Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Damjan Kovač

Obvladovanje zaupanja v storitveno usmerjenih arhitekturah

DOKTORSKA DISERTACIJA

Ljubljana, 2009

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

mag. Damjan Kovač

Obvladovanje zaupanja v storitveno usmerjenih arhitekturah

DOKTORSKA DISERTACIJA

Mentor: prof. dr. Denis Trček

Ljubljana, 2009

Napisano v sistemu LaTeX.

Povzetek

Zaupanje predstavlja pomemben element človekovega obnašanja in je dostikrat glavni pogoj za medsebojne interakcije in sodelovanje v družbi. Tudi sprejemanje odločitev temelji na zaupanju; to je osnovano na ocenah in izkušnjah, prek katerih verjamemo in smo prepričani, da zaupnik lahko izpolni nalogo, ki mu je zaupana. Rečemo lahko, da je zaupanje širši družbeni in hkrati tudi subjektivni fenomen. Največkrat se izoblikuje na podlagi osebne izkušnje oz. osebnega kontakta. V primeru, da osebni kontakt ni mogoč, je zaupanje lahko osnovano na podlagi mnenj in priporočil drugih – v tem primeru govorimo o ugledu.

Tudi v porazdeljenih računalniških okoljih je zaupanje velikokrat pogoj za interakcije oz. transakcije med entitetami. Tu mislimo predvsem na okolja in storitve e-poslovanja, kjer so entitete prostorsko ločene in so osebni kontakti praktično nemogoči. Informacijska varnost s svojimi storitvami v teh okoljih še ne zagotavlja zaupanja; je lahko potreben pogoj, ni pa zadosten. Zaupanje ni varnostni mehanizem, ampak subjektiven pojem, ki vključuje socialne in psihološke vidike. V literaturi se često označuje kot *mehki* varnostni mehanizem, ki v nasprotju z tradicionalnimi varnostnimi mehanizmi (t. i. overjanje, kontrola dostopa), omogoča varovanje pred zlonamernimi ponudniki virov in storitev. To pomeni predvsem povečanje kakovosti storitev ter zanesljivosti ponudnikov oz. drugih entitet znotraj omrežja. Posledično je obvladovanje zaupanja in ugleda eno od temeljnih področij za podporo uporabnikov pri uporabi porazdeljenih storitev e-poslovanja in njihovo širšo uporabo.

Pričujoča disertacija podaja pregled obstoječega stanja mehanizmov zaupanja; večina avtorjev jih obravnava kot običajne varnostne mehanizme, kjer manjka socialna in interaktivna komponenta. To pomanjkljivost odpravimo s formalizacijo zaupanja kot mehkega varnostnega mehanizma na podlagi upoštevanja različnih psiholoških in sociioloških dejavnikov. Tako definiramo abstraktni model zaupanja $ATM = (\mathbf{G}, \Omega, \mathcal{T}, \mathcal{F}, \mathcal{O}, \Gamma, \Pi)$, ki vsebuje gradnike, kamor lahko vključimo različne računske modele za določanje stopnje zaupanja. Ta stopnja predstavlja osnovo za odločanje o prihodnjih interakcijah med entitetami, posredno pa omogoča tudi identifikacijo entitet z zlonamernim obnašanjem.

Preko gradnikov modela ATM predstavimo kvantitativni pristop Jøsangove subjektivne logike, na novo pa nadgradimo kvalitativni model s pripadajočo kvalitativno algebro, ki modelira obnašanje agentov. Tu velja omeniti pomembno težavo, da v splošnem zaupanje ni tranzitivno. Z vpeljavo zaupanja v priporočilo, uporabo določenih algoritmov teorije grafov (tranzitivne ovojnice, iskanje v globino) in algebrajskimi strukturami (polkolobarji, distributivne mreže),

predstavimo način za izračun stopnje zaupanja med entitetami, ki niso bile v neposredni interakciji.

Zahteve o stopnji zaupanja sodelujočih entitet (uporabniki, viri, storitve), so določene z interakcijskimi politikami zaupanja, ki temeljijo na zgodovini interakcij. Omenjene politike formalno predstavimo v jeziku temporalne logike PPLTL ter definiramo algoritme za preverjanje skladnosti.

Storitveno usmerjene arhitekture (SOA) predstavljajo nov koncept za gradnjo porazdeljenih rešitev e-poslovanja na osnovi povezovanja (kompozicije) spletnih storitev kot samostojnih funkcionalnih enot. V delu predstavimo princip orkestracije in koreografije storitev. Večji poudarek je na orkestraciji z jezikom BPEL, ker je sprejet in podprt industrijski standard. Jezik BPEL preslikamo v formalni jezik Petrijevih mrež ter razvijemo algoritme za izračun stopnje zaupanja kompozita storitev glede na model ATM. Tako vključimo mehanizme zaupanja ne samo za atomarne storitve, ampak tudi za njihove kompozite (t. i. poslovne procese).

Obstaja množica OASIS in W3C standardov WS-*, ki pokrivajo predvsem interoperabilnost in varnost spletnih storitev, vendar noben od njih ne upošteva zaupanja kot mehkega varnostnega mehanizma. Posledično je prispevek naloge tudi razširitev odprtih standardov (WS-Trust, WS-Security), preko katerih v infrastrukturo SOA integriramo kvantitativni in kvalitativni model zaupanja kot standardne gradnike. Preko njih definiramo arhitekturno rešitev sistema za obvladovanje zaupanja in ugleda v infrastrukturo SOA. Kot prototip implementiramo sistem *trustGuard* v okoljih WCF (Windows Communication Foundation) in JEE (Java Enterprise Edition), ki ga je mogoče uporabiti tudi v realnem okolju.

Ključne besede: zaupanje, ugled, varnost, obvladovanje zaupanja, e-poslovanje, poslovni procesi, storitveno usmerjene arhitekture, spletne storitve, standardizacija, SOA, BPEL, WS-Trust

Abstract

Trust represents a significant aspect of human behavior and is often a precondition for mutual interactions and cooperation in a society. Decision making is trust-based and the trust is founded on experiences and assessments indicating beliefs about the relying party to perform a given task. Ultimately, trust may be considered a social and subjective phenomenon based on various factors, usually on personal experience or direct interactions. In the absence of personal experience, however, trust often has to be based on referrals from others that can be indicated as a reputation.

In addition, trust is usually a precondition for interactions or transactions among parties in distributed computer environments. These are mostly considered to be environments and services of e-business where parties are generally separated in space thus making personal experience practically impossible. Information security does not assure trust in such environments; mostly, it is a required precondition although insufficient. Trust is not a security mechanism but a subjective concept that includes both social and psychological aspects. Various authors have denoted trust as a soft security mechanism that may, unlike traditional security mechanisms (e.g., authentication, access control), provide protection against the information and service providers who act deceitfully. This, in turn, improves the quality of services and the reliability of providers and other entities in the network. Therefore, trust and reputation management is an essential area when supporting consumers who apply e-business services.

This dissertation examines the existing trust mechanisms in computing environments; most authors denote them as standard security mechanisms where the social and interactive components are lacking. This weakness is overcome by formalizing trust as a soft security mechanism and by taking numerous psychological and social factors into consideration. We define an abstract trust model $ATM = (\mathbf{G}, \Omega, \mathcal{T}, \mathcal{F}, \mathcal{O}, \Gamma, \Pi)$ with the appropriate components that allow different computational models to determine the degree of trust or trustworthiness. The trust degree represents a basis to make decisions about the future interactions among entities and also helps to identify malicious behavior.

Next, the components of the trust model ATM are formally defined in order to present the quantitative approach of Jøsang's subjective logic and extend the qualitative approach with the appropriate qualitative algebra that models the behavior of agents. It must be stressed, however, that trust is not transitive in general. Assuming recommendation trust, different algorithms of the graph theory (e.g. transitive closures, depth first search) and the appropriate algebraic

structures (e.g. semirings, distributive lattices), the method of how to compute the trust degree among the entities that have not interacted directly is presented.

The trust requirements of the interacting entities (e.g. users, resources, services) are represented through the interaction trust polices that are based on the interaction history. Those policies are formally defined by the pure past temporal logic language PPLTL with suitable algorithms for compliance checking.

Service-oriented architectures (SOAs) represent a new concept of building distributed e-business software solutions based on the composition of web services as autonomous functional units. The principle of the orchestration and choreography of web services is presented. The importance of the orchestration with BPEL language is stressed as it is a widely accepted and supported industrial standard. The BPEL language is mapped into the formal language of Petri Nets. The appropriate algorithms for trust degree computation of service composites are defined according to the trust model ATM. In this way, trust mechanisms are added not only to atomic, but also to composite services (i.e. BPEL business processes).

The number of WS-* standards from OASIS and W3C organizations exist that mostly cover the interoperability and security of web services but none of them consider trust as a soft security mechanism. Consequently, the contribution of this dissertation is also an extension of open standards (WS-Trust, WS-Security) through which we incorporate a qualitative and quantitative trust model in the SOA infrastructure as standard building blocks. With them the architecture of trust and the reputation management system using SOA infrastructure is designed. A prototype called *trustGuard* using WCF (Windows Communication Foundation) and JEE (Java Enterprise Edition) technologies is also implemented.

Key words: trust, reputation, security, trust management, e-business, business processes, service-oriented architectures, web services, standardization, SOA, BPEL, WS-Trust

Zahvala

Iskreno se zahvaljujem svojemu mentorju, prof. dr. Denisu Trčku, za vso pomoč in strokovno vodstvo pri izdelavi doktorske disertacije. Zelo sem hvaležen za vse strokovne nasvete, konstruktivne ideje, komentarje, popravke, njegov čas in razpoložljivost. Hvala tudi prof. dr. Francu Solini, prof. dr. Marjanu Krisperju in doc. dr. Branku Kavšku za njihov čas in trud pri preliminarnem pregledu disertacije, njihove opombe in komentarje.

Zahvaljujem se kolegoma – sodelavcema Laboratorija za e-medije asistentu Iztoku Starcu in Davidu Jelencu. Poleg prijetnih konstruktivnih debat sta mi s številnimi koristnimi nasveti in idejami dala dodaten motiv in pomoč pri delu.

Največjo podporo pri nastajanju disertacije pa so mi izkazovali moji domači. Iskrena hvala moji mami, očetu in bratu za vso skrb, spodbude, moralno podporo in motivacijo, da sem vztrajal do konca študija. Hvala za vse, ampak prav vse!

Lepo se zahvaljujem prijateljici Poloni Godina za njeno pripravljenost, čas in trud pri jezikovnem lektoriranju člankov.

Za podporo se zahvaljujem tudi vsem ostalim prijateljem, ki so mi ob času pisanja doktorata stali ob strani in me spodbujali ob trenutkih krize. Hvala vsem, ki ste verjeli vame!

Not everything that counts can be counted,
and not everything that can be counted counts.

A. Einstein

◊ ◊ ◊

Kazalo

Kazalo **xi**

1 Uvod **1**

1.1 Motivacija in cilji 1

1.2 Pregled vsebine 2

2 Pregled področja **3**

2.1 Zaupanje, ugled 3

2.2 Zaupanje in varnost 3

2.3 Definicije zaupanja in ugleda 4

2.4 Obvladovanje zaupanja in ugleda 6

2.4.1 Razlika med sistemi OZ in OU 7

2.4.2 Arhitektura sistemov OZ in OU 7

2.5 Tradicionalni sistemi obvladovanja zaupanja 8

2.5.1 X.509 9

2.5.2 PGP 9

2.5.3 PolicyMaker 10

2.5.4 KeyNote 11

2.5.5 W3C PICS 12

2.5.6 Referee 13

2.5.7 IBM Trust Establishment Framework 14

2.6 Modeli zaupanja 15

2.6.1 Kvantitativni modeli 15

2.6.1.1 Modeli z enostavnimi vsotami in povprečji 15

2.6.1.2 Verjetnostni modeli 16

2.6.1.3 Modeli prepričanja 18

2.6.1.4 Pretočni modeli 20

2.6.1.5 Modeli teorije iger 21

2.6.2 Kvalitativni modeli 24

3 Modeliranje zaupanja in ugleda **27**

3.1 Zasnova modela . . . 27
3.1.1 Dejavniki zaupanja . . . 27
3.1.2 Formalizacija . . . 34
3.1.3 Tranzitivnost . . . 42
3.1.3.1 Pravila za izračun . . . 44
3.1.3.2 Distributivne mreže . . . 47
3.1.3.3 Iskanje tranzitivnih verig . . . 48
3.1.4 Kvalitativni model . . . 52
3.1.5 Kvantitativni model . . . 58
3.1.6 Interakcijske politike . . . 61
3.1.6.1 Sintaksa PPLTL . . . 61
3.1.6.2 Semantika . . . 62
3.1.6.3 Verifikacija . . . 64

