tag{6.8c}$$

$$(R^P_N)^{\complement} = R^P_Y \cup R^P_{N/A} \tag{6.8d}$$

$$R^P_{N/A} = (R^P_Y)^{\complement} \cap (R^P_N)^{\complement} \tag{6.8e}$$

$$(R^P_{N/A})^{\complement} = R^P_Y \cup R^P_N \tag{6.8f}$$

Kao i polise, i binarni i unarni operatori se mogu definisati dvojako. Binarni operator je funkcija koja preslikava Dekartov proizvod $E \times E$ u skup svih efekata E, dok je unarni operator funkcija koja preslikava skup svih efekata E u samog sebe (Poglavlje 6.9). Operatori su istovremeno i pravila za određivanje skupova zahteva $R^P_Y,\ R^P_N,\ R^P_{N/A}$, pri čemu je $R^{P_I}_Y$ skup zahteva koje integrisana polisa P_I preslikava u efekat Y, $R^{P_I}_N$ skup zahteva koje integrisana polisa P_I preslikava u efekat N i $R^{P_I}_{N/A}$ skup zahteva koji integrisana polisa P_I preslikava u efekat N/A, odnosno za koje

integrisana polisa P_I nije primenljiva. Definicije operatora mogu biti predstavljene i tabelama koje određuju efekat u koji će integrisana polisa P_I preslikati zahtev r u zavisnosti od efekata u koje zahtev r preslikavaju polise operandi.

$$+ : E \times E \rightarrow E \qquad \& : E \times E \rightarrow E \tag{6.9a}$$

$$\neg : E \rightarrow E \qquad \Pi_{dc} : E \rightarrow E \tag{6.9b}$$

Negacija polise P je polisa P_I za koju važi:

$$P_I = \neg P \equiv \begin{cases} R_Y^{P_I} = R_N^P \\ R_N^{P_I} = R_Y^P \\ R_{N/A}^{P_I} = R_{N/A}^P \end{cases} \tag{6.10}$$

što je prikazano u Tabeli 6.1.

Tabela 6.1 Operator $\neg$

P	$\neg P$
Y	N
N	Y
N/A	N/A

Projekcija polise P, u oznaci $\Pi_{dc}(P)$ je polisa P_I za koju važi:

$$P_I = \Pi_{dc}(P) \equiv \begin{cases} R_Y^{P_I} = \{r \in R_Y^P \; i \; r \; zadovoljava \; dc\} \\ R_N^{P_I} = \{r \in R_N^P \; i \; r \; zadovoljava \; dc\} \\ R_{N/A}^{P_I} = (R_Y^{P_I})^{\complement} \cap (R_N^{P_I})^{\complement} \end{cases} \tag{6.11}$$

pri čemu ograničenje dc ograničava dozvoljene vrednosti atributa $a \in A$ na prave podskupove skupova vrednosti atributa $range_a \subset V_a$:

$$\{(a_i, range_{a_i}) \mid a_i \in A, \; range_{a_i} \subset V_a, \; i = 1..k\} \tag{6.12}$$

Unarni operator Π_{dc} je određen Tabelom 6.2.

Tabela 6.2 Operator Π_{dc}

P	$Zahtev\ r\ zadovoljava\ dc$	$\Pi_{dc}(P)$
Y	da	Y
Y	ne	N/A
N	da	N
N	ne	N/A
N/A	–	N/A

Sabiranje polisa P_1 i P_2, u oznaci $P_1 + P_2$ je definisano kao integracija polisa P_1 i P_2 u rezultujuću polisu P_I za koju važi:

$$P_I = P_1 + P_2 \equiv \begin{cases} R_Y^{P_I} & = R_Y^{P_1} \cup R_Y^{P_2} \\ R_N^{P_I} & = (R_Y^{P_I})^{\complement} \cap (R_{N/A}^{P_I})^{\complement} \\ R_{N/A}^{P_I} & = R_{N/A}^{P_1} \cup R_{N/A}^{P_2} \end{cases} \tag{6.13}$$

Tabela koja određuje efekat u koji će integrisana polisa $P_I = P_1 + P_2$ preslikati zahtev r u zavisnosti efekata u koje zahtev r preslikavaju polise P_1 i P_2 je prikazana u Tabeli 6.3.

Tabela 6.3 Operator $+$

P_1 \ P_2	Y	N	N/A
Y	Y	Y	Y
N	Y	N	N
N/A	Y	N	N/A

Presek polisa P_1 i P_2, u oznaci $P_1 \ \& \ P_2$ je definisan kao integracija polisa P_1 i P_2 u rezultujuću polisu P_I za koju važi:

$$P_I = P_1 \ \& \ P_2 \equiv \begin{cases} R_Y^{P_I} & = R_Y^{P_1} \cap R_Y^{P_2} \\ R_N^{P_I} & = R_N^{P_1} \cap R_N^{P_2} \\ R_{N/A}^{P} & = (R_Y^{P})^{\complement} \cap (R_N^{P})^{\complement} \end{cases} \tag{6.14}$$

Tabela koja određuje efekat u koji će integrisana polisa $P_I = P_1 \& P_2$ preslikati zahtev r u zavisnosti efekata u koje zahtev r preslikavaju polise P_1 i P_2 je prikazana u Tabeli 6.4.

Tabela 6.4 Operator $\&$

P_1 \ P_2	Y	N	N/A
Y	Y	N/A	N/A
N	N/A	N	N/A
N/A	N/A	N/A	N/A

Skup operatora $\{+, \&, \neg, \Pi_{dc}\}$ je *potpun* (*complete*)[16] tj. bilo koja kombinacija polisa $P_1, P_2, ...P_n$ se može izraziti pomoću operatora iz skupa $\{+, \&, \neg, \Pi_{dc}\}$ i skupa polisa konstanti $\{P_Y, P_N\}$ [97]. Navedeni operatori takođe čine i minimalan skup operatora [97].

6.3.1.2 Proces generisanje integrisane FIA polise

FIA izraz (eng. *FIA expression*) se definiše rekurzivno:

- Ako je P FIA polisa, P je FIA izraz.
- Ako je f FIA izraz i $\neg f$ je FIA izraz.
- Ako je f FIA izraz i dc ograničenje i $\Pi_{dc}(f)$ je FIA izraz.
- Ako su f_1 i f_2 FIA izrazi, onda je i $f_1 + f_2$ FIA izraz.
- Ako su f_1 i f_2 FIA izrazi, onda je i $f_1 \& f_2$ FIA izraz.

Proces generisanja integrisane polise za FIA izraz $f(P_1, P_2, .., P_n)$ je prikazan Algoritmom 1. Osnovne faze generisanja integrisane polise su:

- Transformacija polisa $P_1, P_2, .., P_n$ FIA izraza $f(P_1, P_2, .., P_n)$ u *složene Boolean izraze* pri čemu se opsezi vrednosti atributa ne preklapaju na nivou pojedinačnog atributa,
- Enkodovanje *atomičnih Boolean izraza* u jedinstvene *Boolean promenljive*,
- Transformacija polisa u funkcije koje mapiraju jedinstvene Boolean promenljive u skup efekata,
- Konstrukcija *multi-terminal binary decision tree (MTBDD)* za svaku polisu,
- Konstrukcija *multi-terminal binary decision tree (MTBDD)* integrisane polise i
- Generisanje integrisane polise.

[16] Skup operatora $\{+, \&, \neg\}$ je potpun u smislu da njim može biti izražena bilo koja kombinacija FIA polisa, ali je operator Π_{dc} potreban jer generisanje integrisane polise može koristiti različite atribute za različite zahteve r

Algoritam 1: Integrated FIA policy generation

Data: $f(P_1, P_2, .., P_n)$
Result: P_I

Transform policies in Compound Boolean expressions over their attributes with non-overlaping sets/ranges of attribute values for matching attribute names

Encode atomic Boolean expressions in unique Boolean variables

Transform policies in functions which map vector of unique Boolean variables into set of effects

Construct multi-terminal binary decision diagram (MTBDD) for every policy

Construct multi-terminal binary decision diagram (MTBDD) of the integrated policy

Generate integrated policy

6.3.1.2.1 Transformacija FIA polisa u složene *Boolean* izraze

Složeni Boolean izraz je izraz koji se sastoji od atomičnih Boolean izraza i logičkih operatora $\{\wedge,\ \vee\}$. Atomični Boolean izraz AE je *one-variable equality constraint* ili *one-variable inequality constraint* pri čemu je:

- one-variable equality constraint izraz oblika $a \triangleright c$ gde su a naziv atribut, c konstanta i $\triangleright \in \{=,\ \neq\}$
- one-variable inequality constraint izraz oblika $c_1 \triangleleft a \triangleright c_2$ gde su a naziv atribut, c_1 i c_2 konstante i $\triangleleft,\ \triangleright \in \{<,\ \leq,\ >,\ \geq\}$

Ukoliko nakon transformacije polisa u kompozitne Boolean izraze postoje atomični Boolean izrazi $AE_1,\ AE_2,\ ..,\ AE_n$ koji za isti naziv atributa a imaju opsege vrednosti $range_1,\ range_2, .., range_n$ koji se preklapaju, potrebno ih je transformisati u sekvencu atomičnih Boolean izraza $AE'_1,\ AE'_2,\ ..,\ AE'_{n'}$ u kojima se opsezi vrednosti $range'_1,\ range'_2, .., range'_{n'}$ za naziv atributa a ne preklapaju. To se može uraditi tako što se unija svih skupova vrednosti atributa $V_a^+ = \bigcup_{i=1}^{n} range_i$ podeli na međusobno disjunktne opsege $range'_j,\ j = 1..n'$ tako da važi:

$$\begin{array}{l} \bigcup_{i=1}^{n} range_i = \bigcup_{j}^{n'} range'_j \\ \bigcap_{j=1}^{n'} range'_j = \emptyset \end{array} \tag{6.15}$$

6.3.1.2.2 Enkodovanje atomičnih *Boolean* izraza u jedinstvene *Boolean* promenljive

FIA polisa transformisana u kompozitni Boolean izraz, bez atomičnih Boolean

izraza u kojima se opsezi vrednosti istog atributa a preklapaju, se može enkodovati tako što se svaki atomični Boolean izraz AE_i enkoduje u Boolean promenljivu:

$$AE_i \ \rightarrow \ x_i : \begin{cases} x_i \ = \ 0 & ako\ je\ AE_i\ false \\ x_i \ = \ 1 & ako\ je\ AE_i\ true \end{cases} \tag{6.16}$$

Nakon enkodovanja atomičnih Boolean izraza u Boolean promenljive, FIA polisa se može predstaviti kao preslikavanje vektora Boolean promenljivih $\overrightarrow{x} \ = \ (x_1, x_2, .., x_n)$ u skup efekata E:

$$P \ : \ \{0, \ 1\}^n \ \rightarrow \ \{Y, \ N, \ N/A\} \tag{6.17}$$

Pri enkodovanju FIA izraza $f(P_1, P_2, .., P_n)$, ekvivalentni atomični Boolean izrazi AE i AE' se enkoduju u istu Boolean promenljivu x. Dva atomična Boolean izraza AE_x i AE_X su ekvivalentna ako i samo ako važi:

$$\begin{array}{rl} & (\forall \langle a_i, V_{a_i} \rangle \in AE_x)(\exists \langle a_j, V_{a_j} \rangle \in AE_y)(a_i = a_j, V_{a_i} = V_{a_j}) \\ \wedge & (\forall \langle b_l, V_{b_l} \rangle \in AE_y)(\exists \langle b_k, V_{b_k} \rangle \in AE_x)(b_l = b_k, V_{b_l} = V_{b_k}) \end{array} \tag{6.18}$$

Ako u FIA izrazu $f(P_1, P_2, .., P_n)$ postoje polise P_x i P_y sa atomičnim Boolean izrazima u kojima se opsezi vrednosti istog atributa a preklapaju, potrebno je te atomične Boolean izraze transformisati u sekvence atomičnih Boolean izraza kao što je opisano u 6.3.1.2.1.

6.3.1.2.3 Konstrukcija MTBDD za FIA polisu

Polisa transformisana u kompozitni Boolean izraz može biti predstavljena kao *Multi-Terminal Binary Decission Diagram (MTBDD)* koji omogućava kompaktno predstavljanje mapiranja vektora konačnog skupa promenljivih u konačan skup rezultata.

MTBDD reprezentacija polise P je usmereno stablo u kom su čvorovi Boolean promenljive $x_1, x_2, .., x_m$ u koje su enkodovani atomični Boolean izrazi $AE_1, AE_2, .., AE_m$ od kojih se sastoji FIA polisa P. Svaki čvor ima dve grane: jednu označenu sa 0, a drugu sa 1. Listovi MTBDD stabla su vrednosti iz skupa efekata E. Pošto svaka putanja u MTBDD predstavlja dodelu vrednosti Boolean promenljivama koje predstavljaju čvorovi, svaka putanja u MTBDD predstavlja jedan zahtev $r \ \in \ R_\Sigma$, dok je list efekat u koji polisa P preslikava zahtev r. Da bi se konstruisalo MTBDD stablo polise, potrebno je da je skup Boolean promenljivih parcijalno uređen, na primer:

$$x_1 \ \succ x_2 \ \succ \ \ldots \ \succ \ x_n \tag{6.19}$$

6.3.1.2.4 Konstrukcija MTBDD integrisane FIA polise

Konstrukcija MTBDD integrisane FIA polise za FIA izraz $f(P_1,\ P_2,\ \ldots,\ P_n)$ se obavlja iterativno primenom operatora iz FIA izraza na MTBDD reprezentacije polisa $P_1,\ P_2,\ \ldots,\ P_n$. Pravila za primenu operatora na čvor $Node$, odnosno čvorove $Node_1$ i $Node_2$, MTBDD stabla su:

- unarni operator negacija $\neg$: primena operatora $\neg$ na čvor MTBDD $Node$ rezultuje zamenom listova Y i N u podstablu kome je koren čvor $Node$.

- unarni operator projekcija Π_{dc}: da bi se primenio operator Π_{dc} na čvor $Node$ potrebno je obići podstablo sa korenom u čvoru $Node$ i u svakom čvoru $Node_i$:

Algoritam 2: Primena operatora Π_{dc} na MTBDD FIA polise

```
foreach Node_i u podstablu sa korenom u Node do
    if Node_i sadrži atribute specificirane ograničenjem dc then
        zameniti opsege atributa opsezima određenim ograničenjem dc
    else
        podstablo određeno čvorom Node_i će u integrisanoj polisi P_I biti
        preslikano u efekat N/A, pa ga treba ukloniti
```

- binarni operator sabiranje $+$ i binarni operator presek &: binarna operacija se obavlja prema proceduri *Apply* [97] prikazanoj na Slici 6.3.

Procedura Apply koristi relaciju parcijalnog uređenja skupa promenljivih 6.19 pri obilasku MTBDD stabala polisa P_1 i P_2. $Node_1$ je čvor MTBDD stabla polise P_1, $Node_2$ je čvor stabla P_2, a OP binarna operacija iz FIA izraza $f(P_1, P_2)$.

Procedure Apply($Node_1$**,** $Node_2$**,** OP**)**
Input : $Node_1$, $Node_2$ are MTBDD nodes,
OP is a policy operation

1. initiate $Node_I$ // $Node_I$ is the combination result
2. **if** $Node_1$ and $Node_2$ are terminals **then**
3. $Node_I \leftarrow (Node_1\ OP\ Node_2, null, null)$
4. **else**
5. **if** $Node_1.var = Node_2.var$ **then**
6. $Node_I.var \leftarrow Node_1.var$
7. $Node_I.left \leftarrow$ Apply($Node_1.left, Node_2.left, OP$)
8. $Node_I.right \leftarrow$ Apply($Node_1.right, Node_2.right, OP$)
9. **if** $Node_1.var$ precedes $Node_2$.var **then**
10. $Node_I.var \leftarrow Node_1.var$
11. $Node_I.left \leftarrow$ Apply($Node_1.left, Node_2, OP$)
12. $Node_I.right \leftarrow$ Apply($Node_1.right, Node_2, OP$)
13. **if** $Node_2.var$ precedes $Node_1$.var **then**
14. $Node_I.var \leftarrow Node_2.var$
15. $Node_I.left \leftarrow$ Apply($Node_2.left, Node_1, OP$)
16. $Node_I.right \leftarrow$ Apply($Node_2.right, Node_1, OP$)
17. return $Node_I$

end Apply.

Slika 6.3 Procedura Apply[17]

6.3.1.2.5 Generisanje integrisane FIA polise

Algoritam za generisanje integrisane FIA polise P_I na osnovu MTBDD stabla polise P_I je prikazan Algoritmom 3.

Algoritam je inverzan generisanju MTBDD stabla FIA polise i sastoji se od sledećih koraka:

- odrediti sve putanje od korena MTBDD do svih listova
- podeliti skup svih putanja u dva disjunktna skupa R_Y i R_N. Skup R_Y treba da sadrži sve putanje koje se završavaju u listovima Y, a R_N putanje koje se završavaju u listovima N
- Predstaviti svaku putanju kompozitnim Boolean izrazom
- Dekodovati Boolean promenljive u atomične Boolean izraze
- Transformisati kompozitne Boolean izraze u polise

[17][97], [98]

Algoritam 3: Generisanje FIA polise iz MTBDD

Data: $MTBDD\ FIA\ polise$
Result: $FIA\ polisa$

Find all paths through MTBDD for integrated policy

Split set of all paths into two disjoint sets:

- one containing paths that lead to the terminal Y
- and others that lead to terminal N

Represent each path as a (compound) Boolean expression

Decode Boolean variables in compound Boolean expressions into atomic Boolean expressions

Translate compound Boolean expression representing path into policy over actual policy attributes

Ukoliko, pored autorizacije, polise treba da specificiraju i aktivnosti koje treba da budu izvršene, umesto autorizacionih polisa (eng. *authorization policy*) se koriste obligacione polise (eng. *obligation policy*) i izvršavaju obaveze (eng. *obligation*) specificirane integrisanom polisom [97] [133].

6.3.1.2.6 Kompleksnost algoritama za generisanje integrisane FIA polise u opštem slučaju

Konstrukcija MTBDD integrisane polise FIA izraza $f(P_1,\ P_2,\ \ldots,\ P_n)$ koji ima n polisa ima $n-1$ iteraciju, dok je kompleksnost algoritma za obilazak uređenog grafa, tj. MTBDD stabla, srazmerna kvadratu broja čvorova MTBDD stabla [119].

Da bi se generisala integrisana FIA polisa P_I za FIA izraz $f(P_1,\ P_2,\ \ldots,\ P_n)$ potrebno je odrediti skupove R_Y i R_N integrisane FIA polise P_I. Pošto svaka putanja od korena MTBDD do nekog od terminalnih listova predstavlja zahtev r, određivanje skupova R_Y i R_N se svodi na određivanje svih putanja od korena MTBDD do svakog lista. Kompleksnost algoritma za određivanja svih putanja od korena MTBDD do svih listova u najopštijem slučaju eksponencijalno zavisi od broja čvorova MTBDD, pri čemu je pri konstrukciju MTBDD moguće primeniti tehnike za smanjenje broja putanja u MTBDD kao što su *Espresso heuristic logic minimizer* [101] ili algoritmi predloženi u [41] i [42]. U pojedinim oblastima primene i u specijalnim slučajevima kada nije potrebno odrediti sve putanje do svih listova, kao što je raspodela i korišćenje spektra prikazana u [5] [6], ovi algoritmi se mogu svesti na algoritme linearne kompleksnosti.

6.4 *Environmental Federated Policy-Based Access Control Framework (ECAPBACF)*

Federated policy-based access control framework (ECAPBACF) je policy-based framework za kontrolu prava pristupa koji omogućava *fine-grained* specifikaciju pravila za korišćenje resursa ustupljenih federaciji (federativnih resursa). Ukoliko određivanje odluke o pristupu federativnom resursu treba da specificira i parametre ili aktivnosti potrebne za korišćenje resursa, umesto autorizacionih polisa treba koristiti obligacione polise. Pošto se integrisana polisa računa na isti način i za autorizacione i za obligacione polise, u nastavku teksta će termin *(federativna) ECAPBACF polisa* označavati polisu, bilo autorizacionu ili obligacionu, koja se koristi za kontrolu pristupa resursima namenjenim pružanju višedomenskih usluga, tj. federativnim resursima.

Federativna ECAPBACF polisa P je funkcija koja mapira skup svih zahteva (za korišćenje resursa) R_Σ u skup svih mogućih efekata $E = \{Y,\ N,\ N/A\}$ gde Y znači "Dozvoli akciju nad resursom", N "Zabrani akciju nad resursom" i N/A da polisa nije primenljiva za zahtev.

$$P : R \to E, \quad E = \{Y,\ N,\ N/A\} \tag{6.20}$$

U FIA terminologiji, zahtev za pristupom r je skup uređenih parova $\langle a, v\rangle$, gde je a *naziv atributa*, a $v \in V_a$ *vrednost atributa*:

$$r \equiv \{\langle a_i, v_i\rangle \,|i = 1..m\} \tag{6.21}$$

U ECAPBACF postoje tri tipa polisa: *statičke polise* koja sadrže statička ograničenja koja su domeni vlasnici resursa postavili za korišćenje svojih resursa ustupljenih federaciji, *environmental polise* koje opisuju kontekst akcije i stanje okruženja i *dinamičke polise* koje se koriste za oglašavanje sposobnosti domena na osnovu *statičkih* i *environmental* polisa. Za tipizaciju polisa se koristi atribut $policy\_type$ sa skupom mogućih vrednosti $\{STATIC,\ ENVIRONMENTAL,\ DYNAMIC\}$.

Rečnik Σ je skup svih atributa i skupova njihovih vrednosti koji se koriste za izražavanje polisa:

$$\Sigma = \{\langle a, V_a\rangle \ \mid a \in A\} \tag{6.22}$$

Prema opisu FRSI komponente (Poglavlje 5.2.2), rečnik Σ sistema je određen tipovima višedomenskih usluga koje federacija pruža. Statičke polise određuju koje atribute zahtev r treba da ima da bi akcija nad resursom $Resource_m^{Domain_i}$ bila dozvoljena, tako da statička polisa za autorizaciju pristupa resursu $Resource_m^{Domain_i}$

može biti napisana kao projekcija $\Pi_{dc} = \{\langle a_i, range_i\rangle \mid a_i \in A,\ range_i \subset V_a\}$ koja ograničava skup mogućih vrednosti atributa $\{a_l \mid l = 1..Q\}$ na pravi podskup skupa njegovih mogućih vrednosti $\{range_{a_l} \subset V_{a_l} \mid l = 1..Q\}$ polise konstante P_Y koja skup svih zahteva preslikava u Y:

$$P^{static}_{Resource_m^{Domain_i}} \equiv \Pi_{dc_m}^{Domain_i}(P_Y) \tag{6.23}$$

Domeni specificiraju statičke polise $P^{static}_{Resource^{Domain_i}}$ da bi opisali uslove pod kojima udaljeni domeni mogu koristiti njihove resurse namenjene višedomenskim servisima, dok je pristup svim ostalim resursima domena podrazumevano zabranjen. U FIA terminologiji se prethodno iskazuje sa: postoji implicitna polisa P_{deny} koja obezbeđuje negativnu autorizaciju za sve resurse domena, osim za one kojima je pristup eksplicitno pozitivno autorizovan nekom od statičkih polisa. To znači da statičke polise koje pod određenim uslovima preslikavaju zahteve u N, tj. eksplicitno brane pristup resursima, mogu biti izostavljene iz ECAPBACF. Pošto statičke polise koje eksplicitno dozvoljavaju pristup federativnim resursima, tj. preslikavaju zahteve u Y pod određenim uslovima, nisu primenljive za zahteve za koje su primenljive statičke polise koje eksplicitno brane pristup resursima, skup svih mogućih ishoda statičkih polisa je redukovan na $\{Y, N/A\}$:

$$P^{static} : R_\Sigma \ \rightarrow\ \{Y, N/A\} \tag{6.24}$$

Svakoj instanci višedomenske usluge $MD\_service$ je dodeljen skup resursa potrebnih za njeno pružanje. Skup svih federativnih resursa u svim domenima dodeljen višedomenskoj instanci usluge $MD\_service$ se opisuje environmental polisom $P^{environmental}_{MD\_service}$ koja opisuje skup federativnih resursa rezervisanih za uslugu $MD\_service$:

$$P^{enviromental}_{MD\_service} \ =\ F(\bigcup_{x=1}^{y} Resource_b^{Domain_a}) \tag{6.25}$$

Skup svih resursa u svim domenima koje su rezervisale sve višedomenske usluge je određen integrisanom polisom (eng. *integrated policy*) zbira svih environmental polisa:

$$P^{environmental}_{MD\_service_1} \ +\ P^{environmental}_{MD\_service_2} \ +\ \ldots\ +\ P^{environmental}_{MD\_service_Q} \tag{6.26}$$

Environmental polise određuju kontekst akcija, tako da su na nivou pojedinačnog zahteva r one ili zadovoljene i mapiraju zahtev u efekat Y ili nisu primenljive. Preciznije, skup mogućih efekata je za environmental polise redukovan na $\{Y,\ N/A\}$:

$$P^{environmental} : R_{\Sigma} \ \rightarrow \ \{Y,\ N/A\} \tag{6.27}$$

Analogno statičkim polisama, dinamičke polise određuju koje atribute zahtev r treba da ima da bi akcija nad resursom $Resource_m^{Domain_i}$ bila dozvoljena samo što uključuju i ograničenja nametnuta kontekstom akcije, tj. uzimaju u obzir i sve rezervacije federativnih resursa (6.27):

$$\begin{aligned} P^{dynamic}_{Resource_m^{Domain_i}} &\equiv \Pi^{Domain_i}_{dc_m}(P_Y) \\ &= f(P^{static}_{Resource_m^{Domain_i}},\ P^{environmental}_{MD\_service_1} \ + \ \ldots \ + \ P^{environmental}_{MD\_service_Q}) \end{aligned} \tag{6.28}$$

Kao i za statičke i za environmental polise, skup mogućih efekata je i za dinamičke polise redukovan na $\{Y, N/A\}$ jer dinamička polisa ili pozitivno autorizuje zahtev r, tj. preslikava ga u Y, ili je za konkretan zahtev neprimenljiva:

$$P^{dynamic} : R_{\Sigma} \ \rightarrow \ \{Y,\ N/A\} \tag{6.29}$$

FSI-D/C/A-E koristi dinamičke polise da bi odredio skraćenu topologiju za dizajn nove instance višedomenske usluge jer dinamičke polise $P^{dynamic}_{Resource_m^{Domain_i}}$ za $Resource_m^{Domain_i}$, koje se generišu na osnovu statičkih i environmental polisa, predstavljaju sposobnosti domena. Skraćena topologija, odnosno skup svih federativnih resursa raspoloživih za novu instancu višedomenske usluge, je određen integrisanom polisom zbira svih dinamičkih polisa za sve resurse koje zahtev za novom instancom višedomenske usluge r preslikavaju u efekat Y:

$$P_{constraint\_topology} \equiv P^{dynamic}_{Resource_1^{Domain_1}} \ + \ \ldots \ + \ P^{dynamic}_{Resource_O^{Domain_N}} \tag{6.30}$$

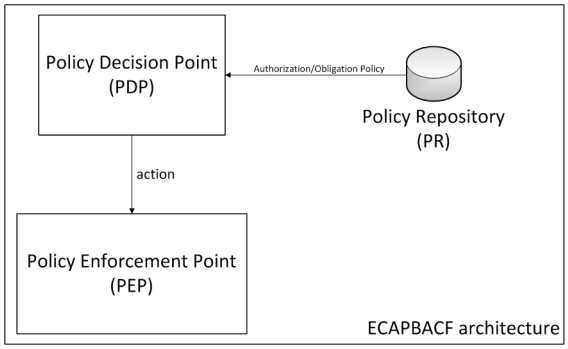
U slučajevima kada su, osim skupa resursa, za dizajn instance višedomenske usluge potrebni i dodatni parametri, umesto autorizacionih polisa se koriste obligacione polise i izvršavaju obligacije specificirane integrisanom polisom.

6.4.1 Kompleksnost generisanja integrisane ECAPBACF polise

Skup resursa koje može koristiti instanca višedomenske usluge je određen zadovoljenim dinamičkim polisama koje za konkretan zahtev r dozvoljavaju pristup resursu, tj. preslikavaju ga u Y. To znači da će samo dinamičke polise koje su zadovoljene za zahtev r biti transformisane u MTBDD. Pošto za svaki resurs $Resource_b^{Domain_a}$

može postojati najviše jedna zadovoljena dinamička polisa, za konkretan zahtev r je zadovoljeno $M_{dynamic} \leq M_{federated} = M_1 + M_2 + \ldots + M_N$ dinamičkih polisa, gde je $M_{i,\ i=1..N}$ broj federativnih resursa domena i, a $M_{federated}$ ukupan broj federativnih resursa federacije. Pošto je skup mogućih efekata dinamičkih polisa redukovan na $\{Y,\ N/A\}$, MTBDD zadovoljenih dinamičkih polisa nemaju listove označene sa N, a grane sa oznakom 0 i njihova podstabla mogu biti izostavljeni pri konstrukciji MTBDD pojedinačnih polisa. To znači da će MTBDD stablo svake zadovoljene dinamičke polise biti svedeno na putanju. Sa druge strane, sve dinamičke polise su oblika $P^{dynamic} = \Pi_{dc_m}^{Domain_i}(P_Y)$, pa će njihova MTBDD stabla, tj. putanje, imati samo koren i list označen sa Y. Prema proceduri Apply (6.3) integrisana dinamička polisa $P_I^{dynamic}$ će biti degenerisano nebalansirano stablo dubine $M_{dynamic}$ sa $M_{dynamic}$ neterminalnih čvorova i $M_{dynamic}$ listova. Svaki neterminalni čvor u MTBDD stablu integrisane polise $P_I^{dynamic}$ će imati tačno jedno dete list i najviše jedno dete neterminalni čvor i do svakog lista će postojati tačno jedna putanja od korena stabla. Da bi se odredio skup zahteva R_Y za koje je integrisana polisa $P_I^{dynamic}$ zadovoljena, odnosno skup svih putanja od korena do svih listova, dovoljan je jedan prolazak kroz stablo koji će imati $M_{dynamic}$ iteracija jer na svakom nivou stabla postoji tačno jedan list. Pošto je $M_{dynamic} \leq M_{federated}$, broj iteracija u jedinom obilasku MTBDD stabla integrisane putanje $P_I^{dynamic}$ ne može biti veći od ukupnog broja federativnih resursa federacije, odnosno vremenska složenost algortima je $O(n)$.

6.4.2 ECAPBACF *framework* - arhitektura



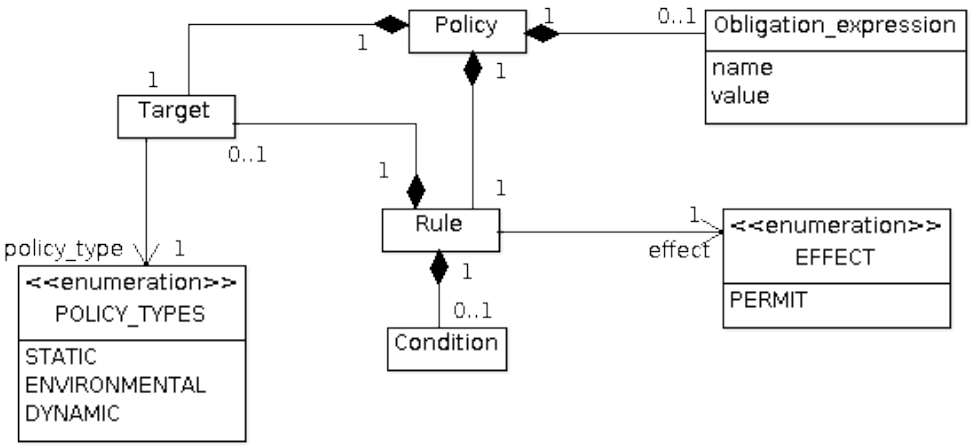
Slika 6.4 ECAPBACF arhitektura

Arhitektura ECAPBACF *framework*-a prikazana na Slici 6.4 je bazirana na *policy-management arhitekturi za policy-based networking* koju je razvila IETF Policy Framework (POLICY) radna grupa [7] [11]. Ključne komponente su *Repozitorijum polisa* (eng. *Policy Repository*) u kom se čuvaju polise, *Policy Decision Point (PDP)* komponenta koja donosi odluku o pristupu resursu i *Policy Enforcement Point (PEP)*

komponenta koja izvršava odluku o pristupu. *Policy-management* arhitektura za *policy-based networking* je detaljnije opisana u Poglavlju 6.2.1.

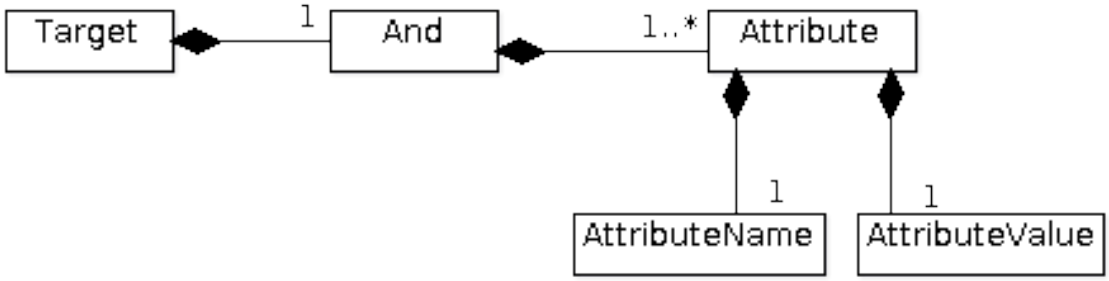
6.4.3 ECAPBACF implementacija

ECAPBACF framework je implementiran korišćenjem *eXtensible Access Control Markup Language Version 3 (XACMLv3)* [94] jezika za izražavanje polisa.



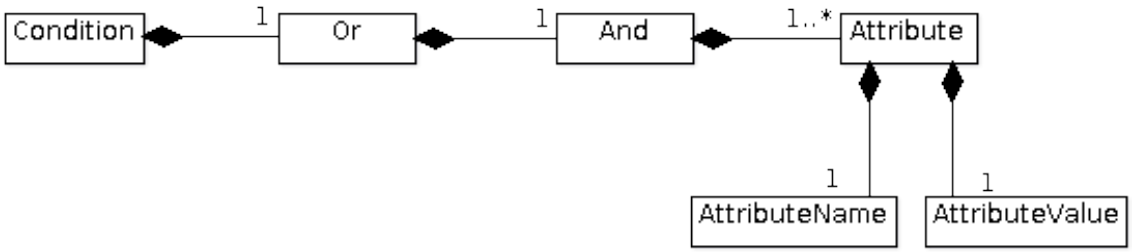
Slika 6.5 Meta-model ECAPBACF XACML polise

Model ECAPBACF XACML polise je prikazan na Slici 6.5. Svaka *Policy* ima *Target* koji opisuje federativni resurs. Pored atributa *policy_type* koji određuje tip polise, *Target* mora specificirati potrebne vrednosti atributa određene zajedničkim modelom podataka. Ako atribut *policy_type* polise ima vrednost *STATIC* ili *DYNAMIC*, polisa može imati i dodatne uslove *Conditions*. Sve polise mogu biti kompozicija *ObligationExpression* koji definišu obaveze u formi akcija i parametara potrebnih za dizajn instance višedomenske usluge. Kao i federativni resurs, i dodatni uslovi (Condition) i obaveze (ObligationExpression) se opisuju atributima zajedničkog modela podataka. Na primer, u modelu polise za višedomenske putanje garantovanog kapaciteta prikazanom na Slici 7.5, Target sadrži atribute *priority* i *reservable_capacity*. Za polise tipa STATIC ili DYNAMIC, Target takođe određuje *source_domain*, *destination_domain* da li će se resurs koristi za incoming_traffic i/ili *outgoing_traffic*, dok se dodatni uslovi polisa (Condition) mogu specificirati koristeći atribute prikazane u Tabeli 7.2 i XACML funkcije [94] prikladne tipovima podataka atributa.



Slika 6.6 Meta-model elementa Target ECAPBACF XACML polise

U [1] je pokazano da svaka *XACMLv2* polisa može biti transformisana u složeni Boolean izraz u kom se koriste samo operatori $\vee$ i $\wedge$. Poređenje XACML verzija 2 i 3[18]pokazuje da je verzija 3 opštija od verzije 2, pa isto tvrđenje važi i za *XACMLv3*. Sa druge stane, pošto su Boolean algebra i iskazna logika ekvivalentne u određenom smislu [84], a za svaku iskaznu formulu postoji njena disjunktivna normalna forma [54], i XACML polisa, kao i njen Condition element (Slika 6.7), mogu biti transformisani u disjunktivnu normalnu formu.



Slika 6.7 Meta-model elementa Condition ECAPBACF XACML polise

ECAPBACF XACML polisa ne koristi sve entitete, atribute i efekte meta-modela *XACMLv3* polise [94], već samo njihov minimalan podskup potreban za specifikaciju prava pristupa federativnim resursima. Na primer, pošto ne postoje ECAPBACF polise sa efektom N, ne postoje ni ECAPBACF XACML polise sa efektom $Deny$. Takođe, ECAPBACF XACML polise nemaju *AdviceExpressions* niti je potrebno grupisati ih u *PolicySet*.

[18]https://wiki.oasis-open.org/xacml/DifferencesBetweenXACML2.0AndXACML3.0

Glava 7

Demonstracija primene ECAPBACF *framework*-a

7.1 Višedomenska putanja garantovanog kapaciteta

Automatizacija omogućena ECAPBACF sistemom će biti opisana na primeru konfiguracije i aktivacije višedomenske putanje garantovanog kapaciteta između tačaka u različitim domenima u skladu sa prioritetima i eventualno drugim ograničenjima domena. Svaki domen alocira određeni kapacitet i stavlja ga na raspolaganje federaciji uz jasno definisana pravila i uslove pod kojima kapacitet može biti korišćen. Implicitno se podrazumeva da postoji osam nivoa prioriteta u rasponu od 1 do 8 i da putanje nižeg prioriteta mogu biti raskinute kada je kapacitet potreban za novu putanju višeg prioriteta. Takođe se pretpostavlja da je kapacitet kroz domen dostupan samo ako postoji polisa koja izričito dozvoljava korišćenje kapaciteta za višedomenske putanje garantovanog kapaciteta. Najjednostavnija polisa ima oblik:

"xGbps are available on link from domain A to domain B for incoming traffic with priority y"

dok najkompleksnija polisa ima oblik:

"xGbps are available on link from domain A to domain B for outgoing traffic with priority y if source domain is C and destination domain is not D and packet headers contain field K in period from 11:00AM to 17:30PM"

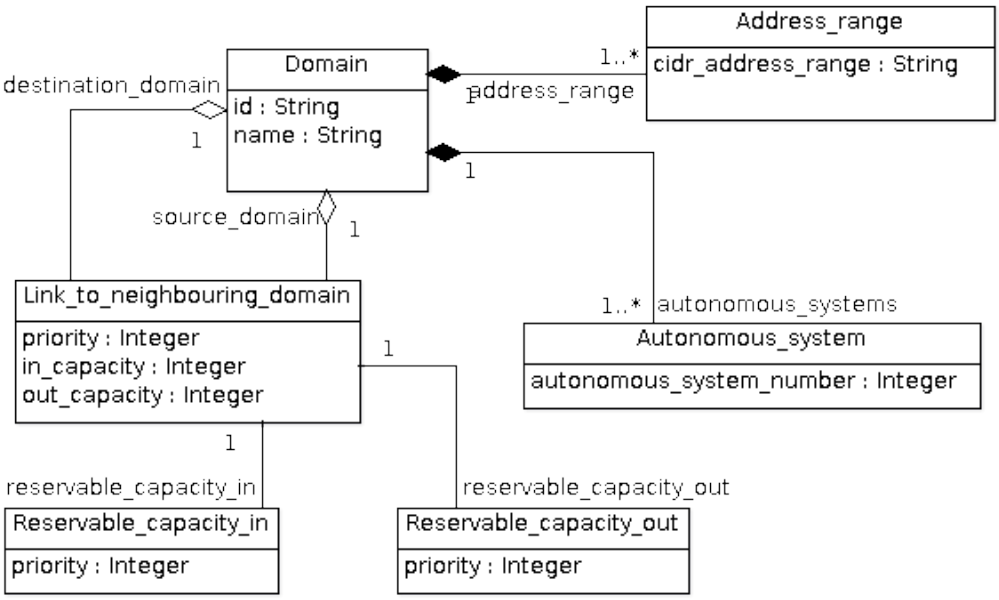
U Tabeli 7.1 su prikazane polise domena labavo spregnute federacije prikazane na Slici 5.3 i neka je uspostavljena višedomenska putanja A-B-C-D prioriteta 2 sa kapacitetom 20Gbps.