4 Storitveno usmerjene arhitekture 69
4.1 Splošno o SOA . . . 69
4.2 Storitve . . . 70
4.2.1 Kompozicija storitev . . . 72
4.2.2 BPEL . . . 73
4.2.3 Formalna predstavitev . . . 74
4.2.4 Petrijeve mreže . . . 75
4.2.5 Preslikava BPEL v PN . . . 78
4.2.6 Analiza PN . . . 80
4.2.7 Stopnja zaupanja kompozita storitev . . . 82
4.2.7.1 Izračun stopnje zaupanja kompozita . . . 85
4.2.7.2 (a) Usmerjeni acikličen graf . . . 88
4.2.7.3 (b) Usmerjeni cikličen graf . . . 92

5 Sistem za obvladovanje zaupanje in ugleda 99
5.1 Splošno . . . 99
5.2 Infrastruktura SOA . . . 100
5.2.1 Spletna storitev zaupanja . . . 100
5.2.2 Varnostni mehanizmi spletnih storitev . . . 101
5.2.2.1 Zaupnost . . . 103
5.2.2.2 Celovitost (integriteta) . . . 105
5.2.2.3 Verodostojnost . . . 107
5.2.2.4 Varna seja sporočil SOAP . . . 110
5.2.3 WS-Trust . . . 111
5.2.3.1 Operacije WS-Trust . . . 113

5.2.4 Žetoni zaupanja . . . 115
5.2.4.1 TrustRequest . . . 116
5.2.4.2 TrustToken . . . 118
5.2.4.3 InsertRequest . . . 119
5.2.4.4 InsertToken . . . 121
5.2.5 Arhitektura . . . 122
5.2.5.1 Implementacija . . . 123
5.2.5.2 Scenarij interakcij . . . 125
5.2.5.3 Interakcijske politike . . . 126
5.2.5.4 Mehanizem za preverjanje skladnosti . . . 132
5.2.5.5 Protokol interakcij . . . 136

6 Zaključek **139**
6.1 Nadaljnje delo . . . 140
6.2 Prispevki k znanosti . . . 142

A Simulacije **143**
A.1 Parametri . . . 143
A.1.1 Generator naključnih števil . . . 143
A.1.2 Agenti . . . 143
A.1.3 Simulacija . . . 144
A.1.3.1 Interakcijska zgodovina . . . 145
A.1.3.2 Vektorji zaupanja . . . 146
A.1.4 Časovni obseg interakcij . . . 147
A.2 Primeri simulacij . . . 147
A.2.1 Simulacija: interakcijska zgodovina . . . 147
A.2.1.1 Primer 1: število interakcij . . . 148
A.2.1.2 Primer 2: deleži operatorjev . . . 151
A.2.2 Simulacija: vektorji zaupanja . . . 151
A.3 Simulacije *Monte Carlo* . . . 155
A.3.1 Primeri simulacij . . . 156
A.3.1.1 Primer 1: $\overline{s}(\mathrm{D})$, $\overline{s}(\mathrm{PD})$ . . . 158
A.3.1.2 Primer 2: $\overline{s}(\mathrm{U})$ . . . 158
A.3.1.3 Primer 3: $\overline{s}(\mathrm{PT})$, $\overline{s}(\mathrm{T})$ . . . 159

B Pregled BPEL **163**
B.1 Struktura . . . 163
B.1.1 Osnovne aktivnosti . . . 164
B.1.1.1 Invoke . . . 165
B.1.1.2 Receive . . . 165

B.1.1.3 Reply . . . 165
B.1.1.4 Throw . . . 165
B.1.1.5 Compensate . . . 166
B.1.2 Strukturirane aktivnosti . . . 166
B.1.2.1 Sequence . . . 166
B.1.2.2 Flow . . . 166
B.1.2.3 Links . . . 166
B.1.2.4 Switch . . . 167
B.1.2.5 Pick . . . 167
B.1.2.6 Scope . . . 168
B.1.2.7 CompensationHandler . . . 169
B.1.2.8 FaultHandler . . . 169
B.1.2.9 EventHandler . . . 169

C Pregled WS-CDL 171
C.1 Struktura . . . 171
C.1.1 Aktivnosti za kontrolo toka (control-flow) . . . 172
C.1.2 Aktivnost delovne enote (workunit) . . . 172
C.1.3 Osnovne aktivnosti . . . 172

D Primer preslikave procesa BPEL v PN 175
D.1 Proces BPEL . . . 175

E Prototip sistema *trustGuard* 179

Kratice, okrajšave, simboli 181

Slike 183

Tabele 185

Algoritmi 186

Primeri 187

Literatura 189

Poglavje 1

Uvod

1.1 Motivacija in cilji

Zaupanje (angl. trust) pomeni enega od osnovnih dejavnikov človekovega obnašanja in je pogoj za mnoge medsebojne interakcije in sodelovanje v družbi. Vsakodnevno sprejemamo odločitve, ki temeljijo na zaupanju (npr. izbira zdravnika, izbira mehanika za popravilo avtomobila, izbira dobavitelja, ...); te odločitve temeljijo na ocenah in izkušnjah, preko katerih verjamemo in smo prepričani, da izbrani subjekt (zaupnik) lahko izpolni nalogo, ki mu je zaupana. Zaupanje je obči družbeni in hkrati tudi subjektiven fenomen, ki se oblikuje največkrat na podlagi osebne izkušnje (osebnega kontakta), pa tudi preko priporočil in mnenj drugih. Odvisno je od mnogih dejavnikov in ga je težko definirati na formalen način; velja omeniti delo S.P. Marsha [1], kjer je med prvimi podal matematično formalizacijo zaupanja z različnih vidikov drugih znanosti (sociologija, psihologija, filozofija).

Dandanes večino poslovanja v sodobni družbi prehaja na t. i. *elektronski* način, ki temelji na informacijskih tehnologijah (IT) oz. omrežju internet. Govorimo o *e-poslovanju* oz. *digitalni* ekonomiji. S hitrim razvojem informacijsko-komunikacijskih tehnologij (IKT) se pojavljajo novi porazdeljeni sistemi in okolja za aplikacije e-poslovanja: omrežja vsak z vsakim (angl. peer-to-peer P2P), mobilna in ad hoc omrežja. Moderni komunikacijski mediji izpodrivajo tradicionalne načine interakcij in poslovanja, ki so sama po sebi omogočala neko stopnjo zaupanja med subjekti oz. poslovnimi partnerji. Veliko tradicionalnih prijemov za določanje stopnje zaupanja v fizičnem svetu ni uporabnih preko porazdeljenih komunikacijskih okolij in medijev. Poslovanje se odvija s subjekti in organizacijami, o katerih dostikrat ne vemo ničesar. V ospredju so interakcije med različnimi vpletenimi subjekti – entitetami (uporabniki, agenti, storitve, računalniki), ki medsebojno sodelujejo, izmenjujejo sporočila in izvajajo transakcije. Odločitve za medsebojno sodelovanje temeljijo na tveganju. Posledica tega dejstva je, da postaja zaupanje poleg varnosti temeljni pojem za učinkovito in zanesljivo izvajanje ter odločanje o (poslovnih) transakcijah v porazdeljenih sistemih e-poslovanja. Posledično postaja obvladovanje zaupanja (angl. trust management) eden temeljnih problemov za podporo uporabnikov pri uporabi porazdeljenih storitev in njihovo širšo uporabo, česar se zaveda tudi EU – rezultat

tega so dodatna sredstva za raziskave omenjenih področij v njenih razvojnih programih [2]. E-poslovanje je močno odvisno od uporabljene informacijske tehnologije; trenutno je v ospredju nov koncept za gradnjo porazdeljenih rešitev – storitveno usmerjena arhitektura (angl. service oriented architecture – SOA) [3, 4], ki je poglavitna tehnologija za današnje in jutrišnje rešitve e-poslovanja. Posledično je zaupanje nujen gradnik SOA.

Cilj tega dela je formalizirati zaupanje in ob podpori obstoječih varnostnih mehanizmov nuditi učinkovito orodje za odločanje o interakcijah (transakcijah) z drugimi entitetami v porazdeljenih računalniških okoljih. Posledično skuša integrirati formalne mehanizme zaupanja v infrastrukturo SOA kot standarden gradnik za sodobne rešitve e-poslovanja.

1.2 Pregled vsebine

Delo obsega 6 poglavij. V drugem poglavju je predstavljeno področje zaupanje in ugleda z ustreznimi definicijami pojmov različnih avtorjev. Opisani so nekateri obstoječi tradicionalni sistemi za obvladovanje zaupanja in ugleda, ki so pravzaprav varnostni mehanizmi. Predstavljeni so tudi nekateri izmed obstoječih kvalitativnih in kvantitativnih modelov za izračun stopnje zaupanja in ugleda.

Tretje poglavje opisuje množico dejavnikov zaupanja iz psihološko-socialne perspektive ter jih poveže v formalni model zaupanja. Predstavljen je princip tranzitivnosti ter uporaba za izračun stopnje zaupanja v okviru kvalitativnega in kvantitativnega modela zaupanja. Vsak model ima definirane ustrezne algoritme. Predstavljen je tudi formalni model za interakcijske politike zaupanja med entitetami.

Četrto poglavje obravnava storitveno usmerjeno arhitekturo z vidika kompozicije spletnih storitev na osnovi orkestracije. Predstavlja formalni opis orkestracije s Petrijevimi mrežami in definira algoritme za izračun stopnje zaupanja kompozitnih storitev (t. i. poslovnih procesov BPEL).

Peto poglavje predstavlja infrastrukturo storitveno usmerjenih arhitektur. Opisuje značilnosti varnosti spletnih storitev na osnovi standardov in specifikacij WS-Security in WS-Trust. Definira razširitev ogrodja WS-Trust za integracijo kvalitativnega in kvantitativnega modela zaupanja. Poglavje obravnava tudi preslikavo formalnega jezika interakcijskih politik zaupanja v specifikacijo WS-Policy. Poleg tega je podan arhitekturni pristop za implementacijo sistema za obvladovanje zaupanja in ugleda.

Šesto poglavje podaja sklepne ugotovitve in sklepne usmeritve za nadaljnje raziskave. Navaja tudi izvirne prispevke znanosti tega dela. V dodatku so predstavljeni rezultati simulacij na kvalitativnem modelu zaupanja. Opisani so tudi osnovni gradniki jezika za orkestracijo (BPEL) in koreografijo (WS-CDL).

Poglavje 2

Pregled področja

2.1 Zaupanje, ugled

Zaupanje je vitalna komponenta vsake transakcije pri e-poslovanju. Entiteta A mora namreč zaupati entiteti B, če hoče z njo stopiti v interakcijo, da izpolni določeno nalogo – kupec npr. mora zaupati prodajalcu pri elektronskem nakupu določenega izdelka ali storitve in obratno – prodajalec mora zaupati kupcu, da bo izdelek oz. storitev plačana. Splošno velja, da bolj ko so udeležene strani v transakciji časovno in prostorsko ločene, večje je tveganje. V tem primeru se transakcija ne izvrši vse dokler stran, ki začne transakcijo, ne poseduje določeno stopnjo zaupanja o strani, ki vstopi v transakcijo kot druga, da bo resnično izvedla svoje obveze.

Zaupanje je močno povezano s pojmom ugleda (angl. reputation), ki se navezuje na zanesljivost in posredno določa stopnjo zaupanja na osnovi priporočil in ocen članov skupnosti socialne mreže (socialna struktura vozlišč). To pa ima konkretne ekonomske učinke – ponudniki e-storitev s pomočjo sistemov za upravljanje ugleda želijo povečati uporabo tovrstnih storitev. Tipičen primer je spletno mesto eBay za elektronsko licitiranje in trgovanje (angl. bidding and e-commerce), ki ima očitno precej učinkovit model za obvladovanje ugleda (angl. reputation management system).

2.2 Zaupanje in varnost

Zaupanje je v tesni povezavi s pojmom varnosti (angl. security). Področje informacijske varnosti [5] s svojimi storitvami (overjanje, zaupnost, celovitost, ne-zanikanje, nadzor dostopa in beleženje z alarmiranjem) samo po sebi še ne zagotavlja zaupanja; varnost je često potreben pogoj, ni pa zadosten. Veliko avtorjev razume pojem zaupanja kot varnostni mehanizem [6, 7], ki pokriva različne tehnike overjanja in avtorizacije (npr. mehanizmi kontrole dostopa).

Varnost je v nekaterih standardih definirana kot mehanizem za ohranjanje *zaupnosti*, *celovitosti* in *razpoložljivosti* informacij (angl. Confidelity, Integrity, Availability – CIA). Glavna naloga informacijske varnosti je zagotavljati lastnosti CIA za informacijske vire znotraj določene domene. CIA se zagotavlja z uporabo različnih kriptografskih protokolov in mehanizmov [8]. To

se lahko označi kot tradicionalni varnostni mehanizmi, katerih namen je varovanje informacij in informacijskih virov od zlonamernih uporabnikov preko omejevanja dostopa samo avtoriziranim uporabnikom. Lastnik informacijskih virov avtorizira neko množico uporabnikov za izvajanje določenih akcij preko varnostne politike. Velikokrat pa se pojavi tudi obratna situacija, kjer se morajo uporabniki varovati pred ponudniki virov, informacij in e-storitev, ker lahko le-ti delujejo zlonamerno in ponujajo napačne ali zavajajoče informacije. Omenjeni tradicionalni varnostni mehanizmi pri taki vrsti groženj odpovejo, medtem ko sistemi za upravljanje zaupanja in ugleda lahko varujejo uporabnike pred takimi grožnjami. Tukaj je bistvena razlika med varnostnimi mehanizmi in sistemi za upravljanje zaupanja in ugleda. Razlika med tema dvema pristopoma k pojmu varnosti se često označi s pojmi *"trdna" varnost* (angl. hard security) za tradicionalne mehanizme, kot npr. overjanje in kontrola dostopa ter *"mehka" varnost* (angl. soft security) za socialne mehanizme, npr. sisteme za obvladovanje zaupanja in ugleda [9, 10, 11].

Poglavitni razlog, ki ločuje zaupanje od varnosti, je v tem, da je zaupanje močno subjektiven pojem in vključuje tudi socialne in psihološke vidike [11, 12, 13, 14]; v osnovi je torej osebni, subjektivni fenomen. Trdna varnost je lahko pogoj za zaupanje, ni pa nujna. Tradicionalni varnostni mehanizmi predstavljajo način za širjenje zaupanja od točke izvora (začetna predpostavka) do točke ponora (končni sklep).

2.3 Definicije zaupanja in ugleda

V literaturi obstaja veliko različnih definicij zaupanja. ECJRC (European Commision Joint Research Centre) [15] definira zaupanje pri e-poslovanju kot "*lastnost poslovnega razmerja, kjer obstaja zanesljivost med poslovnimi partnerji in med njimi ustvarjenimi transakcijami*". Izpostavlja predvsem lastnost infrastrukture – zanesljivost, razpoložljivost, celovitost, preprečevanje nepooblaščene uporabe ter zagotovljeno raven kakovosti storitev.

Slovar Oxford Reference Dictionary [15] navaja zaupanje kot "*trdno prepričanje v zanesljivost, iskrenost, sposobnost osebe ali stvari*". Zaupanja vredna entiteta mora torej imeti visoko zanesljivost in ne sme odpovedati v času interakcije, izvesti mora storitev v razumnem časovnem okviru, delovati mora iskreno in pošteno. Iz teh definicij sledi, da sestavlja zaupanje več različnih lastnosti (zanesljivost, pravočasnost, poštenost, resničnost, varnost, sposobnost), ki jih je potrebno obravnavati glede na okolje.

Grandison in Sloman [16] podajata definicijo zaupanja kot "*trdno prepričanje v sposobnost entitete, da deluje zanesljivo, varno in verodostojno znotraj določenega konteksta*". Še bolj specifična izpeljava za področje e-poslovanja je podana v [15] – "*zaupanje entitete A v entiteto B za storitev X je merljivo prepričanje, ki ga ima A v B za zanesljivo delovanje znotraj določenega časovnega okvira v določenem kontekstu v relaciji do storitve X*".

Zaupanje je eden od pomembnih dejavnikov pri odločanju; nekateri avtorji upoštevajo tudi pojem tveganja [17, 9]. Iz naštetih definicij sledi, da je zaupanje kompleksen pojem in se nanaša na prepričanje v poštenost, verodostojnost, sposobnost, zanesljivost opazovane entitete.

Zaupanje se formalno lahko predstavi kot razmerje (usmerjena relacija) med dvema entitetama – izvorno in ciljno [18]; izvorna entiteta se lahko označi kot *upnik* (ang. trustor), ciljna pa kot *zaupnik* (angl. trustee). Izvorna entiteta poseduje neko stopnjo zaupanja (vrednost zaupanja; angl. trustworthiness) v ciljno entiteto, da lahko izvede ustrezno nalogo/storitev v določenem kontekstu. Predpostavka je, da je izvorna entiteta "razumna" (angl. cognitive) oz. ima kognitivne sposobnosti – ima zmožnost ocenjevanja in odločanja o ciljni entiteti na podlagi prejetih informacij in preteklih izkušenj; lahko jo označimo tudi s pojmom agent(1). Ciljna entiteta je lahko abstrakten pojem – oseba, organizacija, računalnik, storitev, kriptografski ključ, itd. Kontekst zaupanja pomeni specifičen namen ali področje, kot npr. verodostojnost v primeru zaupanja v kriptografski ključ. O vzajemnem zaupanju (angl. mutual trust) govorimo v primeru, ko obe entiteti (izvorna in ciljna) zaupata ena drugi znotraj istega konteksta; to je seveda možno samo v primeru, ko imata obe entiteti kognitivne sposobnosti. Literatura uporablja pojem zaupanja za različne pomene; npr. Jøsang [11] omenja *zaupanje v zanesljivost* (angl. reliability trust) in *odločitveno zaupanje* (angl. decision trust). Prvi pomen obravnava zaupanje kot zaznano zanesljivost (verjetnost) izvorne entitete do ciljne, drugi pa kot odločitev izvorne entitete za začetek določene situacije kljub možnosti negativnih posledic. Marsh [1] loči tri vidike: osnovno, splošno in situacijsko zaupanje; poudarek je na situacijskem zaupanju – situacija se tukaj pojmuje kot kontekst. Grandison in Sloman [16] pa klasificirata zaupanje celo v šest različnih razredov.

Castelfranchi in Falcone [19] navajata pojem zaupanja kot socialno-kognitivni model, ki vsebuje kompleksno strukturo prepričanj in ciljev, vrednotenj in pričakovanj, ki se nanašajo na ciljno entiteto (njene zmožnosti, samozavest, pripravljenost, vztrajnost, moralnost, cilji). Celotna struktura določa stopnjo zaupanja, oceno tveganja in posledično odločitev ali zaupati ciljni entiteti ali ne; ta odločitev temelji na subjektivnem pragu za (ne)sprejem povezanega tveganja. Prepričanja znotraj omenjene strukture niso pogojena samo z osebno izkušnjo in interakcijami, ampak tudi z drugimi viri: opazovanje, sklepanje, socialni stereotipi, komunikacija, širjenje ugleda, signali, itd. Zaupanje je subjektivna ocena (lahko podana kot verjetnost) in kompleksen mentalni proces oz. stanje (t. i. "state of mind") izvorne entitete do ciljne entitete za določeno akcijo in cilj (namen). Proces temelji na notranjih (prepričanja) in zunanjih faktorjih (okolje, situacija) [19].

Ugled (angl. reputation) je tesno povezan s stopnjo zaupanja v ciljno entiteto. Slovar Concise Oxford Dictionary definira ugled kot "*tisto, kar se običajno govori ali verjame o značilnosti osebe ali stvari*". Ta splošna definicija se sklada s trditvijo, da je ugled pridobljena lastnost socialne mreže in je javno vidna vsem njenim članom [9]. Socialna mreža (skupnost) se označuje kot socialna struktura vozlišč, ki so običajno posamezniki ali organizacije. Med vozlišči obstajajo povezave, ki predstavljajo različne tipe relacij. Eden izmed tipičnih primerov socialnih mrež so omrežja P2P [20]. Povezavo med zaupanjem in ugledom je težko točno definirati, predvsem

(1) V nadaljevanju (kognitivne) entitete označujemo tudi z agenti.

zato, ker temelji zaupanje na osebnem in subjektivnem odnosu do ciljne entitete. To je razvidno iz sledečih dveh izjav:

(a) "*Zaupam ti, ker imaš dober ugled.*"

(b) "*Zaupam ti, čeprav imaš slab ugled.*"

Izjava (a) pravi, da je zaupanje pogojeno z ugledom ciljne entitete, izjava (b) pa da obstaja zaupanje, čeprav je ugled ciljne entitete slab – torej mora obstajati subjektivno znanje (mnenje) izvorne entitete do ciljne. To znanje je posledica osebne izkušnje med entitetama in ima večjo težo kot (negativen) ugled. Če osebna izkušnja ne obstaja, se zaupanje lahko oblikuje na podlagi ugleda oz. priporočil in ocen drugih.

Ugled se lahko pojmuje kot celotna mera zanesljivosti in stopnje zaupanja na osnovi priporočil ocen članov skupnosti. Posledično je individualno subjektivno zaupanje kombinacija dobljenih priporočil drugih in osebne izkušnje [9]. Pri oblikovanju priporočil je pomembno, da so rezultat neposredne interakcije s ciljno entiteto in ne ocena drugih priporočil. V nasprotnem primeru pride do odvisnosti in zank v relacijah zaupanja [21].

2.4 Obvladovanje zaupanja in ugleda

Pojma zaupanja in ugleda sta praviloma povezana tudi z njunim obvladovanjem (upravljanjem). V splošnem so entitete, ki nastopajo v okoljih e-poslovanja, razpršene v prostoru in posledično nimajo direktnega medsebojnega stika in izkušenj. Obstajati mora ustrezen mehanizem – sistem za vzpostavitev razmerij (relacij) zaupanja. Govorimo o sistemih za obvladovanje zaupanja(2) (angl. trust management – TM) in ugleda(3) (angl. reputation management – RM), ki so primer t. i. "mehkega" varnostnega mehanizma oz. "mehke" varnosti.

Obvladovanje zaupanja pomeni zbiranje potrebnih informacij za vzpostavitev relacij zaupanja ter dinamičen nadzor in prilagajanje obstoječih relacij zaupanja med entitetami [22]. Sistemi za OZ in OU imajo sledeče lastnosti oz. značilnosti [23]:

a) Entitete morajo biti v sistemu prisotne daljši čas, torej se z vsako interakcijo pričakujejo tudi interakcije v prihodnosti.

b) Ocene (mnenja) o trenutnih interakcijah se shranijo na eno ali več lokacij.

c) Ocene (mnenja) preteklih interakcij so vodilo za odločitve o trenutnih interakcijah.

Značilnost a) pomeni, da entiteta ne more oz. težko spremeni svojo identiteto ali psevdonim; s spremembo se prekine vez s preteklim obnašanjem. Za lastnost b) je potrebna pripravljenost entitet, da podajo svoje ocene preko ustreznega protokola za ocenjevanje (centralizirani, porazdeljeni sistem). Značilnost c) določa uporabnost sistema v različnih okoljih.

(2) V nadaljevanju je uporabljena kratica OZ.
(3) V nadaljevanju je uporabljena kratica OU.

2.4.1 Razlika med sistemi OZ in OU

Glavna razlika med sistemom za obvladovanje zaupanja ugleda je v tem, da pri zaupanju sistem vrne rezultat, ki odraža subjektiven pogled na stopnjo zaupanja izbrane entitete (t. i. *privatna informacija* kot odraz neposredne izkušnje), pri ugledu pa sistem vrne rezultat, ki je odraz celotne mreže (socialne strukture) in je javno dostopen (t. i. *javna informacija*, pridobljena od tretjih entitet). Druga razlika je tudi v tem, da pri zaupanju sistem vzame kot vhod subjektivno mero o zanesljivosti, medtem ko pri ugledu predstavlja vhod informacija oz. ocene (angl. ratings) o specifičnih objektivnih dogodkih (npr. transakcije). Velja pa poudariti, da obstajajo sistemi OZ, ki vključujejo elemente OU in obratno, zato ni vedno jasnih pravil, kako klasificirati določen sistem. Skupna točka sistemom OZ in OU je podpora odločanju.

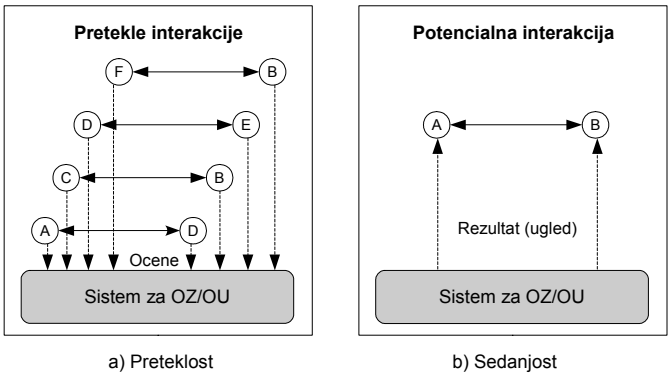
2.4.2 Arhitektura sistemov OZ in OU

V tem razdelku bomo prikazali tehnične principe za implementacijo sistemov OZ in OU. Arhitektura določa način širjenja ocen in rezultatov med člani mreže. Obstajata dva glavna tipa arhitekture sistemov OZ in OU – *centralizirana* in *porazdeljena*. V obeh primerih sta pomembna dva vidika:

- *Komunikacijski protokol*, ki omogoča članom skupnosti podajanje ocen o interakcijah, kot tudi pridobivanje rezultatov (ugled oz. stopnja zaupanja) o izbranih entitetah.
- *Algoritem za izračun* ugleda oz. stopnje zaupanja, temelji na dobljenih ocenah in lahko tudi na drugih informacijah.