Tabela 7.1 Polise domena

Link	Ograničenja domena
A ⟶ B	70Gbps are available for incoming and outgoing traffic with priority 2
A ⟶ B	60Gbps are available for outgoing traffic with priority 3 and packet header field M
A ⟶ F	40Gbps are available for incoming and outgoing traffic with priority 5 and packet header field N
B ⟶ A	50Gbps are available for incoming and outgoing traffic with priority 3
B ⟶ C	80Gbps are available for incoming and outgoing traffic with priority 4 and packet source is not F domain
B ⟶ E	20Gbps are available for incoming and outgoing traffic with priority 5 and packet source or packet destination is domain B
C ⟶ B	90Gbps are available for outgoing traffic with priority 5 and packet header fields M or N or L
C ⟶ D	50Gbps are available for incoming and outgoing traffic with priority 4 and packet header field M
C ⟶ F	30Gbps are available for incoming and outgoing traffic with priority 8
D ⟶ C	60Gbps are available for incoming and outgoing traffic with priority 8 and packet source is not domain E
D ⟶ E	50Gbps are available for incoming and outgoing traffic with priority 8
E ⟶ B	20Gbps are available for incoming and outgoing traffic with priority 4 and packet header field N
E ⟶ D	40Gbps are available for incoming and outgoing traffic with priority 8 and packet header fields M or N
E ⟶ F	50Gbps are available for incoming and outgoing traffic with priority 7 and packet header field N
F ⟶ A	50Gbps are available for incoming and outgoing traffic with priority 4 and packet header field L or N
F ⟶ C	40Gbps are available for incoming and outgoing traffic with priority 4
F ⟶ E	60Gbps are available for incoming and outgoing traffic with priority 5 and packet header field N

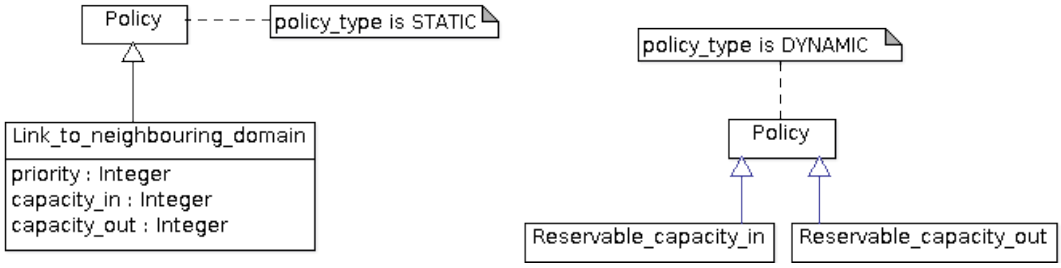
7.1.1 Model podataka višedomenskog servisa, resursa i domena

Model domena u višedomenskoj usluzi *višedomenska putanja garantovanog kapaciteta* je prikazan na Slici 7.1. *Domain* ima jedinstveni *id*, *name* i *autonomous_systems*. Adresni prostor domena *address_space* se sastoji od jedinstvenih adresnih opsega *address_ranges* koji se koriste da bi se odredili domeni kojima pripadaju početna i krajnja tačka višedomenske putanje.



Slika 7.1 Domain - model podataka

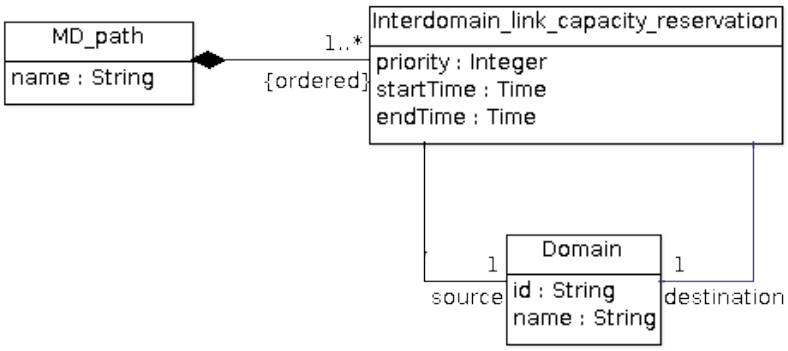
Link_to_neighbouring_domain (Slika 7.2) je polisa (Policy) koje opisuju pravila za korišćenje resursa ustupljenih federaciji i svaki Domain je agregacija Link_to_neighbouring_domains. Za svaki Link_to_neighbouring_domain Domain specificira fizičke kapacitete za dolazni (*in_capacity*) i odlazni saobraćaj (*out_capacity*), kao i sposobnosti, posebno za dolazni saobraćaj (*Reservable_capacity_in*) i odlazni saobraćaj (*Reservable_capacity_out*). Sposobnosti se definišu za svaki prioritet (*priority*) ponaosob. Sposobnosti *Reservable_capacity_in* i *Reservable_capacity_out* predstavljaju dinamičke ECAPBACF polise što je prikazano na Slici 7.2.



Slika 7.2 Sposobnosti domena

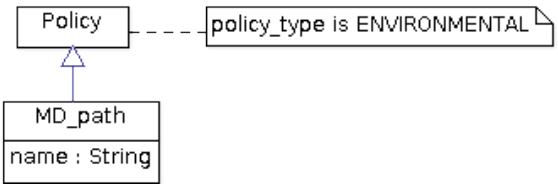
Domeni mogu postaviti ograničenja *Constraints* koja dodatno specificiraju uslove za korišćenje Links_to_neighbouring_domains, kao što je objašnjeno u Poglavlju 5.3. Constraint se izražavaju atributima iz zajedničkog ECAPBACF vokabulara Σ definisanjem pojedinačnih vrednosti ili opsega vrednosti koje zahtev za korišćenje resursa mora imati. Links_to_neighbouring_domains je kompozicija obligacija (*Obligation*) koje specificiraju parametre i akcije potrebne za dizajn instance višedomenske

usluge autorizovane da koristi Links_to_neighbouring_domains. Obligation se takođe izražavaju korišćenjem naziva atributa i vrednosti atributa iz zajedničkog ECAPBACF vokabulara Σ.



Slika 7.3 Višedomenska putanja garantovanog kapaciteta - data model

Na Slici 7.3 su prikazane apstrakcije potrebne za evidenciju rezervacija federativnih resursa. *MD_path* ima dve uređenje agregacije *Inter_domain_link_capacity_reservations*. Jedna agregacija se koristi za čuvanje topologije putanje za dolazni saobraćaj (*incoming_traffic*), a druga za čuvanje topologije putanje za odlazni saobraćaj (*outgoing_traffic*). Svaka Inter_domain_link_capacity_reservation ima prioritet (*priority*), rezervisan kapacitet (*reserved_capacity*), izvorišni (*source*) Domain, odredišni (*destination*) Domain, vreme početka (*start_time*) i vreme završetka (*end_time*). Smatra se da je saobraćaj na linku odlazni za izvorišni Domain, a dolazni za odredišni Domain.

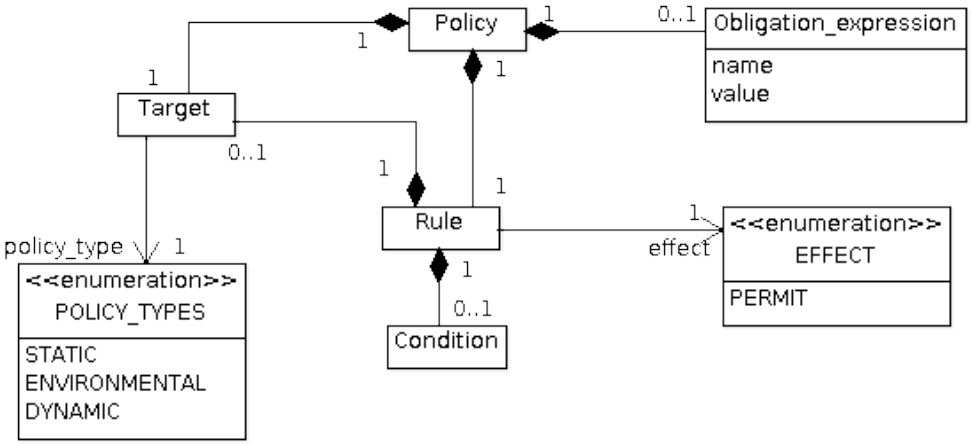


Slika 7.4 Uspostavljena putanja‘

Na osnovu opisanog modela domena, višedomenskih usluga i federativnih resursa, identifikovan je rečnik (vokabular) Σ za specifikaciju federativnih ECAPBACF polisa koji je prikazan u Tabeli 7.2. U ECPABCF terminima, Link_to_neighbouring_domains je ECAPBACF statička polisa, Reservable_capacity_in i Reservable_capacity_out su interpretacije ECAPBACF dinamičkih polisa, dok je MD_paths environmental ECAPBACF polisa.

Tabela 7.2 ECAPBACF vokabular Σ

Atribut	Skup mogućih vrednosti
capacity[Mbps]	positive integer
priority	{1, 2, 3, 4, 5, 6, 7, 8}
source_domain	Enumeration of domain names
destination_domain	Enumeration of domain names
packet_header_fields	Subset of enumeration of possible packet header fields
Service start time	An instant in time represented by a millisecond value that is an offset from the Epoch, January 1, 1970 00:00:00.000 GMT (Gregorian)
Service end time	An instant in time represented by a millisecond value that is an offset from the Epoch, January 1, 1970 00:00:00.000 GMT (Gregorian)
Time period	Period of time represented as $\langle day\_of\_week, beginning\_moment, ending\_moment\rangle$; $day\_of\_week \in$ {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday} and $beginning\_moment$ and $ending\_moment$ are millisecond offsets from 00:00:00.000 GMT
incoming_traffic	{true, false}
outgoing_traffic	{true, false}
policy_type	STATIC, DYNAMIC, ENVIRONMENTAL



Slika 7.5 XACML policy model

Model XACML polise kojom se opisuju pravila za pristup federativnim resursima koji se koriste u višedomenskim putanjama garantovanog kapaciteta je prikazan na Slici 7.5. Za izražavanje polisa se koriste identifikovani zajednički rečnik za opis

višedomenskih usluga i resursa (Tabela 7.2) i XACML funkcije [94] prikladne tipovima podataka atributa. Sve polise imaju *Obligation_expressions* koji definiše obligacije u formi parametara koji se kasnije koriste za izračunavanje najkraće višedomenske putanje.

7.1.2 Dizajn instance višedomenske usluge

Za dizajn instance višedomenske usluge FSI-D/C/A-E je potreban podskup federativnih resursa određen parametrima zahteva za instancom višedomenske usluge i ograničenjima koja su domeni definisali. Ti resursi se određuju na osnovu obligacija specificiranih zadovoljenim dinamičkim ECAPBACF polisama. Ukoliko domeni oglašavaju različite vrednosti deljenih resursa, pri dizajnu instance višedomenske usluge se koristi restriktivnija vrednost. Na primer, domeni mogu specificirati različite fizičke kapacitete međudomenskog linka i, shodno tome, oglašavati različite sposobnosti na tom linku. Ako Domen A dozvoljava 60Gbps za odlazni saobraćaj, a Domen B na istom linku samo 50Gbps dolaznog saobraćaja, FSI-D/C/A-E će prilikom dizajna višedomenske usluge koristiti manju vrednost, tj. 50Gbps.

Algoritam za dizajn višedomenske usluge je prikazan u pseudokodu Algoritmom 4. FSI-D/C/A-E prvo na osnovu parametara navedenih u zahtevu za instancom višedomenske usluge kreira zahtev za pristup raspoloživim federativnim resursima. Kreirani zahtev prosleđuje FRSI komponenti lokalnog domena kako bi dobio listu slobodnih federativnih resursa određenu obligacijama zadovoljenih dinamičkih ECAPBACF polisa. Na dobijeni skup federativnih resursa FSI-D/C/A-E primenjuje Dijkstra algoritam [119] da bi izračunao sve moguće višedomenske putanje. Ako putanja postoji, FSI-D/C/A-E obaveštava krajnjeg korisnika o izračunatoj putanji. Ukoliko višedomenska putanja ne postoji, FSI-D/C/A-E traži od FRSI komponente lokalnog domena da odredi listu fizički raspoloživih federativnih resursa, tj. da odredi listu resursa pod pretpostavkom da su sve putanje nižih prioriteta raskinute. Ovaj skup resursa FRSI određuje na osnovu obligacija zadovoljenih statičkih ECAPBACF polisa. FSI-D/C/A-E potom primenjuje Dijkstra algoritam na skup fizički raspoloživih resursa i, ukoliko višedomenska putanja postoji, traži od FRSI listu svih uspostavljenih višedomenskih putanja nižeg prioriteta kako bi odredio koje putanje moraju biti raskinute.

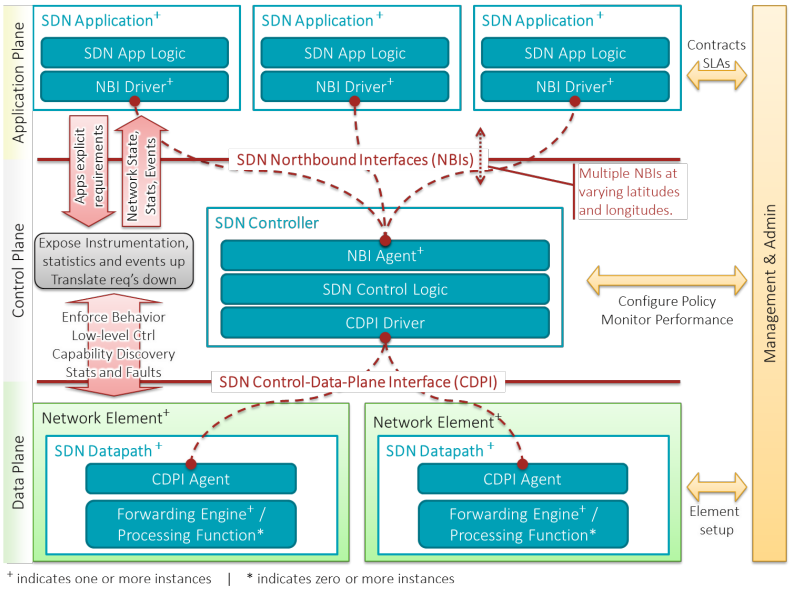
Algoritam 4: MD_path design algorithm

Data: $service\_instance\_request$
Result: $service\_instance$, $\{path\_to\_be\_broken_i | i = 0..R\}$
$federated\_resources\_request = Create(service\_instance\_request)$
$availabe\_resources =$
$FRSI.access\_decision('DYNAMIC', federated\_resources\_request)$
$\{available\_path_j | j = 0..S\} =$
$Dijkstra(available\_federeated\_resources, service\_instance\_request)$
if $\{available\_path_j | j = 0..S\} \neq \emptyset$ then
    notify user that path can be created
else
    $physically\_available\_federated\_resources\_request =$
    $FRSI.access\_decision('STATIC', federated\_resources\_request)$
    $\{available\_path_j | j = 0..S\} =$
    $Dijkstra(physically\_available\_resources, service\_instance\_request)$
    if $\{available\_path_j | j = 0..S\} = \emptyset$ then
        notify user that path cannot be created
    else
        $\{path\_with\_lesser\_priority_k | k = 0..T\} =$
        $FRSI.access\_decision('ENVIRONMENTAL', servie\_instance\_request.priority)$
        $OrderDescending(priority, capacity, \{path\_with\_lesser\_priority_k | k =$
        $0..T\})$
        foreach $\{path\_with\_lesser\_priority_k | k = 0..T\}$ do
            if $path\_with\_lesser\_priority_k$ must be broken then
                add $path\_with\_lesser\_priority_k$ to output set

7.1.3 Sistem za konfiguraciju i aktivaciju višedomenske putanje - implementacioni detalji

Prototip sistema za konfiguraciju i aktivaciju višedomenske putanje koji koristi ECAPBACF framework je razvijen i implementiran u mrežama sa softverski definisanim načinom upravljanja (eng. *SDN – Software Defined Networks*).

7.1.3.1 Mreže sa softverski definisanim načinom upravljanja *(SDN – Software Defined Networks)*



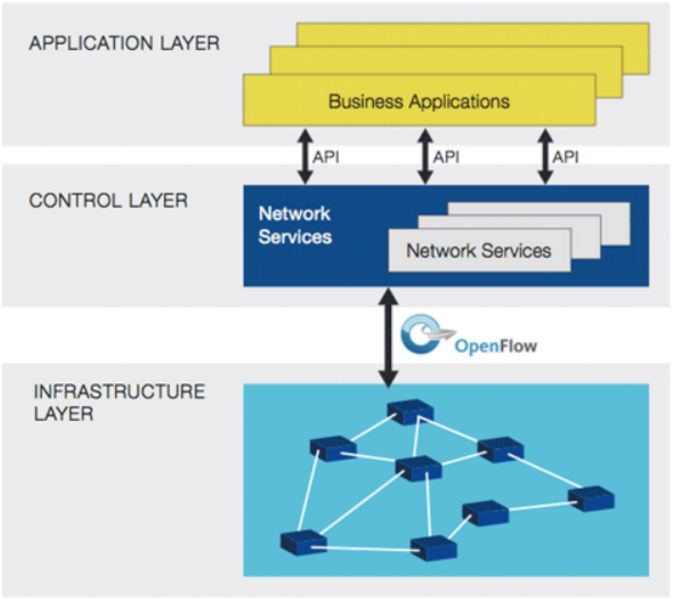
Slika 7.6 SDN mreže [1]

Softverski definisane mreže (eng. *Software Defined Networks (SDN)*) su koncept upravljanja računarskim mrežama čiji je cilj obezbeđivanje fleksibilne i agilne komunikacione infrastrukture. Konvencionalne računarske mreže se tipično sastoje od mrežnih uređaja u kojima su prisutne i kontrolna ravan (eng. *control plane*) i ravan podataka (eng. *data plane*) i ravan upravljanja (eng. *management plane*), tj. vertikalno su integrisane (eng. *vertical integration*). U tako koncipiranim mrežama konfiguracija je distribuirana (eng. *distributed configuration*), što znači da je svaki uređaj potrebno posebno konfigurisati, najčešće korišćenjem komandi niskog nivoa koje zavise od proizvođača uređaja. Pored distribuirane konfiguracije, većina IP mreža je hijerarhijski organizovana u stablo, što je pogodno za tradicionalne *client-server* arhitekture, ali nije skalabilna i sigurna mrežna infrastruktura sa fleksibilnom

[1] [91]

kontrolom saobraćaja potrebnom za podršku dinamične promene obrazaca mrežnog saobraćaja (eng. *traffic pattern*) u velikim data centrima, cloud okruženjima, pri virtualizaciji mrežnih funkcija (eng. *Network Function Virtualization (NVF)*) ili za podršku BYOD (eng. *Bring Your Own Device*) trenda [89].

Fizičkim razdvajanjem kontrolne ravni i ravni podataka (Slika 7.6) SDN eliminiše vertikalnu integraciju jer je kontrolna logika izmeštena iz mrežnih uređaja u logički centralizovan sloj.[2] Mrežni uređaji tako postaju prosti "prosleđivači" paketa (eng. *forwarding device*) [67] koji odlučuju šta treba da urade sa paketom na osnovu uparivanja pravila iz svoje flow tabele (eng. *flow table*) i skupa polja paketa. Za razmenu podataka između slojeva se koriste Northd (eng. *Northbound Interface (NI)*), odnosno Southbound interfejsi (eng. *Southbound Interface (SI)*). Preporuka organizacije Open Networking Foundation (ONF)[3] je da interfejsi koriste zajedničke modele podataka: *Core information model* [92] i *Common information model* [90].



Slika 7.7 SDN arhitektura [4]

Jedan od osnovnih principa SDN-a su "otvoreni interfejsi" (eng. *open interfaces*) čime se postiže nezavisnost od proizvođača mrežne opreme [93]. OpenFlow je

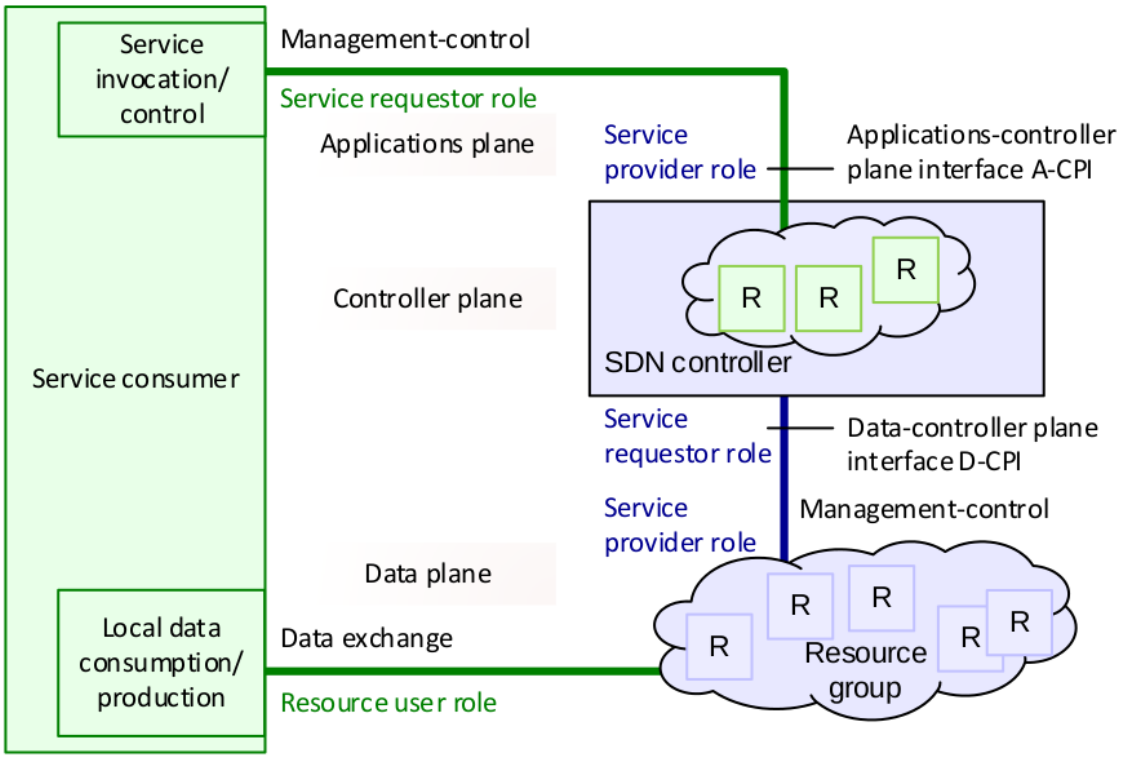
[2]Logički centralizovan sloj ne znači da je kontrolna logika i fizički centralizovana. Naprotiv, produkcijske SDN mreže imaju fizički distribuiranu kontrolnu ravan da bi bile skalabilne, pouzdane i garantovale potrebne QoS [52] [65].

[3]Open Networking Foundation (ONF) je neprofitna organizacija koja razvija, standardizuje i promoviše SDN protokole i tehnologije. Među kompanijama koje finansiraju ONF su Deutsche Telekom, Facebook, Google, Microsoft, Verizon i Yahoo!

[4]https://www.opennetworking.org/sdn-resources/sdn-definition

prvi standard za razmenu podataka između OpenFlow kontrolera (eng. *OpenFlow controller*) i OpenFlow uređaja[5] dizajniran posebno za SDN (Slika 7.7).

Iako je mogućnost programiranja mreže postojala i pre pojave SDN-a [38], programiranje mreže i mrežnih servisa se smatra osnovnom karakteristikom SDN-a [93], [67], [108]. Korisnik aplikacije u aplikativnoj ravni (eng. *application plane*) može posredstvom SDN kontrolera kontrolisati uređaje u ravni podataka (Slika 7.6).



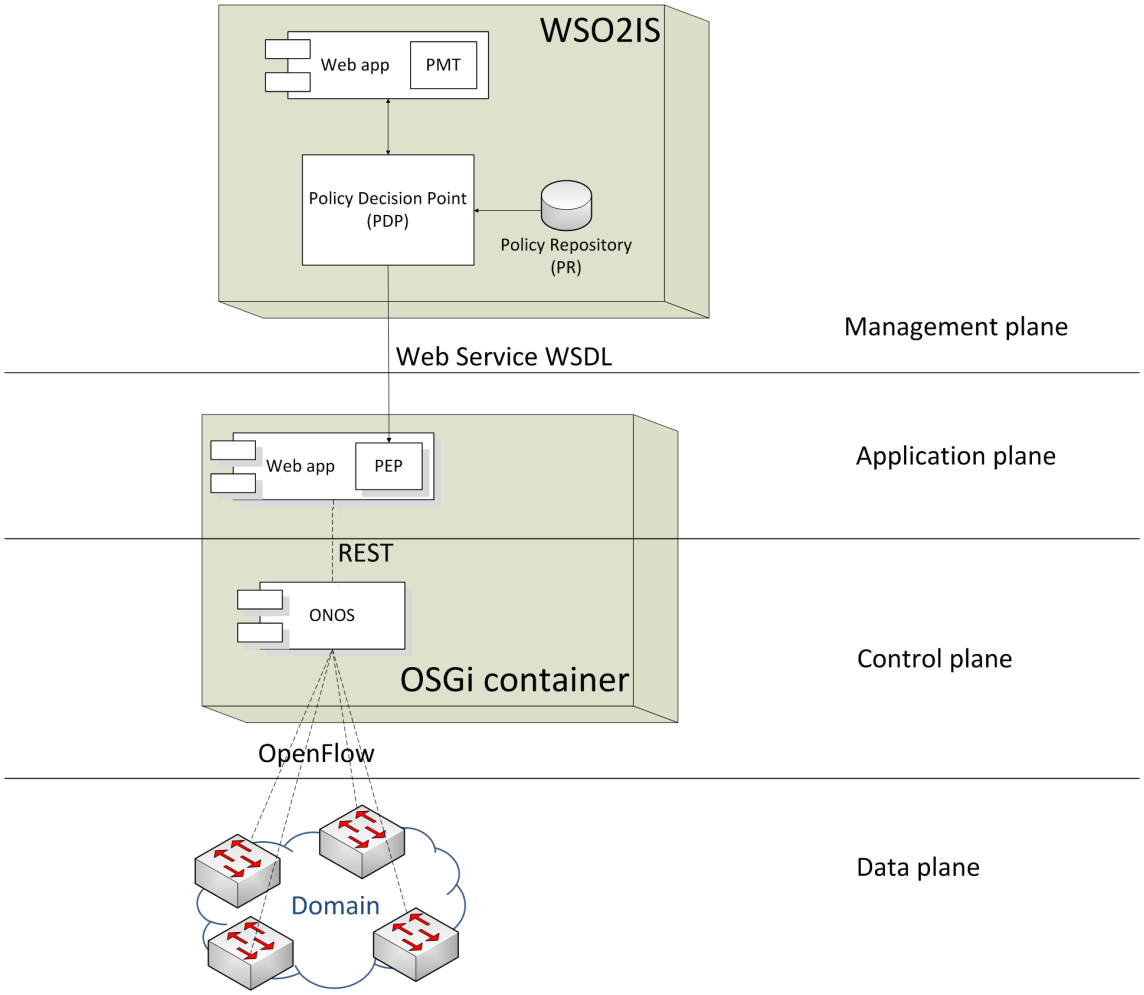
Slika 7.8 Model SDN servisa[6]

Na Slici 7.8 je dat predlog osnovnog modela SDN arhitekture [93] koja, između ostalog, treba da obezbedi okruženje za efikasan razvoj novih mrežnih servisa. SDN kontroler je zadužen za orkestraciju skupa resursa koje korisnik mrežne usluge vidi kao "svoje". Osnovne uloge su administrator (eng. *administrator*), naručilac mrežne usluge (eng. *service requestor*), pružalac mrežne usluge (eng. *service provider*) i korisnik resursa (eng. *resource user*).

[5] [91] koristi termin *SDN Datapath* za logički mrežni uređaj koji oglašava svoje sposobnosti u vezi prosleđivanja i obrade paketa

[6] [93]

7.1.4 Izbor softverskih komponenata



Slika 7.9 Softverske komponente

Sistem za upravljanje višedomenskim putanjama je potpuno decentralizovan i svaki domen ima svoje instance opisanih komponenata sistema (Poglavlje 5.2). Softverske komponente koje čine sistem u okviru jednog domena su WSO2 Identity Server (WSO2IS) 5.1.0 kao federated policy management sistem, OpenFlow kontroler domena[7] je ONOS 1.5.1 (Falcon)[8] dok su kao OpenFlow uređaji korišćeni Open vSwitch-evi 2.5.0[9]. Implementacioni detalji i uloge izabranih softverskih komponenata u predloženom sistemu su prikazane na Slici 7.9.

Aplikativna ravan svakog SDN kontrolera sadrži Web aplikaciju koja programira tokove (eng. *flow*) potrebne za rezervaciju kapaciteta. Moduli za upravljanje XACML polisama WSO2IS i Web aplikacija koja obavlja sinhronizaciju polisa slanjem upita

[7]Domen može koristiti jedan OF kontroler ili klaster OF kontrolera
[8]https://github.com/opennetworkinglab/onos/tree/1.5.1
[9]http://openvswitch.org/releases/openvswitch-2.5.0.tar.gz

repozitorijumima polisa drugih domena i ažuriranjem lokalnog repozitorijuma polisa su u ravni upravljanja SDN kotrolera. Aplikacija u ravni upravljanja omogućava unos polisa u obliku ciljeva na visokom nivou, a aplikacija u kontrolnoj ravni izvršava njihovu rafinaciju (eng. *refinement*) u OpenFlow tokove. FISE koristi Web servise pri Eastbound/Westbound sinhronizaciji podataka, dok se *intent framework*-u ONOS kontrolera pristupa korišćenjem *northbound REST* interfejsa i API-a.

Radi pojednostavljenja softverskog steka i smanjenja broja softverskih komponenata, opšti podaci o Domenu opisani ključnom apstrakcijom *Domain* na Slici 7.1 se čuvaju u repozitorijumu polisa i oglašavaju kao polise tipa "DOMAIN_INFO", a WSO2IS se koristi i kao aplikativni server Web aplikacija iz kontrolne ravni i ravni upravljanja.

7.1.4.1 Ažuriranje i sinhronizacija federativnih polisa

Ažuriranje repozitorijuma federativnih polisa može biti asinhrono, kao posledica aktivacije ili deaktivacije višedomenske usluge, ili sinhrono, odnosno rezultat međudomenske sinhronizacije federativnih polisa. I sinhrona i asinhrona modifikacija repozitorijuma federativnih polisa se izvršavaju kao nedeljive logičke operacije, tj. atomične (eng. *atomic*) su, konzistentne (eng. *consistent*), izolovane (eng. *isolated*) i trajne (eng. *durable*). Da bi se izbeglo korišćenje kompleksnih mehanizama za upravljanje transakcijama ili korišćenje distribuiranih baza podataka, u prototipu je korišćen *optimistic locking pattern* u kom svaka polisa ima verziju koja se inkrementira pri svakom ažuriranju. *Optimistic locking* podrazumeva da se više operacija može završiti bez međusobne interferencije. Stoga operacije ne zaključavaju polise koje koriste. Pre svakog upisa ažurirane polise, operacija proverava da li je neka druga operacija modifikovala polisu tako što poredi svoju verziju polise sa verzijom u repozitorijumu polisa. Ako poređenje otkrije da je polisa modifikovana, radi se *roll-back* operacije i polise vraćaju u prethodno konzistentno stanje.

7.1.5 Realizovani prototip

Kada je mreža konvergirala svi domeni imaju istu sliku topologije federativne mreže i FRSI komponente svih domena sadrže iste verzije svih federativnih polisa. Neka je krajnji korisnik iz domena E zatražio novu višedomensku putanju od tačke X u domenu E do tačke Y u domenu A kapaciteta 20Gbps i prioriteta 1.

Da bi započeo dizajn nove višedomenske putanje, FSI-D/C/A-E treba da odredi izvorišni i odredišni domen na osnovu IP adresa tačaka X i Y, odnosno na osnovu obligacija zadovoljenih polisa tipa "DOMAIN_INFO". FSI-D/C/A-E zatim formira

XACML request da bi odredio skraćenu topologiju iz obligacija zadovoljenih polisa tipa "DYNAMIC" (Slika 7.10).

```
−<Policy PolicyId="domain-A_domain-B" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable" Version="1.0">
   <Description>Domain-A_capabilities</Description>
 −<Target>
  −<AnyOf>
   −<AllOf>
    −<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">dynamic</AttributeValue>
       <AttributeDesignator AttributeId="policy_type" Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"/>
      </Match>
    −<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
       <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:priority" Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource" DataType="http://www.w3.org/2001/XMLSchema#integer"
      MustBePresent="true"/>
      </Match>
    −<Match MatchId="urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal">
       <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">70</AttributeValue>
       <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:reservable_capacity" Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource" DataType="http://www.w3.org
      /2001/XMLSchema#integer" MustBePresent="true"/>
      </Match>
    </AllOf>
  </AnyOf>
 </Target>
 <Rule Effect="Permit" RuleId="domain-A_domain-B"/>
−<ObligationExpressions>
 −<ObligationExpression FulfillOn="Permit" ObligationId="domain_name">
  −<AttributeAssignmentExpression AttributeId="source_domain">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">domain-A</AttributeValue>
    </AttributeAssignmentExpression>
  </ObligationExpression>
 −<ObligationExpression FulfillOn="Permit" ObligationId="destination_domain">
  −<AttributeAssignmentExpression AttributeId="domain_1">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">domain-B</AttributeValue>
    </AttributeAssignmentExpression>
  </ObligationExpression>
 −<ObligationExpression FulfillOn="Permit" ObligationId="in_reservable_capacity">
  −<AttributeAssignmentExpression AttributeId="in_reservable_capacity">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">70</AttributeValue>
    </AttributeAssignmentExpression>
  </ObligationExpression>
 −<ObligationExpression FulfillOn="Permit" ObligationId="out_reservable_capacity">
  −<AttributeAssignmentExpression AttributeId="out_reservable_capacity">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">70</AttributeValue>
    </AttributeAssignmentExpression>
  </ObligationExpression>
 </ObligationExpressions>
</Policy>
```

Slika 7.10 *XACML dynamic polisa*

Nakon dizajna nove višedomenske putanje, FSI-D/C/A-E inicira konfiguraciju u udaljenim domenima pozivom Web servisa *configure_new_path_request* domena koji će biti uključeni u novu višedomensku putanju. U prototipu je atomičnost poziva Web servisa obezbeđena korišćenjem sinhronih, tj. blokirajućih poziva Web servisa. Za promenu konfiguracije lokalnih resursa FSI-D/C/A-E koristi REST API Intent Framework-a ONOS kontrolera. Pošto su svi domeni potvrdili alokaciju i konfiguraciju svojih resursa, FSI-D/C/A-E obaveštava korisnika u uspešnoj aktivaciji tražene putanje.

Domeni koji učestvuju u višedomenskoj putanji unose "ENVIRONMENTAL" polisu koja opisuje novokreiranu višedomensku putanju i menjaju svoje sposobnosti sadržane u polisama tipa "DYNAMIC" posle alokacije i konfiguracije lokalnih resursa. Mreža će ponovo konvergirati i svi domeni će imati konzistentnu sliku mreže kada i domeni koji nisu uključeni u višedomensku putanju ažuriraju oglašene sposobnosti drugih domena tokom narednog ciklusa sinhronizacije polisa.

7.1.6 Poređenje sa sličnim SDN sistemima

Opisani prototip ne treba mešati sa pokušajima da se kontrolna ravan SDN kontrolera distribuira i sinhronizuje kako bi se poboljšala skalabilnost i eliminisala *single point of failure* koju predstavlja centralizovani SDN kontroler [62] ili sa distribuiranom

kontrolnom platformom koja transformiše centralizovane module u kontrolnoj ravni u distribuirane korišćenjem projektnih obrazaca (eng. *design patterns*) i Beehive *framework*-a [131], niti sa *framework*-om za sinhronizaciju kontrolnih ravni SDN kontrolera na osnovu atomičnih transakcija [104]. Svi oni nude rešenja na nivou SDN mreže jednog domena.

GEANT Autobahn servis[10] u kombinaciji sa DynPac *framework*-om [79] se može koristiti za obezbeđivanje višedomenske putanje garantovanog propusnog opsega tokom željenog vremenskog intervala. Međutim, razmena sposobnosti domena je zasnovana na Network Services Interface-Connection Service (NSI-CS) protokolu [100] koji omogućava samo rudimentarnu specifikaciju prava pristupa resursima u udaljenim domenima [76].

Svi spomenuti sistemi imaju zajedničkih elemenata i dodirnih tačaka sa sistemom za upravljanje višedomenskim putanjama, ali ni jedan ne omogućava granularnu kontrolu prava pristupa i korišćenja federativnih resursa u udaljenim domenima koja je u [128] identifikovana kao jedna od ključnih prepreka automatizaciji pružanja mrežnih usluga.

[10] Autobahn - home page

Glava 8

Zaključak

Internet, koji je i sam federativno mrežno okruženje, je vremenom evoluirao od *best-effort packet forwarding* servisa do arhitektura orijentisanih ka pružanju mrežnih servisa. Tradicionalno upravljanje mrežama koje podrazumeva ljudske aktivnosti u većini poslovnih procesa nije dovoljno fleksibilno da se izbori sa porastom kompleksnosti mreža, heterogenom mrežnom infrastrukturom i zahtevima krajnjih korisnika koji često imaju rigorozne zahteve u pogledu kvaliteta servisa (*eng. Quality of Service (QoS)*) i kvaliteta iskustva (*eng. Quality of Experience (QoE)*). Stoga je neophodan što veći stepen automatizacije upravljanja računarskim mrežama.

Komparativnom analizom trenutnog stanja na polju upravljanja višedomenskim mrežnim uslugama i definicije problema upravljanja ovom vrstom usluga utvrđeno je da ne postoje standardi i preporučeni primeri dobre prakse koji bi se mogli primeniti na upravljanje uslugama i resursima u federativnim mrežnim okruženjima. Naime, mrežna usluga se uvek posmatra iz ugla jednog, tipično komercijalnog, pružaoca usluge, a predložena proširenja ne uzimaju u obzir sve aspekte federativnog okruženja na pružanje višedomenske usluge.

U slučajevima kada mrežnu uslugu zajednički pruža više od jednog entiteta (pružalaca usluga, tj. domena) pri čemu su domeni koji učestvuju u zajedničkom pružanju usluga autonomni i nezavisno definišu pravila pristupa za resurse koji učestvuju u pružanju multi-domen usluga, potpuna automatizacija procesa je nemoguća jer je protivrečna suverenosti domena. Naime, jedan od zaključaka je da je automatizovano pružanje višedomenske usluge u labavo spregnutom federativnom okruženju u suprotnosti sa autonomijom domena, jer automatizovano pružanje usluga podrazumeva da se resursi domena mogu konfigurisati iz udaljenih domena kroz autonomnu povratnu petlju. Takva autonomna petlja je u labavo spregnutoj federaciji raskinuta i ljudska intervencija je neophodna u nekim aktivnostima procesa pružanja višedomenske mrežne usluge. Isti zaključak ne važi samo za proces

konfiguracije višedomenske usluge, već i za procese rešavanja problema koji se tiču višedomenskih servisa i upravljanje kvalitetom višedomenske usluge.

Diskutabilno je da li će napredak i unapređenje različitih tehnika i tehnologija virtuelizacije omogućiti udaljenu rezervaciju virtuelnih delova infrastrukture domena koji će biti korišćeni za pružanje višedomenske usluge. Skup resursa (fizičkih ili virtuelnih) koje samostalno administrira jedan entitet je slučaj u kom uslugu pruža jedan domen, bez obzira gde se ti resursi fizički nalaze. Čak i kada je zajedničko korišćenje resursa radi pružanja višedomenske usluge dogovoreno i regulisano na nivou federacije, na operativnom nivou svakim resursom upravlja samo jedan domen.

Dekompozicija procesa pružanja višedomenske usluge je identifikovala ključne komponente i procese potrebne za automatizaciju pružanja višedomenske usluga. Analiza je takođe pokazala da je automatska razmena podataka između domena maksimum automatizacije procesa pružanja višedomenske usluge u labavo spregnutom federativnom okruženju potpuno autonomnih domena. Opisani *Federated Trouble Ticket System (FTTS)* je jedan od načina na koji se može postići automatska razmena ključnih međudomenskih poruka tokom pružanja višedomenske mrežne usluge.

Uporedna analiza modela različitih višedomenskih mrežnih usluga, mrežnih resursa i skupa ograničenja koje domeni mogu da postave je pokazala da je za pojedine mrežne usluge sa dobro definisanom deljenom semantikom usluge moguće realizovati sistem koji bi omogućio svakom domenu da definiše skup pravila koja bi ograničila korišćenje resursa u federativnom okruženju, ali i omogućio prihvatanje zahteva striktno prema definisanim ograničenjima. Sa druge strane, zaključak komparativne analiza i pregleda arhitektura i modela višedomenskog upravljanja uslugama i *policy-based management* arhitektura, jezika i *framework*-a je da se pojedina rešenja mogu primeniti samo u slučajevima kada se federativnim resursima upravlja centralizovano, što, u krajnjoj liniji, svodi upravljanje višedomenskim resursima na slučaj jednog pružaoca mrežne usluge. U slučaju da domeni zadrže potpuni suverenitet nad svojim federativnim resursima, postojeća rešenja nisu dovoljno izražajna da bi obezbedila finu kontrolu prava pristupa potrebnu pri svakodnevnom korišćenju.

Policy-based jezici su utemeljeni na logici, pri čemu je matematička logika prvog reda najopštija od svih korišćenih logika. Ovo za posledicu ima mogućnost atributivnog opisivanja potrebnih ograničenja koje domeni specificiraju za svoje federativne resurse, kao i definisanje meta-modela baziranog na atributima i mogućim skupovima vrednosti atributa. Svaki tip servisa ima svoju semantiku potrebnih atributa i pokazano je instanciranje generičkih polisa za slučaj tipa usluge "višedomenska putanja sa garantovanim kapacitetom". Izborom adekvatnih atributa i skupova njihovih vrednosti je kreiran minimalan skup podataka koji je potreban da

bi se opisali: višedomenska mrežna usluga, resursi i skup ograničenja koje domen postavlja.

Nakon pregleda i analize relevantne literature kao i primera dobre prakse koji se bave višedomenskim mrežnim uslugama, matematičkom zasnovanošću *policy-based management*-a i postojećih jezika, arhitektura i *framework*-a za specifikaciju polisa, pregleda literature o problemima i metodama predikatske logike, pregleda literature o tehnikama virtualizacije mrežnih resursa i SDN-a, primenjene su agilne metodologije za realizaciju prototipa sistema za upravljanje višedomenskim mrežnim uslugama (uspostavljanje putanje sa kraja na kraj) prema realizovanom modelu i protokolima. U simuliranom okruženju je demonstriran prototip za uspostavljanje i održavanje višedomenskih usluga u SDN mrežama. Konfiguracija SDN kontrolera se radi na osnovu semantike instanciranih polisa korišćenjem *northbound* interfejsa kontrolera.

Kreirani sistem polisa u kom polise koriste meta-model podataka sistema se može koristiti u mnogim oblastima gde je ključno obezbediti kontrolu prava pristupa nad deljenim resursima. Predložena arhitektura sistema za pristup resursima u udaljenim domenima je proširiva i na druge probleme, jer je veoma čest slučaj da je za pružanje neke IT usluge potrebno angažovanje više različitih entiteta sa različitim nivoima prava pristupa resursima.

Rešavanje problema federativnog upravljanja uslugama ili makar pronalaženje mehanizama kojima bi se moglo doći do automatizacije procesa upravljanja uslugama u specijalnim slučajevima bi se stvorio prostor za realizaciju mnogih novih usluga na Internetu, jednostavniju i pouzdaniju saradnju različitih entiteta i viši kvalitet usluga pruženih posredstvom Interneta. Takođe, pošto je problem federativnog upravljanja generalnim IT uslugama u višedomenskom okruženju u kojem različiti entiteti imaju različita prava pristupa resursima čest, generalizacijom rešenja problema federativnog upravljanja mrežnim servisima bi moglo da se nađe rešenje za širi skup problema.

Literatura

[1] Anderson, A. (2003). Evaluating XACML as a Policy Language. Technical report, OASIS.

[2] Ascom (2003). OSS/J-facilitated Trouble Ticketing at Vodafone.

[3] Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and Xiao, X. (2015). Overview and Principles of Internet Traffic Engineering.

[4] Bagchi, A., Caruso, F., Mayer, A., and Roman, R. (2009). Framework to Achieve Multi-Domain Service Management. 08854:287–290.

[5] Bahrak, B., Deshpande, A., and Park, J.-m. J. (2009). A POLICY REASONER FOR POLICY-BASED DYNAMIC SPECTRUM ACCESS. In *Software Defined Radio Forum Technical Conference and Product Exposition*.

[6] Bahrak, B., Deshpande, A., Whitaker, M., and Jung-Min Park (2010). BRESAP: A Policy Reasoner for Processing Spectrum Access Policies Represented by Binary Decision Diagrams. In *2010 IEEE Symposium on New Frontiers in Dynamic Spectrum (DySPAN)*, pages 1–12. IEEE.

[7] Bandara, A., Damianou, N., Lupu, E., Sloman, M., and Dulay, N. (2008). Policy-based management. In *Handbook of Network and System Administration*, pages 507–563.

[8] Belinga, J. B., Oumina, H., and Ranc, D. (2007). A vision for reliable network services architecture. *IFIP International Federation for Information Processing*, 229:327–337.

[9] Bertino, E., Bonatti, P. A., and Ferrari, E. (2001). TRBAC: A temporal role-based access control model. *ACM Transactions on Information and System Security*, 4(3):191–233.

[10] Boote, J., Boyd, E., Brown, A., Grigoriev, M., Metzger, J., Swany, M., Tierney, B., Zekauskas, M., Li, Y.-T., and Zurawski, J. (2009). Instantiating a Global Network Measurement Framework. In *4th Workshop on Real Overlays and Distributed Systems (ROADS'09) Co-located with the 22nd ACM Symposium on Operating Systems Principles (SOSP)*, pages 1–10.

[11] Boutaba, R. and Aib, I. (2007). Policy-based Management: A Historical Perspective. *Journal of Network and Systems Management*, 15(4):447–480.

[12] Bradshaw, J. M., Uszok, A., and Montanari, R. (2014). Policy-Based Governance of Complex Distributed Systems: What Past Trends Can Teach Us about Future Requirements. In *Agile Computing*, chapter Progress a, pages 259–284. CRC Press.

[13] Bradshaw, J.M. A. Uszok, M. Breedy, L. Bunch, T.C. Eskridge, P.J. Feltovich, M. Johnson, J. Lott and Vignati., M. (2013). The KAoS Policy Services Framework. *Eighth Cyber Security and Information Intelligence Research Workshop (CSIIRW 2013). Oak Ridge, TN: Oak Ridge National Labs.*

[14] Brennan, R., Feeney, K., Keeney, J., O'Sullivan, D., Fleck, J., Foley, S., and van der Meer, S. (2010). Multidomain IT architectures for next-generation communications service providers [Next-Generation Telco IT Architectures. *IEEE Communications Magazine*, 48(8):110–117.

[15] Brucker, A. D. and Petritsch, H. (2009). Extending access control models with break-glass. In *Proceedings of the 14th ACM symposium on Access control models and technologies - SACMAT '09*, page 197, New York, New York, USA. ACM Press.

[16] Bussard, L., Nano, A., and Pinsdorf, U. (2009). Delegation of access rights in multi-domain service compositions. *Identity in the Information Society*, 2(2):137–154.

[17] Carlos, J. and Delgado, M. (2014). *Continued Rise of the Cloud.* Computer Communications and Networks. Springer London, London.

[18] Ceri, S., Gottlob, G., and Tanca, L. (1989). What you always wanted to know about Datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166.

[19] Chadha, R. (2004). Applications of policy-based network management. *2004 IEEE/IFIP Network Operations and Management Symposium (IEEE Cat. No.04CH37507)*, 1.

[20] Chadwick, D. and Otenko, A. (2004). Implementing Role Based Access Controls using X. 509 Privilege Management-the PERMIS Authorisation Infrastructure. *Security and privacy in advanced*, pages 1–13.

[21] Chadwick, D., Zhao, G., Otenko, S., Laborde, R., Su, L., and Nguyen, T. A. (2008). PERMIS: a modular authorization infrastructure. *Concurrency and Computation: Practice and Experience*, 20(11):1341–1357.

[22] Chadwick, D. W. and Otenko, A. (2003). The PERMIS X.509 role based privilege management infrastructure. *Future Generation Computer Systems*, 19(2):277–289.

[23] Chadwick, D. W., Otenko, S., and Nguyen, T. A. (2009). Adding support to XACML for multi-domain user to user dynamic delegation of authority. *International Journal of Information Security*, 8(2):137–152.

[24] Chen, F. and Sandhu, R. S. (1996). Constraints for role-based access control. In *Proceedings of the first ACM Workshop on Role-based access control - RBAC '95*, pages 14–es, New York, New York, USA. ACM Press.

[25] Chen, L., Gasparini, L., and Norman, T. (2013). XACML and risk-aware access control. In *Proceedings of the 10th International Workshop on Security in Information ...*, pages 66–75.

[26] Clocksin, W. F. and Mellish, C. S. (2003). *Programming in Prolog*, volume 53. Springer-Verlag, New York, fifth edit edition.

[27] Crampton, J. and Morisset, C. (2012). Towards A Generic Formal Framework for Access Control Systems. *arXiv:1204.2342*.

[28] Damianou, N., Bandara, A. K., Sloman, M., and Lupu, E. C. (2002). A survey of policy specification approaches. Technical Report April.

[29] Damianou, N., Dulay, N., Lupu, E., and Sloman, M. (2000). Ponder: A Language for Specifying Security and Management Policies for Distributed Systems. *Computing*, 13(DoC 2000/1):1–49.

[30] Davy, S., Jennings, B., and Strassner, J. (2007). The policy continuum–a formal model. *. . . of the Second IEEE International . . . .*

[31] Della-Libera, G., Gudgin, M., Hallam-Baker, P., Hondo, M., and Granqvist, H. (2005). Web Services Security Policy Language (WS-SecurityPolicy).

[32] Desai, P. (2012). *The Break-the-Glass (BtG) principle in Access Control.* PhD thesis.

[33] Donmez, B., Pina, P. E., and Cummings, M. L. (2008). Evaluation criteria for human-automation performance metrics. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems - PerMIS '08*, page 77, New York, New York, USA. ACM Press.

[34] Douville, R., Le Roux, J.-L., Rougier, J.-L., and Secci, S. (2008). A service plane over the PCE architecture for automatic multidomain connection-oriented services. *IEEE Communications Magazine*, 46(6):94–102.

[35] ENNAHBAOUI, M. and ELHAJJI, S. (2013). Study of Access Control Models. In *Proceedings of the World Congress on Engineering*, volume 2. Proceedings of the World Congress on Engineering 2013 Vol II, WCE 2013, July 3 - 5, 2013, London, U.K. ISBN: 978-988-19252-8-2 ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online) WCE.

[36] Estrin, D. L. (1985). Inter-organizational networks: Stringing wires across administrative boundaries. *Computer Networks and ISDN Systems*, 9(4):281–295.

[37] Famaey, J. and De Turck, F. (2012). Federated management of the Future Internet: Status and challenges. *International Journal of Network Management*, 22(6):508–528.

[38] Feamster, N., Rexford, J., and Zegura, E. (2014). The road to SDN. *ACM SIGCOMM Computer Communication Review*, 44(2):87–98.

[39] Feeney, K., Brennan, R., Keeney, J., Thomas, H., Lewis, D., Boran, A., and O'Sullivan, D. (2010). Enabling decentralised management through federation. *Computer Networks*, 54(16):2825–2839.

[40] Ferreira, A., Correia, R., Brito, M., and Antunes, L. (2011). Usable access control policy and model for healthcare. In *2011 24th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 1–6. IEEE.

[41] Fey, G. and Drechsler, R. (2006). Minimizing the number of paths in BDDs: Theory and algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(1):4–11.

[42] Fey, G. and Drehsler, R. (2003). A Hybrid Approah Combining Symbolic and Strutural Tehniques for Disjoint SOP Minimization. In *In Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI)*, pages 54–60.

[43] Foster, N., Guha, A., Reitblatt, M., Story, A., Freedman, M. J., Katta, N. P., Monsanto, C., Reich, J., Rexford, J., Schlesinger, C., Walker, D., and Harrison, R. (2013). Languages for software-defined networks. *IEEE Communications Magazine*, 51(2):128–134.

[44] Foster, N., Harrison, R., and Freedman, M. (2011). Frenetic: A network programming language. *Proc. ACM ICFP Conf.*, pages 279–291.

[45] Gandía Carriedo, Maria Isabel Litröm, S., Limberg, S., Vuletic, P., and Szegedi, P. (2012). TF-NOC Software Tools.

[46] Gasparini, L. (2013). *Risk-Aware Access Control and XACML*. PhD thesis.

[47] Gredler, H., Medved, J., Previdi, S., Farrel, A., and Ray, S. (2016). North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP.

[48] Gurevich, Y. (2012). Datalog: A perspective and the potential. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7494 LNCS, pages 9–20.

[49] Halpern, J. Y. and Weissman, V. (2008). Using First-Order Logic to Reason about Policies. V(February):1–39.

[50] Hardy, N. (1988). The Confused Deputy. *ACM SIGOPS Operating Systems Review*, 22(4):36–38.

[51] Hayton, R. and Moody, K. (1997). An Open Architecture for Secure Interworking Services. In *October*, pages 315–321.

[52] Jain, S., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., Vahdat, A., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., and Zhou, J. (2013). B4: Experience with a Globally-Deployed Software DefinedWA. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM - SIGCOMM '13*, page 3.

[53] Jajodia, S., Samarati, P., and Subrahmanian, V. S. (1997). A logical language for expressing authorizations. *Proceedings. 1997 IEEE Symposium on Security and Privacy (Cat. No.97CB36097)*, pages 31–42.

[54] Janičić, P. (2008). *Matematička logika u računarstvu*.

[55] Jennings, B., Brennan, R., Donnelly, W., Foley, S. N., Lewis, D., O'Sullivan, D., Strassner, J., and Van Der Meer, S. (2009). Challenges for federated, autonomic network management in the future internet. In *2009 IFIP/IEEE International Symposium on Integrated Network Management-Workshops, IM 2009*, pages 87–92.

[56] Jürjens, J. (2002). UMLsec: Extending UML for secure systems development. *Proceedings of the 5th International Conference on The Unified Modeling Language*, pages 412–425.

[57] Kagal, L. (2002). Rei : A Policy Language for the Me-Centric Project. Technical Report HPL-2002-270, HP Labs.

[58] Kagal, L. and Berners-Lee, T. (2005). Rein: Where policies meet rules in the semantic web. *Computer Science and Artificial ....*

[59] Kandala, S., Sandhu, R., and Bhamidipati, V. (2011). An Attribute Based Framework for Risk-Adaptive Access Control Models. In *2011 Sixth International Conference on Availability, Reliability and Security*, pages 236–241. IEEE.

[60] Karp, A. H., Haury, H., and Davis, M. H. (2010). From ABAC to ZBAC : The Evolution of Access Control Models. *ISSA Journal*, (April):22–30.

[61] Katz, D., Kompella, K., and Yeung, D. (2003). Traffic Engineering (TE) Extensions to OSPF Version 2 Status.

[62] Kevin Phemius, M. B. and Leguay, J. (2014). DISCO: Distributed Multi-domain SDN Controllers. *IEEE*.

[63] Khamadja, S., Adi, K., and Logrippo, L. (2013). Designing flexible access control models for the cloud. *SIN 2013 - Proceedings of the 6th International Conference on Security of Information and Networks*, pages 225–232.

[64] Kolovski, V. (2008). a Logic-Based Framework for Web Access Control Policies. *Dissertation*, page 258.

[65] Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., Others, and Shenker, S. (2010). Onix: A distributed control platform for large-scale production networks. *OSDI, Oct*, pages 1—6.

[66] Krautsevich, L., Lazouski, A., Martinelli, F., and Yautsiukhin, A. (2013). Towards Policy Engineering for Attribute-Based Access Control. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8292 LNCS, pages 85–102.

[67] Kreutz, D., Ramos, F. M. V., Esteves Verissimo, P., Esteve Rothenberg, C., Azodolmolky, S., and Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):14–76.

[68] Lampson, B. W. (1974). Protection. *ACM SIGOPS Operating Systems Review*, 8(1):18–24.

[69] Latré, S., Famaey, J., and Turck, F. D. (2014). A semantic context exchange process for the federated management of the future internet. *International Journal of Network Management*, 24(1):1–27.

[70] Lazou, C. (2008). Grid Infrastructure Organization. *European Parliament Magazine*, (265):8–9.

[71] Li, N. and Mitchell, J. C. (2002). Datalog with Constraints: A Foundation for Trust Management Languages. *Padl*, 2562(Chapter 6):58–73.

[72] Lodderstedt, T., Basin, D. A., and Doser, J. (2002). SecureUML: A UML-Based Modeling Language for Model-Driven Security. In *UML*, volume 2460, pages 426–441.

[73] Loo, B. T., Condie, T., Hellerstein, J. M., Maniatis, P., Roscoe, T., and Stoica, I. (2005a). Implementing declarative overlays. *ACM SIGOPS Operating Systems Review*, 39(5):75.

[74] Loo, B. T., Hellerstein, J. M., Stoica, I., and Ramakrishnan, R. (2005b). Declarative routing. In *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '05*, page 289, New York, New York, USA. ACM Press.

[75] Lutoff, D. (2007). OSS common API—final release.

[76] MacAuley, J., Kudoh, T., Guok, C., and Roberts, G. (2016). Applying Policy in the NSI environment. *Grid Forum Document (GFD), Recommendation (R)*.

[77] Maw, H., Xiao, H., Christianson, B., and Malcolm, J. (2014). A Survey of Access Control Models in Wireless Sensor Networks. *Journal of Sensor and Actuator Networks*, 3(2):150–180.

[78] McGraw, R. (2009). Risk-adaptable access control (radac). *NIST Privilege (Access) Management Workshop*.

[79] Mendiola, A., Astorga, J., Jacob, E., Stamos, K., Juszczyk, A., Dombek, K., Vuleta-Radoičić, J. J., and Aznar, J. e. I. (2016). SDN-enabled AutoBAHN. Envisioning future Bandwidth on Demand services. In *TNC16 - Networking Conference (12-16 June 2016, Prague, Czech Republic)*.

[80] Meskill, B., Balasubramaniam, S., Brennan, R., Feeney, K., and Jennings, B. (2013). Federation Lifecycle Management Incorporating Coordination of Bio-inspired Self-management Processes. *Journal of Network and Systems Management*, 21(4):650–676.

[81] Meyners, M., Driss, B., and Feger, U. (2008). TMF615 "OSS Identity Management".

[82] Mitra, N. and Lafon, Y. (2007). SOAP Version 1.2 Part 0: Primer (Second Edition) W3C Recommendation 27 April 2007. *W3C*, (April):1–57.

[83] Moniruzzaman, M. and Barker, K. (2011). Delegation of access rights in a privacy preserving access control model. *2011 9th Annual International Conference on Privacy, Security and Trust, PST 2011*, pages 124–133.

[84] Monk, J. D. (1976). *Mathematical Logic*. Springer New York, New York, NY.

[85] Monsanto, C., Reich, J., Foster, N., Rexford, J., and Walker, D. (2013). Composing software-defined networks. *Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation*, pages 1–14.

[86] Moore, B., Ellesson, E., Strassner, J., and Westerinen, A. (2001). RFC3060 - Policy Core Information Model – Version 1 Specification Status.

[87] NEMA/COCIR/JIRA (2004). Break-Glass – An Approach to Granting Emergency Access to Healthcare Systems. Technical Report December.

[88] Nguyen, P. H., Nain, G., Klein, J., Mouelhi, T., and Le Traon, Y. (2014). Modularity and dynamic adaptation of flexibly secure systems: Model-driven adaptive delegation in access control management. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8400(JANUARY):109–144.

[89] Open Networking Foundation (2012). Software-Defined Networking: The New Norm for Networks [white paper]. *ONF White Paper*, pages 1–12.

[90] Open Networking Foundation (2013a). Common Information Model Overview version 1.1. *ONF Technical Recommendation*, pages 1–9.

[91] Open Networking Foundation (2013b). SDN Architecture Overview. *ONF Specification*, pages 1–5.

[92] Open Networking Foundation (2015). Core Information Model (CoreModel). *ONF Technical Recommendation*, pages 1–46.

[93] Open Networking Foundation (2016). SDN Architecture 1.1. *ONF Technical Reference*, pages 1–59.

[94] Parducci, B. and Lockhart, H. (2013). eXtensible Access Control Markup Language (XACML) Version 3.0. *OASIS Standard*, (January):1–154.

[95] Phan, T., Han, J., Schneider, J.-g., Ebringer, T., and Rogers, T. (2008). A Survey of Policy-Based Management Approaches for Service Oriented Systems. *19th Australian Conference on Software Engineering*, pages 392–401.

[96] Ramli, C. D. P. K., Nielson, H. R., and Nielson, F. (2014). The logic of XACML. *Science of Computer Programming*, 83:80–105.

[97] Rao, P., Lin, D., Bertino, E., Li, N., and Lobo, J. (2009). An algebra for fine-grained integration of XACML policies. In *Proceedings of the 14th ACM symposium on Access control models and technologies.New York, NY, USA*, pages 63–72.

[98] Rao, P., Lin, D., Bertino, E., Li, N., and Lobo, J. (2011). Fine-grained integration of access control policies. *Computers & Security*, 30(2-3):91–107.

[99] Rekhter, Y., Li, T., and Hares, S. (2006). A Border Gateway Protocol 4 (BGP-4). *RFC4271*.

[100] Roberts, G., Kudoh, T., Monga, I., Sobieski, J., MacAuley, J., and Guok, C. (2014). NSI Connection Service v2.0. *Gfd*, pages 1–119.

[101] Rudell, R. L. (1986). Multiple-Valued Logic Minimization for PLA Synthesis. *Memorandum No. UCB/ERL M86-65*, (June).

[102] Sahafizadeh, E. and Parsa, S. (2010). Survey on access control models. *Future Computer and Communication (ICFCC), 2010 2nd International Conference on*, 1:V1–1–V1–3.

[103] Schefer-Wenzl, B. S., Bukvova, H., and Strembeck, M. (2014). A review of delegation and break-glass models for flexible access control management. *Lecture Notes in Business Information Processing*, 183:93–104.

[104] Schiff, L., Schmid, S., and Kuznetsov, P. (2016). In-Band Synchronization for Distributed SDN Control Planes. *ACM SIGCOMM Computer Communication Review*, 46(1):37–43.

[105] Schulzrinne, H., Tschofenig, H., Morris, J., Cuellar, J., Polk, J., and Rosenberg, J. (2007). Common Policy: A Document Format for Expressing Privacy Preferences.

[106] Serrano, M., Davy, S., Johnsson, M., Donnelly, W., and Galis, A. (2011). *Review and designs of federated management in future internet architectures*.

[107] Sevasti, A., Vuletic, P., Kalogeras, D., Giertych, M., Parniewicz, D., and Pajin, D. (2011). Deliverable D.J2.1.1: Information Schemas and Workflows for Multi-Domain Control and Management Functions.

[108] Shinde, M. B. and Tamhankar, S. G. (2013). Review : Software Defined Networking and OpenFlow. *International Journal of Scientific Research in Network Security and Communication*, 1(2):18–20.

[109] Singh, M. and Patterh, M. (2010). Formal specification of common criteria based access control policy model. *International Journal of Network Security*, 11(3):139–148.

[110] Sloman, M. (1994). Policy driven management for distributed systems. *Journal of Network and Systems Management*, 2(4):333–360.

[111] Strassner, J. (2003). *Policy-Based Network Management: Solutions for the Next Generation*. Morgan Kaufmann, 1st edition.

[112] Strassner, J., Souza, J. N., Meer, S., Davy, S., Barrett, K., Raymer, D., and Samudrala, S. (2009a). The Design of a New Policy Model to Support Ontology-Driven Reasoning for Autonomic Networking. *Journal of Network and Systems Management*, 17(1-2):5–32.

[113] Strassner, J., van der Meer, S., O'Sullivan, D., and Dobson, S. (2009b). The Use of Context-Aware Policies and Ontologies to Facilitate Business-Aware Network Management. *Journal of Network and Systems Management*, 17(3):255–284.

[114] Thomas, R. K. and Sandhu, R. S. (1998). Task-based authorization controls (TBAC): a family of models for active and enterprise-oriented authorization management. In Lin, T. Y. and Qian, S., editors, *Securosis*, number April 2016 in IFIP Advances in Information and Communication Technology, pages 166–181. Springer US, Boston, MA.

[115] Tmf, B., Forum, T. M., and Version, A. (2012). Federated Service Management for Defense (Catalyst). pages 1–87.

[116] TMForum (2009). Business Process Framework-Addendum B: Business to Business Integration.

[117] TMForum (2010a). Business Process Framework (eTOM) - Addendum D: Process Decompositions and Descriptions.

[118] TMForum (2010b). OSS/J trouble ticket API, TMF890.

[119] Tomašević, M. (2008). *ALGORITMI I STRUKTURE PODATAKA*. Akademska Misao/Academic Mind.

[120] Toninelli, A., Montanari, R., Kagal, L., and Lassila, O. (2007). Proteus: A semantic context-aware adaptive policy model. *Proceedings - Eighth IEEE International Workshop on Policies for Distributed Systems and Networks, POLICY 2007*, pages 129–138.

[121] Tonti, G., Bradshaw, J. M., Jeffers, R., Montanari, R., Suri, N., and Uszok, A. (2003). Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder. In Fensel, D., Sycara, K., and Mylopoulos, J., editors, *The Semantic Web - ISWC 2003*, volume 2870/2003, pages 419–437. Springer.

[122] Twidle, K., Dulay, N., Lupu, E., and Sloman, M. (2009). Ponder2: A policy system for autonomous pervasive environments. *Proceedings of the 5th International Conference on Autonomic and Autonomous Systems, ICAS 2009*, pages 330–335.

[123] Uszok, A., Bradshaw, J., and Johnson, M. (2004). KAoS policy management for semantic web services. *Intelligent Systems, IEEE*, 19(4):32–41.

[124] Uszok, A. and Bradshaw, J. M. (2013). KAoS Policy Services Framework: User Guide. (January):1–65.

[125] Vacca, J. R. (2009). *Computer and Information Security Handbook*, volume XXXIII. Morgan Kaufmann Publishers.

[126] Verma, D. (2002). Simplifying network administration using policy-based management. *IEEE Network*, 16(2):20–26.

[127] Vuletić, P. A. and Sevasti, A. G. (2010). A Network Management Architecture proposal for the GEANT NREN environment. *TERENA Networking Conference2*.

[128] Vuletić, P. V., Vuleta-Radoičić, J. J., and Kalogeras, D. (2015). Federated trouble ticket system for service management support in loosely coupled multi-domain environments.

[129] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., and Waldbusser, S. (2001). Terminology for Policy-Based Management. *RFC 3198*, pages 1–22.

[130] Yampolskiy, M., Hommel, W., Liu, F., König, R., Metzker, M. G., and Schiffers, M. (2012). Link repair in managed multi-domain connections with end-to-end quality guarantees. *International Journal of Network Management*, 22(6):494–507.

[131] Yeganeh, S. H. and Ganjali, Y. (2014). Beehive. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks - HotNets-XIII*, pages 1–7, New York, New York, USA. ACM Press.

[132] Zhang, R. and Vasseur, J.-P. (2005). MPLS Inter-Autonomous System (AS) Traffic Engineering (TE) Requirements.

[133] Zhao, H., Lobo, J., and Bellovin, S. M. (2008). An Algebra for Integration and Analysis of Ponder2 Policies. In *2008 IEEE Workshop on Policies for Distributed Systems and Networks*, pages 74–77. IEEE.

[134] Zisiadis, D., Kopsidas, S., Tsavli, M., and Cessieux, G. (2011). The Network Trouble Ticket Data Model (NTTDM).

Dodatak A

FTTS realizacija

Integrated Trouble Ticket System – installation instructions

Table of Contents

Preface

The document presents integrated trouble ticket system (ITTS) installation and configuration manual. The described installation steps are based on experience gained during design and development of a prototype deployed on QA Testbed virtual machines: sim-test (https://issues.geant.net/jira/browse/QATB-31), jira1test.qalab.geant.net (https://issues.geant.net/jira/browse/QATB-32), rt1test.qalab.geant.net (https://issues.geant.net/jira/browse/QATB-30), rt2test.qalab.geant.net (https://issues.geant.net/jira/browse/QATB-49), rt3test.qalab.geant.net (https://issues.geant.net/jira/browse/QATB-50) and jira2test.qalab.geant.net (https://issues.geant.net/jira/browse/QATB-64). After the prototype has been created and tested snapshots of these virtual machines are made.

Integrated trouble ticket system overview

The integrated trouble ticket system consists of trouble ticket systems (TTSes) and a Service inventory management (SIM) prototype. Interconnection among ITTS components is achieved by appropriate configuration of software that is already in use as a service supporting tool by majority of NRENs and implementation of standard interfaces proposed by OSS/J initiative. ITTS architecture, shown in Figure 1., is fully decentralized as every NREN has its own trouble ticket system, not necessarily exclusively dedicated to interdomain problems. Interconnection mechanism among TTSes is email exchange.

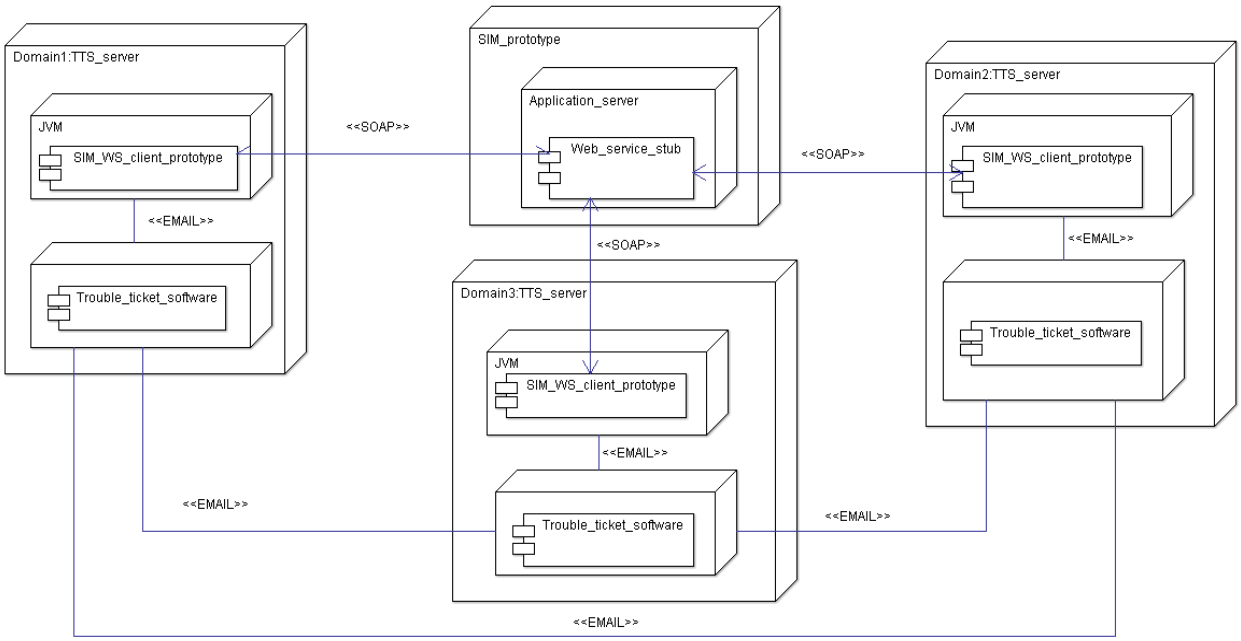


Figure 1: Integrated trouble ticket system - architectural overview

Integrated trouble ticket system is based on common features supported by most of trouble ticket software. In order to be a part of a ITTS, NRENs' TTSes must have following features:

- Ticket types – several types of tickets can exist within TTS, i. e. custom ticket types can be configured.
- Custom fields – Customizable set of ticket attributes (fields).
- Customizable ticket lifecycle – ticket lifecycle (i. e. state machine which determines ticket lifecycle that is defined by workflow within the institution that ticket follows) is customizable and different types of tickets can have different workflows.
- Custom conditions – conditions used to determine whether a ticket's transition to next state is possible can be customized.
- Customizable post-functions – functions that are performed after the transition is complete (known as post functions or scrips) are customizable.
- Ticket creation can be triggered by reception of the email or TTS can be easily customized or extended to create a ticket when it receives a email.

- State transition can be triggered by email.
- Post-functions can be customized to send emails.
- TTS can create tickets and notification emails according to customized templates.

Propagation of pieces of information about status of the particular service instance is achieved by automatic creation and resolution of tickets in TTSes that belong to NRENs included in the service instance and custom configuration of trouble resolution workflow (lifecycle, state machine) in TTS. All custom workflows, shown in Fugure 2. and Figure 3. are extensions of the “Base trouble ticket state model” specified in JSR091.

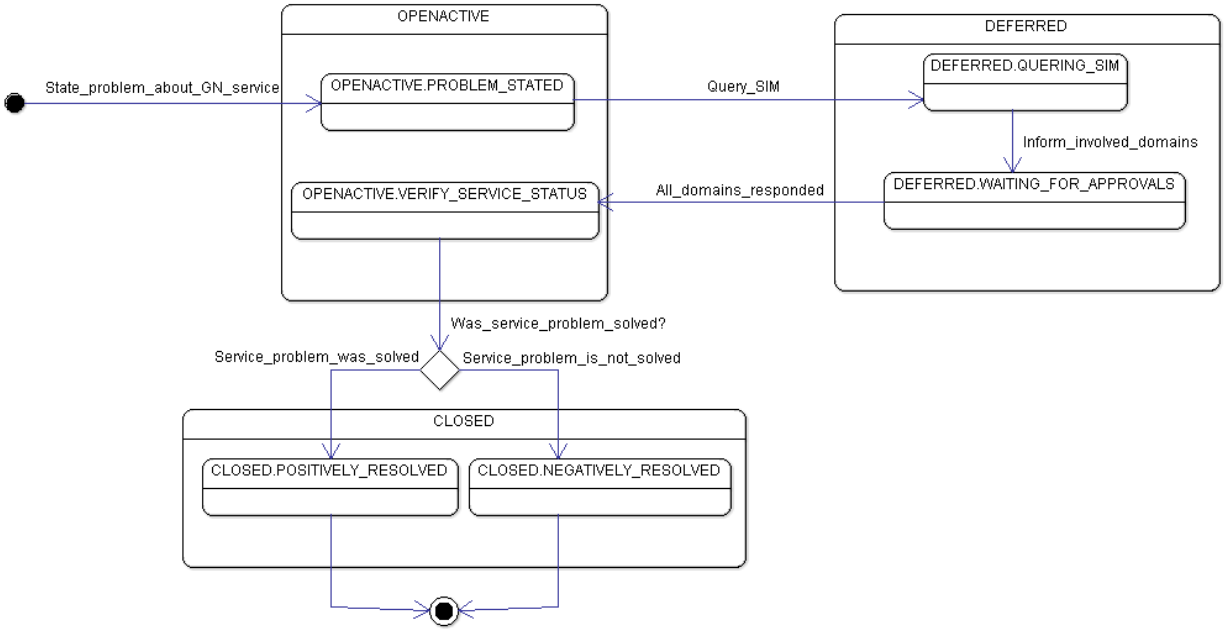


Figure 2: Lifecycle of the ticket whose creation initiates creation of tickets in other domains

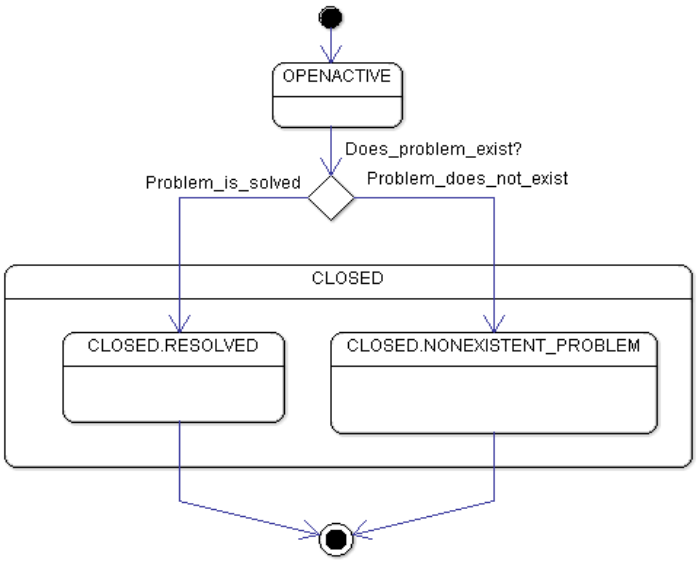


Figure 3: Lifecycle of the automatically created ticket

The ITTS supports following scenario:

- A trouble ticket about some GN3 service instance problem is somehow (manually or automatically) created. State of the ticket is “OPEN.ACTIVE.PROBLEM_STATED”.
- Post function which is executed after ticket creation consults SIM stub about other domains that participate in the affected service instance, gets addresses of TTSes within the domains participating in

the service instance and sends mails to the acquired addresses. All necessary state transitions from “OPEN.ACTIVE.PROBLEM_STATED” to “DEFERRED.WAITING_FOR_APPROVALS” are triggered automatically.