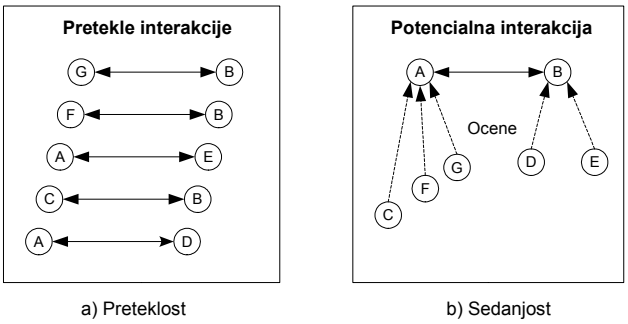
V centraliziranem sistemu se zbirajo podatki (ocene), ki jih imajo člani skupnosti o izbrani entiteti, s katero so bili v neposredni interakciji. Sistem shranjuje vse ocene na eno centralno lokacijo ter vrača javno dostopen rezultat (ugled oz. stopnja zaupanja) za vsako entiteto. Rezultat služi kot informacija za odločanje ali stopiti v interakcijo (transakcijo) z izbrano entiteto ali ne. Po vsaki interakciji sistem sprejme ocene vseh entitet in ažurira rezultat – ugled kot funkcijo sprejetih ocen, ki je ves čas javno dostopen vsem članom skupnosti. Centralizirani sistemi so najbolj pogosto komercialno uporabljeni sistemi (npr. eBay, Amazon). Njihova slabost je majhna skalabilnost, lahko odpovejo v primeru prevelikega števila interakcij. Slika 2.1 prikazuje primer centraliziranega sistema OU, kjer A in B predstavljata entiteti s svojo zgodovino interakcij, trenutno pa hočeta stopiti v medsebojno interakcijo (transakcijo).

V porazdeljenem sistemu ne obstaja centralna lokacija za shranjevanje ocen in pridobivanje rezultatov o ugledu. Namesto tega se uporabljajo porazdeljene lokacije za shranjevanje ocen; te so lahko tudi pri samih vozliščih (entitetah) omrežja, ki shranjujejo svoje ocene o izkušnjah z drugimi. Entiteta, ki želi v interakcijo z določeno ciljno entiteto, mora najti ustrezne porazdeljene lokacije oz. pridobiti ocene od čimveč članov skupnosti, ki so imeli neposredno izkušnjo (oceno) s ciljno entiteto. Primer prikazuje slika 2.2. Izvorna entiteta izračuna ugled



Slika 2.1: Centraliziran sistem za obvladovanje zaupanja/ugleda.

na podlagi dobljenih ocen. V primeru, da že obstaja neposredna izkušnja s ciljno entiteto, ima ta večjo težo od izračunanega ugleda. Porazdeljeni sistem je bistveno bolj kompleksen od



Slika 2.2: Porazdeljen sistem za obvladovanje zaupanja/ugleda.

centraliziranega. Zelo pomemben je učinkovit komunikacijski protokol za porazdeljene poizvedbe in ažuriranje ocen. Prednost je v skalabilnosti in robustnosti pri velikem številu entitet oz. agentov. Porazdeljeni pristop se uporablja v omrežjih P2P [20, 24]. Omrežja P2P predstavljajo okolje za učinkovito uporabo porazdeljenih sistemov OU [22, 25, 26, 27]. Entitete nastopajo istočasno kot odjemalci in strežniki. Cilj sistema OU v omrežju P2P je ugotoviti, katera entiteta je najbolj zanesljiva pri ponujanju kvalitetnih storitev in virov ter katera entiteta ponuja najbolj zanesljive informacije [11]. V porazdeljenih okoljih tipa P2P je vsaka entiteta odgovorna za zbiranje in agregacijo ocen drugih entitet. Vendar je zaradi narave porazdeljenega omrežja ponavadi nemogoče ali časovno prezahtevno pridobiti ocene vseh interakcij z izbrano entiteto. Zaradi tega se ugled izračuna samo na podmnožici ocen, ponavadi na lokalni soseščini izvorne entitete.

2.5 Tradicionalni sistemi obvladovanja zaupanja

Ta razdelek prikazuje nekatere tipične obstoječe sisteme za obvladovanje zaupanja, ki jih lahko označimo kot trdne varnostne mehanizme, saj ne vključujejo socio-kognitivne dimenzije zaupanja. Osredotočajo se na različne protokole za izvajanje varnostnih storitev v specifičnem

kontekstu; nekateri uporabljajo ustrezne jezike za opis politik zaupanja, ki specificirajo zahteve in kriterije za ciljne entitete, da so vredne zaupanja. Zgodnji tradicionalni sistemi OZ so bili sistemi za kontrolo dostopa s principom preverjanja uporabniških poverilnic na podlagi varnostnih politik za omejevanje dostopa do virov in storitev.

Tradicionalni sistemi OZ tako temeljijo na različnih mehanizmih overjanja in kontrole dostopa; novejši vsebujejo tudi formalizacijo (opis politk dostopa in preverjanje skladnosti [6], formalni mehanizmi družine jezikov RT [28], Crypto-CCS [29, 30], relacijske algebre [31]). Ne upoštevajo časovne dinamike in psihološko-socialne dimenzije zaupanja, ampak le tehnološke osnove, zato jih lahko označimo kot *delne sisteme* za obvladovanja zaupanja. Velja omeniti nekatere tradicionalne sisteme OZ, katerih večina temelji na infrastrukturi javnih ključev (angl. Public Key Infrastructure – PKI) [8, 5]. PKI temelji na principu digitalnih potrdil (vsebujejo javni ključ asimetričnega kriptografskega sistema RSA [8]), ki so digitalno podpisana s strani pooblaščenih overoviteljev (angl. Certificate Authority – CA). Digitalno potrdilo povezuje identiteto uporabnika (lastnika) in javni ključ preko CA in predstavlja elektronsko identifikacijo. Pri PKI gre za tehnično in organizacijsko strukturo procesa, ki zajema izdajanje in preklicevanje digitalnih potrdil. Z vzpostavljeno infrastrukturo PKI je možno digitalno podpisovanje, enkripcija sporočil, overjanje ter vzpostavitev mreže zaupanja med lastniki digitalnih potrdil.

Omeniti velja še TRUSTe [32], ki je izjema, saj upošteva samo socialno dimenzijo zaupanja in ima glavni namen pospeševati ter povečevati e-poslovanje. TRUSTe ni model zaupanja, ampak zbirka uporabnih smernic, priporočil in prakse na področju varnosti spletnih aplikacij in mobilnih naprav. Njen glavni cilj je zmanjšati ranljivost in tveganja znotraj področja varnosti ter posledično povečati vrednost zaupanja med sodelujočimi entitetami znotraj e-poslovanja.

2.5.1 X.509

X.509 [33] je hierarhičen model za podporo zaupanja preko overjanja. Vsaka entiteta mora imeti digitalno potrdilo, ki je podpisano s strani vrhovnega (korenskega) uradnega overovitelja (angl. CA root) ali druge entitete, ki je overjena s strani vrhovne CA posredno ali neposredno. Ta model podpira zaupanje v obliki hierarhičnega drevesa; zaupanje med dvema entitetama v drevesu obstaja, če med njima obstaja pot skozi graf preko vrhovne CA (veriga overjanja). Obstajajo različni algoritmi za iskanje poti in računanje stopnje zaupanja skozi verigo overjanja [34, 15]. X.509 je standard za PKI, poleg tega določa tudi format in strukturo digitalnih potrdil, seznam preklicanih potrdil (angl. Certificate Revocation List – CRL) [5].

2.5.2 PGP

PGP (angl. Pretty Good Privacy) [16, 11, 35] je model za podporo zaupanja preko overjanja in za razliko od X.509 nima hierarhične strukture. Največkrat se uporablja pri e-pošti med uporabniki interneta. Model predvideva, da vsaka entiteta lahko digitalno podpiše javne ključe

(potrdi povezavo identitete in javnega ključa) drugih entitet in tako nastopa tudi v vlogi CA. Model tudi predvideva implicitno zaupanje entitete v lastni javni ključ [16].

Zaupanje nastopa v treh vidikih [36], ki so *"Zaupanje lastnika"* (Owner Trust) – določa zaupanje v lastnika javnega ključa, *"Zaupanje podpisa"* (Signature Trust) – določa zaupanje v prejeto digitalno potrdilo in *"Veljavnost ključa"* (Key Validity) – določa zaupanje v javni ključ. Vsak tip zaupanja ima določeno diskretno vrednost:

$$\begin{matrix}\text{Zaupanje lastnika}\\ \text{Zaupanje podpisa}\end{matrix} = \begin{cases} always\ trusted \\ usually\ trusted \\ not\ trusted \\ unknown\ trust \end{cases} \qquad \text{Veljavnost ključa} = \begin{cases} complete \\ marginal \\ undefined \end{cases}$$

Privatni ključ uporabnika je shranjen v tabeli *Secret Key Ring* in se uporablja za digitalno podpisovanje in dekripcijo kriptiranih sporočil. Tabela *Public Key Ring* se uporablja za shranjevanje javnih ključev drugih uporabnikov skupaj s parametri 'Veljavnost ključa' in 'Zaupanje lastnika' za vsak ključ. Ti ključi služijo enkripciji sporočil, ki se pošiljajo drugim uporabnikom in preverjanje digitalnih podpisov prejetih sporočil.

Ko uporabnik prejme javni ključ preko digitalnega potrdila, se najprej preveri vrednost 'Veljavnost ključa', ki mora biti enaka *complete*, v nasprotnem primeru se potrdilo ignorira. Po sprejetju potrdila se njegova vrednost 'Zaupanje podpisa' priredi na vrednost 'Zaupanje lastnika' lastnika, ki je digitalno podpisal potrdilo. Privzeto model PGP zahteva eno vrednost *always trusted* ali dve *usually trusted* za potrdilo, da priredi vrednost *complete* parametru 'Veljavnost ključa' prejetega javnega ključa. Premajhno število vrednosti *always trusted* ali *usually trusted* parametra 'Zaupanje podpisa' povzroči vrednost *marginal* oz. *undefined*.

PGP izračuna samo vrednost parametra 'Veljavnost ključa', medtem ko za parameter 'Zaupanje lastnika' vpraša uporabnika s kakšno stopnjo zaupa lastniku prejetega javnega ključa (subjektivna odločitev). Oba parametra predstavljata zaupno informacijo in nista na voljo za druge uporabnike. Ko sta določeni vrednosti 'Veljavnost ključa' in 'Zaupanje lastnika' za določen javni ključ, ga uporabnik lahko digitalno podpiše in doda v tabelo *Public Key Ring*. Uporabnik lahko sedaj posreduje ta javni ključ ostalim, ki ga ovrednotijo po enakem postopku.

Slabost tega modela je v tem, da nima dobro definiranih mehanizmov za ustvarjanje, pridobivanje in distribucijo potrdil, zato ni primeren za okolja e-poslovanja, ampak za osebno komunikacijo.

2.5.3 PolicyMaker

PolicyMaker [16] je sistem za podporo zaupanja, osnovan na modelu X.509, ki določa dovoljene akcije v povezavi z javnim ključem. Identiteta (ime) uporabnika je povezana z javnim ključem preko digitalnega potrdila. Sistem preveri ali je zahtevana akcija v skladu z lokalno politiko. Lokalna politika je množica trditev za zaupanje. PolicyMaker uporablja pojem poverilnic, ki so digitalno podpisane trditve entitet. Trditev je lahko politika ali poverilnica. Vhod v sistem so

lokalna politika, prejete poverilnice in imena akcij, ki jih javni ključ lahko izvede. Rezultat, ki ga vrne sistem, je logična vrednost (ustreza, ne ustreza) ali seznam omejitev, da se akcija lahko izvrši.

Politike in poverilnice so opisane v *jeziku trditev* (angl. assertion language), ki ima sledečo sintakso:

Source ASSERTS *AuthorityStruct* WHERE *Filter*

Source predstavlja vir trditve, *AuthorityStruct* predstavlja javni ključ uporabnika, na katerega se trditev nanaša, *Filter* je predikat, ki ga mora izpolnjevati akcija, da trditev drži. Filtri so opisani v posebnem jeziku, lahko so tudi zunanji programi. Za vhod vzamejo trenutni akcijski niz in okolje, kjer so na voljo podatki o trenutnem kontekstu (npr. datum, čas, ...).