- Tickets in TTSes in other domains included in the service will be automatically created after the emails from 2. have been received. Initial state of these tickets is “OPENACTIVE”.
- A network administrator responsible for service maintenance checks whether problem exists in his/her domain, checks the status of the service instance in his/her domain, probably fixes the problem in his/her domain and, at the end, changes the state of the ticket to: “CLOSED.RESOLVED” or “CLOSED.NONEXISTENT_PROBLEM”. After the state transition, post function sends email to the TTS in the domain that initiated creation of the ticket.
- After reception of all emails from 4, the state of the first ticket is automatically changed to “OPENACTIVE.VERIFY_SERVICE_STATUS” an the owner of the first ticket has to check the status of the service. According to the status of the service, the state of the ticket will be changed to “CLOSED.POSITIVELY_RESOLVED” or “CLOSED.NEGATIVELY_RESOLVED”.

ITTS prototype

The developed ITTS prototype is comprised of several instances of JIRA 5.1.5 and Request Tracker 4.0.5 as trouble ticket systems and Service inventory management (SIM) prototype. SIM prototype is an implementation of Web integration profile specified by OSS Common API. Web service stub is described by OSSJ-Common-v1-5.wsdl. SIM prototype is deployed on application server JBoss 5.1.0.GA, while a SIM_WS_client_prototype is a stand-alone Java application. Interface between Web service stub and its client is OSSJ-Common-v1-5.wsdl, while TTSes use only emalis in order to exchange information with the rest of the world.
SIM stub and its client are the only “new” software. Everything else was done by appropriate configuration of the trouble ticket software and mail transfer agents. SIM stub and its client were generated automatically by Eclipse's plugin “JBoss Toll”.

Request Tracker – installation and configuration

Assumptions:

- Virtual host name is rt2test.qalab.geant.net
- Operating system is Linux

Postfix configuration:

1. Disable selinux: in file `/etc/sysconfig/selinux` change line
   `SELINUX=enforcing`
   to
   `SELINUX=disabled`
2. Reboot virtual machine
   `reboot now`
3. Copy service inventory management web service client implementation (`wsKlijent.jar`) to
   `/Jovana/wsKlijent.jar` with appropriate permissions:
   `chmod 777 /Jovana/wsKlijent.jar`
4. Change postfix configuration:
   - Change file `/etc/postfix/main.cf` insert line:
     `myhostname = rt2test.qalab.geant.net`
   - Change file `/etc/postfix/main.cf` comment line:
     `inet_interfaces = localhost`
   - Change file `/etc/postfix/main.cf` uncomment line:
     `inet_interfaces = all`
   - Change file `/etc/postfix/main.cf` comment line:
     `mydestination = $myhostname, localhost.$mydomain, localhost`
   - Change file `/etc/postfix/main.cf` insert line:
     `mydestination = $myhostname, localhost.localdomain, localhost`
   - Modify `/etc/aliases` insert lines:

```
sim_client: "|/usr/lib/jvm/jre-1.6.0-openjdk.x86_64/bin/java -jar
-Djava.endorsed.dirs=/Jovana/lib/endorsed /Jovana/wsKlijent.jar > /test1.txt"

# RT aliases
reply_query_sim: "|/opt/rt4/bin/rt-mailgate --queue Query_SIM --action
correspond --url http://rt2test.qalab.geant.net/"

comment_query_sim: "|/opt/rt4/bin/rt-mailgate --queue Query_SIM
--action comment --url http://rt2test.qalab.geant.net/"

interconnection_prototype_reply: "|/opt/rt4/bin/rt-mailgate --queue
Tickets_in_other_TTSes --action correspond --url http://rt2test.qalab.geant.net/"

interconnection_prototype_comment: "|/opt/rt4/bin/rt-mailgate --queue
Tickets_in_other_TTSes --action comment --url http://rt2test.qalab.geant.net/"

reply_others: "|/opt/rt4/bin/rt-mailgate --queue
Service_problems_reported_by_others --action correspond --url
http://rt2test.qalab.geant.net/"

comment_others: "|/opt/rt4/bin/rt-mailgate --queue
Service_problems_reported_by_others --action comment --url
http://rt2test.qalab.geant.net/"
```

   - Create file `/test1.txt` with appropriate permissions:
     `touch /test1.txt`

```
chmod 777 /test1.txt
```

Request Tracker insallation and configuration:

1. Install or upgrade Perl to 5.1.16 or later:
   ```
   wget http://www.cpan.org/src/5.0/perl-5.16.1.tar.gz
   tar -xzvf perl-5.16.1.tar.gz
   cd perl-5.16.1
   sh Configure -de
   make
   make test
   make install
   ```
2. Install and start MySQL:
   ```
   yum install mysql mysql-server mysql-devel
   /etc/init.d/mysqld start
   ```
3. Grant all privileges on database `rt4` to user `` `rt_user` ``:
   ```
   mysql
   grant all privileges on rt4.* to 'rt_user'@'localhost' identified by
'rt_user';
   exit
   ```
4. Install Request Tracker 4.0.5 or later (follow the instructions from `README`):
   ```
   wget http://download.bestpractical.com/pub/rt/release/rt-4.0.5.tar.gz
   tar -xzvf rt.tar.gz -C /Jovana
   cd /Jovana/rt-4.0.5
   ./configure
   make testdeps
   make fixdeps
   yum install expat-devel
   make fixdeps
   ```
5. Create and initialize database:
   ```
   make initialize-database
   ```
6. Add necessary configuration changes to `/opt/rt4/etc/RT_SiteConfig.pm`:
   ```
   Set( $WebDomain, 'rt2test.qalab.geant.net');
   Set( $WebPort, '80' );
   Set( $rtname, 'rt2test.qalab.geant.net');
   Set( $Organization, 'GN3-TTS_integration' );

   # Database settings
   Set( $DatabaseType, 'mysql' );
   Set( $DatabaseHost, 'localhost' );
   Set( $DatabaseName, 'rt4' );
   Set( $DatabaseAdminPassword, '' );
   Set( $DatabaseUser, 'rt_user' );
   Set($DatabasePassword, 'rt_user' );

   #Ticket lifecycles
   set(%Lifecycles,
         'service_level_ticket' => {
               initial  => [ 'OPENACTIVE.PROBLEM_STATED' ],
               active   => [ 'DEFERRED.QUERING_SIM',
'DEFERRED.WAITING_FOR_APPROVALS', 'OPENACTIVE.VERIFY_SERVICE_STATUS' ],
               inactive => [ 'CLOSED.POSITIVELY_RESOLVED' ,
'CLOSED.NEGATIVELY_RESOLVED', 'DELETED' ],

               defaults => {  on_create => 'OPENACTIVE.PROBLEM_STATED' },

               transitions => {
                     '' => [ qw(OPENACTIVE.PROBLEM_STATED) ],
                     'OPENACTIVE.PROBLEM_STATED' =>
   ```

```
[ qw(DEFERRED.QUERING_SIM) ],
                        'DEFERRED.QUERING_SIM' =>
[ qw(DEFERRED.WAITING_FOR_APPROVALS) ],
                        'DEFERRED.WAITING_FOR_APPROVALS' =>
[ qw(OPENACTIVE.VERIFY_SERVICE_STATUS) ],
                        'OPENACTIVE.VERIFY_SERVICE_STATUS' =>
      [ qw(CLOSED.POSITIVELY_RESOLVED CLOSED.NEGATIVELY_RESOLVED) ],
                        'CLOSED.POSITIVELY_RESOLVED' => [ qw(DELETED) ],
                        'CLOSED.NEGATIVELY_RESOLVED' => [ qw(DELETED) ]
                  },

                  rights => {
                        '* -> DELETED'  => 'DeleteTicket',
                        '* -> *' => 'ModifyTicket'
                  },

                  actions => [
                        'OPENACTIVE.VERIFY_SERVICE_STATUS ->
CLOSED.POSITIVELY_RESOLVED' => { label => 'Fixed!', update => 'Comment' },
                        'OPENACTIVE.VERIFY_SERVICE_STATUS ->
CLOSED.NEGATIVELY_RESOLVED' => { label => 'Failed!', update => 'Comment' },
                        'CLOSED.POSITIVELY_RESOLVED -> DELETED' => { label =>
'Delete' },
                        'CLOSED.NEGATIVELY_RESOLVED -> DELETED' => { label =>
'Delete' }
                  ]
            },

            'received_service_ticket' => {
                        initial  => [ '' ],
                        active   => [ 'OPENACTIVE' ],
                        inactive => [ 'CLOSED.RESOLVED' ,
'CLOSED.NONEXISTENT_PROBLEM' ],

                        defaults => { on_create => 'OPENACTIVE' },

                        transitions => {
                              '' => [ qw(OPENACTIVE) ],
                              OPENACTIVE => [ qw(CLOSED.RESOLVED
CLOSED.NONEXISTENT_PROBLEM) ],
                              CLOSED.RESOLVED => [ qw(DELETED) ],
                              CLOSED.NONEXISTENT_PROBLEM => [ qw(DELETED) ]
                        },

                        rights => {
                              '* -> *' => 'ModifyTicket',
                        },

                        actions => [
                              'OPENACTIVE -> CLOSED.RESOLVED' => { label =>
'Fixed!', update => 'Respond' },
                              'OPENACTIVE -> CLOSED.NONEXISTENT_PROBLEM' => { label
=> 'Failed!', update => 'Respond' }
                        ],
            }
      );
```

7. Start RT and log in as a `root` (default password is `password`):
   `/opt/rt4/sbin/rt-server`

Figure 4: Request Tracker login page

8. Create queue `Detected_service_problems` (`Tools -> Configuration -> Queues -> Create`):

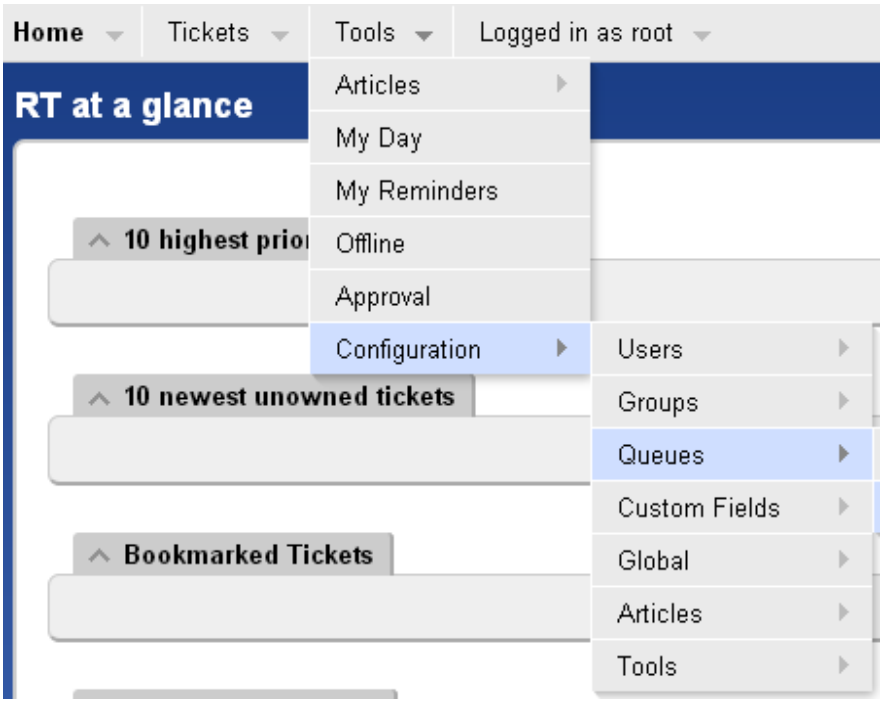


Figure 5: Queue creation

`Queue Name: Detected_service_problems`

```
Description: Queue for tickets concerning GN services
Lifecycle: service_level_ticket
Reply Address: reply_service_problems@rt2test.qalab.geant.net
Comment Address: comment_service_problems@rt2test.qalab.geant.net
Checkbox "Enabled" should be checked
```

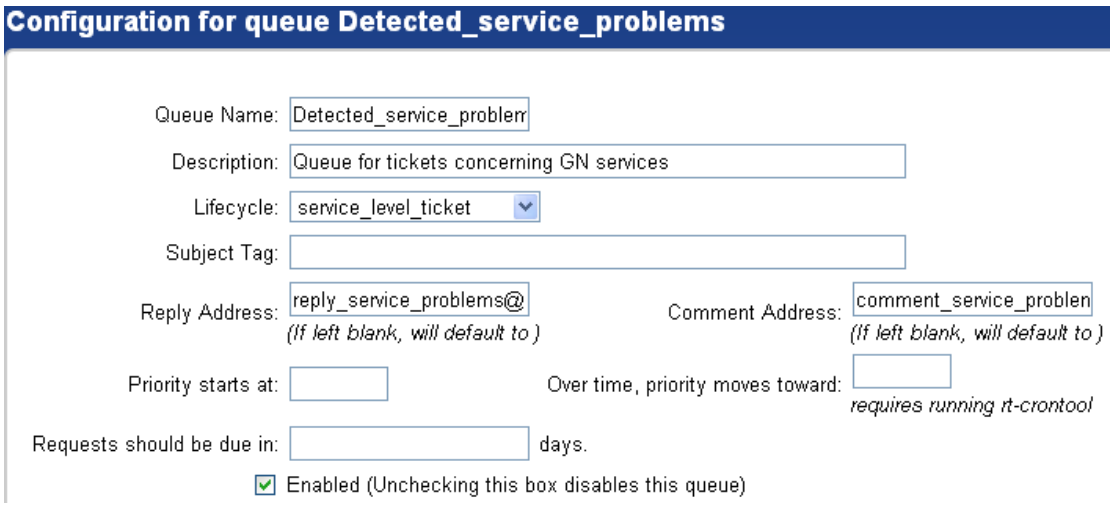


Figure 6: Queue Detected_service_problem

9. Add template `Query_SIM` for queue `Detected_service_problems (Tools -> Configuration -> Queues -> Select,` select queue `Detected_service_problems, Templates -> Create`)

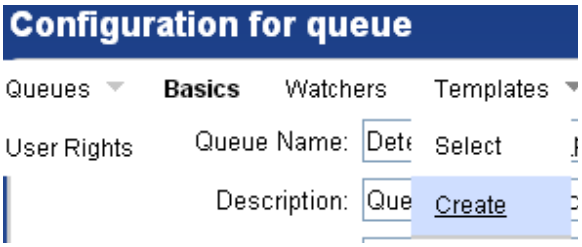


Figure 7: Creating a template for queue

```
Name: Query_SIM
Type: Perl
Content:
===Create-Ticket: Query SIM
Subject: {$Tickets{'TOP'}->Id}
Depended-On-By: {$Tickets{'TOP'}->Id}
Queue: Query_SIM
Content:
Unique service id: { $Tickets{'TOP'}->CustomFieldValues('Unique Service ID')->Next-
>Content }
{ $Tickets{'TOP'}->CustomFieldValues('Message for others')->Next->Content }
endofcontent
ENDOFCONTENT
```

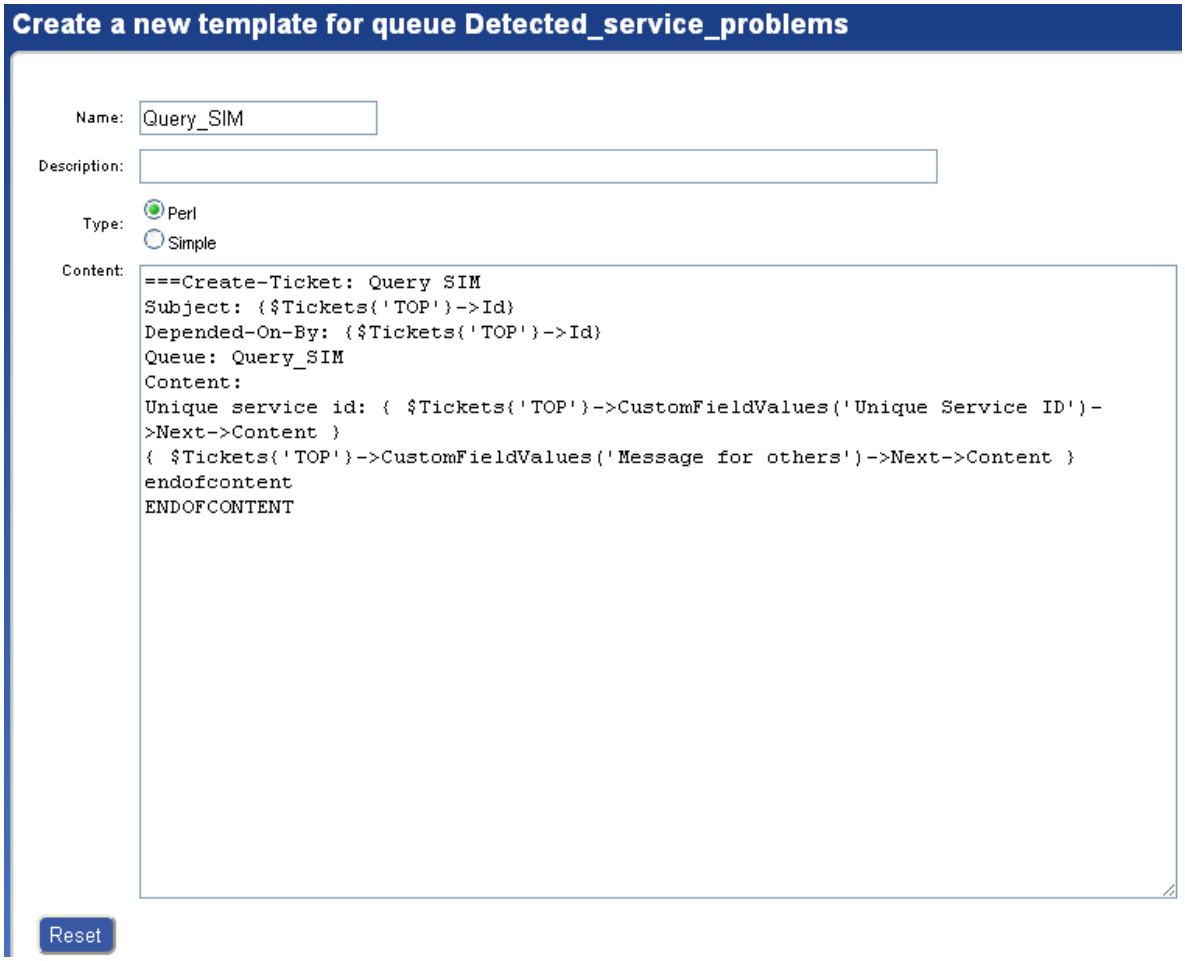


Figure 8: Template Query_SIM

10. Add scrip `Create_ticket_for_following_SIM_query` for queue `Detected_service_problems` (`Tools -> Configuration -> Queues -> Select`, select queue `Detected_service_problems`, `Scrips -> Create`):

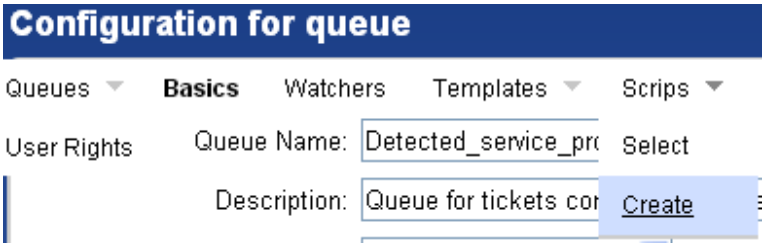


Figure 9: Creating a scrip for queue

```
Description: Create_ticket_for_following_SIM_query
Condition: On Create
Action: Create Tickets
Template: Query_SIM
Stage: TransactionCreate

Custom condition:
Custom action preparation code:
Custom action cleanup code:
```

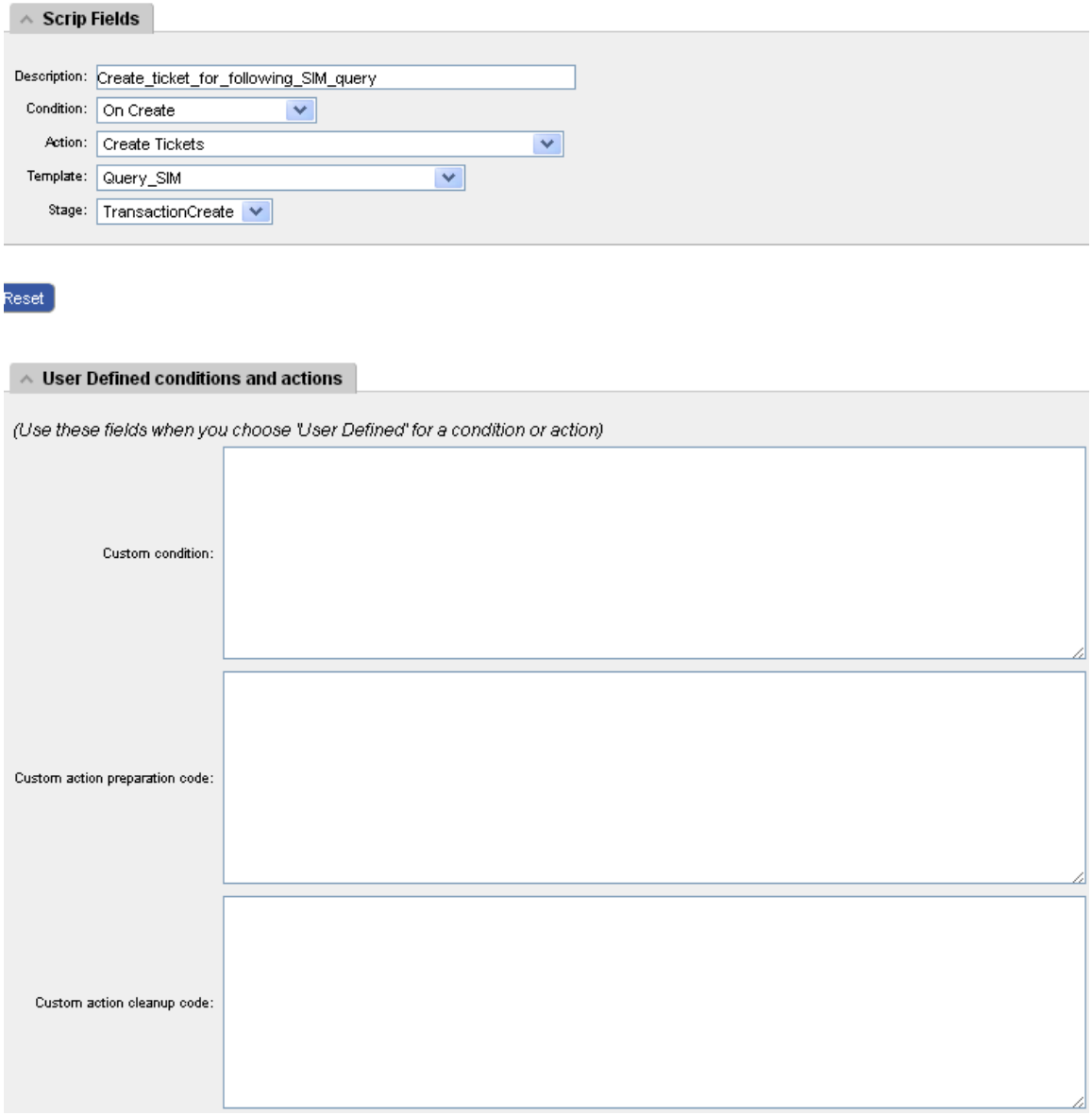


Figure 10: Scrip Create_ticket_for_following_SIM_query

11. Add scrip `OnCreateChangeMyState` for queue `Detected_service_problems` (`Tools -> Configuration -> Queues -> Select`, select queue `Detected_service_problems`, `Scrips -> Create`):

```
Description: OnCreateChangeMyState
Condition: On Create
Action: User Defined
Template: Global template: Blank
Stage: TransactionCreate

Custom condition:

Custom action preparation code:
return 1;

Custom action cleanup code:
$self->TicketObj->SetStatus('DEFERRED.QUERING_SIM');
return 1;
```

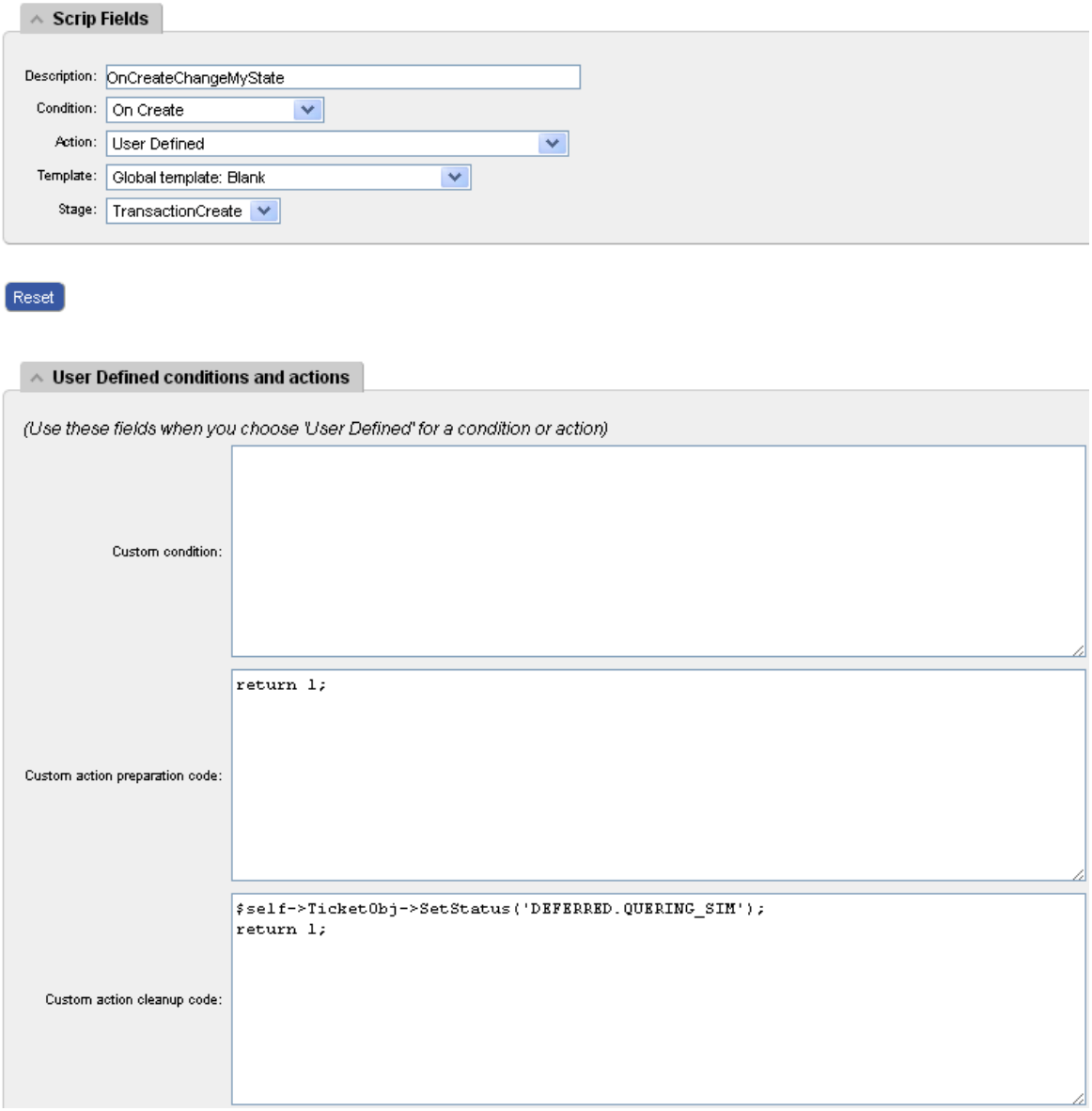


Figure 11: Scrip OnCreateChangeMyState

12. Create custom field `Unique Service ID` (`Tools -> Configuration -> Custom Fields -> Create`)

```
Name: Unique Service ID
Description: Unique Service ID in service inventory
Type: Enter one value
Applies to: Tickets
Validation: (?#Mandatory)
Link values to:
Include page:
Checkbox "Enabled" should be checked
```

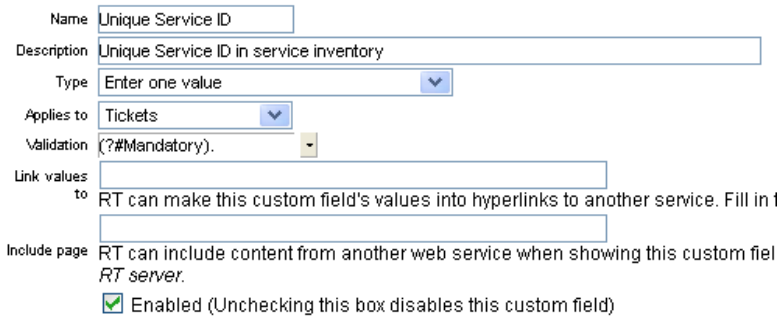


Figure 13: Custom field Unique sevice ID

13. Create custom field `Message for others (Tools -> Configuration -> Custom Fields -> Create)`

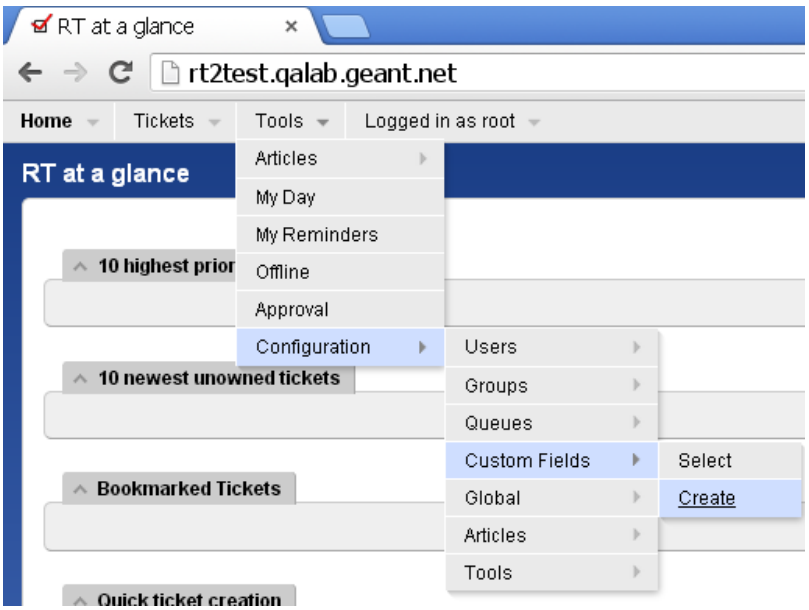


Figure 12: Creating custom field

```
Name: Message for others
Description: Ticket content in other TTSes
Type: Fill in one text area
Applies to: Tickets
Validation: (?#Mandatory)
Link values to:
Include page:
Checkbox "Enabled" should be checked
```

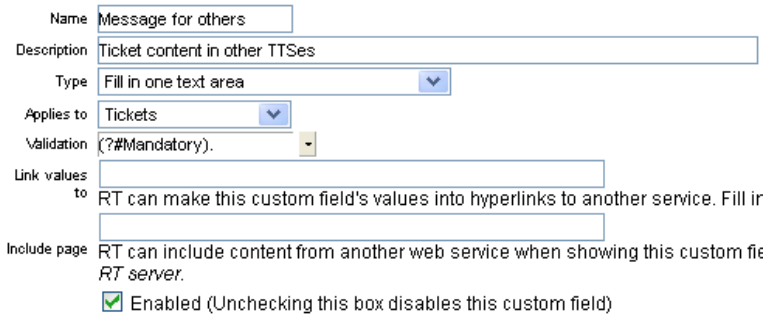


Figure 14: Custom field Message for others

14. Assign custom fields `"Unique Service ID"` and `"Message for others"` to queue `Detected_service_problems`:
    `Tools -> Configuration -> Queues -> Select`, select queue `Detected_service_problems, Ticket Custom Fields`,
    check custom fields "`Unique Service ID`" and `"Message for others`" and press `Submit`
15. Create queue `Query_SIM (Tools -> Configuration -> Queues -> Create)`:
    `Queue Name: Query_SIM`
    `Description: Queue for tickets concernig SIM queries`
    `Lifecycle: default`
    `Reply Address: reply_query_sim@rt2test.qalab.geant.net`
    `Comment Address: comment_query_sim@rt2test.qalab.geant.net`

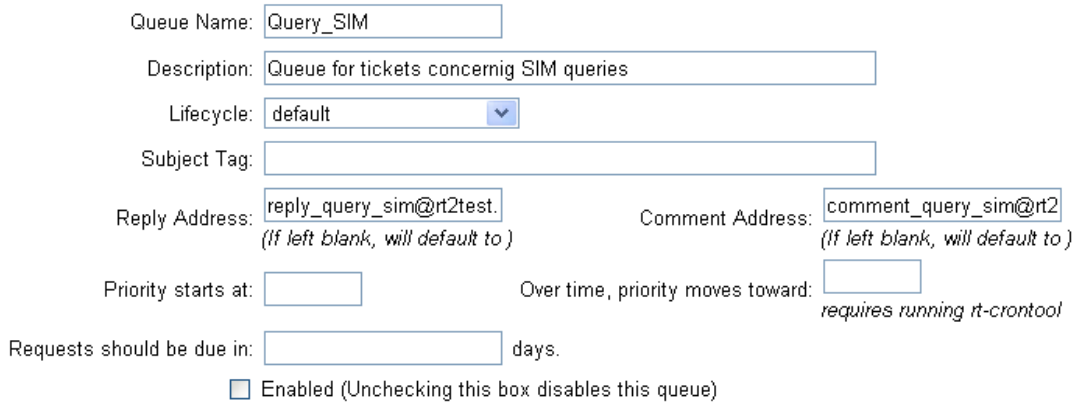


Figure 15: Queue Query_SIM

16. Create a privileged user `SIM_client (Tools -> Configuration -> Users -> Create)`:

`Username: SIM_client`
`Email: sim_client@localhost.localdomain`
`Checkboxes "Let this user access RT" and "Let this user be granted rights (Privileged)" should be checked`

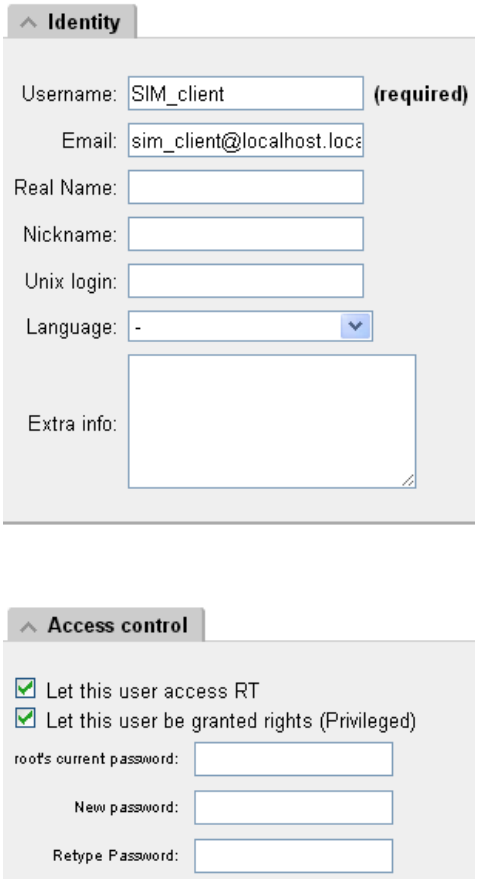


Figure 16: User SIM_client

17. Add `SIM_client` as `AdminCC` for queue `Query_SIM`: `Tools -> Configuration -> Queues -> Select`, select queue `Query_SIM, Watchers`, find `SIM_client` using form `"Find people whose"`, select `AdminCC` from select list which is in front of `SIM_client` in search results and press "`Save changes`"

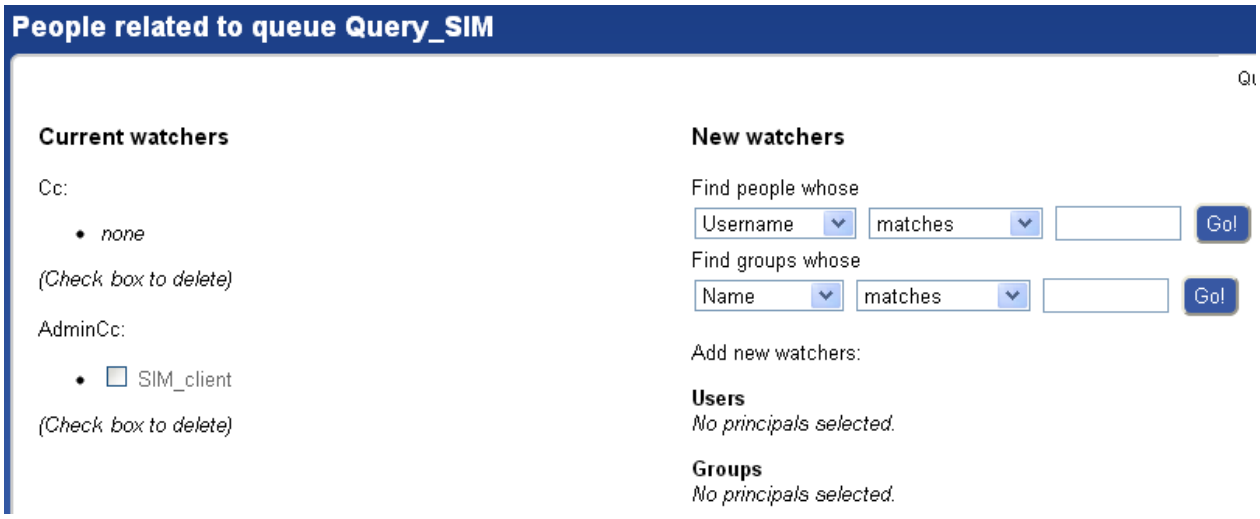
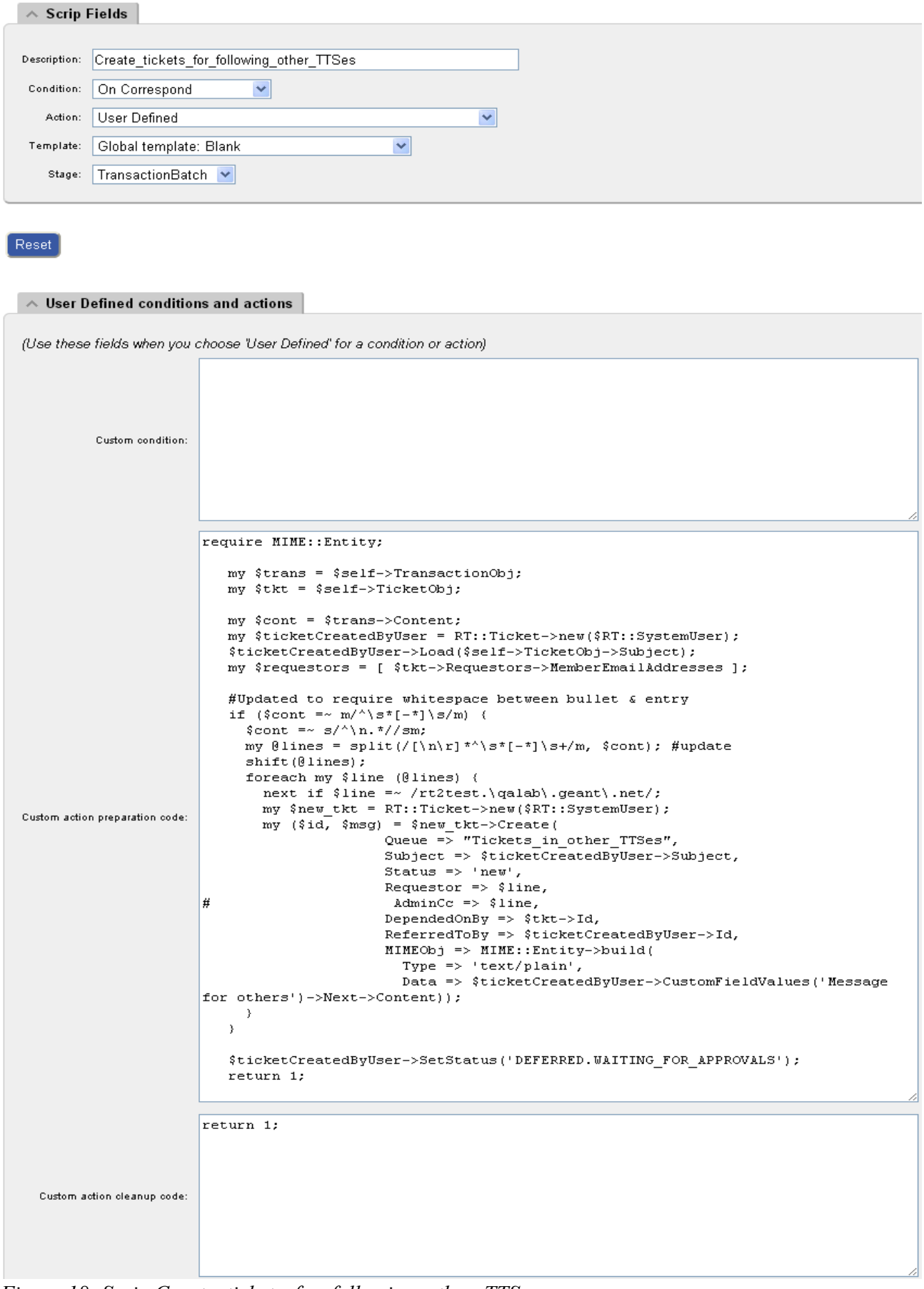


Figure 17: Query_SIM watchers

18. Add scrip `Create_tickets_for_following_other_TTSes` for queue `Query_SIM` (`Tools ->`

Configuration -> Queues -> Select, select queue Query_SIM, Scrips -> Create):

```
Description: Create_tickets_for_following_other_TTSes
Condition: On Correspond
Action: User Defined
Template: Global template: Blank
Stage: TransactionBatch

Custom condition:

Custom action preparation code:
require MIME::Entity;

   my $trans = $self->TransactionObj;
   my $tkt = $self->TicketObj;

   my $cont = $trans->Content;
   my $ticketCreatedByUser = RT::Ticket->new($RT::SystemUser);
   $ticketCreatedByUser->Load($self->TicketObj->Subject);
   my $requestors = [ $tkt->Requestors->MemberEmailAddresses ];

   #Updated to require whitespace between bullet & entry
   if ($cont =~ m/^\s*[-*]\s/m) {
     $cont =~ s/^\n.*//sm;
     my @lines = split(/[\n\r]*^\s*[-*]\s+/m, $cont); #update
     shift(@lines);
     foreach my $line (@lines) {
       next if $line =~ /rt2test.\qalab\.geant\.net/;
       my $new_tkt = RT::Ticket->new($RT::SystemUser);
       my ($id, $msg) = $new_tkt->Create(
                     Queue => "Tickets_in_other_TTSes",
                     Subject => $ticketCreatedByUser->Subject,
                     Status => 'new',
                     Requestor => $line,
                     DependedOnBy => $tkt->Id,
                     ReferredToBy => $ticketCreatedByUser->Id,
                     MIMEObj => MIME::Entity->build(
                       Type => 'text/plain',
                       Data => $ticketCreatedByUser->CustomFieldValues('Message for
others')->Next->Content));
     }
   }

   $ticketCreatedByUser->SetStatus('DEFERRED.WAITING_FOR_APPROVALS');
   return 1;

Custom action cleanup code:
return 1;
```



Scrip Fields

Description: Create_tickets_for_following_other_TTSes

Condition: On Correspond

Action: User Defined

Template: Global template: Blank

Stage: TransactionBatch

Reset

User Defined conditions and actions

(Use these fields when you choose 'User Defined' for a condition or action)

Custom condition:

Custom action preparation code:

```
require MIME::Entity;

   my $trans = $self->TransactionObj;
   my $tkt = $self->TicketObj;

   my $cont = $trans->Content;
   my $ticketCreatedByUser = RT::Ticket->new($RT::SystemUser);
   $ticketCreatedByUser->Load($self->TicketObj->Subject);
   my $requestors = [ $tkt->Requestors->MemberEmailAddresses ];

   #Updated to require whitespace between bullet & entry
   if ($cont =~ m/^\s*[-*]\s/m) {
     $cont =~ s/^\n.*//sm;
     my @lines = split(/[\n\r]*^\s*[-*]\s+/m, $cont); #update
     shift(@lines);
     foreach my $line (@lines) {
       next if $line =~ /rt2test.\qalab\.geant\.net/;
       my $new_tkt = RT::Ticket->new($RT::SystemUser);
       my ($id, $msg) = $new_tkt->Create(
                        Queue => "Tickets_in_other_TTSes",
                        Subject => $ticketCreatedByUser->Subject,
                        Status => 'new',
                        Requestor => $line,
#                        AdminCc => $line,
                        DependedOnBy => $tkt->Id,
                        ReferredToBy => $ticketCreatedByUser->Id,
                        MIMEObj => MIME::Entity->build(
                          Type => 'text/plain',
                          Data => $ticketCreatedByUser->CustomFieldValues('Message
for others')->Next->Content));
     }
   }

   $ticketCreatedByUser->SetStatus('DEFERRED.WAITING_FOR_APPROVALS');
   return 1;
```

Custom action cleanup code:

```
return 1;
```

Figure 18: Scrip Create_tickets_for_following_other_TTSes

19. Modify global template "Transaction" (Tools -> Configuration -> Global -> Templates -> Select, Transaction)

```
Name: Transaction
Description: Default transaction template
Type: Perl
Content:
RT-Attach-Message: yes

{$Transaction->CreatedAsString}: Request {$Ticket->id} was acted upon.
 Transaction: {$Transaction->Description}
       Queue: {$Ticket->QueueObj->Name}
     Subject: {$Transaction->Subject || $Ticket->Subject || "(No subject given)"}
       Owner: {$Ticket->OwnerObj->Name}
  Requestors: {$Ticket->RequestorAddresses}
      Status: {$Ticket->Status}
 Ticket <URL: {RT->Config->Get('WebURL')}Ticket/Display.html?id={$Ticket->id} >
 Subject tag: { $Ticket->SubjectTag }

{$Transaction->Content()}
```

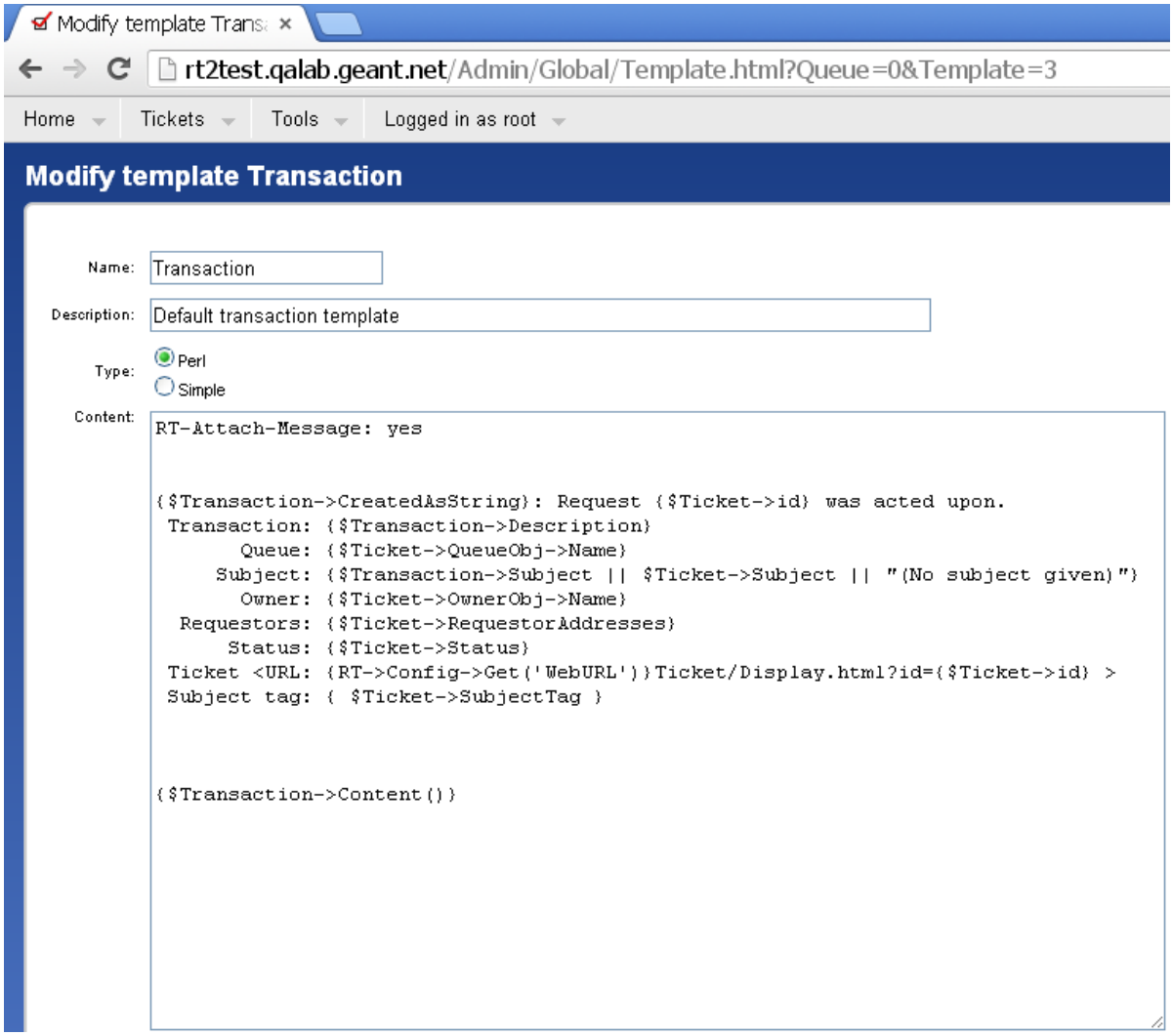


Figure 19: Changing global template "Transaction"

20. Add scrip `OnCreateNotifyAdminCcs` for queue `Query_SIM` (Tools -> Configuration -> Queues -> Select, select queue `Query_SIM`, Scrips -> Create):

```
Description: OnCreateNotifyAdminCcs
Condition: On Create
Action: Notify AdminCCs
Template: Global template: Transaction
Stage: TransactionCreate

Custom condition:
Custom action preparation code:
Custom action cleanup code:
```

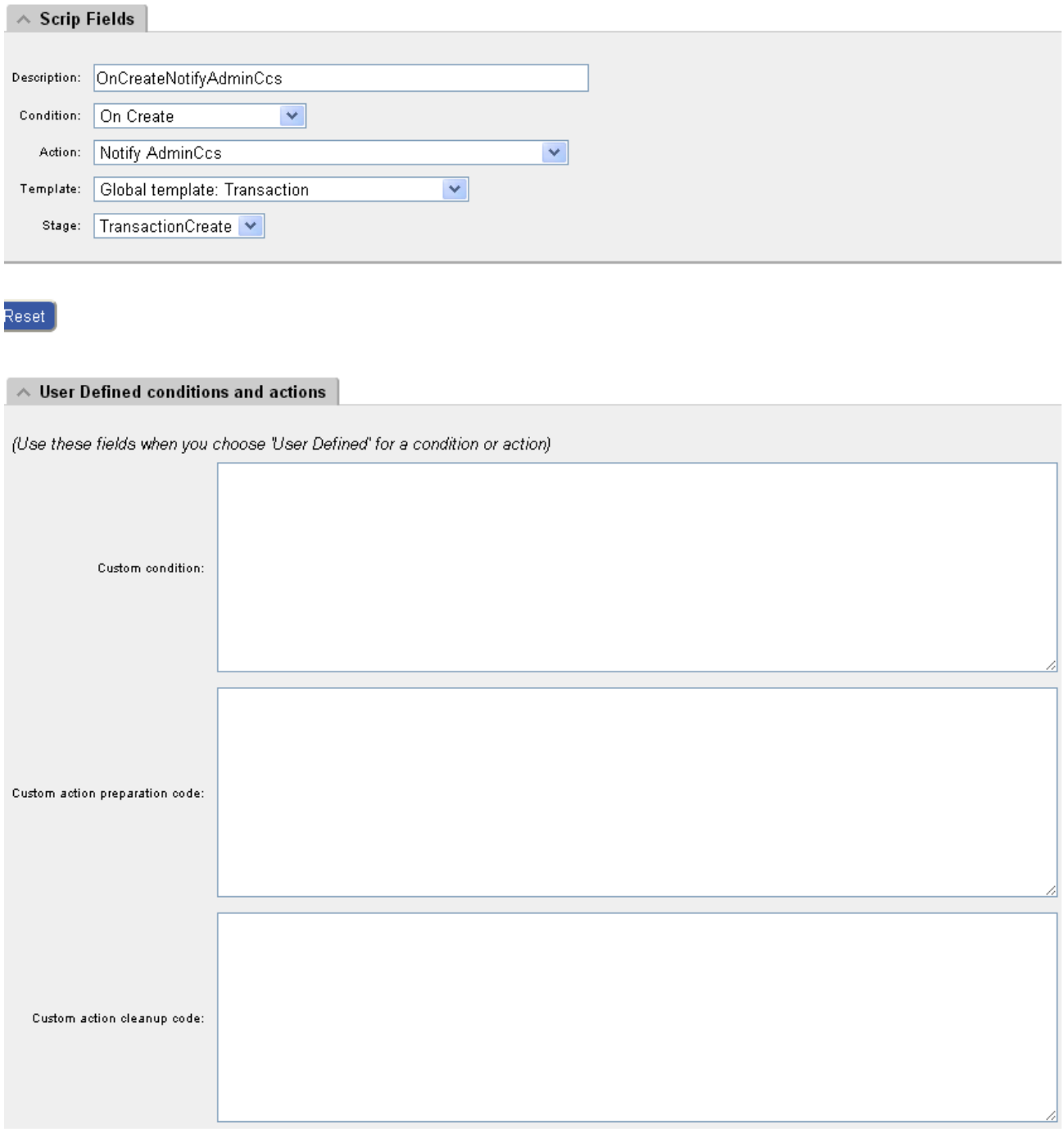


Figure 20: Scrip OnCreateNotifyAdminCcs

21. Add scrip `OnResolveChangeStatesOfDependentOnByTickets` for queue `Query_SIM` (`Tools -> Configuration -> Queues -> Select`, select queue `Query_SIM`, `Scrips -> Create`):

Description: OnResolveChangeStatesOfDependentOnByTickets
Condition: On Resolve
Action: User Defined
Template: Global template: Blank
Stage: TransactionCreate

Custom condition:

Custom action preparation code:

```
return 1;
```

Custom action cleanup code:

```
my $DepOnBy = $self->TicketObj->DependedOnBy;
while( my $l = $DepOnBy->Next ) {
  next unless( $l->BaseURI->IsLocal );

  $l->BaseObj->SetStatus('OPENACTIVE.VERIFY_SERVICE_STATUS');
}
$DepOnBy = undef;
return 1;
```

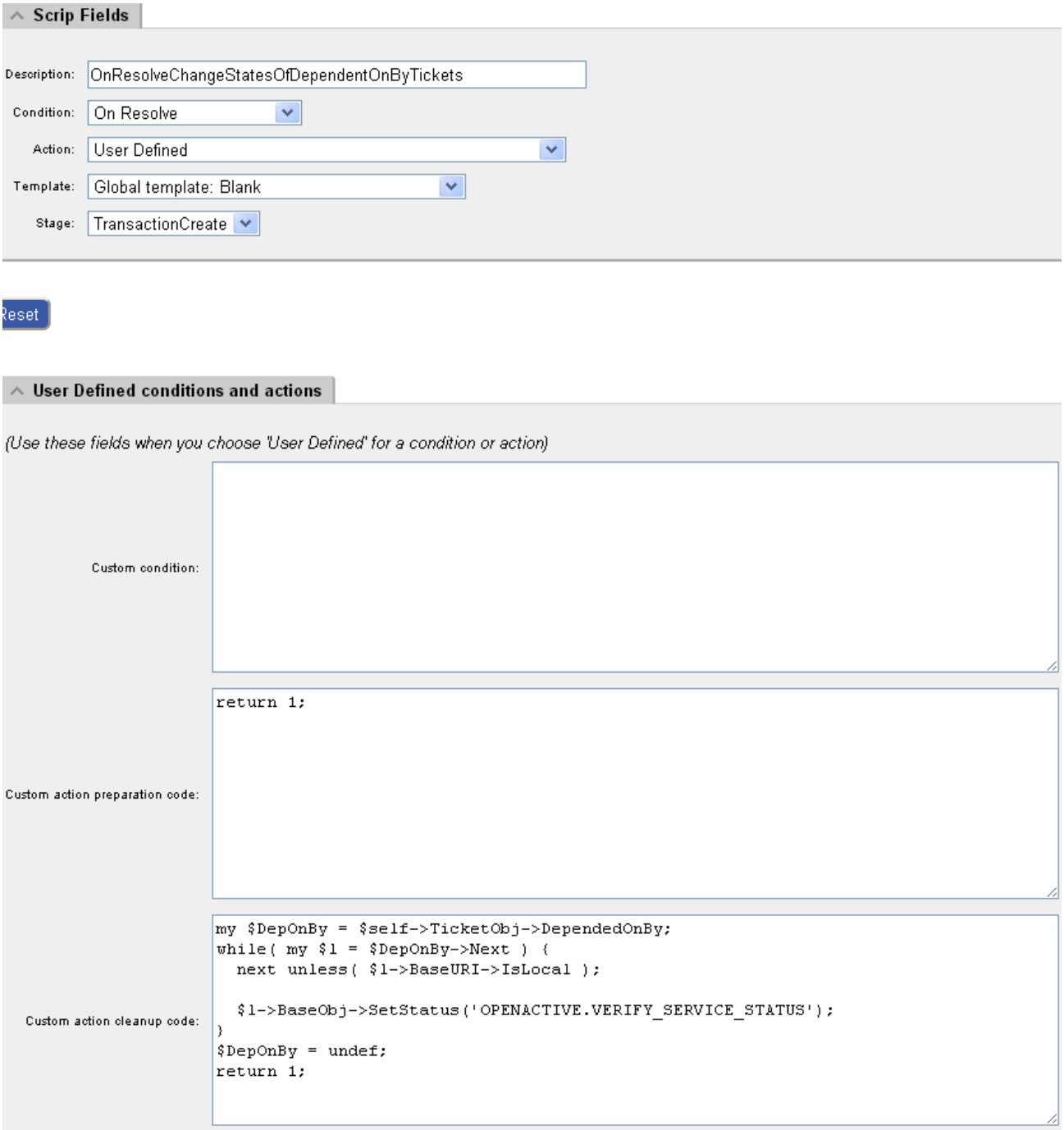


Figure 21: Scrip OnResolveChangeStatesOfDependentOnByTickets

22. Create queue `Tickets_in_other_TTSes` (`Tools -> Configuration -> Queues -> Create`):
    `Queue Name: Tickets_in_other_TTSes`
    `Description: Tickets that initiated creation of tickets in other TTSes`
    `Lifecycle: default`
    `Reply Address: interconnection_prototype_reply@rt2test.qalab.geant.net`
    `Comment Address: interconnection_prototype_comment@rt2test.qalab.geant.net`

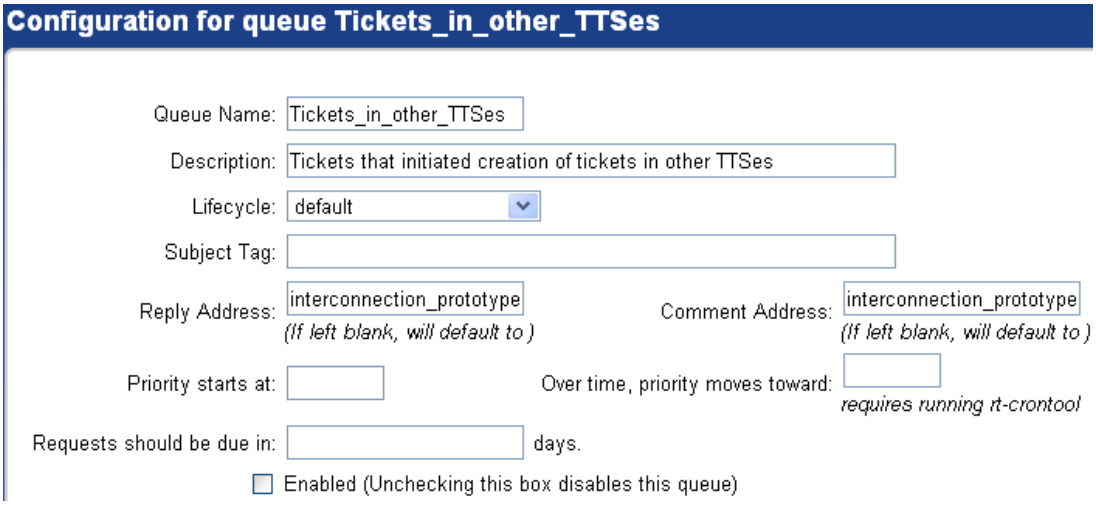


Figure 22: Queue Tickets_in_other_TTSes

23. Add scrip `OnCreateNotifyRequestor` for queue `Tickets_in_other_TTSes` (`Tools -> Configuration -> Queues -> Select, select queue Tickets_in_other_TTSes, Scrips -> Create`):

```
Description: OnCreateNotifyRequestor
Condition: On Create
Action: Autoreply To Requestors
Template: Global template: Autoreply
Stage: TransactionCreate

Custom condition:
Custom action preparation code:
Custom action cleanup code:
```

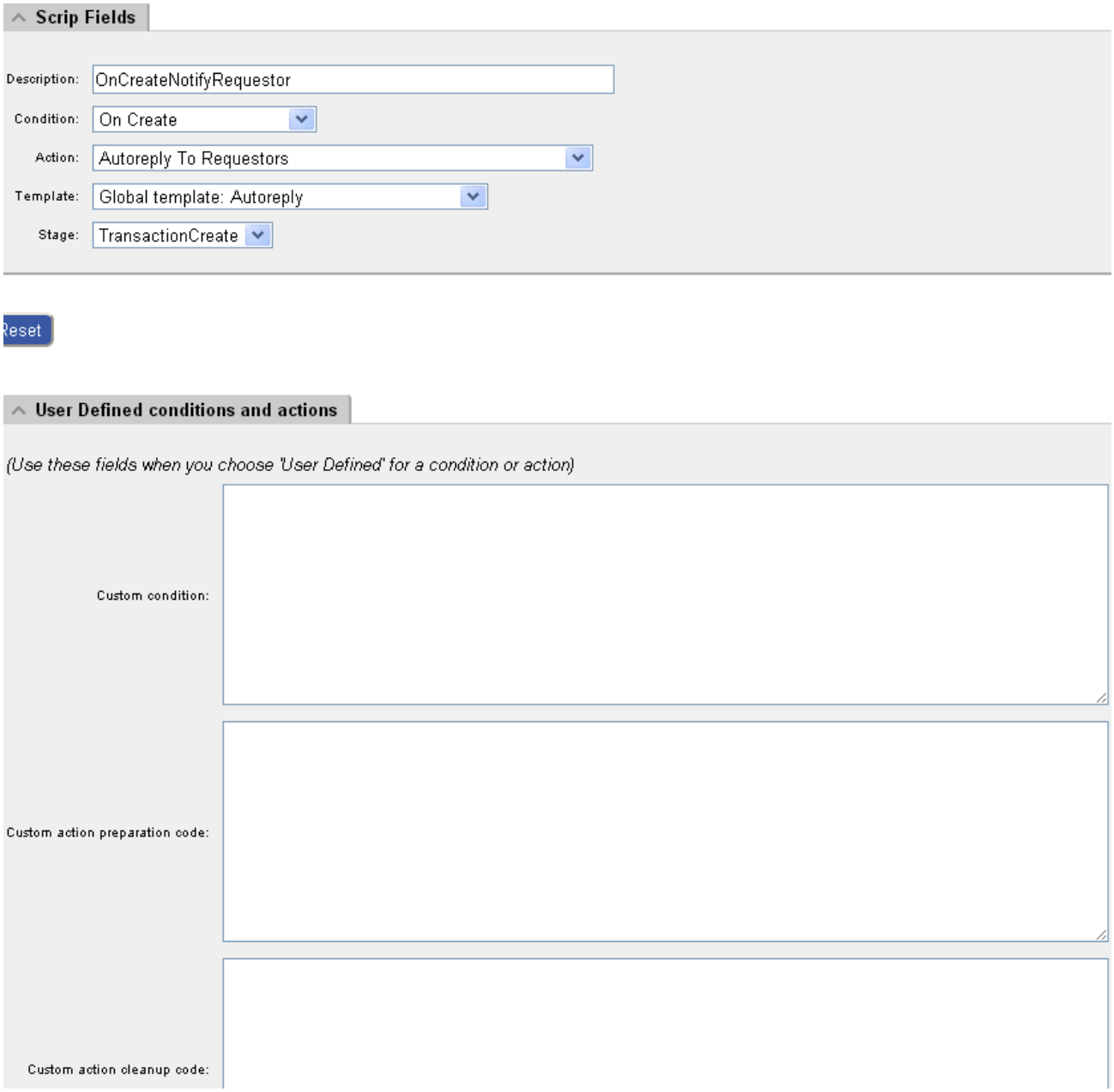


Figure 23: Scrip OnCreateNotifyRequestor

24. Add scrip `OnCorrespondCloseMyself` for queue `Tickets_in_other_TTSes` (`Tools -> Configuration -> Queues -> Select`, select queue `Tickets_in_other_TTSes`, `Scrips -> Create`):

```
Description: OnCorrespondCloseMyself
Condition: On Correspond
Action: User Defined
Template: Global template: Blank
Stage: TransactionCreate

Custom condition:

Custom action preparation code:
$self->TicketObj->SetStatus('resolved');
```

```
return 1;

Custom action cleanup code:
return 1;
```

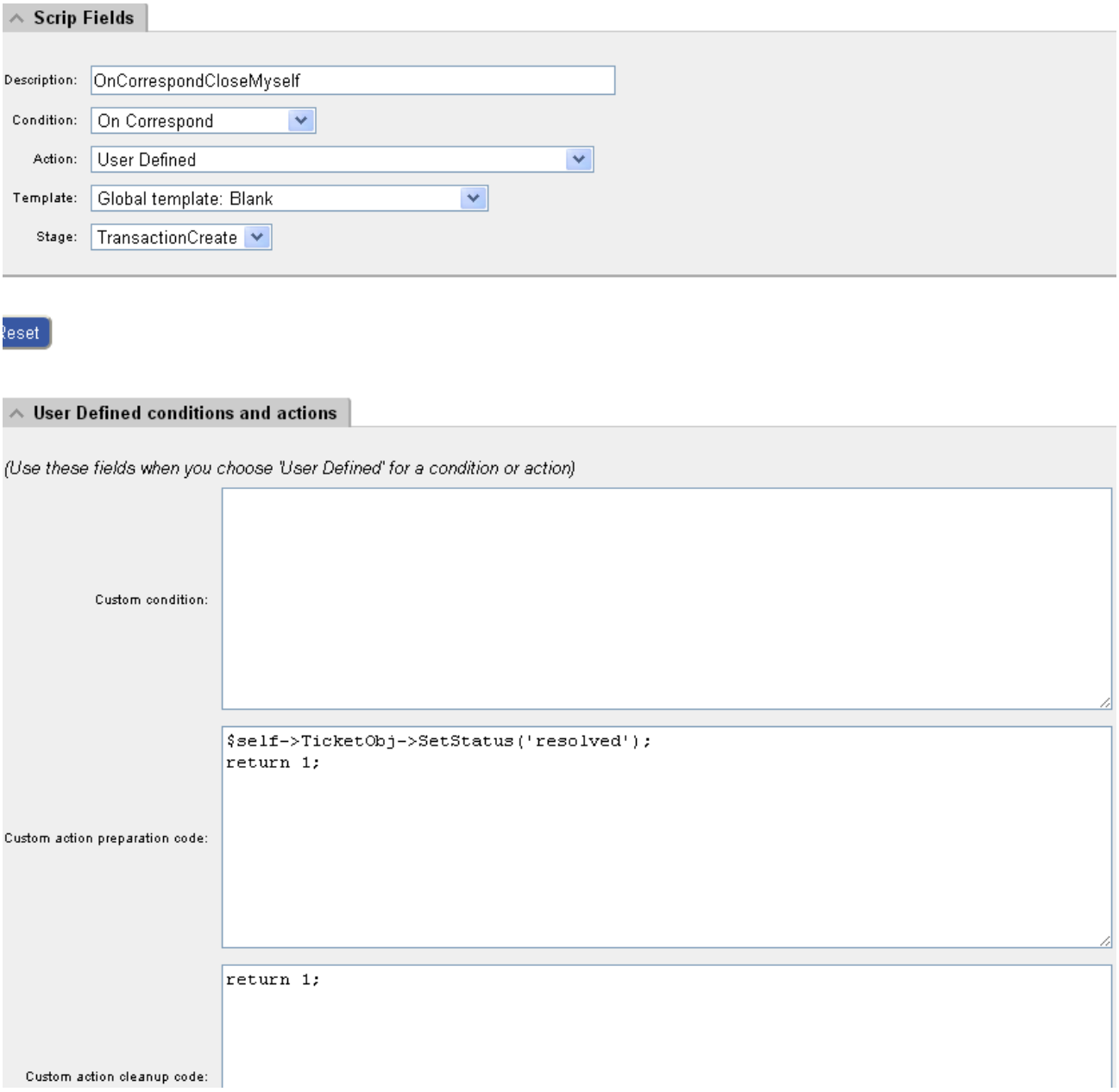


Figure 24: Scrip OnCorrespondCloseMyself

25. Add scrip `OnResolveResolveDependentOnByTickets` for queue `Tickets_in_other_TTSes` (`Tools -> Configuration -> Queues -> Select`, select queue `Tickets_in_other_TTSes`, `Scrips -> Create`):

```
Description: OnResolveResolveDependentOnByTickets
Condition: On Resolve
Action: User Defined
Template: Global template: Blank
Stage: TransactionCreate
```

Custom condition:

Custom action preparation code:

```
return 1;
```

Custom action cleanup code:

```
return 1 if ($self->TransactionObj->NewValue !~ /^(?:resolved|deleted|rejected)$/);

$self->TicketObj->SetStatus('resoved');
my $DepOnBy = $self->TicketObj->DependedOnBy;
while( my $l = $DepOnBy->Next ) {
  next unless( $l->BaseURI->IsLocal );
  next unless( $l->BaseObj->Status =~ /^(?:new|open|stalled)$/ );

    # here you can add any action
    # see also example below
  $l->BaseObj->SetStatus('resolved');
}
$DepOnBy = undef;
return 1;
```

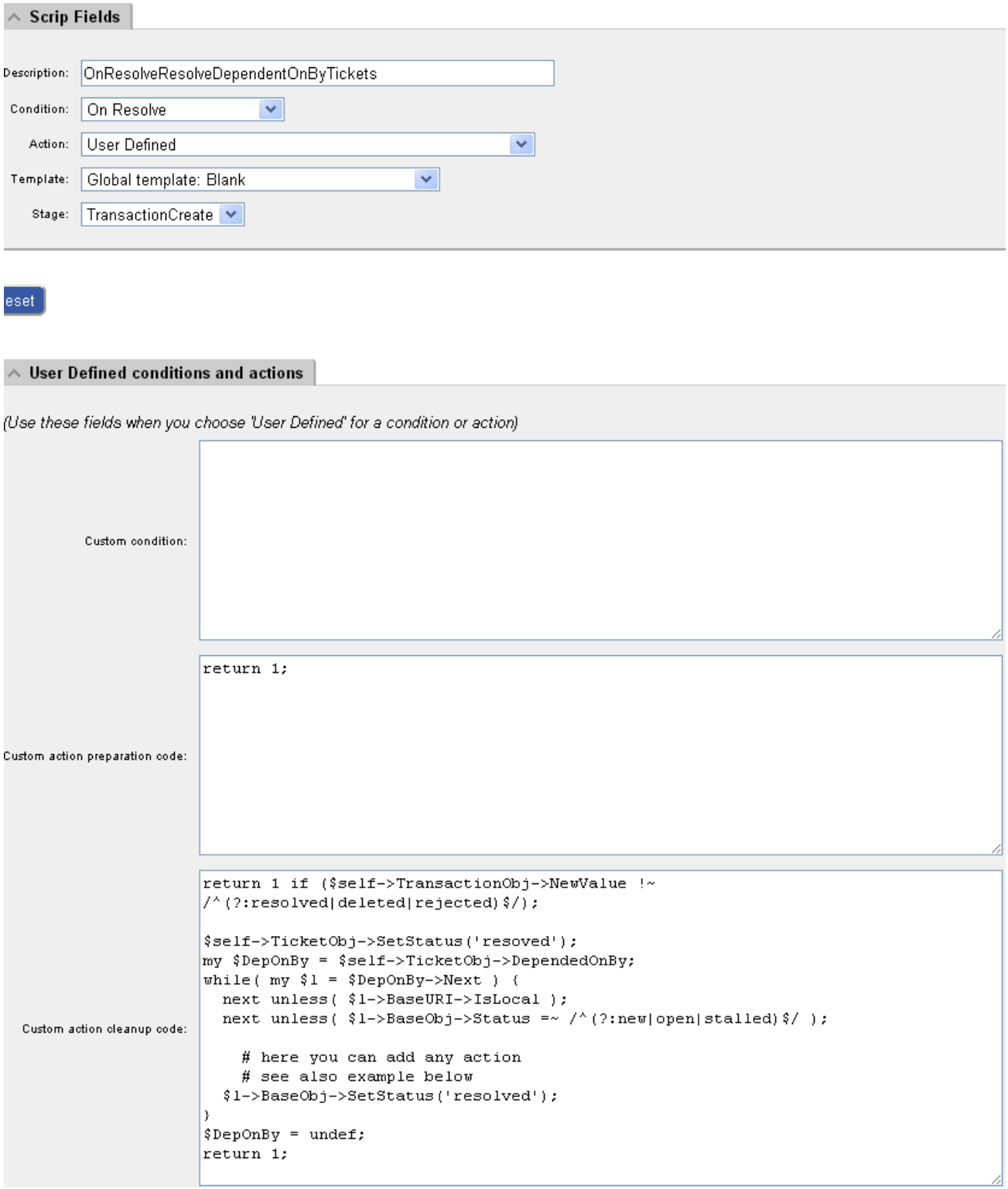


Figure 25: Scrip OnResolveResolveDependentOnByTickets

26. Create queue `Service_problems_reported_by_others` (`Tools -> Configuration -> Queues -> Create`):
    `Queue Name: Service_problems_reported_by_others`
    `Description: Queue for tickets automatically created by other TTSes`
    `Lifecycle: received_service_ticket`

```
Reply Address: reply_others@rt2test.qalab.geant.net
Comment Address: comment_others@rt2test.qalab.geant.net
```

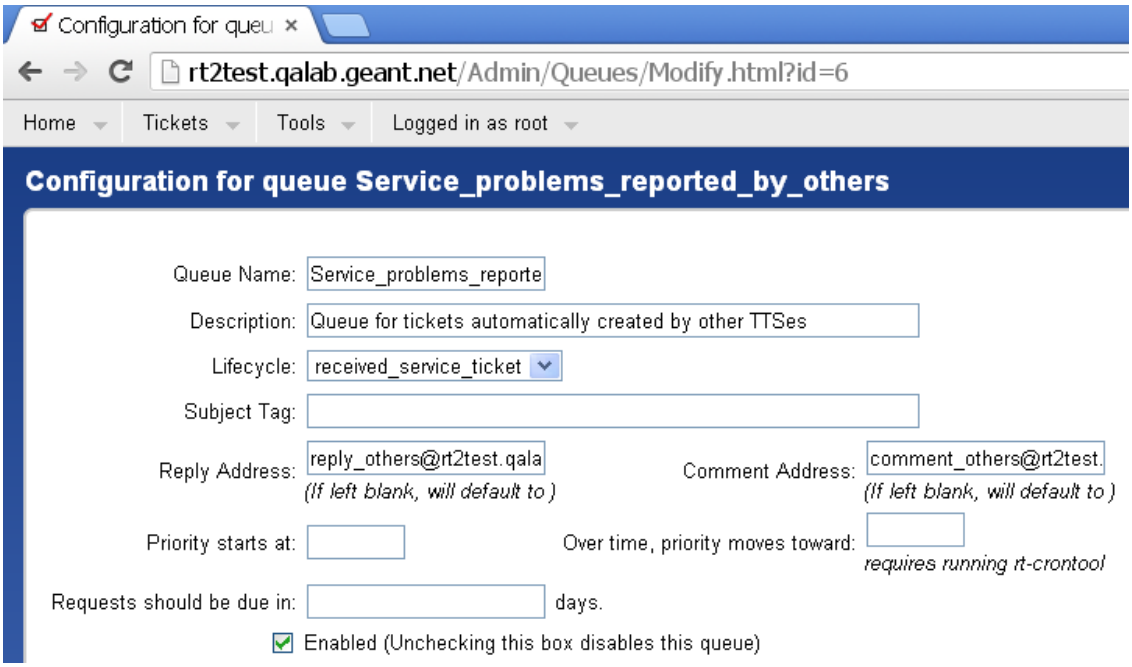


Figure 26: Queue Service_problems_reported_by_others

27. Add scrip `OnResolveNotifyRequestor` for queue `Service_problems_reported_by_others` (`Tools -> Configuration -> Queues -> Select`, select queue `Service_problems_reported_by_others`, `Scrips -> Create`):

```
Description: OnResolveNotifyRequestor
Condition: On Close
Action: Notify Requestors
Template: Global template: Correspondence
Stage: TransactionBatch

Custom condition:
Custom action preparation code:
Custom action cleanup code:
```