PolicyMaker uporablja poverilnice za dokazovanje, da zahtevana akcija ustreza predpisani politiki (proces preverjanja skladnosti). Torej v splošnem sistem vzame množico zahtevanih akcij, množico poverilnic in politiko in na podlagi tega skuša dokazati, da poverilnice vsebujejo dokaz o ustreznosti akcij glede na politiko.

2.5.4 KeyNote

KeyNote [16, 6, 37] je naslednik PolicyMaker-ja in je bil razvit z namenom, da izboljša njegove pomanjkljivosti. Ima isti princip trditev in poizvedb, a ima dve prednosti – standardizacijo in lažjo integracijo [37]. V primerjavi s sistemom PolicyMaker (gre v večini primerov za klicanje zunanjih aplikacij), ima sistem KeyNote večino funkcij (npr. preverjanje digitalnih podpisov) izvedenih v samem jedru. Uporablja samo eno različico jezika za opis trditev (PolicyMaker lahko uporablja različne), kar omogoča lažjo integracijo s strojem za preverjanje skladnosti.

Sistem KeyNote kot vhod sprejme sledeče parametre kličoče aplikacije: seznam poverilnic, politike, javne ključe vira ter vrednosti okolja izvajanja oz. akcij (seznam parov kjuč-vrednost). Okolje izvajanja generira aplikacija in vsebuje vse ključne informacije, ki se nanašajo na zahtevo (vir) in tako odraža varnostne zahteve aplikacije. Rezultat, ki ga vrne KeyNote, je aplikacijsko določen niz, kot npr. *true*, *false*, *authorized*, *admin_access*, *no_access*.

Trditev znotraj KeyNote sistema je lahko poverilnica ali politika. Vsebuje množico polj, ki določajo entiteto za avtorizacijo oz. njen javni ključ (polje *Licensee*), kdo avtorizira (polje *Authorizer*) ter predikat za testiranje seznama akcijskih atributov. Sledi primer KeyNote trditve:

```
Authorizer: rsa-pkcs-hex:"1123acbd"
Licensees: dsa-hex "94532c1" || rsa-pkcs1-hex:"14abcd12"
Conditions: ($file == "etc/passwd" && $access == "read") -> true
Signature: rsa-md5-pkcs1-hex:"d000f5379"
```

KeyNote ima dobro definirano semantiko in je *monoton* (angl. monotonic) – z brisanjem trditve ni možno povečati privilegijev. Omogoča tudi kombinacijo (verigo) večih trditev, vendar ne vsebuje funkcionalnosti za iskanje minimalne množice poverilnic, ki ustreza določeni politiki.

2.5.5 W3C PICS

PICS (Platform for Content Selection) [16, 38] je bil razvit s strani konzorcija W3C. Njegov glavni namen je, da omejuje dostop do določenih spletnih dokumentov. PICS določa standarde za format in distribucijo oznak, ki so meta-dokumenti za opis spletnih dokumentov. Aplikacija, skladna s PICS, prebere oznake, uporabi definirana pravila – filtre in se odloči ali sprejme ali zavrne dokument.

Spletni dokument ima lahko več oznak, ki so določene s strani različnih organizacij. Uporabnik lahko izbere poljubno PICS aplikacijo in oznako – imenuje se *ocenjevalna storitev* (angl. rating service). Ocenjevalni sistem PICS definira atribute oznak in njihovo zalogo vrednosti za ocenjevalno storitev. Sledi primer dokumenta za ocenjevalni sistem [38]:

```
((PICS-version 1.1)
 (rating-system "http://www.gcf.org/ratings")
 (rating-service "http://www.gcf.org/v1.0/")
 (icon "icons/gcf.gif")
 (name "The Good Clean Fun Rating System")
 (description "For demonstration purposes only.")

 (category
  (transmit-as "suds")
  (name "Soapsuds Index")
  (min   0.0)
  (max   1.0))

 (category
  (transmit-as "density")
  (name "suds density")
  (label (name "none") (value 0) (icon "icons/none.gif"))
  (label (name "lots") (value 1) (icon "icons/lots.gif")))

 (category
  (transmit-as "subject")
  (name "document subject")
  (multivalue true)
  (unordered true)
  (label (name "soap") (value 0))
  (label (name "water") (value 1))
  (label (name "soapdish") (value 2))
  (label-only))
))
```

Prva sekcija določa verzijo PICS, ocenjevalni sistem in ocenjevalno storitev. Naslov URL ocenjevalnega sistema in storitve določa lokacijo dokumenta s pripadajočim opisom za uporabnika. Sledi definicija atributov oznak, ki so ločeni s kategorijami (*category*). Vsaka

kategorija vsebuje ime prenosa (*transmit-as*), kateremu sledi zaloga vrednosti za atribut. Primer PICS oznake (v obliki BNF[4]), ki uporablja prej navedeni ocenjevalni sistem:

```
((PICS-version 1.1)
 "http://www.gcf.org/v1.0/"
 labels on "2001.11.05T08:16.0500"
 until "2002.11.05T08:16.0500"
 for "http://www.example.com/index.html"
 by "John Smith"
 ratings (suds 0.5 density "lots" subject "soap"))
```

Primer določa ocenjevalno storitev z definiranimi oznakami, časovni interval veljavnosti, lokacijo spletnega dokumenta za označitev, osebo ter dejanske vrednosti atributov. Podrobnejši opis sintakse oznak PICS, je podan v [38].

2.5.6 Referee

Referee (Rule-Controlled Environment for Evaluation or Rules and Everything Else) [16] je model za podporo zaupanja, ki omogoča odločanje o dostopu na podlagi dokumentov, temelječih na sistemu PolicyMaker. Je tipično sestavljen iz več medsebojno sodelujočih modulov, kjer vsak omogoča odločanje glede na izbrano politiko. Tako kot PolicyMaker in KeyNote mora biti tudi Referee vgrajen v gostiteljsko aplikacijo. Sistem ovrednoti zahteve in vrne rezultat (*true*, *false* ali *unknown* – tri-vrednostna logika) ter seznam izjav, ki upravičujejo dani rezultat.

Seznam izjav vsebuje pridobljene informacije tekom izvajanja modulov. Predstavljajo način komunikacije med moduli. Vse izjave so sestavljene iz dveh elementov oz. izrazov. Prvi izraz določa kontekst, drugi pa vsebino izjave. Primer izjave, da je uporabnik Janez zaupanja nevreden znotraj modula Referee:

```
(("certification module")
 ("Janez" (untrustworthy yes)))
```

Moduli znotraj Referee predstavljajo osnovne bloke, kjer je vsak zadolžen za odločitev glede na določeno politiko. Modul lahko delegira pod-naloge drugim modulom in naredi odločitev na podlagi vrnjenih trditev. Vsi moduli imajo enak vmesnik – sprejmejo ime oz. informacijo o akciji kot vhod ter vrnejo možne tri vrednosti in seznam izjav. Modul je sestavljen iz politike in enega ali več tolmačev (angl. interpreter). Politika je segment kode, ki je napisana v ustreznem jeziku, tolmači pa so programi za interpretacijo politike ali drugih tolmačev. Množica tolmačev znotraj modula je hierarhična; politika modula je interpretirana s tolmačem na najvišjem nivoju, ki je lahko interpretiran s tolmačem na nižjem nivoju.

Življenjski cikel sistema Referee je sestavljen iz dveh faz: *zagon* (angl. bootstrap) in *povpraševanje* (angl. query). V prvi fazi gostiteljska aplikacija posreduje sistemu implicitno

[4] Backus Naur Form.

zaupanja vredne trditve in repozitorij imena akcij (repozitorij je podoben sistemu DNS (Domain Name Server)), ki poveže imena akcij z moduli. V drugi fazi gostiteljska aplikacija priskrbi akcijo in ostale argumente (npr. poverilnice), ki se predajo ustreznemu modulu. REFEREE požene tolmač modula za politiko in ostale argumente (ta lahko proži tudi druge module) in vrne rezultat gostiteljski aplikaciji.

2.5.7 IBM Trust Establishment Framework

Združba IBM vidi zaupanje kot centralni del za e-poslovanje, zato je razvila model za kontrolo dostopa preko vlog (RBAC – Role Based Access Control) na osnovi X.509 v3 digitalnih potrdil: *Java Trust Establishment Module (JTEM)* in *Trust Policy Language (TPL)* [39]. Sistem je podoben PolicyMaker-ju, vendar omogoča tudi negativna pravila za omejevanje dostopa.

JTEM preveri digitalno potrdilo odjemalca in preslika lastnika potrdila v ustrezno vlogo. Lokalna politika, definirana preko TPL, določa, kakšne pravice ima vloga oz. kaj lahko počne. TPL [6] uporablja sintakso jezika XML. Osnovna struktura znotraj TPL je skupina (element `<GROUP>`). Vsaka skupina vsebuje pravila (element `<RULE>`) za članstvo. Pravila v osnovi določajo, katera digitalna potrdila je potrebno preveriti. Pravilo vsebuje zaporedje elementov `<INCLUSION>`, ki določajo osnovne lastnosti (atribute) digitalnega potrdila; npr. tip (*type*) in izdajatelja (*issuer*). Več omejitev na atributih digitalnega potrdila vsebuje element `<FUNCTION>`. TPL podpira osnovne relacijske in logične operatorje ter zunanje funkcije. Sledi primer definicije skupine *VeljavenStudent*, kamor spadajo samo digitalna potrdila študentov, ki so digitalno podpisani od skupine *Univerza*, so tipa *uni-lj* in imajo status enak *"student"*:

```
<POLICY>
  <GROUP NAME="Univerza">
  </GROUP>
  <!-- skupina za veljavne študente -->
  <GROUP NAME="VeljavenStudent">
    <RULE>
      <INCLUSION FROM="Univerza" ID="studcrt" TYPE="uni-lj" />
      <FUNCTION>
        <EQ>
          <FIELD ID="studcrt" NAME="Status" />
          <CONST>"student"</CONST>
        </EQ>
      </FUNCTION>
    </RULE>
  </GROUP>
</POLICY>
```

Ko modul JTEM ugotovi, da lahko entiteta pripada skupini oz. vlogi, pošlje to informacijo drugemu modulu, ki vrne seznam dostopnih pravic, vezanih na to vlogo.

2.6 Modeli zaupanja

V splošnem je opis zaupanja v obliki formalnega modela težka naloga, saj zaupanje vsebuje mnogo različnih dimenzij, ki jih je potrebno upoštevati: subjektivnost, zanesljivost, razni situacijski elementi (npr. korist, profit), faktorji okolja (varnostni mehanizmi, zakonodaja), vidiki tveganja, itd. Nekateri ekonomisti zavračajo zaupanje kot računski koncept; npr. Willimanson [40] nasprotuje uporabi zaupanja v ekonomskih transakcijah, saj so zanesljivost, korist in tveganje (dimenzije zaupanja) zadostni za ta namen.

Sicer ima Williamson do neke mere celo prav, vendar ne upošteva, da zaupanje ni omejeno samo na dimenzije, ki nastopajo v ekonomskih modelih in sistemih za podporo odločanju. Trditev, da socialna komponenta ni prisotna v ekonomski domeni, je napačna. Tudi ekonomski igralci so socialni in se tako tudi obnašajo v ekonomskih transakcijah – vključno s svojimi motivi, idejami, razmerji in zaupanjem, ki ga imajo v poslovne partnerje in združbe, te pa niso nujno racionalno osnovane. Trdimo, da je zaupanje pomemben pojem v ekonomskih modelih, posebno v navezavi z različnimi teorijami kooperacije (npr. teorija iger [41]). Zato je zelo pomembno, kako je zaupanje modelirano kot računski koncept. Znotraj tega razdelka so predstavljani nekateri že izdelani računski modeli za stopnjo zaupanja (ugleda) ter njihova klasifikacija. Nekateri modeli služijo kot izhodišče za formalizacijo našega lastnega modela zaupanja, ki bo razdelan v prihodnjih poglavjih.

Interpretacija zaupanja kot mere zanesljivosti omogoča uporabo različnih merskih sistemov. A. Jøsang [9] npr. deli zaupanje (glej razdelek 2.3) na zaupanje v zanesljivost (diskretna ali zvezna metrika) in odločitveno zaupanje (binarna metrika). Običajno sistemi za OZ predvidevajo predvsem prvo vrsto zaupanja, druga pa je domena orodij za podporo odločanju.

Naša osnovna klasifikacija loči *kvalitativne* in *kvantitativne* modele [42] za izračun stopnje zaupanja. V splošnem se stopnja lahko izračuna na podlagi lastnih izkušenj (privatna informacija), priporočil drugih (javna informacija) ali kot kombinacija obojega. V nadaljevanju sledi prikaz nekaterih modelov z osnovnimi značilnostmi. Nekateri od njih ostajajo zgolj v akademski domeni, drugi pa se uporabljajo tudi v komercialnih aplikacijah – večina so domena različnih spletišč (npr. *eBay*, *Amazon*, *Epinions*, *BizRate*, *Slashdot*,...) [9], prevladuje pa centralizirana arhitektura.