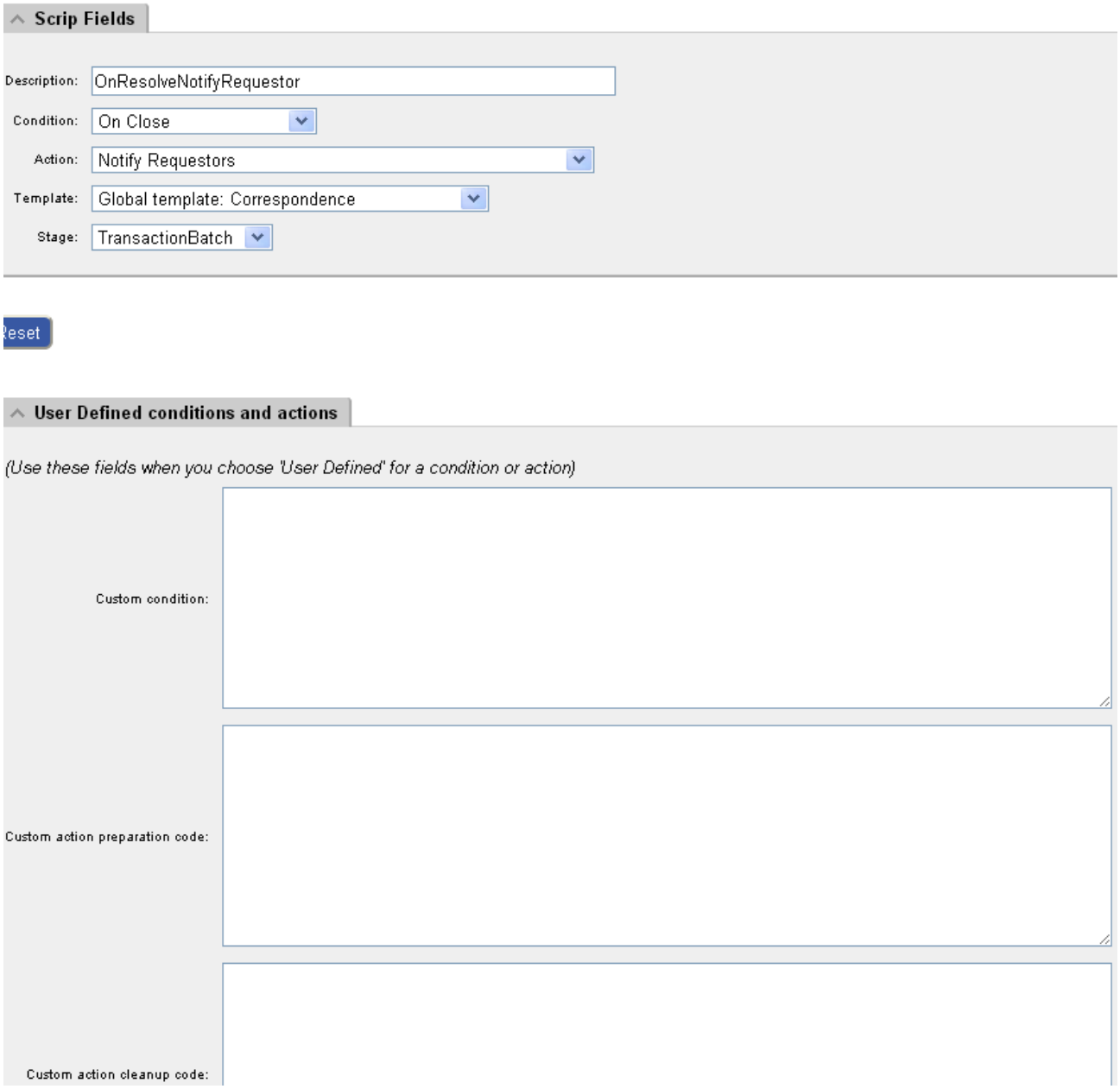


Figure 27: Scrip OnResolveNotifyRequestor

28. Create privileged user `User1`
    `Tools -> Configuration -> Users -> Create,`
    enter `User1` as both `username` and `password`
    check checkboxes `Let this user access RT` and `Let this user be granted rights (Privileged)`
    enter root's pasword
    press button "Save changes"

Figure 28: Creating user User1

29. Define appropriate permissions for queue `Detected_service_problems` `Tools -> Configuration -> Queues -> Select`, select the queue `Detected_service_problems`, click on "`User Rights`", enter username `User1` in the field `Add user`, check "`Comment on tickets`", "`Create tickets`", "`Reply to tickets`", "`Sign up as a ticket Requestor or ticket or queue Cc`", "`View custom field values`", "`View queue`" and "`View ticket summaries`" on tab "`General rights`", check "`Delete tickets`", "`Modify custom field values`", "`Modify tickets`" and "`View exact outgoing email messages and their recipients`" on tab "`Rights for Staff`" and press button "`Save changes`"

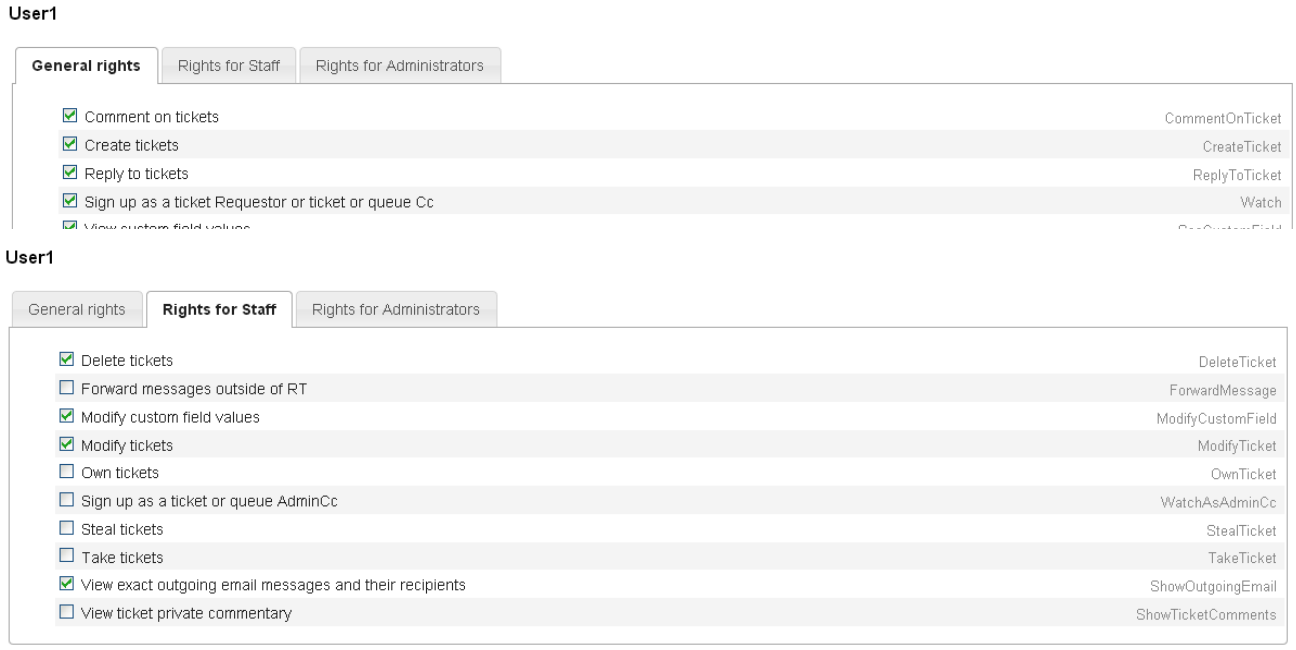


Figure 30: User rights for queue Detected_service_problems - tab Rights for Staff

30. Define appropriate permissions for queue `Query_SIM`

`Tools -> Configuration -> Queues -> Select`, select the queue `Query_SIM`, click on `Group Rights` and click on role `AdminCC`, check "`Create tickets`" and "`Reply to tickets`" on tab "`General rights`" and press button "`Save changes`"

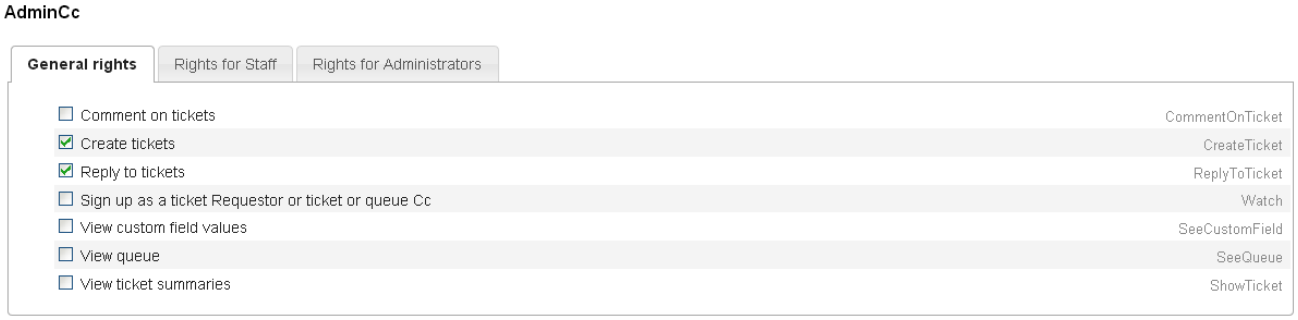


Figure 31: Group rights for queue QuerySIM

31. Define appropriate permissions for queue `Tickets_in_other_TTSes`

`Tools -> Configuration -> Queues -> Select`, select the queue `Tickets_in_other_TTSes`, click on `Group Rights` and click on `Everyone`, check "`Comment on tickets`", "`Reply to tickets`" and "`View ticket summaries`" on tab "`General rights`" and press button "`Save changes`"

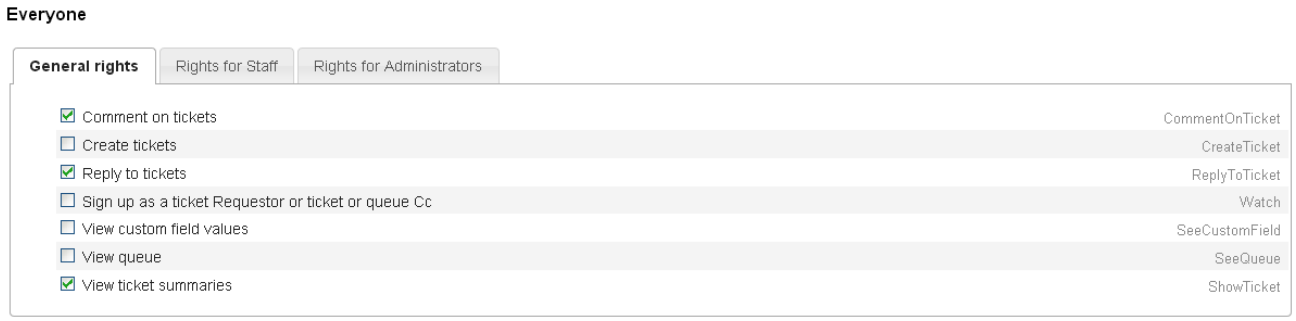


Figure 32: Group rights for queue Tickets_in_other_TTSes

32. Define appropriate permissions for queue `Service_problems_reported_by_others`

`Tools -> Configuration -> Queues -> Select`, select the queue `Service_problems_reported_by_others`, click on `Group Rights` and click on `Unprivileged`, check "`Create tickets`" on tab "`General rights`" and press button "`Save changes`"

`Tools -> Configuration -> Queues -> Select`, select the queue `Service_problems_reported_by_others`, click on `User Rights`, enter username `User1` in the field `Add user`, check "`Comment on tickets`", "`Reply to tickets`", "`View queue`" and "`View ticket summaries`" on tab "`General rights`", check "`Modify tickets`" on tab "`Rights for Staff`" and press button "`Save changes`"

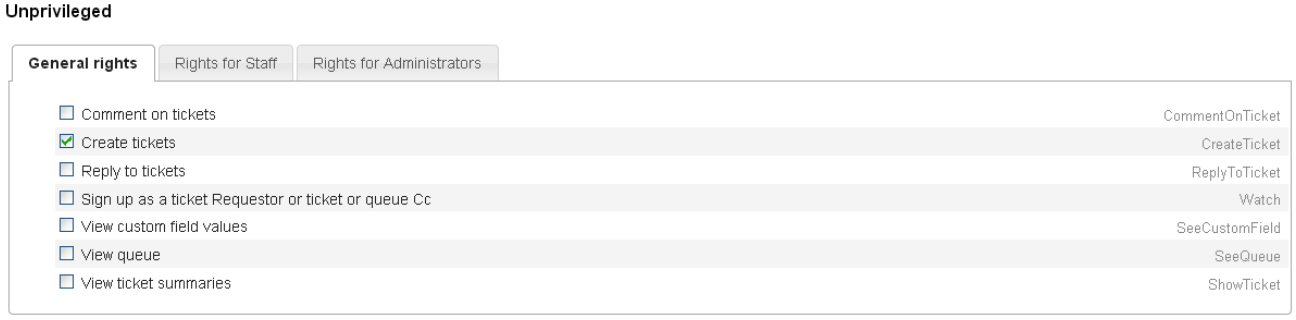


Figure 33: Group rights for queue Service_problems_reported_by_others

JIRA – installation and configuration – version 1

This version requires development of a custom mail handler and JIRA users for each TTS which is part of a integrated trouble ticket system.

Assumptions:

- Virtual host name is jira2test.qalab.geant.net
- Operating system is Linux
- Add Linux users: jira2test and sim_client and reply_others

Postfix configuration:

1. Disable selinux: in file `/etc/sysconfig/selinux` change line
   `SELINUX=enforcing`
   to
   `SELINUX=disabled`
2. Reboot virtual machine
   `reboot now`
3. Copy service inventory management web service client implementation (`wsKlijent.jar`) to
   `/Jovana/wsKlijentJIRA.jar` with appropriate permissions:
   `chmod 777 /Jovana/wsKlijentJIRA.jar`
4. Change postfix configuration:
   Change file `/etc/postfix/main.cf`
   - insert line:
     `myhostname = jira2test.qalab.geant.net`
   - comment line:
     `inet_interfaces = localhost`
   - uncomment line:
     `inet_interfaces = all`
   - comment line:
     `mydestination = $myhostname, localhost.$mydomain, localhost`
   - insert line:
     `mydestination = $myhostname, localhost.localdomain, localhost`

   Modify `/etc/aliases`
   - insert lines:
     `sim_client: "|/usr/lib/jvm/jre-1.6.0-openjdk.x86_64/bin/java -jar -Djava.endorsed.dirs=/Jovana/lib/endorsed /Jovana/wsKlijentJIRA.jar > /test1.txt"`
   - create file `/test1.txt` with appropriate permissions:
     `touch /test1.txt`
     `chmod 777 /test1.txt`

Dovecot installation and configuration

1. Install dovecot:

```
yum install dovecot
```

2. Configure dovecot - in `/etc/dovecot/dovecot.conf` insert line:

```
mail_location = mbox:~/mail:INBOX=/var/mail/%u
```

3. Change permissios for /var/spool/mail/* or dovecot won't work properly(http://dovecot.markmail.org/message/ovjoror2aveauzqy?q=chown+Operation+not+permitted+order:date-backward&page=1)

```
chmod 0600 /var/spool/mail/*
```

JIRA insallation and configuration:

1. Install JIRA – installation process depends on license type. File Readme.txt contains a brief install guide and there is plenty online documentation.
2. Create outgoing SMTP mail server `localhost_jira1test`:

Name: localhost_jira1test
From address: jira@jira1test.qalab.geant.net
Email prefix: JIRA1TEST
Protocol: SMTP
Hostname: jira1test.qalab.geant.net
Username: jira1test

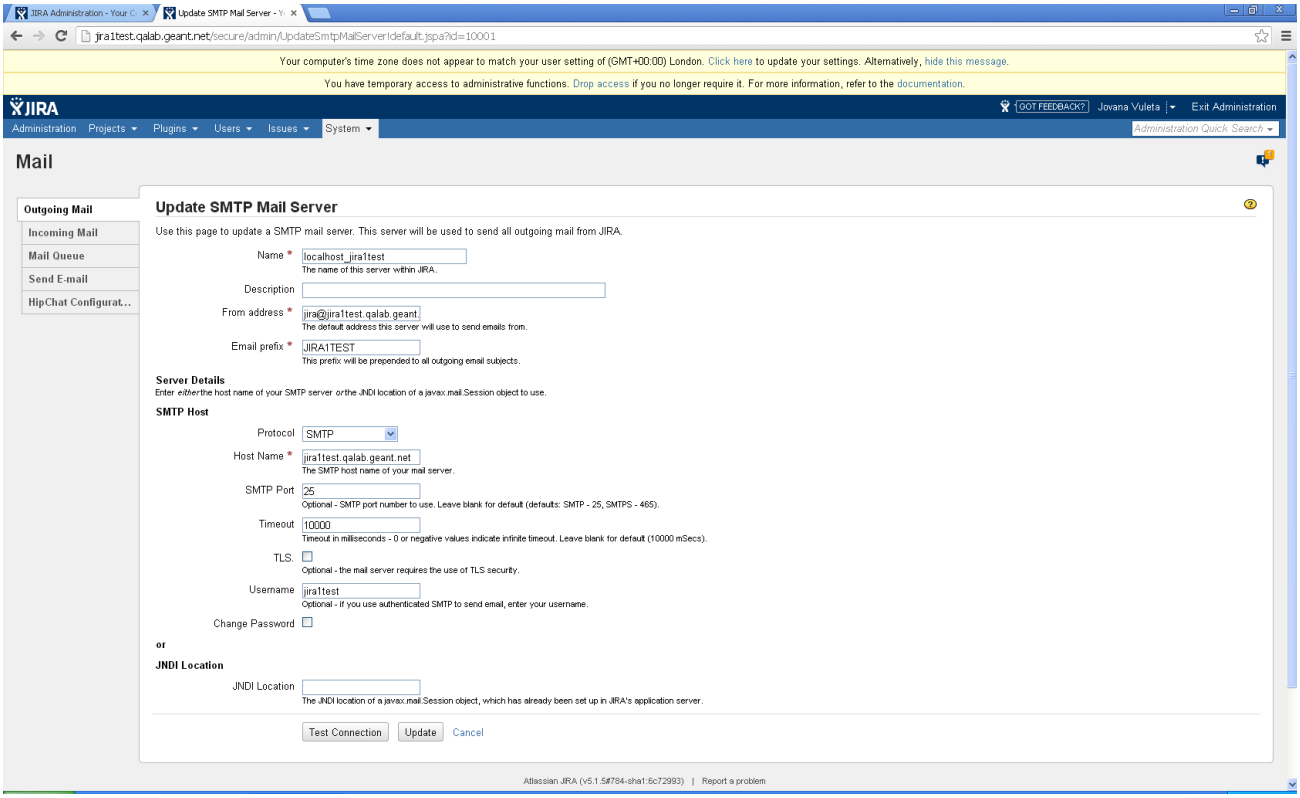


Figure 34: JIRA outgoing mail server

3. Create POP3 mail server `localhost_jira`:

Name: localhost_jira
Protocol: POP
Hostname: jira1test.qalab.geant.net
Username: jira

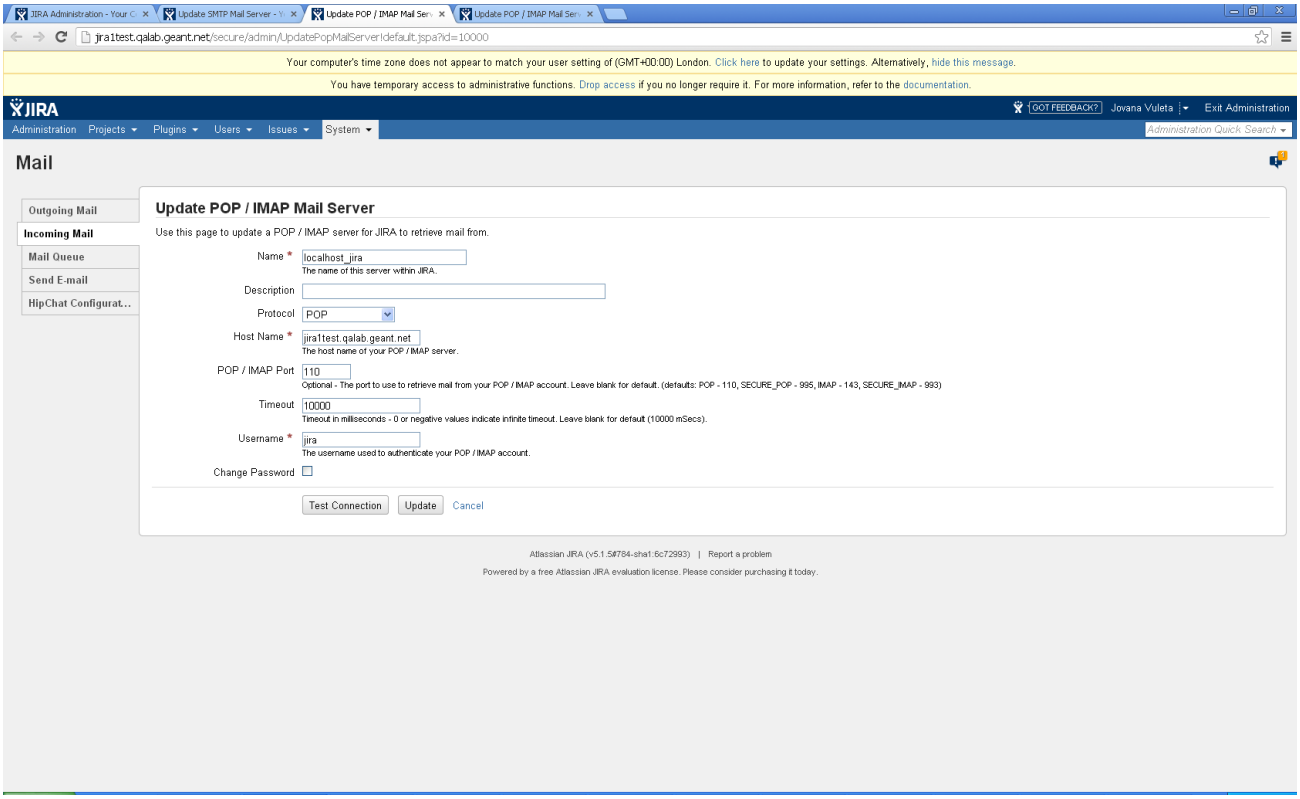


Figure 35: JIRA incoming mail sever

4. Create POP3 mail server `localhost_reply_others`

Name: localhost_reply_others
Protocol: POP
Hostname: jira1test.qalab.geant.net
Username: reply_others

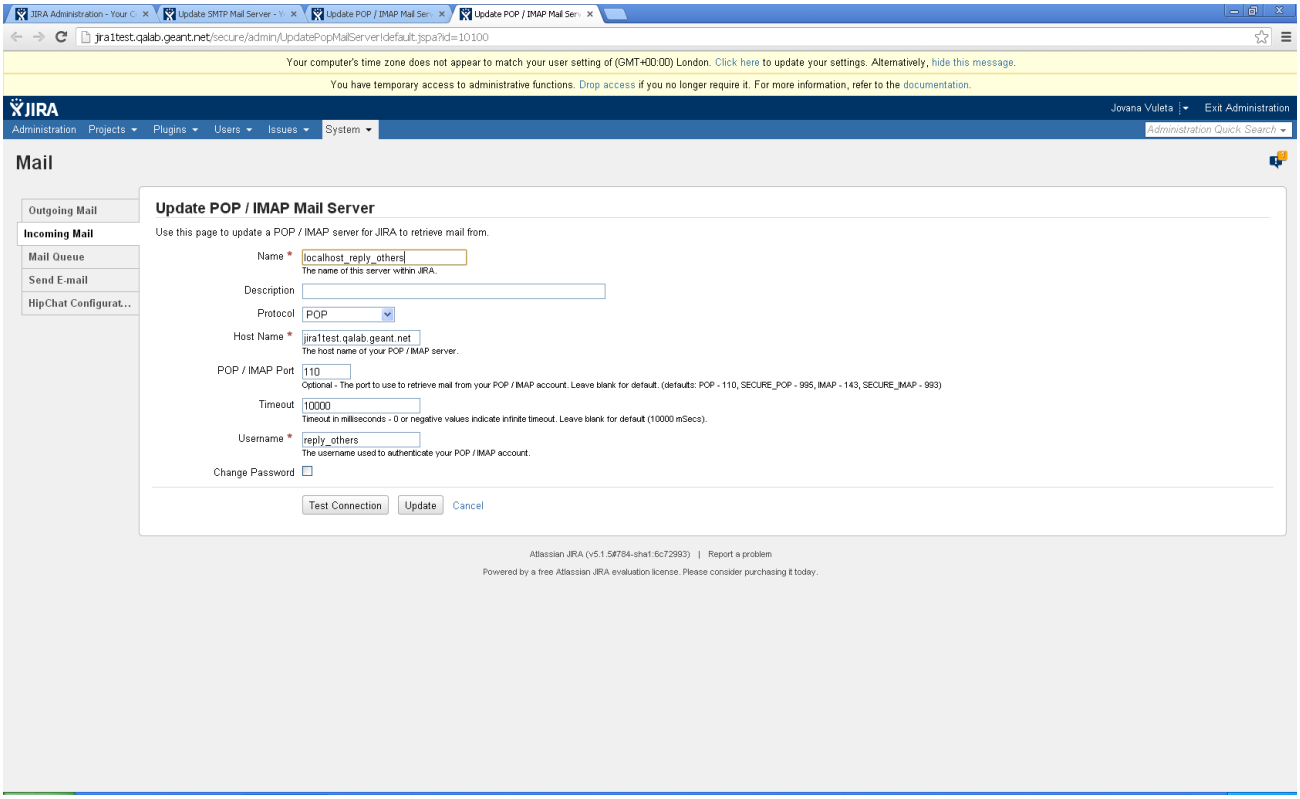


Figure 36: JIRA incoming mail server

5. Create system users for communication with SIM prototype and other TTSes in group `jira-users` sim_client@localhost.localdomain with email sim_client@localhost.localdomain, reply_others@rt1test.qalab.geant.net with email reply_others@rt1test.qalab.geant.net, reply_others@rt2test.qalab.geant.net with email reply_others@rt2test.qalab.geant.net and reply_others@rt3test.qalab.geant.net reply with email others@rt3test.qalab.geant.net
6. In "`User preferences`" set text for "`Default outgoing email format`" and select `YES` radio button "`Notify users of their own changes?`"
7. Create project `Detected_service_problems` with key `SP`

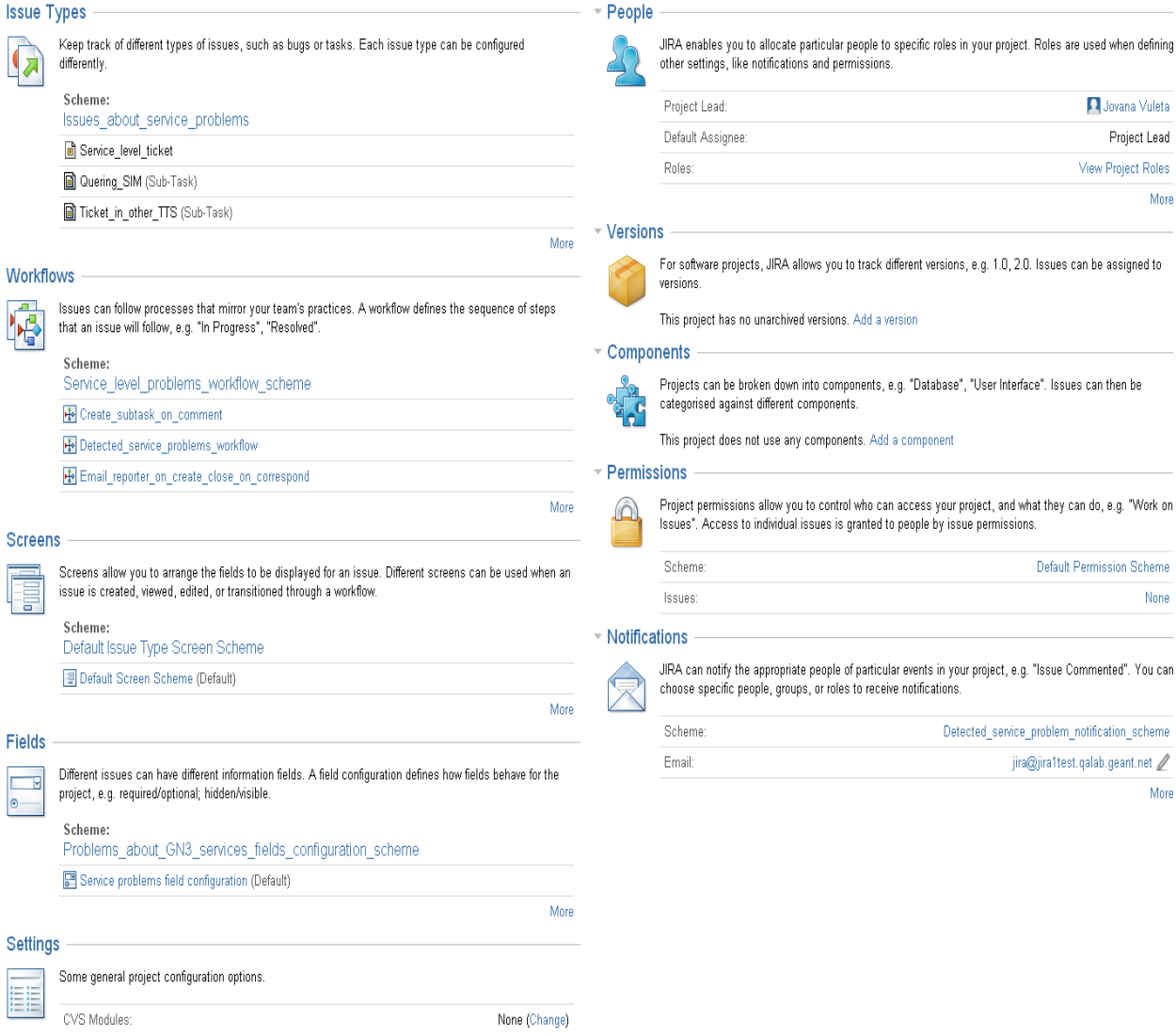


Figure 37: Project Detected_service_problems

8. Create project `Service_problem_reported_by_others` with key `RSP`

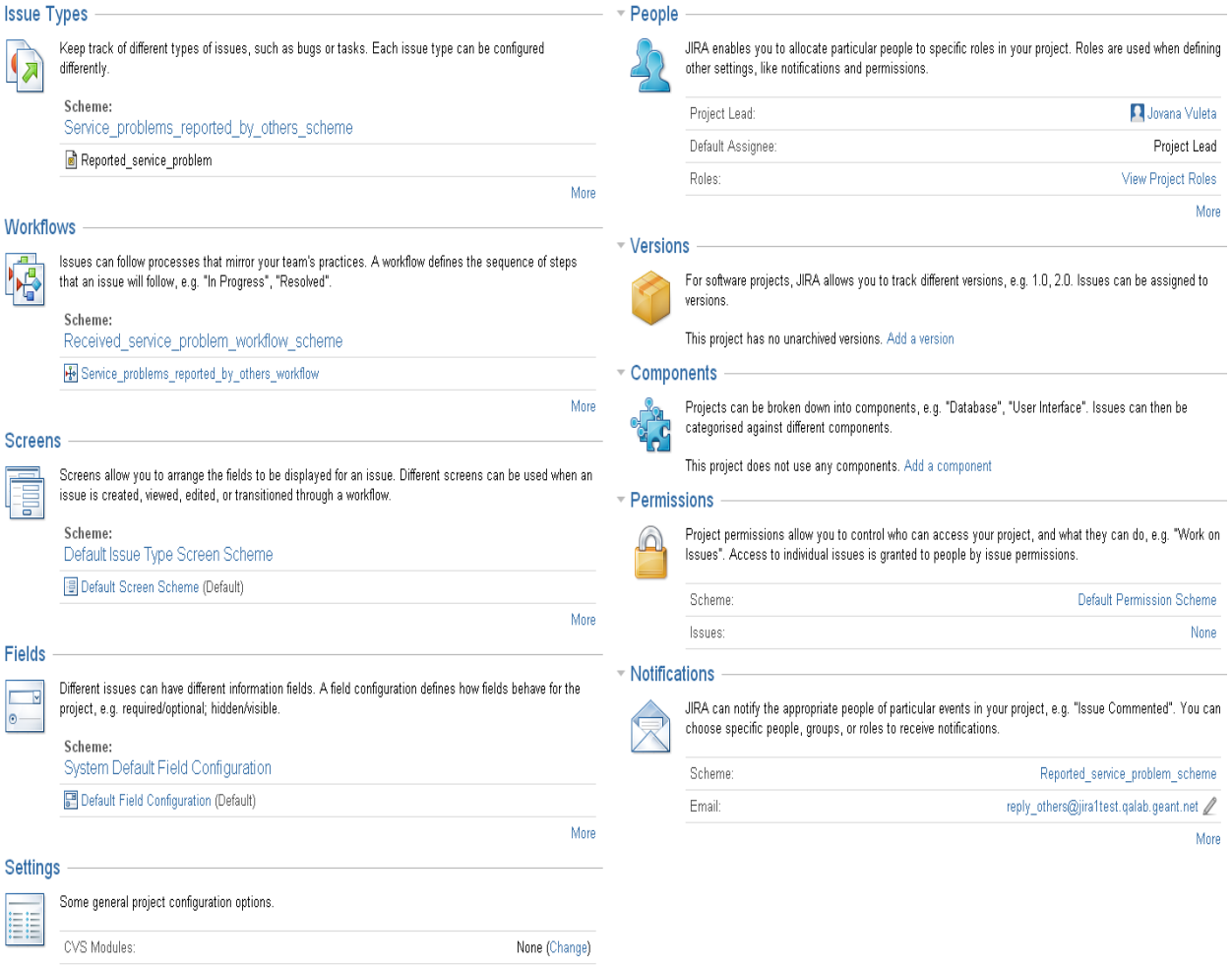


Figure 38: Project Service_problem_reported_by_others

9. Create issue types: `Service_level_ticket` and `Reported_service_problem`, and subtasks `Quering_SIM` and `Ticket_in_other_TTS` as subtasks of `Service_level_ticket`.

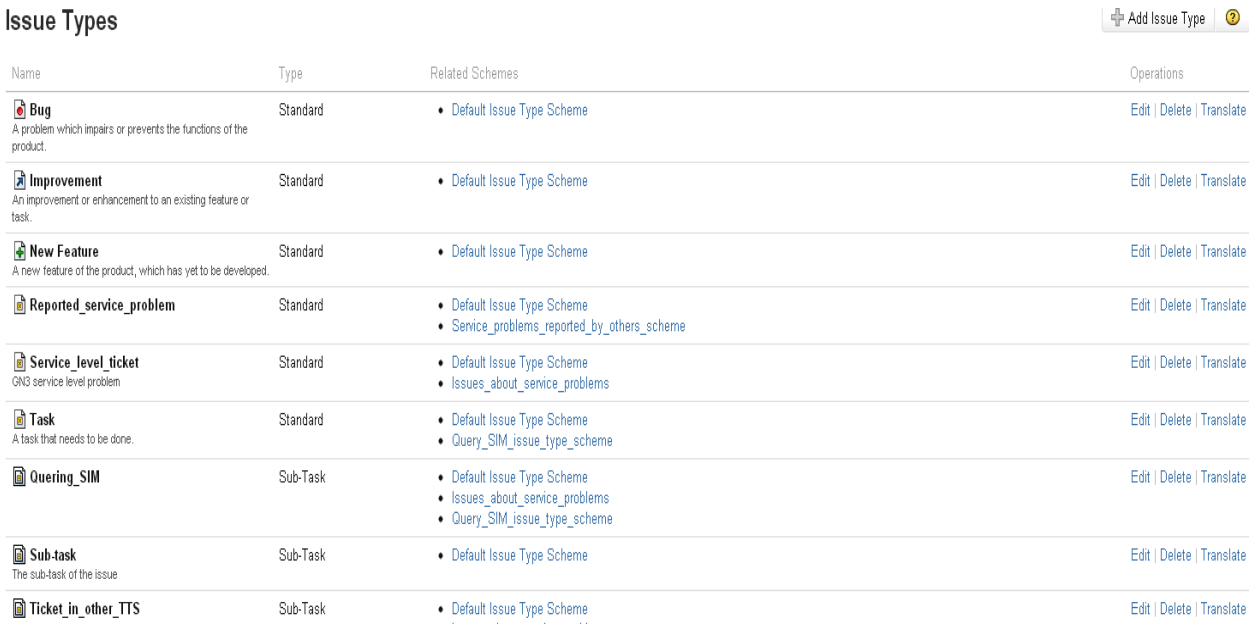


Figure 39: JIRA issue types

10. Create issue type schemes:
    - `Issues_about_service_problems` with issue types `Service_level_ticket` (which should be marked as default issue type), `Quering_SIM` and `Ticket_in_other_TTS`
    - `Service_problems_reported_by_others_scheme` with issue type `Reported_service_problem` (which should be marked as default issue type)
11. Create custom fields:
    - "`Unique Service ID`" of type "`Text Field (< 255 characters)`" for issue type `Service_level_ticket` in project `Detected_service_problems`
    - "`Message for others`" of type "`Free Text Field (unlimited text)`" for issue types `Service_level_ticket` and `Quering_SIM` in project `Detected_service_problems`
    - "`Watcher`" of type "`User Picker`" for issue types `Service_level_ticket` and `Quering_SIM` in project Detected_service_problems

Figure 40: Custom fields

12. Change issue type scheme for project `Detected_service_problems` to `Issues_about_service_problems`
13. Change issue type scheme for project `Service_problem_reported_by_others` to `Service_problems_reported_by_others_scheme`
14. Copy "`Default Field Configuration`" to "`Service problems field configuration`"
15. Make custom fields "`Unique Service ID`" and "`Message for others`" mandatory in "`Service`

problems field configuration": Adminstration -> Field configurations, click on Configure next to "Default Field Configuration" and click on "Required" for custom fields "Unique Service ID", "Message for others" and "Remote_TTS"

16. Create field configuration scheme "Problems_about_GN3_services_fields_configuration_scheme" and associate "Service problems field configuration" to issue types Service_level_ticket and Quering_SIM

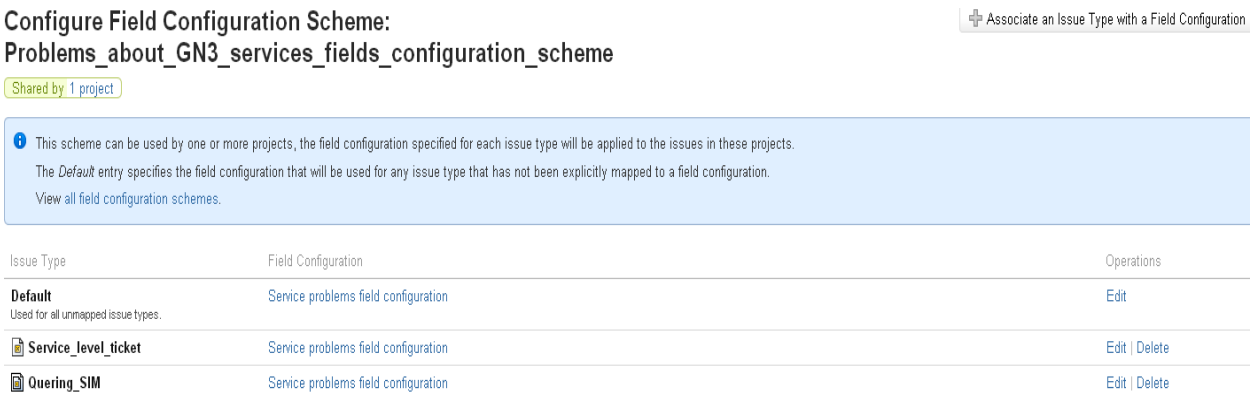


Figure 41: Problems_about_GN3_services_fields_configuration_scheme

17. Change field configuration scheme for project Detected_service_problems to Problems_about_GN3_services_fields_configuration_scheme
18. Copy "Default notification scheme", rename it to "Detected_service_problem_notification_scheme", delete all default notification and add notifications to "Reporter" and "All Wathcers" for event "Issue Created"
19. Copy "Default notification scheme", rename it to "Reported_service_problem_scheme", delete all default notification and add notifications to "Reporter" for event "Issue Resolved"
20. Change notification scheme for project Detected_service_problems to "Detected service problems - Notification Scheme"
21. Change notification scheme for project Service_problem_reported_by_others to "Reported_service_problem_scheme"
22. Install "Create On Transition for JIRA" plugin (https://marketplace.atlassian.com/plugins/org.swift.jira.cot)
23. Install "JIRA Misc Workflow Extensions" plugin
24. Install "JIRA Misc Custom Fields" plugin
25. Install "JIRA Toolkit Plugin " plugin
26. Install "JIRA Workflow Enhancer" plugin
27. Install "Support Tools Plugin " plugin
28. Install "CustomWare JIRA Utilities" plugin (https://marketplace.atlassian.com/plugins/com.keplerrominfo.jira.plugins.jjupin/version/29)
29. Create workflow Detected_service_problem_workflow:
    - add statuses: DEFERRED.QUERING_SIM, DEFERRED.WAITING_FOR_APPROVALS, OPENACTIVE.VERIFY_SERVICE_STATUS, CLOSED.POSITIVELY_RESOLVED, CLOSED.NEGATIVELY_RESOLVED
    - create workflow:

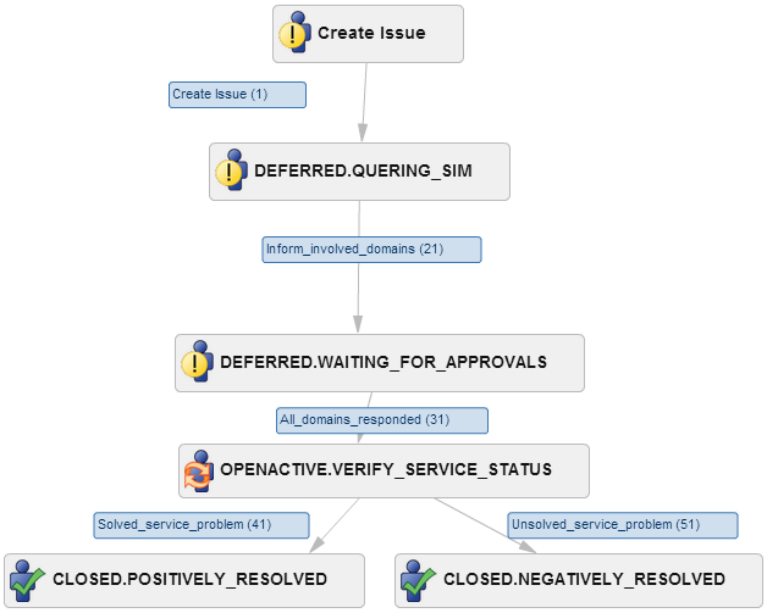


Figure 42: Detected_service_problems_workflow

- add post-functions and conditions:
  - transition `Create_Issue`: after "`Creates the issue originally.`" but before "`Fire a` **`Issue Created`** `event that can be processed by the listeners.`" add post-function with name "`Create sub-task on transition`" and parameters:

Sub-task type: `Quering_SIM`
Sub-task reporter: `Current user`
Sub-task assignee: `Unassigned`
Sub-task summary: `%parent_key%`
Sub-task description:

```
Reply-To:jira@jira1test.qalab.geant.net
Unique service id:  %Unique Service ID%
%description%
endofcontent
```

Set sub-task custom fields:
name: `Watcher`
value: `sim_client`
name: `Message for others`
value: `%Message for others%`

  - transition `All_domains_responded`: add condition "`All sub-tasks must have one of the following statuses to allow parent issue transitions:`**`Resolved`** `or` **`Closed.`**"

30. Create workflow `Create_subtask_on_comment`:
    - add status: `DEFERRED.WAITING`
    - create workflow:

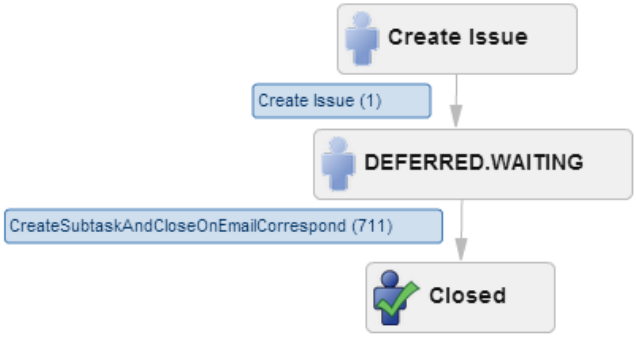


Figure 43: Workflow Create_subtask_on_comment

- add post-functions:
  - transition `CreateSubtaskAndCloseOnEmailCorrespond`: after "`Set issue status to the linked status of the destination workflow step.`" but before "`Fire a` **`Generic Event`** `event that can be processed by the listeners.`" add transition "`Transition` **`Inform_involved_domains`** `will be triggered on the issue's parent issue.`"

31. Create workflow `Email_reporter_on_create_close_on_correspond`
    - add status: `DEFERRED.WAITING_FOR_MAIL`
    - create workflow:

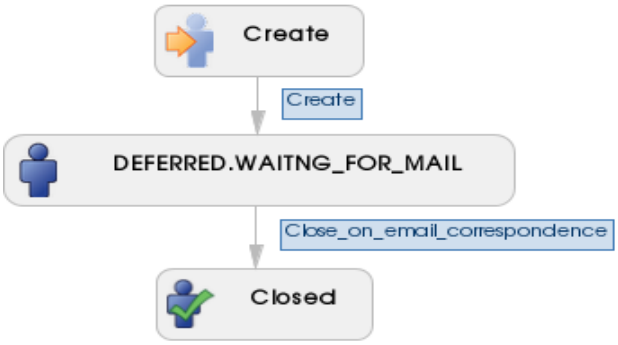


Figure 44: Workflow Email_reporter_on_create_close_on_correspon

- add post-functions:
  - transition `Close_on_email_correspondence`: after "`Set issue status to the linked status of the destination workflow step.`" but before "`Fire a` **`Generic Event`** `event that can be processed by the listeners.`" add post-function with name "`Update Issue Field`" and parameters:

Issue Field: Resolution
Field Value: Fixed

  - transition `Close_on_email_correspondence`: after previously added `Update issue field` post-function but before "`Fire a` **`Generic Event`** `event that can be processed by the listeners.`" add transition "`Transition` **`All_domains_responded`** `will be triggered on the issue's parent issue.`"

32. Create workflow `Service_problem_reported_by_others_workflow`
    - add statuses: `OPENACTIVE`, `CLOSED.RESOLVED`, `CLOSED.NONEXISTENT_PROBLEM`
    - create workflow:

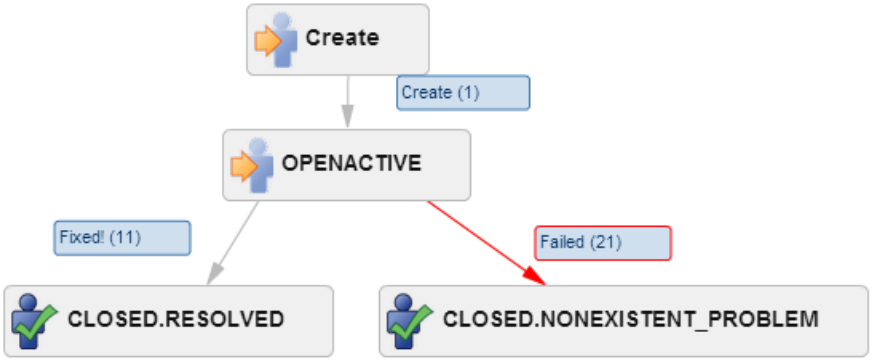


Figure 45: Service_problems_reported_by_others_workflow

- add post-functions:
  - transition `Fixed!`: after "`Set issue status to the linked status of the destination workflow step.`" but before "`Fire a` **`Generic Event`** `event that can be processed by the listeners.`" add post-function with name "`Update Issue Field`" and parameters:

Issue Field: Resolution
Field Value: Fixed

  - transition `Failed!`: after "`Set issue status to the linked status of the destination workflow step.`" but before "`Fire a` **`Generic Event`** `event that can be processed by the listeners.`" add post-function with name "`Update Issue Field`" and parameters:

Issue Field: Resolution
Field Value: Fixed

33. Write custom email handler according to the Plugin tutorial on https://developer.atlassian.com/display/JIRADEV/Plugin+Tutorial+-+Writing+a+Custom+Message+(Mail)+Handler+for+JIRA

with DemoHandler implementation:

```
package com.example.plugins.tutorial.jira.mailhandlerdemo;

import com.atlassian.crowd.embedded.api.User;
import com.atlassian.jira.issue.Issue;
import com.atlassian.jira.service.util.ServiceUtils;
import com.atlassian.jira.service.util.handler.MessageHandler;
import com.atlassian.jira.service.util.handler.MessageHandlerContext;
import com.atlassian.jira.service.util.handler.MessageHandlerErrorCollector;
import com.atlassian.jira.service.util.handler.MessageUserProcessor;
import com.atlassian.mail.MailUtils;

import java.util.Map;
import javax.mail.Message;
import javax.mail.MessagingException;

public class DemoHandler implements MessageHandler {
      private String issueTypeId;
      private MessageUserProcessor messageUserProcessor;

      public DemoHandler(MessageUserProcessor messageUserProcessor) {
            this.messageUserProcessor = messageUserProcessor;

            this.issueTypeId = "8";
      }

    @Override
```

```
    public void init(Map<String, String> params, MessageHandlerErrorCollector
monitor) {
    }

    @Override
    public boolean handleMessage(Message message, MessageHandlerContext context)
throws MessagingException {
            String subject = message.getSubject();
            System.out.print("Message subject: ");
            System.out.println((subject == null) ? "NULL!!!" : "'" + subject +
"'");
            final String body = MailUtils.getBody(message);
            System.out.print("Message body: ");
            System.out.println((subject == null) ? "body je NULL!!!" : "'" + body +
"'");

            Issue commentedIssue = ServiceUtils.findIssueObjectInString(subject);

            if (commentedIssue == null) {
                  return false;
            }
            if (!"8".equals(commentedIssue.getIssueTypeObject().getId())) {
                  return false;
            }

            final User sender = messageUserProcessor.getAuthorFromSender(message);
            context.createComment(commentedIssue, sender, body, true);
            System.out.println("Dodat komentar koji ispaljuje ISSSUE_COMMENTED
event");
            return true;
    }

}
```

34. Add listener "`Close on comment`" `com.atlassian.jira.toolkit.listener.AutoTransitionListener` for event "`Issue Commented`" in status `DEFERRED.WAING_FOR_MAIL` trigger transition Close_on_email_correspondence
35. Add line to template `issuecreated.vm ($JIRA_HOME/atlassian-jira-5.1.5-standalone/atlassian-jira/WEB-INF/classes/templates/email/text/issucreated.vm)`

```
Subject tag: [$issue.key]
```

JIRA – installation and configuration – version 2

This version is based on appropriate configuration of publicly available plugins.

Assumptions:

- Virtual host name is jira2test.qalab.geant.net
- Operating system is Linux
- Add Linux users: jira2test, sim_client and reply_others

Postfix configuration:

1. Disable selinux: in file `/etc/sysconfig/selinux` change line
   `SELINUX=enforcing`
   to
   `SELINUX=disabled`
2. Reboot virtual machine
   `reboot now`
3. Copy service inventory management web service client implementation (`wsKlijent.jar`) to `/Jovana/wsKlijent.jar` with appropriate permissions:
   `chmod 777 /Jovana/wsKlijent.jar`
4. Change postfix configuration:
   Change file `/etc/postfix/main.cf`
   - insert line:
     `myhostname = jira2test.qalab.geant.net`
   - comment line:
     `inet_interfaces = localhost`
   - uncomment line:
     `inet_interfaces = all`
   - comment line:
     `mydestination = $myhostname, localhost.$mydomain, localhost`
   - insert line:
     `mydestination = $myhostname, localhost.localdomain, localhost`

   Modify `/etc/aliases`
   - insert lines:
     `sim_client: "|/usr/lib/jvm/jre-1.6.0-openjdk.x86_64/bin/java -jar -Djava.endorsed.dirs=/Jovana/lib/endorsed /Jovana/wsKlijent.jar > /test1.txt"`
   - create file `/test1.txt` with appropriate permissions:
     `touch /test1.txt`
     `chmod 777 /test1.txt`

Dovecot installation and configuration

1. Install dovecot:

```
yum install dovecot
```

2. Configure dovecot - in `/etc/dovecot/dovecot.conf` insert line:

```
mail_location = mbox:~/mail:INBOX=/var/mail/%u
```

3. Change permissios for /var/spool/mail/* or dovecot won't work properly(http://dovecot.markmail.org/message/ovjoror2aveauzqy?q=chown+Operation+not+permitted+order:date-backward&page=1)

```
chmod 0600 /var/spool/mail/*
```

JIRA insallation and configuration:

1. Install JIRA – installation process depends on license type. File Readme.txt contains a brief install guide and there is plenty online documentation.
2. Create outgoing SMTP mail server:

Name: SMTP – jira2test.qalab.geant.net
From address: jira@jira2test.qalab.geant.net
Email prefix: JIRA2TEST
Protocol: SMTP
Hostname: jira2test.qalab.geant.net
Username: jira2test

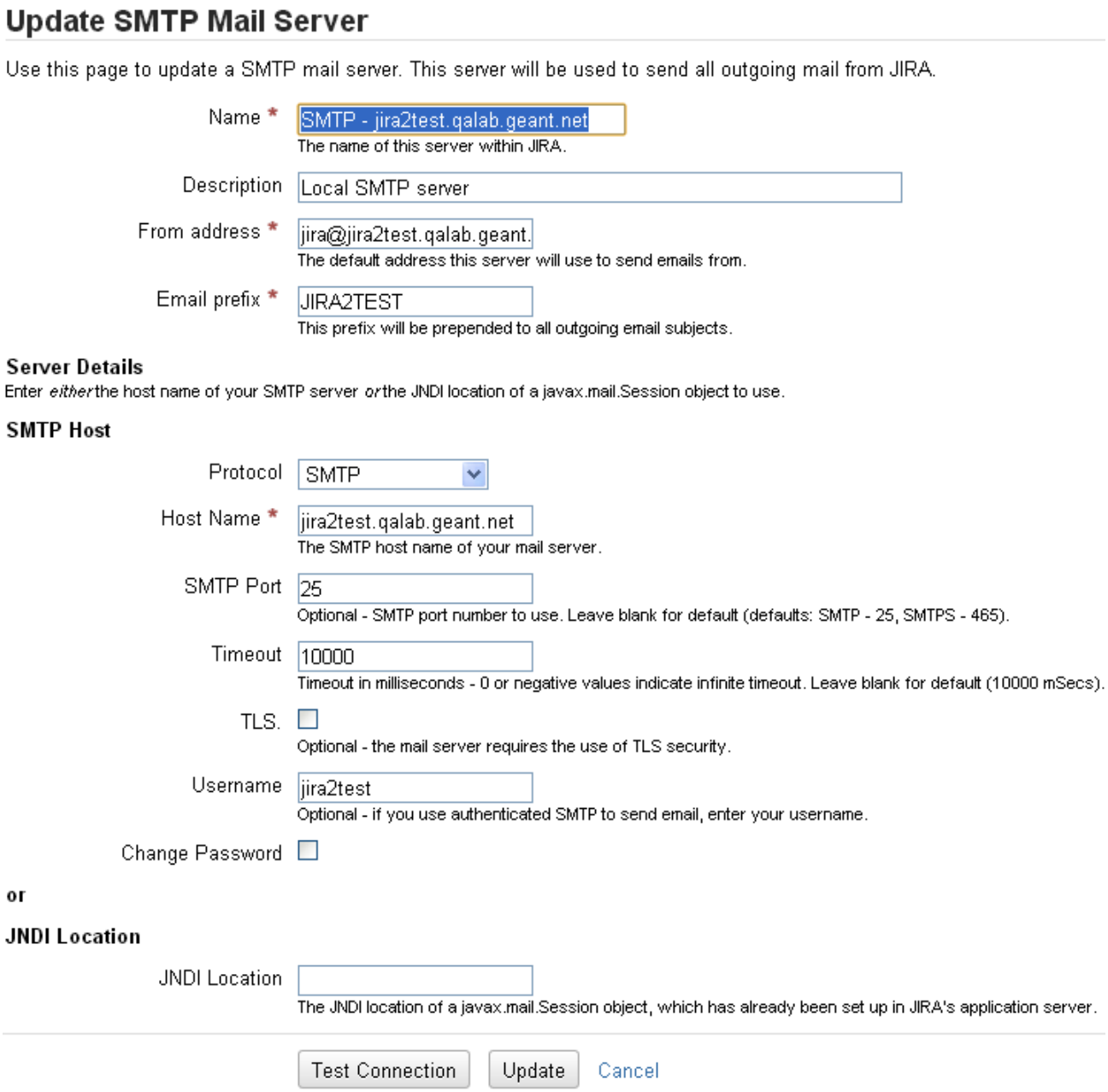


Figure 46: JIRA outgoing mail server

3. Create POP3 mail server `POP3 - detected service problems`:

Name: POP3 - detected service problems

Protocol: POP
Hostname: jira2test.qalab.geant.net
Username: jira2test

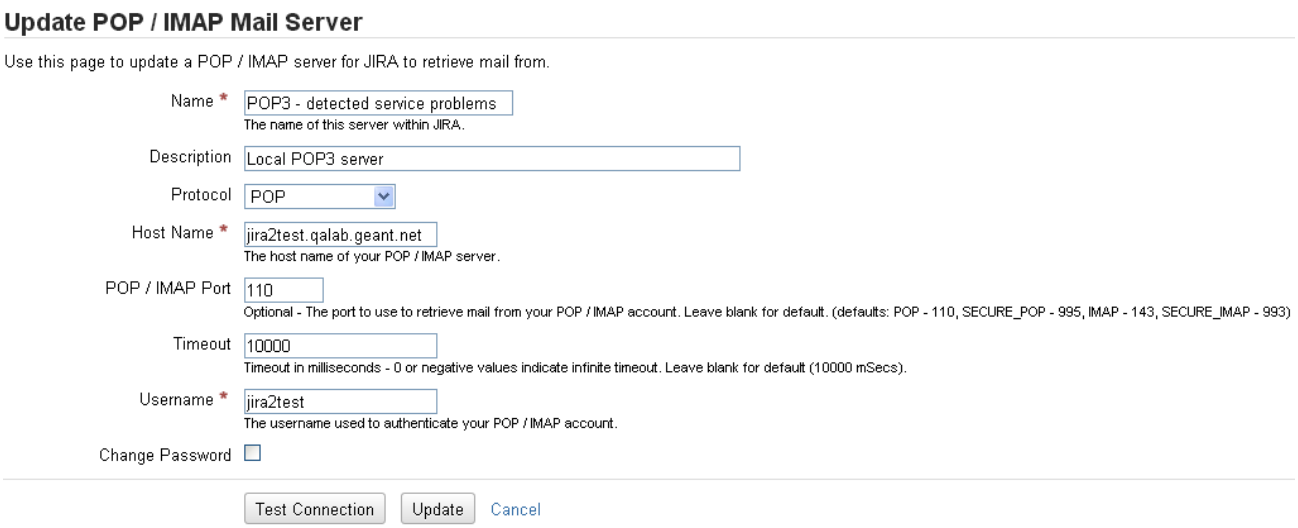


Figure 47: JIRA incoming mail server

4. Create POP3 mail server `POP3 - received service problems`

Name: POP3 - received service problems
Protocol: POP
Hostname: jira2test.qalab.geant.net
Username: reply_others

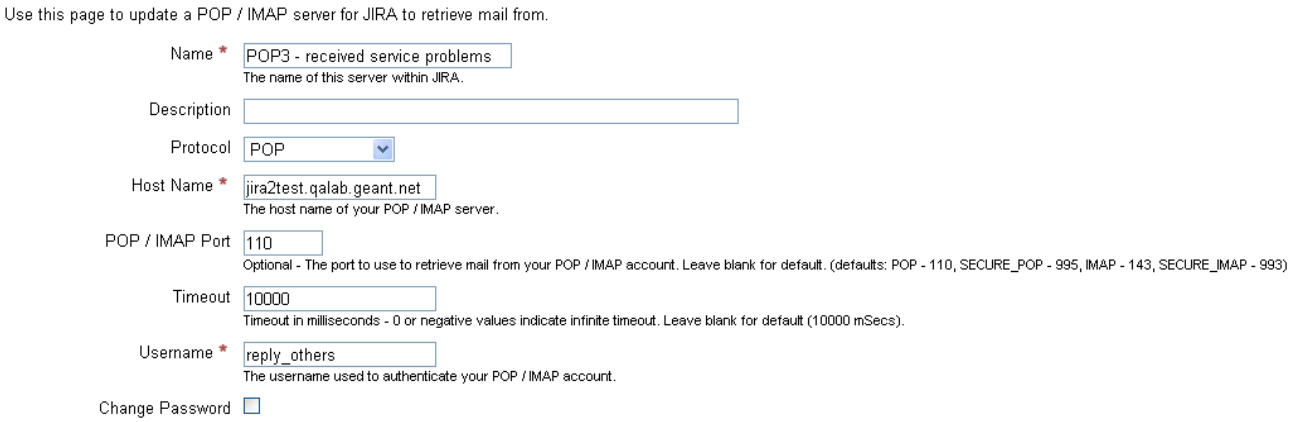


Figure 48: JIRA incoming mail server

5. Create projects: `Detected_service_problems` and `Service_problem_reported_by_others`

Issue Types

Keep track of different types of issues, such as bugs or tasks. Each issue type can be configured differently.

Scheme:
Service_problem_issues_scheme

- Service_level_ticket (Default)
- Quering_SIM (Sub-Task)
- Ticket_in_other_TTS (Sub-Task)

More

Workflows

Issues can follow processes that mirror your team's practices. A workflow defines the sequence of steps that an issue will follow, e.g. "In Progress", "Resolved".

Scheme:
Detected_service_level_problems_workflow_scheme

- Detected_service_problem_workflow
- Email_reporter_on_create_close_on_correspond
- On_create_email_reporter_on_correspond_create_subtasks_and_close

More

Screens

Screens allow you to arrange the fields to be displayed for an issue. Different screens can be used when an issue is created, viewed, edited, or transitioned through a workflow.

Scheme:
Default Issue Type Screen Scheme

- Default Screen Scheme (Default)

More

People

JIRA enables you to allocate particular people to specific roles in your project. Roles are used when defining other settings, like notifications and permissions.

Project Lead:	Root
Default Assignee:	Unassigned
Roles:	View Project Roles

More

Versions

For software projects, JIRA allows you to track different versions, e.g. 1.0, 2.0. Issues can be assigned to versions.

This project has no unarchived versions. Add a version

Components

Projects can be broken down into components, e.g. "Database", "User Interface". Issues can then be categorised against different components.

This project does not use any components. Add a component

Permissions

Project permissions allow you to control who can access your project, and what they can do, e.g. "Work on Issues". Access to individual issues is granted to people by issue permissions.

Scheme:	Default Permission Scheme
Issues:	None

Notifications

JIRA can notify the appropriate people of particular events in your project, e.g. "Issue Commented". You can choose specific people, groups, or roles to receive notifications.

Scheme:	Detected service problem - Notification Scheme

Issue Types

Keep track of different types of issues, such as bugs or tasks. Each issue type can be configured differently.

Scheme:
Service_problems_reported_by_others_scheme

- Reported_service_problem (Default)

More

Workflows

Issues can follow processes that mirror your team's practices. A workflow defines the sequence of steps that an issue will follow, e.g. "In Progress", "Resolved".

Scheme:
Reported_service_problem_scheme

- Service_problem_reported_by_others_workflow

More

Screens

Screens allow you to arrange the fields to be displayed for an issue. Different screens can be used when an issue is created, viewed, edited, or transitioned through a workflow.

Scheme:
Default Issue Type Screen Scheme

- Default Screen Scheme (Default)

More

Fields

Different issues can have different information fields. A field configuration defines how fields behave for the project, e.g. required/optional; hidden/visible.

Scheme:
System Default Field Configuration

- Default Field Configuration (Default)

More

Settings

Some general project configuration options.

CVS Modules:	None (Change)

People

JIRA enables you to allocate particular people to specific roles in your project. Roles are used when defining other settings, like notifications and permissions.

Project Lead:	Root
Default Assignee:	Unassigned
Roles:	View Project Roles

More

Versions

For software projects, JIRA allows you to track different versions, e.g. 1.0, 2.0. Issues can be assigned to versions.

This project has no unarchived versions. Add a version

Components

Projects can be broken down into components, e.g. "Database", "User Interface". Issues can then be categorised against different components.

This project does not use any components. Add a component

Permissions

Project permissions allow you to control who can access your project, and what they can do, e.g. "Work on Issues". Access to individual issues is granted to people by issue permissions.

Scheme:	Default Permission Scheme
Issues:	None

Notifications

JIRA can notify the appropriate people of particular events in your project, e.g. "Issue Commented". You can choose specific people, groups, or roles to receive notifications.

Scheme:	None
Email:	jira@jira2test.qalab.geant.net

More

Figure 50: Project Service_problems_reported_by_others

6. Allow unassigned issues: `System -> General Configuration -> Edit Configuration -> Allow unassigned issues` and select radio `ON`

Figure 51: Allowing unassigned issues in general

7. Allow unassigned issues for project `Service_problem_reported_by_others` (`Administration -> Service_problem_reported_by_others -> People → More`, press pencil next to `Assignee`: and select `Unassigned`)
8. Create issue types: `Service_level_ticket` and `Reported_service_problem`, and subtasks `Quering_SIM` and `Ticket_in_other_TTS` as subtasks of `Service_level_ticket`.
9. Create issue type schemes:
   - `Service_problem_issues_scheme` with issue types `Service_level_ticket` (which should be marked as default issue type), `Quering_SIM` and `Ticket_in_other_TTS`
   - `Service_problems_reported_by_others_scheme` with issue type `Reported_service_problem` (which should be marked as default issue type)
10. Create custom fields:
    - "`Unique Service ID`" of type "`Text Field (< 255 characters)`" for issue type `Service_level_ticket` in project `Detected_service_problems`
    - "`Message for others`" of type "`Free Text Field (unlimited text)`" for issue type `Service_level_ticket` in project `Detected_service_problems`
    - "`SIM_client`" of type "`User picker`" for issue type `Quering_SIM` in project `Detected_service_problems`
    - "`Remote_TTS`" of type "`Text Field (< 255 characters)`" for issue type `Ticket_in_other_TTS` in project `Detected_service_problems`
    - "`Reporter_email `" of type "`Text Field (< 255 characters)`" for issue type `Reported_service_problem` in project `Service_problems_reported_by_others`
    - "`Reporter_name`" of type "`Text Field (< 255 characters)`" for issue type `Reported_service_problem` in project `Service_problems_reported_by_others`
11. Make custom fields "`Unique Service ID`", "`Message for others`", "`Remote_TTS`" mandatory: `Adminstration -> Field configurations`, click on `Configure` next to "`Default Field Configuration`" and click on "`Required`" for custom fields "`Unique Service ID`", "`Message for others`" and "`Remote_TTS`"
12. Change issue type scheme for project `Detected_service_problems` to `Service_problem_issues_scheme`
13. Change issue type scheme for project `Service_problem_reported_by_others` to `Service_problems_reported_by_others_scheme`
14. Set field `Reporter` in field configuration to optional (`Adminstration -> Field configurations`, click on `Configure` next to "`Default Field Configuration`" and click on "`Optional`" for field "`Reporter`")
15. Allow anonymous issue creation by giving permissions `Browse Project` and `Create Issue` to the group `Anyone` in the permission scheme (for the project or default one). This requires field Reporter to be set as optional.
16. Copy "`Default notification scheme`", rename it to "`Detected service problems - Notification Scheme`", delete all default notification and add notifications to "`All watchers`" and "`User Custom Field Value (SIM_client)`" for event "`Issue created`"
17. Change notification scheme for project `Detected_service_problems` to "`Detected service problems - Notification Scheme`"

18. Install "Create On Transition for JIRA" plugin (https://marketplace.atlassian.com/plugins/org.swift.jira.cot)
19. Install "JJUPIN - Simple Issue Language (SIL) for JIRA" plugin (https://marketplace.atlassian.com/plugins/com.keplerrominfo.jira.plugins.jjupin/version/29)
20. Install "JIRA Misc Workflow Extensions" plugin
21. Install "JIRA Misc Custom Fields" plugin
22. Install "JIRA Toolkit Plugin " plugin
23. Install "JIRA Workflow Enhancer" plugin
24. Install "Support Tools Plugin " plugin
25. Install and setup "JIRA Enterprise Mail Handler" plugin (https://marketplace.atlassian.com/plugins/com.javahollic.jira.jemh-ui)
    - Create profile Detected_service_level_problems_profile, as default project set Detected_service_level_problems, as default issue type set Quering_SIM, turn ON regexp field processor and enable At(@) Prefix regexp match configuration
    - Create profile Service_problems_reported_by_others_profile, as default project set Service_problem_reported_by_others, as default issue type set Reported_service_problem, enable JEMH listener and add custom field Remote_TTS as non-JIRA account holder turn ON regexp field processor and enable Basic match configuration
26. Create workflow Detected_service_problem_workflow:
    - add statuses: DEFERRED.QUERING_SIM, DEFERRED.WAITING_FOR_APPROVALS, OPENACTIVE.VERIFY_SERVICE_STATUS, CLOSED.POSITIVELY_RESOLVED, CLOSED.NEGATIVELY_RESOLVED
    - create workflow:

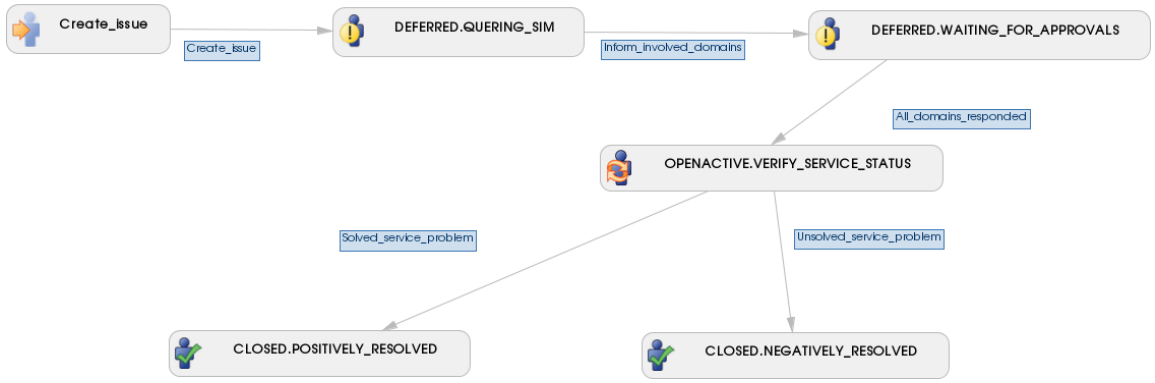


Figure 52: Detected_service_problem_workflow

    - add post-functions and conditions:
      - transition Create_Issue: after "Creates the issue originally." but before "Fire a **Issue Created** event that can be processed by the listeners." add post-function with name "Create sub-task on transition" and parameters:

```
Sub-task type: Quering_SIM
Sub-task reporter: sim_client
Sub-task assignee: Unassigned
Sub-task summary: %parent_summary%
```

Sub-task description:
Reply-To:jira2test@jira2test.qalab.geant.net
Unique service id: %Unique Service ID%
%description%
endofcontent
Set sub-task custom fields:
name: Unique Service ID
value: %Unique Service ID%
name: Message for others
value: %Message for others%

name: SIM_client
value: sim_client

- transition `Inform_involved_domains`: after "`Set issue status to the linked status of the destination workflow step.`" and "`Add a comment to an issue if one is entered during a transition.`" but before "`Fire a` **`Generic Event`** `event that can be processed by the listeners.`" add post-function with name "`(k) SIL Post-function`" and content:

```
//Start your programs with code comments, this way it will show here (max 3 lines)
// addComment(key, "root", "Triggered autotransiotion on " + subtaskKey);

string subtaskKey;

for (subtaskKey in subtasks(key)) {
  if (%subtaskKey%.type == "Ticket_in_other_TTS") {
    autotransition("Send_notification_emails", subtaskKey);
  }
}
```

- transition `All_domains_responded`: add condition "`All sub-tasks must have one of the following statuses to allow parent issue transitions:`**`Resolved`** `or` **`Closed`**`."`

27. Create workflow `On_create_email_reporter_on_correspond_create_subtasks_and_close`:
    - add status: `DEFERRED.WAITING_FOR_MAIL`
    - create workflow:

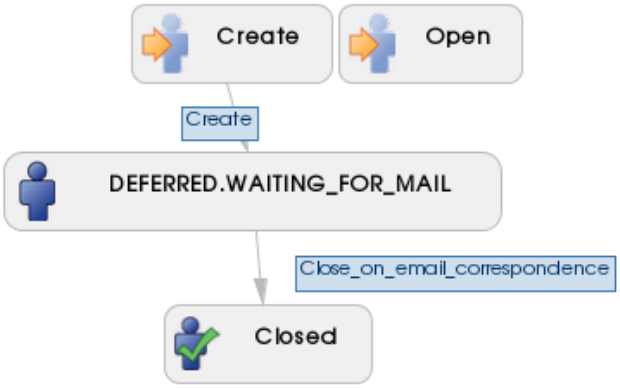


Figure 53: Worflow On_create_email_reporter_on_correspond_create _subtasks_and_close

    - add post-functions:
      - transition `Close_on_email_correspondence`: after "`Set issue status to the linked status of the destination workflow step.`" but before "`Fire a` **`Generic Event`** `event that can be processed by the listeners.`" add post-function with name "`(k) SIL Post-function`" and content:

```
// Subtask creation, one per each e-mail address
// summary is CSV list of e-mail addresses because JEMH postpones custom field
update

string involvedDomains = summary;
string querySIMKey = key;
resolution = "Fixed";

if (isNotNull(involvedDomains)) {
  number commaIndex = indexOf(involvedDomains, ",");
  string currentTTS;
```

```
  while (commaIndex != -1) {

    currentTTS = substring(involvedDomains, 0, commaIndex);

    if (!endsWith(currentTTS, "@jira2test.qalab.geant.net")) {
      string currentSubtaskKey = createIssue("DSP", parent,  "Ticket_in_other_TTS",
%parent%.summary);
      %currentSubtaskKey%.description = %parent%.customfield_10000;
      %currentSubtaskKey%.customfield_10103 = currentTTS;
    } else {
    }

    number involvedDomainsLength = length(involvedDomains);
    involvedDomains = substring(involvedDomains, commaIndex + 1,
involvedDomainsLength);
    commaIndex = indexOf(involvedDomains, ",");
  }


  currentTTS = involvedDomains;
  if (!endsWith(currentTTS, "@jira2test.qalab.geant.net")) {
    string currentSubtaskKey = createIssue("DSP", parent,  "Ticket_in_other_TTS",
%paren%.summary);
    %currentSubtaskKey%.description = %parent%.customfield_10000;
    %currentSubtaskKey%.customfield_10103 = currentTTS;
  } else {
  }

}
```

- transition `Close_on_email_correspondence`: after previously added `(k) SIL` post-function but before "`Fire a` **`Generic Event`** `event that can be processed by the listeners.`" add transition "`Transition` **`Inform_involved_domains`** `will be triggered on the issue's parent issue.`"

28. Create workflow `Email_reporter_on_create_close_on_correspond`
    - add status: `OPENACTIVE.CREATED`
    - create workflow:

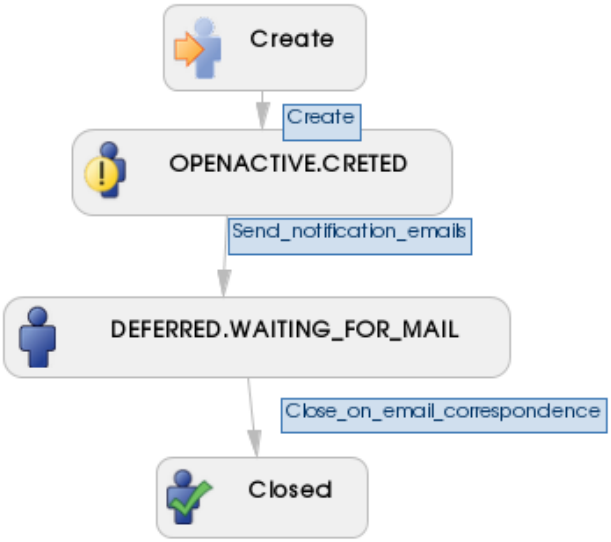


Figure 54: Workflow Email_reporter_on_create_close_on_correspond

- add post-functions:
  - transition `Send_notification_emails`: after "`Set issue status to the linked status of the destination workflow step.`" but before "`Fire a` **`Generic Event`** `event that can be processed by the listeners.`" add post-function with name "`Email This Issue`" and parameters

Recipients (To): Remote_TTS
recipients stored in field: Remote_TTS
Email settings: Text email