2.6.1 Kvantitativni modeli

Pri teh modelih je stopnja zaupanja podana preko kvantitativne, numerične vrednosti. V nadaljevanju sledijo različne razvrstitve glede na uporabljene algoritme za izračun.

2.6.1.1 Modeli z enostavnimi vsotami in povprečji

Ti modeli predstavljajo najenostavnejše pristope pri izračunu stopnje zaupanja in ugleda. V osnovi gre za seštevanje pozitivnih ocen, od česar se odšteje vsota negativnih ocen. Obstajajo

različne variacije, npr. povprečja, dodajanje različnih uteži (npr. ugled ocenjevalca, starost ocene, razlika med oceno in trenutnim rezultatom, itd.).

Te modele uporabljajo zlasti spletna mesta za elektronske licitacije in trgovanje (npr. eBay, Amazon) [9, 23]. *eBay* sestavlja množica kupcev in prodajalcev. Pri vsaki izvedeni transakciji kupec oceni prodajalca in obratno z oceno 1 (pozitivno), 0 (nevtralno) ali -1 (negativno). Ugled uporabnika se izračuna kot skupek vseh ocen, dobljenih v zadnjih 180 dneh. Novi uporabniki dobijo ugled enak 0. Pri *Amazon.com* uporabniki ocenjujejo drug drugega z oceno od 1 do 5. Ugled se izračuna kot povprečje vseh dobljenih ocen v celotnem času delovanja sistema. Novi uporabniki nimajo definiranega ugleda.

2.6.1.2 Verjetnostni modeli

Verjetnostni modeli predvidevajo specifično obnašanje entitete v prihodnosti na podlagi informacij o preteklem obnašanju. Temeljijo na verjetnostnih in statističnih metodah. Predvidevanje določenega obnašanja entitete je izraženo kot verjetnost. Verjetnost pomeni kakšna je stopnja zaupanja oz. ugleda izbrane entitete. Nova – *aposteriorna* (angl. posterior) vrednost se izračuna na podlagi prejšnjih – *apriornih* (angl. prior) vrednosti in trenutne ocene. Ločimo *binomske* (t. i. dvovrednostne) modele za možnost ocenjevanja preko dveh vrednosti (npr. pozitivno, negativno obnašanje) ter *multinomske* (t. i. večvrednostne) za možnost ocenjevanja z več vrednostmi oz. stopnjami (npr. slabo, povprečno, dobro, odlično) [11, 24, 43]. Prednost verjetnostnih modelov je, da izračun stopnje zaupanja in ugleda temelji na matematični teoriji, hkrati pa je slabost, da so preveč kompleksni za razumevanje s strani običajnega uporabnika.

Bayesovi modeli so primer binomskih modelov [44]. Temeljijo na statistični teoriji določanja negotovih binarnih dogodkov θ. Na začetku ima opazovani dogodek apriorno verjetnost $p(\theta)$, ki odraža začetno znanje o dogodku. Ko je na voljo nova informacija $\mathcal{I}$, se nova – posteriorna verjetnost spremeni glede na Bayesovo pravilo (pogojna verjetnost) (2.1):

$$aposteriorna\ verjetnost = p(\theta|\mathcal{I}) = \frac{p(\mathcal{I}|\theta)p(\theta)}{p(\mathcal{I})} \tag{2.1}$$

V primeru, da je apriorna verjetnost $p(\theta)$ binomska verjetnostna *beta* (B) [45] porazdelitev (dogodki neodvisnih poskusov metanja kovanca), potem je tudi posteriorna verjetnost $p(\theta|\mathcal{I})$ izražena kot *beta* porazdelitvena funkcija B [45]. Stopnja zaupanja oz. ugleda se torej izraža kot porazdelitvena funkcija B (beta PDF; angl. probability density function) s parametroma α (pomeni število pozitivnih ocen) in β (pomeni število negativnih ocen). Družina beta porazdelitvenih funkcij je družina zveznih porazdelitvenih funkcij s parametroma α in β. Beta porazdelitvena funkcija $B(\theta|\alpha,\beta)$ se lahko izrazi z gama funkcijo Γ, kjer je $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\mathrm{d}t$ za $z > 0$ (2.2):

$$B(\theta|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}, 0 \leq \theta \leq 1, \alpha > 0, \beta > 0 \tag{2.2}$$

z omejitvijo verjetnostne spremenljivke $\theta \neq 0$, če $\alpha < 1$ in $\theta \neq 1$, če $\beta < 1$ [35]. Matematično upanje se izraža kot $E(\theta) = \frac{\alpha}{\alpha+\beta}$. Ob začetnem stanju, ko ni na voljo nobenih informacij, je apriorna porazdelitev enaka $B(\theta|1,1), \alpha = 1, \beta = 1$. Kasneje po r pozitivnih in s negativnih izidih transakcij, je posteriorna porazdelitev enaka $B(\theta|r+1, s+1), \alpha = r+1, \beta = s+1$. Beta PDF $B(\theta|\alpha, \beta)$ izraža verjetnost, da bo stopnja ugleda v prihodnosti pozitivna. Vendar bolj običajno je, da se ugled izraža kot funkcija matematičnega upanja [9].

Multinomski Bayesovi modeli [11, 43] omogočajo ocenjevanje za k različnih vrednosti oz. stopenj, ki se lahko smatrajo kot množica k ($2 \leq k \in \mathbb{N}$) disjunktnih elementov oz. stopenj. Naj bo ta množica $\mathcal{L} = \{L_1, L_2, \ldots, L_k\}$; ocene se podajajo kot glasovi (angl. votes) za elemente (stopnje) množice $\mathcal{L}$. Samo za ilustracijo lahko ponazorimo obnašanje agenta v transakcijah s petimi različnimi stopnjami; elementi množice $\mathcal{L}$ so lahko ($k = 5$):

$$
\begin{aligned}
L_1 &= \text{zlonamerno obnašanje,}\\
L_2 &= \text{slabo obnašanje,}\\
L_3 &= \text{povprečno obnašanje,}\\
L_4 &= \text{dobro obnašanje,}\\
L_5 &= \text{odlično obnašanje.}
\end{aligned}
$$

Izračun verjetnosti za posamezno stopnjo vodi k *Dirichletovi* $\mathcal{D}$ verjetnostni porazdelitveni funkciji reda k naključnih verjetnostnih spremenljivk $\theta(L_i), i = 1 \ldots k$ na prostoru $[0,1]^k$, kjer velja $\sum_{i=1}^{k} \theta(L_i) = 1$ [11, 43]. Dirichletova porazdelitev temelji na zaporedju neodvisnih poskusov s k različnimi izidi s k pozitivnimi ocenjevalnimi parametri $r(L_i), i = 1 \ldots k$, kjer vsak pripada enemu elementu množice $\mathcal{L}$. Vektor $\vec{\theta} = \{\theta(L_i) \mid 1 \leq i \leq k\}$ označuje verjetnostno spremenljivko s k elementi, vektor $\vec{r} = \{r_i \mid 1 \leq i \leq k\}$ pa označuje spremenljivko ocen s k elementi. Dirichletova porazdelitev $\mathcal{D}$ je določena s k različnimi parametri kot [43] (2.3):

$$\mathcal{D}(\vec{\theta}|\vec{r}) = \frac{\Gamma(\sum_i L_i)}{\prod_i \Gamma(L_i)} \prod_i \theta_i^{L_i - 1} \tag{2.3}$$

Apriorna verjetnost na prostoru $[0,1]^k$ je predstavljena z vektorjem $\vec{a}$. Posteriorna verjetnost oz. ugled je definiran kot matematično upanje vsake naključne verjetnostne spremenljivke glede na elemente iz $\mathcal{L}$ za ocenjevanje. Matematično upanje za vsako izmed k naključnih verjetnostnih spremenljivk je definirano kot (2.4):

$$E(\theta(L_i)|\vec{r}, \vec{a}) = \frac{r(L_i) + Ca(L_i)}{C + \sum_{i=1}^{k} r(L_i)} \tag{2.4}$$

Konstanta C predstavlja apriorno utež, ponavadi ima vrednost $C = 2$ [11]. Izbira večje vrednosti pomeni, da ima nov poskus oz. opazovanje manj vpliva na Dirichletovo porazdelitev. Tako vektor $\vec{a}$ kot utež C predstavljata apriorno informacijo, ki je specifična glede na kontekst.

2.6.1.3 Modeli prepričanja

Modeli prepričanja temeljijo na verjetnostni teoriji. Vsota verjetnosti čez vse možne izide ni nujno enaka 1, ostanek verjetnosti se označuje kot *negotovost* (angl. uncertainty). Primer je Dempster-Shaferjeva teorija evidence [46], ki definira množico možnih stanj kot *okvir razsodnosti* (angl. frame of discernment) Θ. Znotraj tega okvirja se predvideva, da je aktivno samo eno stanje ob vsakem času. Če okvir vsebuje npr. atomarna stanja x_1, x_2, x_3 in sestavljeno stanje $x_4 = \{x_1, x_3\}$, potem je $\Theta = \{x_1, x_2, x_3, \{x_1, x_3\}\}$. Vsakemu stanju se priredi *teža prepričanja* m_Θ (angl. belief mass). Stanje x_4 se interpretira kot prepričanje, da je eno od stanj x_1 ali x_3 resnično, vendar opazovalec ne ve, katero stanje točno je. Teža prepričanj m_Θ je osnova za *funkcijo prepričanja* (angl. belief function) $b : 2^\Theta \mapsto [0, 1]$ definirano kot $b(x) = \sum_{y \subseteq x} m_\Theta(y)$, kjer $x, y \in 2^\Theta$. Funkcija prepričanja se interpretira kot celotno prepričanje, da je določeno stanje resnično, bodisi atomarno ali sestavljeno.

Dempster-Shaferjeva teorija služi kot osnova za Jøsangovo *subjektivno logiko* [47]. *Mnenje* (angl. opinion) je definirano kot $\omega_p^A = (b, d, u, a)$ in določa prepričanje entitete A v resničnost trditve p; b pomeni stopnjo *zaupanja* (angl. belief), d stopnjo *nezaupanja* (angl. disbelief) in u stopnjo *negotovosti* (anlg. uncertainty) kjer velja: $b, d, u \in [0, 1]$, $b + d + u = 1$. Parameter $a \in [0, 1]$ se imenuje *relativna atomarnost* in določa, koliko negotovost vpliva na matematično upanje $E(\omega_p^A) = b + au$. V splošnem je $a = 0.5$ in se lahko izpusti iz zapisa mnenja ω_p^A.

Naj Ω označuje množico mnenj, Φ pa množico porazdelitvenih funkcij B (glej enačbo 2.2). Obstaja bijektivna preslikava med prostorom mnenj Ω in porazdelitvami Φ [35]. V primeru, da parametra α in β v funkciji $B(\theta|\alpha, \beta)$ nadomestimo s parametroma $r \geq 0$ (pozitivni izidi) in $s \geq 0$ (negativni izidi), potem velja $\alpha = r + 1$ in $\beta = s + 1$. Nova porazdelitvena funkcija ψ je sedaj določena s parametri r in s (2.5):

$$\psi(\theta|r, s) = \frac{\Gamma(r + s + 2)}{\Gamma(r + 1)(s + 1)} \theta^r (1 - \theta)^s, \ 0 \leq \theta \leq 1, \ r \geq 0, \ s \geq 0 \tag{2.5}$$

Matematično upanje se sedaj izraža kot $E(\theta) = \frac{r+1}{r+s+2}$. Naj bo $\omega_p = (b, d, u)$ mnenje entitete o binarnem dogodku p, $\psi_p(r, s)$ pa naj bo verjetnostna ocena glede p izražena kot porazdelitvena funkcija. Mnenje ω_p se lahko analitično izrazi kot funkcija $\varphi_p(r, s)$ in obratno (2.6):

$$\left.\begin{array}{l} b = \frac{r}{r+s+1} \\ d = \frac{s}{r+s+1} \\ u = \frac{1}{r+s+1} \end{array}\right\} \iff \left\{\begin{array}{l} r = \frac{b}{1-b-d} = \frac{b}{u} \\ s = \frac{d}{1-b-d} = \frac{d}{u} \\ 1 = b + d + u \end{array}\right. \tag{2.6}$$

Popolna negotovost $\omega = (0, 0, 1)$ ustreza funkciji $\psi(0, 0)$ $[r = 0, s = 0]$, absolutno zaupanje $\omega = (1, 0, 0)$ ustreza $\psi(\infty, 0)$ $[r = \infty, s = 0]$ ter absolutno nezaupanje $\omega = (0, 1, 0)$ ustreza porazdelitveni funkciji $\psi(0, \infty)$ $[r = 0, s = \infty]$.

Subjektivna logika vsebuje okoli 10 različnih operatorjev [47], tu predstavljamo samo nekatere pomembnejše od njih: *konjunkcija* (angl. conjunction), *konsenz* (angl. consensus) in *priporočilo* (angl. recommendation).

(a) **Konjunkcija**. Naj bosta $\omega_p^A = (b_p^A, d_p^A, u_p^A)$ in $\omega_q^A = (b_q^A, d_q^A, u_q^A)$ mnenji entitete A o dveh različnih trditvah p in q. Potem konjunkcija predstavlja skupno mnenje A o trditvah p in q in je definirana kot (2.7):

$$\omega_{p\wedge q}^A = \omega_p^A \wedge \omega_q^A = \{b_{p\wedge q}^A, d_{p\wedge q}^A, u_{p\wedge q}^A\} \text{ kjer } \begin{cases} b_{p\wedge q}^A = b_p^A b_q^A \\ d_{p\wedge q}^A = d_p^A + d_q^A - d_p^A d_q^A \\ u_{p\wedge q}^A = b_p^A u_q^A + u_p^A b_q^A + u_p^A u_q^A \end{cases} \tag{2.7}$$

Konjunkcija mnenj je komutativna in asociativna in zahteva statistično neodvisne argumente (konjunkcija mnenja s samim seboj nima pomena).

(b) **Konsenz**. Naj bosta $\omega_p^A = (b_p^A, d_p^A, u_p^A)$ in $\omega_p^B = (b_p^B, d_p^B, u_p^B)$ mnenji entitet A in B o isti trditvi p. Konsenz predstavlja mnenje navidezne entitete $[A, B]$ o trditvi p (2.8):

$$\omega_p^{A,B} = \omega_p^A \oplus \omega_p^B = \{b_p^{A,B}, d_p^{A,B}, u_p^{A,B}\} \text{ kjer } \begin{cases} b_p^{A,B} = \frac{b_p^A u_p^B + b_p^B u_p^A}{u_p^A + u_p^B - u_p^A u_p^B} \\ d_p^{A,B} = \frac{d_p^A u_p^B + d_p^B u_p^A}{u_p^A + u_p^B - u_p^A u_p^B} \\ u_p^{A,B} = \frac{u_p^A u_p^B}{u_p^A + u_p^B - u_p^A u_p^B} \end{cases} \tag{2.8}$$

Konsenz je komutativen in asociativen in zahteva statistično neodvisne argumente. Namen operatorja je zmanjšati negotovost.

(c) **Priporočilo**. Naj bosta A in B dve entiteti, kjer je $\omega_B^A = \{b_B^A, d_B^A, u_B^A\}$ mnenje A o priporočilih B-ja in naj bo p trditev, kjer je $\omega_p^B = \{b_p^B, d_p^B, u_p^B\}$ mnenje B-ja o trditvi p. Potem je mnenje entitete A o trditvi p rezultat priporočila entitete B in je definirano kot (2.9):

$$\omega_p^{AB} = \omega_p^A \otimes \omega_p^B = \{b_p^{AB}, d_p^{AB}, u_p^{AB}\} \text{ kjer } \begin{cases} b_p^{AB} = b_B^A b_p^B \\ d_p^{AB} = b_B^A d_p^B \\ u_p^{AB} = d_B^A + u_B^A + b_B^A u_p^B \end{cases} \tag{2.9}$$

Operator za priporočilo je asociativen, ni pa komutativen. Posledično se operator lahko uporabi samo na začetku ali na koncu verige mnenj; pomemben je vrstni red kombiniranja mnenj. V verigi, kjer obstaja več kot ena entiteta s priporočilom, je pomembna neodvisnost mnenj (ista entiteta se v verigi ne sme pojaviti več kot enkrat). Pomembno je omeniti, da je uporaba operatorja smiselna samo ob predpostavki, da je priporočilo tranzitivno – entitete v verigi ne spreminjajo vrednosti oz. obnašanja (gledano kot funkcija tistih entitet, s katerimi so v interakciji) [35].

Prednost subjektivne logike je močan matematičen formalizem za računanje sestavljenih mnenj preko različnih operatorjev in posledično stopnje zaupanja in ugleda entitet. Operator za priporočilo se npr. uporablja za izračun stopnje zaupanja v verigah in grafih za overjanje (npr. model X.509) [34, 35]. Slabost subjektivne logike je v tem, da predvideva racionalne entitete za prirejanje ustreznih vrednosti b, d in u. Poleg tega je razumevanje operatorjev za povprečnega uporabnika pretežko; pogoj je razumevanje verjetnosti in logike, ki je velikokrat neizpolnjen.

2.6.1.4 Pretočni modeli

Ti modeli temeljijo na principu izračuna stopnje zaupanja ali ugleda preko tranzitivnih iteracij skozi omrežje (graf) entitet. Nekateri modeli predvidevajo konstantno stopnjo zaupanja/ugleda za celo skupnost, ta stopnja pa se porazdeli med vse člane skupnosti. Entiteta lahko poveča stopnjo zaupanja ali ugled na račun zmanjšanja pri drugih entitetah. V splošnem se stopnja zaupanja ali ugled povečuje kot funkcija *vhodnega toka* in zmanjšuje kot funkcija *izhodnega toka*.

Model *Advogato* temelji na principu računanja maksimalnih pretokov skozi omrežje [14, 9]. Ugotavlja, katere entitete znotraj skupnosti so zaupanja vredne in katere ne. Vhod v model je celo število n, ki predstavlja število entitet, ki se jim zaupa, ter *seme* (angl. seed) zaupanja s, ki je podmnožica vseh entitet $V = \{a_1, \ldots, a_n\}$. Izhod je karakteristična funkcija, ki preslika vsako entiteto v binarno logično vrednost; ta ponazarja stopnjo zaupanja [14] (2.10):

$$\mathrm{Trust}_M : 2^V \times \mathbb{N}_0^+ \mapsto (V \mapsto \{\mathrm{true}, \mathrm{false}\}) \tag{2.10}$$

Model *Appleseed* predstavlja model zaupanja v semantičnih mrežah [14]. Semantična mreža predstavlja pomenske relacije med posameznimi koncepti oz. pojmi; je eden od načinov za predstavitev znanja. Zaupanje v mreži se prenaša oz. širi po metodi širjenja *aktivacijskih modelov* (simulacija človeškega razumevanja). Ta metoda se aplicira tudi za področje iskanja v kontekstnih mrežnih grafih (t. i. semantično indeksiranje) [48], ki ga uporablja tudi Appleseed. Izvorno vozlišče se aktivira z dodatkom energije e, ki se razširi na sosednja vozlišča po določenih zakonitostih(5). Vsaka povezava $(x, y) \in E \subseteq V \times V$ ima določeno utež $W(x, y)$, $W : E \mapsto [0, 1]$ in večja je vrednost, večji delež energije e gre skozi to povezavo. Faktor širjenja $d \in [0, 1]$ določa delež pritoka $\mathrm{in}(x)$ energije v vozlišče x, ki je enako $d \cdot \mathrm{in}(x)$, medtem ko se $(1 - d) \cdot \mathrm{in}(x)$ energije porazdeli med naslednike x. Normalizacija uteži določa količino energije $e_{x \to y}$, ki se porazdeli od vozlišča x preko povezave (x, y) na vozlišče y; odvisno je od relativne teže $W(x, y)$ glede na vsoto uteži vseh odhodnih povezav vozlišča x (2.11):

$$e_{x \to y} = d \cdot \mathrm{in}(x) \cdot \frac{W(x, y)}{\sum_{(x,s) \in E} W(x, s)} \tag{2.11}$$

Vhod v model predstavlja seme zaupanja s iz množice vseh vozlišč (entitet) V, drugi parameter je dodatek oz. "injekcija" zaupanja $e \in \mathbb{R}_0^+$, tretji parameter je faktor širjenja $d \in [0, 1]$, četrti parameter je prag natančnosti T_C, ki služi za kriterij konvergence. Izhod algoritma predstavlja funkcija zaupanja na domeni V [14] (2.12):

$$\mathrm{Trust}_A : V \times \mathbb{R}_0^+ \times [0, 1] \times \mathbb{R}^+ \mapsto (\mathrm{trust} : V \mapsto \mathbb{R}_0^+) \tag{2.12}$$

Appleseed deluje na delnem grafu entitet – do vozlišč se dostopa samo v primeru, ko jih doseže tok energije e. Za vsako vozlišče x, ki ga doseže tok, algoritem v vsaki iteraciji izračuna

(5) Model je blizu realnemu principu širjenja energije skozi omrežja [14].

količino vhodne stopnje zaupanja z izrazom (2.13) [14]. Vhodni tok je določen s skupnim tokom predhodnih vozlišč p, ki ga porazdelijo po sosednjih povezavah (p, x).

$$\mathrm{in}(x) = d \cdot \sum_{(p,x)\in E} \left(\mathrm{in}(p) \cdot \frac{W(p,x)}{\sum_{(p,s)\in E} W(p,s)} \right) \tag{2.13}$$

Model *PageRank* [49, 9] se uporablja za vrednotenje zadetkov pri iskanju spletnih strani na osnovi ugleda strani; ta model uporablja spletni iskalnik *Google*(6). Ugled se izračuna na podlagi števila vhodnih in izhodnih hiperpovezav (angl. hyperlinks). Rezultat spletne strani se določi kot vsota normaliziranih uteži vhodnih povezav; normalizirana utež povezave je določena kot kvocient ugleda strani, ki vsebuje povezavo, in številom vseh povezav iz spletne strani. Naj P predstavlja množico povezanih spletnih strani; $a, b \in P$ predstavljata elementa oz. strani znotraj množice. Naj bo $N^-(a)$ množica strani, ki kažejo na stran a in $N^+(b)$ množica strani, na katere kaže stran b. Naj bo v ustrezen začetni vektor čez množico P, tako da velja $\sum_{x\in P} v(x) = 1$. Potem je *rang* (t. i. ugled) strani a določen kot [49] (2.14):

$$R(a) = d \cdot v(a) + (1-d) \cdot \sum_{b\in N^-(a)} \frac{R(b)}{|N^+(b)|} \tag{2.14}$$

Za konstanto d se predlaga vrednost $d = 0.15$ [49]. Opisani princip se lahko označi kot model zaupanja, ker množica povezav tvori tranzitivne verige zaupanja in predstavlja osnovo za izpeljavo merila zaupanja vsake spletne strani. Posamezna povezava na spletno stran se lahko pojmuje kot relacija zaupanja med izvorno in ciljno spletno stranjo. Omenjeni princip se lahko uporabi za izračun stopnje zaupanja in ugleda v socialnih mrežah.

2.6.1.5 Modeli teorije iger

Ti modeli temeljijo na principu kooperativnih interakcij med racionalnimi entitetami v ekonomskem smislu [41]. To pomeni, da entitete izbirajo med množico različnih opcij (strategij), kjer vsaka prinaša določeno korist (angl. utility). Cilj je izbrati take strategije, da je korist (angl. utility outcome) vedno optimalna. To načelo prinaša teorija iger, kjer za vsak tip igre obstaja ravnotežje (angl. equilibrium) – npr. *Nashevo ravnotežje* [50], ki določa strategije za optimalen izid ob predpostavki, da igralci (entitete) igrajo racionalno. Korist igralca je določena tako z njegovo lastno strategijo, kot tudi istočasno s strategijo nasprotnika. *Strateška igra* je sestavljena iz množice igralcev $\mathcal{N} = \{1, 2, \ldots, n\}$, za vsakega igralca i obstaja čista *strategija* $S_i \in \mathcal{S}$ in za vsakega igralca i obstaja *funkcija koristnosti* (angl. utility function) za vsako kombinacijo strategij – $u_i : \mathcal{S} \mapsto \mathbb{R}$, kjer je $\mathcal{S} = \times_{i\in\mathcal{N}} S_i$. Igra se lahko predstavi na dva načina [41]:

(a) *Normalna* oblika je običajno predstavljena z matriko izidov (angl. payoff matrix), ki prikazuje igralce, strategije in izide (korist). Za vsakega igralca določa vse možne strategije

(6) http://www.google.com/.

in njihov izid. Ponavadi matrika predstavlja 2 igralca; igralec 1 je predstavljen na vrsticah matrike, igralec 2 pa na stolpcih. Vrstice matrike predstavljajo možne strategije igralca 1, stolpci možne strategije igralca 2, celice matrike pa izid ob izbrani kombinaciji strategij.

(b) *Razširjena* oblika je predstavljena kot drevo. Vsako vozlišče predstavlja odločitveno točko za igralca. Igralec je določen s številko (oznako) vozlišča. Povezave iz vozlišča predstavljajo možne strategije za izbranega igralca. Končni izidi za posamezne kombinacije strategij so predstavljene kot listi drevesa. Ta oblika je primerna za igre z nepopolnimi (angl. incomplete) informacijami.