  - transition `Close_on_email_correspondence`: after "`Set issue status to the linked status of the destination workflow step.`" but before "`Fire a` **`Generic Event`** `event that can be processed by the listeners.`" add post-function with name "`Update Issue Field`" and parameters:

Issue Field: Resolution
Field Value: Fixed

  - transition `Close_on_email_correspondence`: after previously added `Update issue field` post-function but before "`Fire a` **`Generic Event`** `event that can be processed by the listeners.`" add transition "`Transition` **`All_domains_responded`** `will be triggered on the issue's parent issue.`"

29. Create workflow `Service_problem_reported_by_others_workflow`
    - add statuses: `OPENACTIVE`, `CLOSED.RESOLVED`, `CLOSED.NONEXISTENT_PROBLEM`
    - create workflow:

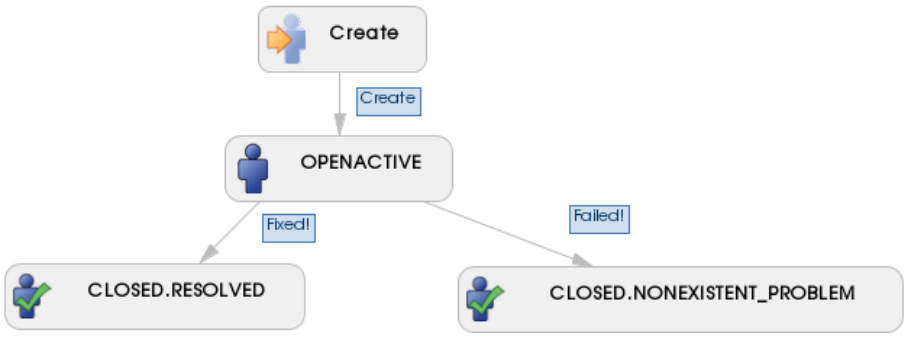


Figure 55: Workflow Service_problem_reported_by_others_workflow

- add post-functions:
  - transition `Fixed!`: after "`Set issue status to the linked status of the destination workflow step.`" but before "`Fire a` **`Generic Event`** `event that can be processed by the listeners.`" add post-function with name "`Email This Issue`" and parameters

Recipients (To): Reporter_email
recipients stored in field: Reporter_email
Email settings: Text email

  - transition `Failed!`: after "`Set issue status to the linked status of the destination workflow step.`" but before "`Fire a` **`Generic Event`** `event that can be processed by the listeners.`" add post-function with name "`Email This Issue`" and parameters

Recipients (To): Reporter_email
recipients stored in field: Reporter_email
Email settings: Text email

30. Add mail handler "`Detected service problems mail handler`" with parameters:

Server: POP3 – detected service problems
Handler: JIRA Enterprise Mail Handler
Profile ID: 3

31. Add mail handler "`Received service problems mail handler`" with parameters:

Server: POP3 – received service problems
Handler: JIRA Enterprise Mail Handler
Profile ID: 7

32. Add line to template `issuecreated.vm ($JIRA_HOME/atlassian-jira-5.1.5-standalone/atlassian-jira/WEB-INF/classes/templates/email/text/issucreated.vm)`

```
Subject tag: [$issue.key]
```

Service Inventory Management Stub – Installation and Configuration

Assumptions:

- Virtual host name is sim-test.qalab.geant.net
- Operating system is Linux

Web service stub generation:

1. Create Eclipse workspace SIM_prototype and set UTF-8 for default text file encoding
2. Add jdk-1.6.0_32 as default Java runtime environment
3. Add server JBoss 5.1.0.GA for jdk-6
4. Create Java project SIM-Prototype-WS and select Dynamic Web Module, Java, JavaScript and JBoss Web Services Core as Project facets and JBoss 5.1 Runtime for project runtime

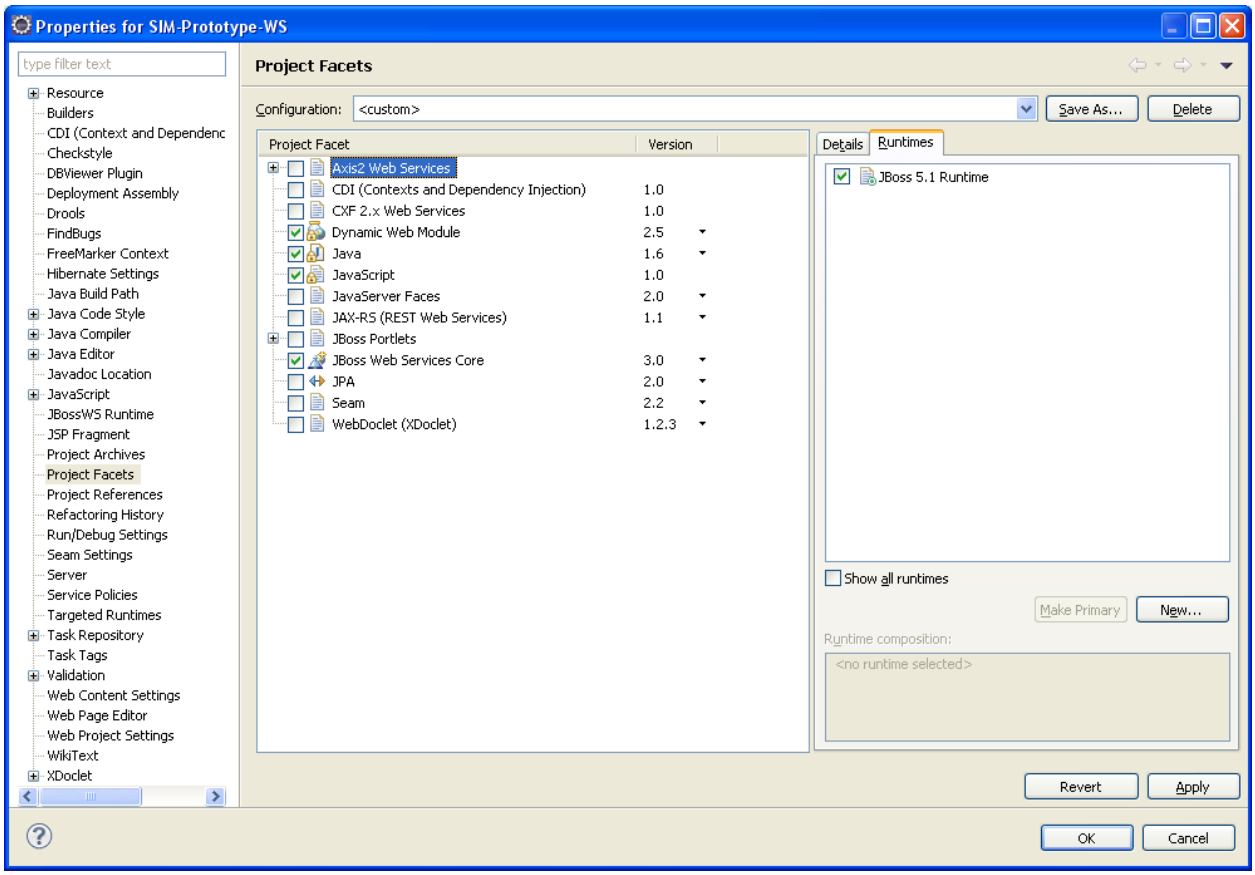


Figure 56: SIM web service project configuration

5. Generate web service using JBossWS: File -> New -> Other -> Web Service

Web service type: Top down Java bean Web Service
Service definition: select OSSJ-Common-v1-5.wsdl
Server runtime: JBoss AS 5.1

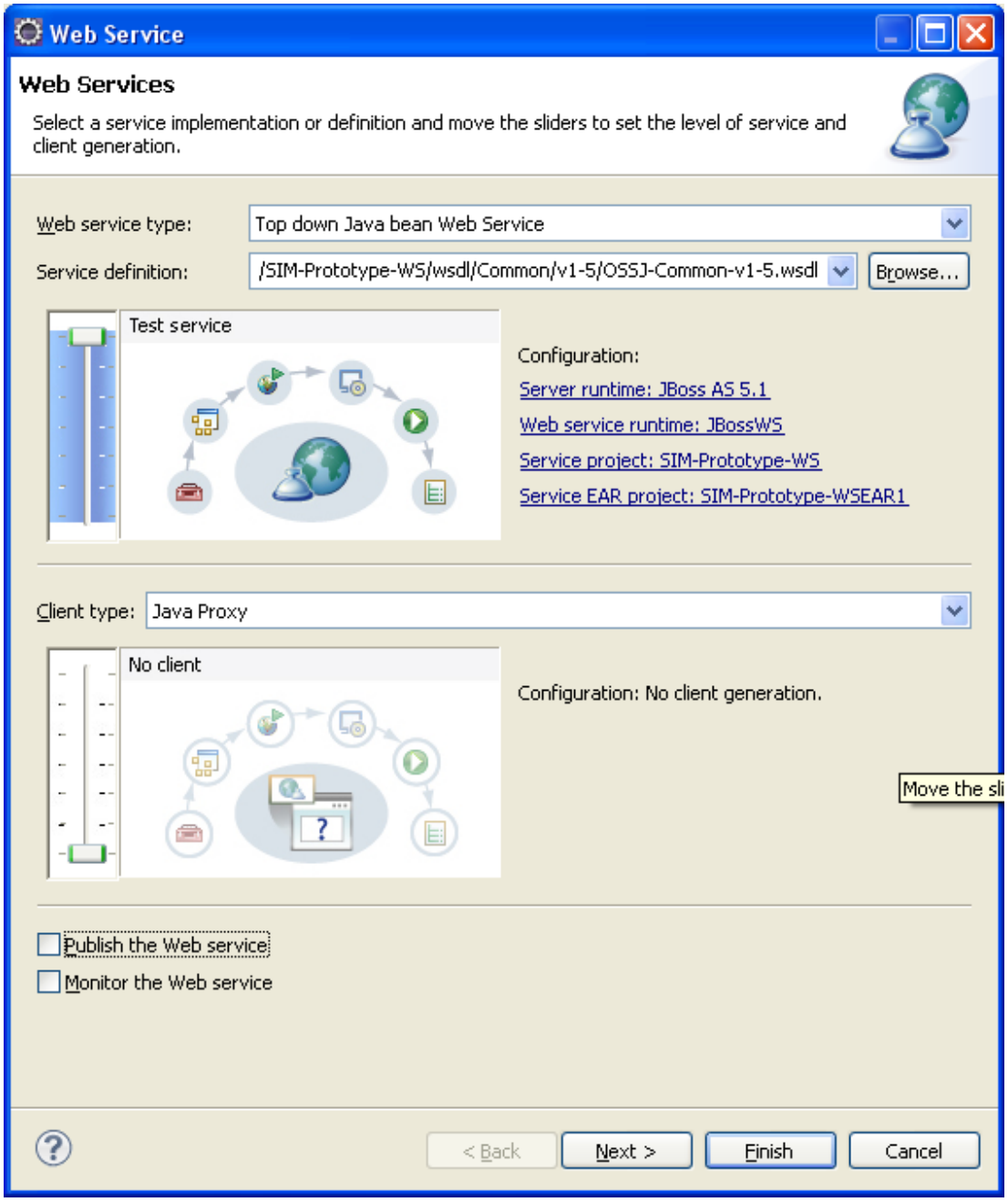


Figure 57: SIM stub - web service generation

Web service runtime: JBossWS

6. Change implementation `org.ossj.wsdl.common.v1_5.impl.JVTSessionWSPortImpl`:

```
package org.ossj.wsdl.common.v1_5.impl;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Properties;
import java.util.List;

import org.ossj.xml.common.v1_5.ArrayOfString;
import org.ossj.xml.common.v1_5.GetEventDescriptorRequest;
import org.ossj.xml.common.v1_5.GetEventDescriptorResponse;
import org.ossj.xml.common.v1_5.GetEventTypesRequest;
```

```
import org.ossj.xml.common.v1_5.GetEventTypesResponse;
import org.ossj.xml.common.v1_5.GetManagedEntityTypesRequest;
import org.ossj.xml.common.v1_5.GetManagedEntityTypesResponse;
import org.ossj.xml.common.v1_5.GetNamedQueryTypesRequest;
import org.ossj.xml.common.v1_5.GetNamedQueryTypesResponse;
import org.ossj.xml.common.v1_5.GetSupportedOptionalOperationsRequest;
import org.ossj.xml.common.v1_5.GetSupportedOptionalOperationsResponse;
import org.ossj.xml.common.v1_5.GetUpdateProcedureTypesRequest;
import org.ossj.xml.common.v1_5.GetUpdateProcedureTypesResponse;
import org.ossj.xml.common.v1_5.NamedQueryResponse;
import org.ossj.xml.common.v1_5.ObjectFactory;
import org.ossj.xml.common.v1_5.QueryRequest;
import org.ossj.xml.common.v1_5.QueryResponse;
import org.ossj.xml.common.v1_5.UpdateRequest;
import org.ossj.xml.common.v1_5.UpdateResponse;
import org.ossj.wsdl.common.v1_5.*;

import javax.jws.WebService;
import javax.xml.bind.JAXBElement;

@WebService(serviceName = "JVTSessionWebService", endpointInterface =
"org.ossj.wsdl.common.v1_5.JVTSessionWSPort", targetNamespace =
"http://ossj.org/wsdl/Common/v1-5")
public class JVTSessionWSPortImpl implements JVTSessionWSPort {

      public JVTSessionWSPortImpl() {
      }

      public GetSupportedOptionalOperationsResponse getSupportedOptionalOperations(
                  GetSupportedOptionalOperationsRequest parameters)
                  throws GetSupportedOptionalOperationsException {
            return null;
      }

      public GetManagedEntityTypesResponse getManagedEntityTypes(
                  GetManagedEntityTypesRequest parameters)
                  throws GetManagedEntityTypesException {
            return null;
      }

      public GetEventTypesResponse getEventTypes(GetEventTypesRequest parameters)
                  throws GetEventTypesException {
            return null;
      }

      public GetEventDescriptorResponse getEventDescriptor(
                  GetEventDescriptorRequest parameters)
                  throws GetEventDescriptorException {
            return null;
      }

      public UpdateResponse update(UpdateRequest parameters)
                  throws UpdateException {
            return null;
      }

      public GetUpdateProcedureTypesResponse getUpdateProcedureTypes(
                  GetUpdateProcedureTypesRequest parameters)
                  throws GetUpdateProcedureTypesException {
```

```
            return null;
      }

      public QueryResponse query(QueryRequest parameters) throws QueryException {

            String uniqueServiceId = null;
            if (parameters != null &&
parameters.getNamedQuery().getUniqueServiceId() != null) {
                  uniqueServiceId =
parameters.getNamedQuery().getUniqueServiceId();
                  System.out.println("Unique service ID: '" + ((uniqueServiceId !=
null) ? uniqueServiceId : "null") + "'");
            } else {
                  System.out.println("Ovo bas i ne radi");
            }

            QueryResponse emailAddresses =
getServiceParticipantsByUniqueServiceId(uniqueServiceId);

            System.out.println("Vracam emailAddresses");
            return emailAddresses;
      }

      private QueryResponse getServiceParticipantsByUniqueServiceId(
                  String uniqueServiceId) {
            ObjectFactory factory = new ObjectFactory();
            QueryResponse emailAddresses = factory.createQueryResponse();


            Properties properties = new Properties();
            List<String> emails = new ArrayList<String>();

            try {


properties.load(this.getClass().getResourceAsStream("ServiceParticipants.properties
"));
                  for (Object key : properties.keySet()) {
                        if ((key != null) && (key instanceof String) && (((String)
key).startsWith(uniqueServiceId + '.'))) {
                              emails.add(properties.getProperty((String) key));
                        }
                  }
            } catch (IOException ioe) {
                  ioe.printStackTrace();
                  emails = null;
            }

            emailAddresses.setEmailAddresses(emails);
            return emailAddresses;
      }

      public GetNamedQueryTypesResponse getNamedQueryTypes(
                  GetNamedQueryTypesRequest parameters)
                  throws GetNamedQueryTypesException {
            return null;
      }
}
```

7. Add properties file `ServiceParticipants.properties`:

- Properties file for JIRA Version 1 configuration:

```
Service1.prviUcesnik=reply_others@rt1test.qalab.geant.net
Service1.drugiUcesnik=reply_others@rt2test.qalab.geant.net
Service1.treciUcesnik=reply_others@rt3test.qalab.geant.net
Service1.cetvrtiUcesnik=reply_others@jira1test.qalab.geant.net

Service2.prviUcesnik=reply_others@rt2test.qalab.geant.net
Service2.drugiUcesnik=reply_others@rt3test.qalab.geant.net
Service2.treciUcesnik=reply_others@jira1test.qalab.geant.net

Service3.prviUcesnik=reply_others@rt1test.qalab.geant.net
Service3.drugiUcesnik=reply_others@rt3test.qalab.geant.net
Service3.treciUcesnik=reply_others@jira1test.qalab.geant.net

Service4.prviUcesnik=reply_others@rt1test.qalab.geant.net
Service4.drugiUcesnik=reply_others@rt2test.qalab.geant.net
Service4.treciUcesnik=reply_others@jira1test.qalab.geant.net

Service5.prviUcesnik=reply_others@rt1test.qalab.geant.net
Service5.drugiUcesnik=reply_others@rt2test.qalab.geant.net
Service5.treciUcesnik=reply_others@rt3test.qalab.geant.net

Service6.prviUcesnik=reply_others@rt2test.qalab.geant.net
Service6.drugiUcesnik=reply_others@rt3test.qalab.geant.net

Service7.prviUcesnik=reply_others@rt1test.qalab.geant.net
Service7.drugiUcesnik=reply_others@rt3test.qalab.geant.net

Service8.prviUcesnik=reply_others@rt1test.qalab.geant.net
Service8.drugiUcesnik=reply_others@rt2test.qalab.geant.net
```

- Properties file for JIRA Version 2 configuration:

```
Service1.prviUcesnik=reply_others@rt1test.qalab.geant.net
Service1.drugiUcesnik=reply_others@rt2test.qalab.geant.net
Service1.treciUcesnik=reply_others@rt3test.qalab.geant.net
Service1.cetvrtiUcesnik=reply_others@jira2test.qalab.geant.net

Service2.prviUcesnik=reply_others@rt2test.qalab.geant.net
Service2.drugiUcesnik=reply_others@rt3test.qalab.geant.net
Service2.treciUcesnik=reply_others@jira2test.qalab.geant.net

Service3.prviUcesnik=reply_others@rt1test.qalab.geant.net
Service3.drugiUcesnik=reply_others@rt3test.qalab.geant.net
Service3.treciUcesnik=reply_others@jira2test.qalab.geant.net

Service4.prviUcesnik=reply_others@rt1test.qalab.geant.net
Service4.drugiUcesnik=reply_others@rt2test.qalab.geant.net
Service4.treciUcesnik=reply_others@jira2test.qalab.geant.net

Service5.prviUcesnik=reply_others@rt1test.qalab.geant.net
Service5.drugiUcesnik=reply_others@rt2test.qalab.geant.net
Service5.treciUcesnik=reply_others@rt3test.qalab.geant.net

Service6.prviUcesnik=reply_others@rt2test.qalab.geant.net
Service6.drugiUcesnik=reply_others@rt3test.qalab.geant.net

Service7.prviUcesnik=reply_others@rt1test.qalab.geant.net
Service7.drugiUcesnik=reply_others@rt3test.qalab.geant.net

Service8.prviUcesnik=reply_others@rt1test.qalab.geant.net
```

```
Service8.drugiUcesnik=reply_others@rt2test.qalab.geant.net
```

Web service client generation:

1. Create Eclipse workspace SIM_WS_client and set UTF-8 for default text file encoding
2. Set system property `java.endorsed.dirs` to `$JBOSS_HOME/lib/endorsed` by adding `-Djava.endorsed.dirs=$JBOSS_HOME/lib/endorsed` in "`Default VM arguments`" where `JBOSS_HOME` is environment variable that points to the JBoss Application Server installation directory
3. Add server JBoss 5.1.0.GA for jdk-6
4. Create Java project SIM_CLIENT and select Dynamic Web Module, Java, JavaScript and JBoss Web Services Core as Project facets and JBoss 5.1 Runtime for project runtime
5. Generate web service client using JBossWS: File -> New -> Other -> Web Service Client -> Next

Service definition: select OSSJ-Common-v1-5.wsdl
Client type: Java Proxy
Server runtime: JBoss AS 5.1
Web service runtime: JBossWS

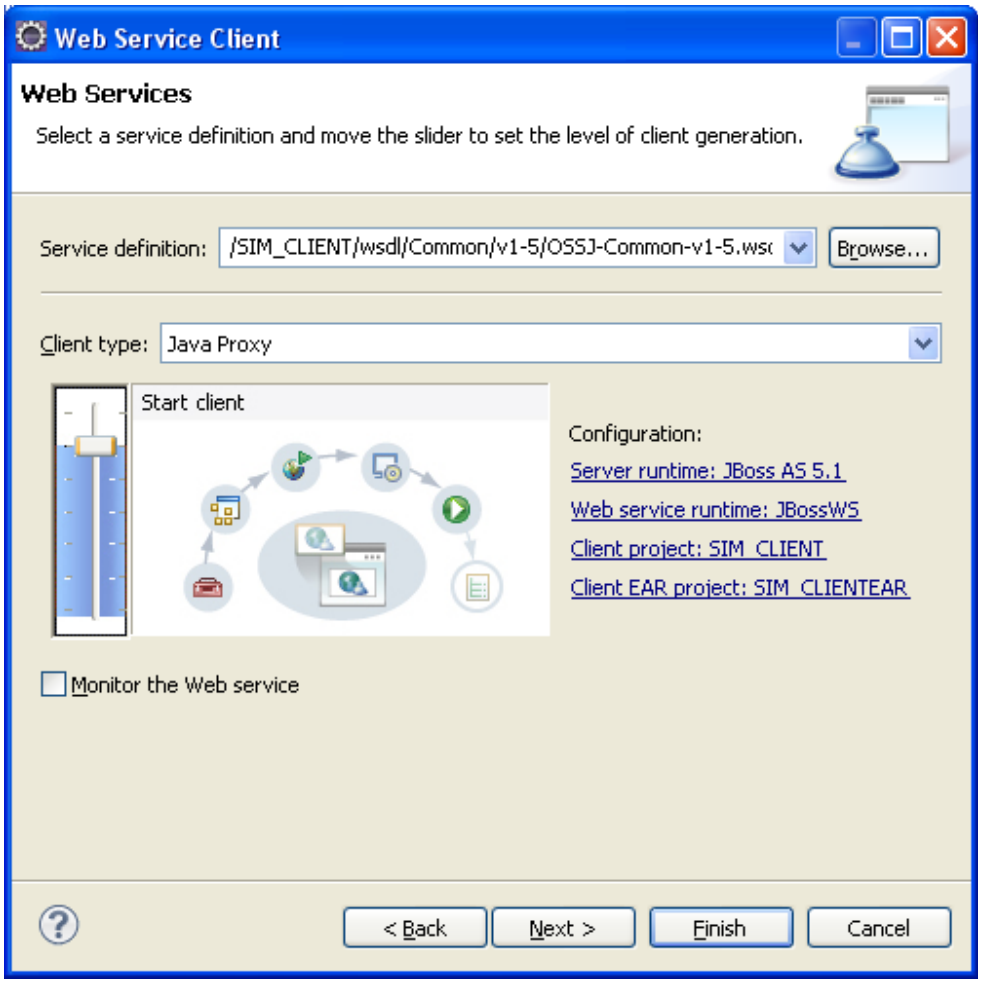


Figure 58: SIM stub WS client generation

6. Change implementation `org.ossj.wsdl.common.v1_5.clientsample.ClientSample`
   - WS client for Request tracker and JIRA version 1 configuration:

```
package org.ossj.wsdl.common.v1_5.clientsample;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
```

```
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Properties;

import javax.mail.Folder;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Store;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

import org.ossj.wsdl.common.v1_5.*;
import org.ossj.xml.common.v1_5.NamedQueryValue;
import org.ossj.xml.common.v1_5.ObjectFactory;
import org.ossj.xml.common.v1_5.QueryRequest;
import org.ossj.xml.common.v1_5.QueryResponse;

public class ClientSample {

	public static void main(String[] args) {

		Properties receivedProperties = ClientSample.readMail();
		List<String> remoteTTSes = ClientSample.querySIM(receivedProperties);
		System.out.println("SIM je vratio ucesnike:");
		for (String remoteTTS : remoteTTSes) {
			System.out.println('\t' + remoteTTS);
		}
		if (ClientSample.sendMail(receivedProperties.getProperty("reply.to"),
				receivedProperties.getProperty("subject.tag"),
remoteTTSes)) {
			System.out.println("true");
		} else {
			System.out.println("false");
		}

	}

	private static Properties readMail() {
		Properties readProperties = new Properties();
		try {
			BufferedReader br = new BufferedReader(new InputStreamReader(
					System.in));
			String line = br.readLine();
			while (line != null && !"endofcontent".equalsIgnoreCase(line)) {
			if (line.contains("Unique service id:")
						&& line.length() > line.indexOf("Unique service
id:")) {
				readProperties.put(
						"unique.service.id",
						line.substring(
								line.indexOf("Unique service
id:") + 18)
								.trim());
			}
```

```
                        if (line.startsWith("Reply-To:")) {
                              readProperties.put("reply.to",
                                          line.substring(line.indexOf("Reply-To:")
+ 9)
                                                      .trim());
                        }

                        if (line.contains("Subject tag: [")
                                    && (line.indexOf("Subject tag: [") <
line.indexOf(']',
                                                line.indexOf("Subject tag: [")))) {
                              readProperties
                                          .put("subject.tag",
                                                      line.substring(

      line.indexOf("Subject tag: [") + 13,
                                                                  line.indexOf(']',
line
                                                                              .index
Of("Subject tag: [")) + 1)
                                                                  .trim());

                        }

                        line = br.readLine();
                  }
                  br.close();

            } catch (IOException e) {
                  System.out.println("Error while reading line from console : " +
e);
                  readProperties.clear();
            }
            readProperties.put("all.properties.are.present", (readProperties
                        .keySet().size() == 3) ? "true" : "false");
            System.out.println("procitano:");
            for (Object key : readProperties.keySet()) {
                  System.out.println("\t" + key + ':'
                              + readProperties.getProperty((String) key));
            }
            return readProperties;
      }

      private static boolean sendMail(String recipient, String subject,
                  List<String> remoteTTSes) {

            // Get system properties
            Properties props = System.getProperties();

            // Setup mail server
            props.put("mail.smtp.host", "localhost");

            // Get session
            Session session = Session.getDefaultInstance(props, null);

            // Define message
            MimeMessage message = new MimeMessage(session);
            try {
                  message.addRecipient(Message.RecipientType.TO, new
```

```
InternetAddress(
                              recipient));
                  message.setFrom(new
InternetAddress("sim_client@localhost.localdomain"));
                  StringBuilder messageBody = new StringBuilder();
                  for (String remoteTTS : remoteTTSes) {
                        messageBody.append("-
").append(remoteTTS).append('\r').append('\n');
                  }
                  System.out.println(messageBody.toString());
                  message.setSubject("Test"); //subject);
                  message.setText(messageBody.toString());


                  // Send message
                  Transport.send(message);
                  System.out.println("Mejl je poslat: " + message.getSubject() + '
' + message.getSender());

                  return true;
            } catch (AddressException e) {
                  e.printStackTrace();
                  return false;
            } catch (MessagingException e) {
                  e.printStackTrace();
                  return false;
            }
      }

      private static List<String> querySIM(Properties receivedProperties) {
            try {
                  System.out.println("**********************");
                  System.out.println("Create Web Service Client...");
                  JVTSessionWebService service1 = new JVTSessionWebService();
                  System.out.println("Wsdl location: ");
                  System.out.println(service1.getWSDLDocumentLocation());
                  System.out.println("Create Web Service...");
                  JVTSessionWSPort port1 = service1.getJVTSessionWSPort();
                  System.out.println("Prepare parameters...");
                  String uniqueServiceId = receivedProperties
                              .getProperty("unique.service.id");
                  QueryRequest involvedDomainsByUniqueServiceId = (new
ObjectFactory())
                              .createQueryRequest();
                  involvedDomainsByUniqueServiceId.setNamedQuery(new
NamedQueryValue(
                              uniqueServiceId));
                  System.out.println("Call Web Service Operation...");
                  QueryResponse emails = port1
                              .query(involvedDomainsByUniqueServiceId);
                  System.out.println("**********************");
                  System.out.println("Call Over!");
                  return emails.getEmailAddresses();
            } catch (QueryException qe) {
                  System.out.println("Jovana");
                  qe.printStackTrace();
                  return null;
            }
```

```
	}

}
```

- WS client for JIRA version 2 configuration:

```
package org.ossj.wsdl.common.v1_5.clientsample;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Properties;

import javax.mail.Folder;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Store;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

import org.ossj.wsdl.common.v1_5.*;
import org.ossj.xml.common.v1_5.NamedQueryValue;
import org.ossj.xml.common.v1_5.ObjectFactory;
import org.ossj.xml.common.v1_5.QueryRequest;
import org.ossj.xml.common.v1_5.QueryResponse;

public class ClientSample {

	public static void main(String[] args) {

		Properties receivedProperties = ClientSample.readMail();
		List<String> remoteTTSes = ClientSample.querySIM(receivedProperties);
		System.out.println("SIM je vratio ucesnike:");
		for (String remoteTTS : remoteTTSes) {
			System.out.println('\t' + remoteTTS);
		}
		if (ClientSample.sendMail(receivedProperties.getProperty("reply.to"),
				receivedProperties.getProperty("subject.tag"),
remoteTTSes)) {
			System.out.println("true");
		} else {
			System.out.println("false");
		}

	}

	private static Properties readMail() {
		Properties readProperties = new Properties();
		try {
			BufferedReader br = new BufferedReader(new InputStreamReader(
					System.in));
			String line = br.readLine();
			while (line != null && !"endofcontent".equalsIgnoreCase(line)) {
				if (line.contains("Unique service id:")
						&& line.length() > line.indexOf("Unique service
```

```
id:")) {
                              readProperties.put(
                                          "unique.service.id",
                                          line.substring(
                                                      line.indexOf("Unique service
id:") + 18)
                                                      .trim());
                        }

                        if (line.startsWith("Reply-To:")) {
                              readProperties.put("reply.to",
                                          line.substring(line.indexOf("Reply-To:")
+ 9)
                                                      .trim());
                        }

                        if (line.contains("Subject tag: [")
                                    && (line.indexOf("Subject tag: [") <
line.indexOf(']',
                                                line.indexOf("Subject tag: [")))) {
                              readProperties
                                          .put("subject.tag",
                                                      line.substring(

      line.indexOf("Subject tag: [") + 13,
                                                                  line.indexOf(']',
line
                                                                              .index
Of("Subject tag: [")) + 1)
                                                                  .trim());

                        }

                        line = br.readLine();
                  }
                  br.close();

            } catch (IOException e) {
                  System.out.println("Error while reading line from console : " +
e);
                  readProperties.clear();
            }
            readProperties.put("all.properties.are.present", (readProperties
                        .keySet().size() == 3) ? "true" : "false");
            System.out.println("procitano:");
            for (Object key : readProperties.keySet()) {
                  System.out.println("\t" + key + ':'
                              + readProperties.getProperty((String) key));
            }
            return readProperties;
      }

      private static boolean sendMail(String recipient, String subject,
                  List<String> remoteTTSes) {

            // Get system properties
            Properties props = System.getProperties();

            // Setup mail server
```

```
            props.put("mail.smtp.host", "localhost");

            // Get session
            Session session = Session.getDefaultInstance(props, null);

            // Define message
            MimeMessage message = new MimeMessage(session);
            try {
                  message.addRecipient(Message.RecipientType.TO, new
InternetAddress(
                              recipient));
                  message.setFrom(new
InternetAddress("sim_client@localhost.localdomain"));

                  // SIM response as a string
                  StringBuilder simResponse = new StringBuilder("\r\n");
                  // CSV list of remote TTSes
                  StringBuilder messageBody = new StringBuilder("@summary=");
                  for (String remoteTTS : remoteTTSes) {
                        messageBody.append(remoteTTS).append(',');
                        simResponse.append(remoteTTS).append('\r').append('\n');
                  }
                  messageBody.deleteCharAt(messageBody.lastIndexOf(","));

                  messageBody.append('\r').append('\n');
                  messageBody.append('\r').append('\n');
                  messageBody.append("Service inventory management responded: ");
                  messageBody.append(simResponse.toString());

                  System.out.println(messageBody.toString());
                  message.setSubject(subject);
                  message.setText(messageBody.toString());



                  // Send message
                  Transport.send(message);
                  System.out.println("Mejl je poslat: " + message.getSubject() + '
' + message.getSender());

                  return true;
            } catch (AddressException e) {
                  e.printStackTrace();
                  return false;
            } catch (MessagingException e) {
                  e.printStackTrace();
                  return false;
            }
      }

      private static List<String> querySIM(Properties receivedProperties) {
            try {
                  System.out.println("**********************");
                  System.out.println("Create Web Service Client...");
                  JVTSessionWebService service1 = new JVTSessionWebService();
                  System.out.println("Wsdl location: ");
                  System.out.println(service1.getWSDLDocumentLocation());
                  System.out.println("Create Web Service...");
                  JVTSessionWSPort port1 = service1.getJVTSessionWSPort();
```

```
                  System.out.println("Prepare parameters...");
                  String uniqueServiceId = receivedProperties
                              .getProperty("unique.service.id");
                  QueryRequest involvedDomainsByUniqueServiceId = (new
ObjectFactory())
                              .createQueryRequest();
                  involvedDomainsByUniqueServiceId.setNamedQuery(new
NamedQueryValue(
                              uniqueServiceId));
                  System.out.println("Call Web Service Operation...");
                  QueryResponse emails = port1
                              .query(involvedDomainsByUniqueServiceId);
                  System.out.println("************************");
                  System.out.println("Call Over!");
                  return emails.getEmailAddresses();
            } catch (QueryException qe) {
                  System.out.println("Jovana");
                  qe.printStackTrace();
                  return null;
            }

      }

}
```

Biografski podaci kandidata

Jovana Vuleta je rođena 27. 7. 1976. godine Beogradu. Matematičku gimnaziju „Veljko Vlahović“ je završila 1995. godine, Elektrotehnički fakultet Univerziteta u Beogradu je upisala školske 1995./96. godine, završila ga 2003. godine sa prosečnom ocenom tokom studija 9.10 i ocenom 10 na diplomskom radu „Okruženje za modelovanje upravljanja poslovima“. Školske 2003./04. godine je upisala postdiplomske studije na Elektrotehničkom fakultetu, položila sve ispite sa prosečnom ocenom 10 i 2010. godine stekla zvanje magistra elektrotehničkih nauka za oblast Softverski sistemi odbranom magistarskog rada „Razvoj bolničkog inforamcionog sistema primenom višeslojne arhitekture“ pod mentorstvom profesora dr Zorana Jovanovića. Stipendista Fondacije za razvoj naučnog i umetničkog podmlatka Ministarstva nauke je postala početkom školske 1996./97. godine, a stipendista Fondacije za studije nauka i umetnosti Srpske akademije nauka i umetnosti nekoliko meseci nakon toga. Na Dan Univerziteta u Beogradu 2000. godine je dobila nagradu Beogradske banke kao najbolji student četvrte godine Elektrotehničkog fakulteta. Iste godine je dobila i jednokratnu stipendiju „Za generaciju koja obećava“ Kraljevske norveške ambasade.

Od januara 2004. godine do danas je zaposlena u Računarskom centru Univerziteta u Beogradu kao inženjer za informatičku podršku. Kao član ekipe za informacione sisteme, učestvovala je u izradi više projekata iz oblasti informacionih sistema i Web razvoja.

Projekti:

2004: „Razvoj informacionih servisa NIOnet naučne i obrazovne informacione mreže“ – inovacioni projekat Ministarstva za nauku i zaštitu životne sredine Republike Srbije

2004: Razvoj distribuiranog sistema za evidenciju i kontrolu prepisivanja recepata za potrebe Republičkog zavoda za zdravstveno osiguranje koji pokriva 28 filijala sa dvosmernom XML komunikacijom i preko 200 miliona podataka u centralnoj bazi

2004: OpIS – Opštinski informacioni sistem implementiran u preko 30 opština u Srbiji uz pomoć donatara kao što su USAID, UNDP i SIDA

2004 – 2013: Univerzitetska dečja klinika u Beogradu, klinički centri Kraljevo, Valjevo i Vranje, Institut za zdravstvenu zaštitu dece i omladine Vojvodine u Novom Sadu: bolnički informacioni sistem HELIANT – sistem sa integrisanim podsistemima za zakazivanje, prozivanje i praćenje pregleda, elektronskim kartonima i elektronskim fakturama.

2006: Softver za komunikaciju sa “koridor” (MTC) uređajima putem GSM (GPRS) mreže za potrebe MTC doo, Bela zemlja bb, Užice

2008 – 2013: Dom zdravlja Stari Grad, Dom zdravlja Vračar, Dom zdravlja Voždovac, Dom zdravlja Zvezdara, Dom zdravlja Rakovica, Dom zdravlja Valjevo, Dom zdravlja Novi Sad, Dom zdravlja Niš, ... (preko 30 ustanova primarnog nivoa zdravstvene zaštite): informacioni sistem doma zdravlja HELIANT – sistem sa integrisanim podsistemima za zakazivanje, prozivanje i praćenje pregleda, elektronskim kartonima i elektronskim fakturama

2012 – 2013: FP7 GEANT3 projekat

2013 – 2015: FP7 GEANT3plus projekat

2015 – : GEANT2020 projekat

2015 – : GEANT2020 projekat

2015 – : CaSA (Building capacity of Serbian Agricultural Education to link with Society)

(Tempus programme – Structural Measures-Higher Education and Society subprogramme)

2015 – : RODOS (Restructuring of Doctoral Studies in Serbia)

(TEMPUS programme (Structural Measures-Governance Reform subprogramme)

Koautor je više radova u domaćim i stranim časopisima i konferencijama.

Прилог 1.

I. Изјава о ауторству

Потписани-а Јована Вулета-Радоичић

број уписа _____

Изјављујем

да је докторска дисертација под насловом

„Управљање мрежним услугама и ресурсима у федеративним мрежним окружењима“

_____

_____

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

Потпис докторанда

У Београду, 07.04.2016.

Јована Вулета-Радоичић

Прилог 2.

II. Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Јована Вулета-Радоичић

Број уписа _____

Студијски програм _____

Наслов рада „Управљање мрежним услугама и ресурсима у федеративним мрежним окружењима“

Ментор др Зоран Јовановић

Потписани Јована Вулета-Радоичић

изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла за објављивање на порталу **Дигиталног репозиторијума Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанда

У Београду, 07.04.2016.

Јована Вулета-Радоичић

Прилог 3.

III. Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

„УУправљање мрежним услугама и ресурсима у федеративним мрежним окружењима “

_____

_____

која је моје ауторско дело.

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

Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

(1.) Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

5. Ауторство – без прераде

6. Ауторство – делити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда

У Београду, _07.04.2016._____

Јована Вулетић-Радоичић

1. Ауторство - Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. Ауторство – некомерцијално. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. Ауторство - некомерцијално – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. Ауторство - некомерцијално – делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. Ауторство – без прераде. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. Ауторство - делити под истим условима. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.