Obstajajo različni tipi strateških iger [41, 51]: (a) *simultane* (angl. simultaneous), kjer igralci izbirajo strategije istočasno in samo predvidevajo o strategijah nasprotnika (npr. igra *zaporniška dilema*) in (b) *zaporedne* (angl. sequential), kjer igralci izbirajo strategije v določenem zaporedju. Ločimo tudi *enkratne* (angl. one-shot) igre in *ponavljajoče* (angl. repeated) igre, kjer se igra lahko ponovi končno ali neskončno krat. Glede na popolnost informacij o igri, igralcih, strategijah in izidih ločimo sledeče tipe [41]:

- *Informacijsko dovršena igra* (angl. complete information game): igralec, ki je na izbiri strategije (potezi) ima vsaj toliko informacij, kot njegov nasprotnik. Obstajata dve vrsti teh iger:
  - *popolna igra* (angl. perfect information game): igralec pozna celotno zgodovino strategij (potez) in izidov igre;
  - *nepopolna igra* (angl. imperfect information game): igralec pozna le izide, ne pa tudi strategije nasprotnika.
- *Informacijsko nedovršena igra* (angl. incomplete information game): igralec, ki izbira strategijo (potezo) ima manj informacij kot njegov nasprotnik.

Primer informacijsko nedovršenih iger so *Bayesove* igre, ki se uporabljajo za modeliranje kooperativnih interakcij (npr. e-trgovanje, eBay[(7)]); posledica teh interakcij je efekt ugleda med agenti oz. igralci [52]. Bayesove igre so kombinacija teorije iger in verjetnostne teorije, kjer so prisotne nepopolne informacije o igralcih. Vsak igralec ima svoje privatne informacije (t. i. svoj tip obnašanja), ki lahko vplivajo na izid igre. Drugi igralci poznajo tip obnašanja drugih igralcev le z določeno verjetnostjo (verjetnostna porazdelitev na podlagi Bayesovega pravila). Igralci igrajo optimalno (dosežejo t. i. Bayes-Nashevo ravnotežje) glede na vse svoje tipe obnašanja in verjetnost poznavanja tipov obnašanja drugih igralcev. Primer ponavljajočih Bayesovih iger za modeliranje interakcij (posledično ugleda) spletišča eBay je podal P. Nurmi [44] in z eksperimenti dokazal, da je bolj učinkovit kot obstoječi eBay sistem za OU.

K. Aberer [51] teoretično opisuje kako uporabiti področje teorije iger za modeliranje ugleda in njihovo uporabo v skupnostih za elektronsko licitiranje in trgovanje z namenom povečati

(7) Gre za interakcije med enim velikim igralcem (prodajalec) in populacijo majhnih igralev (kupci).

zaupanja vredno obnašanje entitet. Gre za ponavljajoče strateške modele iger z nepopolnimi informacijami. Za učinkovito modeliranje stopnje zaupanja in ugleda, je potrebno identificirati vzorec interakcij med entitetami, določiti pravi model igre, povratne informacije in način njihove agregacije. Ti koraki so zelo odvisni od mnogih dejavnikov in jih je težko optimalno določiti.

Pri modeliranju ugleda pri ekonomskih interakcijah je pomemben pogoj *zaporedno ravnotežje* (angl. sequential equlibrium) [53], ki določa izbiro optimalne strategije kupca glede na prejšnjo interakcijo in prepričanja (verjetnosti) kupca o obnašanju prodajalca. To ravnotežje predvideva, da so prepričanja kupcev zaporedoma racionalna in posledično konvergirajo k pravilni verjetnostni porazdelitvi tipov obnašanj prodajalca in tako določajo njegov ugled (Faingold in Sanikov [52] sta npr. podala zaporedno ravnotežje kot rešitev dveh diferencialnih enačb). Ta model temelji na zaporedni racionalnosti v ponavljajočih Bayesovih igrah [54].

Drugi pristop pri modeliranju ugleda je *evolucijska teorija iger*, ki jih je preučeval R. Axelrod [55]. Večina pristopov temelji na iterativni igri *zaporniške dileme* (angl. iterated prisoner's dilemma – IPD) [55, 56, 51, 57]. Igralca, ki igrata to igro, imata na voljo dve strategiji: *sodelovanje* (S) (angl. cooperate) ali *goljufanje* (G) (angl. defect). Če oba sodelujeta, imata enako korist, medtem če eden sodeluje, drugi pa goljufa, ima večjo korist tisti, ki goljufa. Igra temelji na teoriji, da ima igralec motivacijo sodelovati (strategija S) le v primeru, če ne-sodelovanje (nasprotna strategija G) povzroči v prihodnosti izgubo koristi (trenutna uporaba strategije G ima večjo korist) [44]. Igro PD lahko predstavimo z normalno obliko (matriko izidov) – tabela 2.1, kjer je $\mathcal{N} = \{1, 2\}$, $\mathcal{S} = \{\mathrm{S}, \mathrm{G}\}$, $u_1(\mathrm{S}, \mathrm{S}) = u_2(\mathrm{S}, \mathrm{S}) = U_1$, $u_1(\mathrm{S}, \mathrm{G}) = u_2(\mathrm{S}, \mathrm{G}) = U_2$, $u_1(\mathrm{G}, \mathrm{S}) = u_2(\mathrm{G}, \mathrm{S}) = U_3$, $u_1(\mathrm{G}, \mathrm{G}) = u_2(\mathrm{G}, \mathrm{G}) = U_4$. Med

	S	G
S	U_1, U_1	U_2, U_3
G	U_3, U_2	$\underline{U_4, U_4}$

Tabela 2.1: Matrika izidov za igro *Zaporniška dilema.*

vrednostmi za izide (koristi) veljajo sledeče omejitve [55, 56]: $U_3 > U_1 > U_4 > U_2$ in $2U_1 > U_2 + U_3$ (npr. $U_1 = 5, U_2 = 0, U_3 = 6, U_4 = 1$). V igri obstaja eno Nashevo ravnotežje in to je, ko oba igralca izvedeta strategijo G (goljufija). To pa ni po pričakovanju socialnega izida, kjer oba igralca sodelujeta (strategija S). PD je torej abstrakcija socialne situacije, kjer pomeni sodelovanje S socialno odgovornost, goljufija G pa trenutno maksimalno korist, ne glede na to, kakšno škodo pomeni to za nasprotnika. Korist za oba igralca je torej največja, če oba goljufata in se ne ozirata na nasprotnika. Vendar je socialna korist največja, ko oba igralca sodelujeta. Iz teh situacij torej izhaja dilema ali sodelovati ali ne.

PD je primer igre za proučevanje kratkoročnega (trenutnega) odločanja. Izpostaviti je potrebno pojem iracionalnosti (teorija iger temelji na racionalnosti igralcev). Če bi bila oba igralca v popolnosti racionalna, ne bi nikoli sodelovala. Racionalni proces odločanja pomeni, da igralec sprejme odločitev, ki je zanj najboljša, ne glede na odločitev nasprotnika. Sodelovanje je

potemtakem iracionalna strategija, kljub temu, da korist za posameznika ni maksimalna. Pojavi se problem, kakšno je najustreznejše razmerje med racionalnimi in iracionalnimi strategijami igralcev, da se doseže dober ugled.

Ponavljajoča (angl. repeated) PD z enim stalnim (angl. long-run) igralcem (npr. prodajalec) proti populaciji enkratnih (angl. short-run) igralcev (npr. kupci), modelira kooperacijo med igralci v okoljih e-poslovanja (npr. e-trgovina, eBay, okolja P2P) [51]. Stalni igralec skozi igro pridobiva ugled glede na strategije, ki jih igra; torej je ugled oz. obnašanje igralca evolucija njegovih strategij. Če pogledamo enkratno igro PD s stališča racionalnosti igralcev, lahko ugotovimo sledeče: če prvi igralec goljufa, v odgovor goljufa tudi drugi; če prvi igralec sodeluje, drugi posledično goljufa, ker ima večjo korist. V splošnem ima drugi igralec vedno težnjo, da goljufa. Tudi za prvega igralca je boljše, da goljufa in se s tem posledično izogne "izdaji" drugega. Potemtakem, če se oba igralca odločita za goljufijo, ne glede na odločitev drugega, vedno končata v slabši situaciji, kot če bi oba sodelovala. V primeru ponavljajoče igre PD pa morata igralca odločati o trenutni in prihodnjih strategijah. Če igra igralec kratkoročno, ima še vedno motiv, da goljufa. Nasprotno pa igralec, ki igra dolgoročno, skuša s strategijo sodelovanja zgraditi boljši ugled in zaupanje. V ta namen obstaja več determinističnih strategij igranja (npr. vzajemna strategija *milo za drago* "Tit for Tat" – TFT, *vedno sodeluj* "Always Cooperate" – ALLC , *vedno goljufaj* "Always Defect" – ALLD, Pavlov, Grim, ...) [55, 57, 58].

Analiza ponavljajoče igre PD kaže na močno povezavo med zaupanjem kooperativnimi akcijami (strategijami) ter kako se vedenje in akcije spremenijo ob ugotovitvi, da je nekdo (ne)vreden zaupanja. V splošnem lahko rečemo, da prisotnost popolnega nezaupanja povzroči eliminacijo kooperativnih akcij. Na drugi strani pa prisotnost zaupanja zmanjša strah za negativne posledice pri uporabi kooperativnih akcij.

Za modeliranje zaupanja in ugleda velikega igralca (npr. prodajalca), proti kateremu igra (t. i. izvaja interakcije) populacija majhnih igralcev (npr. kupcev), se lahko uporabijo ekonomske igre s ponavljanji (končno ali neskončno krat):

- Bayesove igre [44] z zaporednim ravnotežjem, kjer prepričanje oz. verjetnost majhnih igralcev o obnašanju velikega igralca konvergira k določeni vrednosti in tako določa njegov ugled.
- Zaporniška dilema [55] z uporabo različnih determinističnih strategij igranja.

2.6.2 Kvalitativni modeli

Ljudje oz. uporabniki mnogokrat raje ocenjujemo stvari z diskretnimi kot pa z zveznimi merami. Pri kvalitativnim modelih je stopnja zaupanja (tudi ugled) do ciljne entitete določena s kvalitativno opisno vrednostjo iz vnaprej določene diskretne množice, npr. *zaupanja vreden*, *zaupanja delno vreden*, *zaupanja nevreden*. Tudi model PGP npr. uporablja kvalitativne vrednosti (glej razdelek 2.5.2). Ponavadi obstaja tudi ustrezna preslikava v kvantitativne

(numerične) vrednosti za ustrezne računske algoritme; ti lahko vračajo kvalitativni ali kvantitativni rezultat. V večini primerov pa se za izračun uporablja hevristika.

Kvalitativni model [18, 13] izhaja iz socialne dimenzije zaupanja. Relacija zaupanja med izvorno entiteto A in ciljno entiteto B je predstavljena kot relacija $\omega_{A,B}$, stopnja zaupanja je predstavljena kot ocena (mnenje), ki ima lahko sledeče kvalitativne vrednosti s pripadajočimi numeričnimi vrednostmi: *zaupanja vreden* (1), *neodločen* (0) in *zaupanja nevreden* (-1). Relacija zaupanja je refleksivna (entiteta zaupa sama sebi, torej $\omega_{A,A}$), asimetrična (v splošnem $\omega_{A,B} \neq \omega_{B,A}$) in praviloma ni tranzitivna (če obstaja $\omega_{A,B}$ in $\omega_{B,C}$, ni nujno da obstaja $\omega_{A,C}$). Zaupanje v socialni mreži (skupnosti), ki vsebuje n entitet, je predstavljeno z *matriko zaupanja* M (2.15) z elementi $\omega_{i,j}$, ki določa stopnjo zaupanja entitete i do entitete j. Znak "–" pomeni, da relacija zaupanja ne obstaja ali pa ni poznana.

$$M = \begin{bmatrix} \omega_{1,1} & \omega_{1,2} & \cdots & \omega_{1,n} \\ \omega_{2,1} & \omega_{2,2} & \cdots" & \omega_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{n,1} & \omega_{n,2} & \cdots & \omega_{n,n} \end{bmatrix} \tag{2.15}$$

Poudariti je potrebno, da nad matriko ne veljajo enake operacije kot v linearni algebri. Vrstice predstavljajo zaupanje izbrane entitete do ostalih, stolpci pa predstavljajo zaupanje (t. i. *vektor zaupanja*) celotne skupnosti do izbrane entitete. Nad elementi matrike obstajajo operacije v obliki $\omega^+_{i,k} = op_i(\omega^-_{i,k}, \omega^-_{j,k})$, kjer je op_i prefiksni operator za izračun stopnje zaupanja entitete i in j do entitete k. Oznaka "$^-$" označuje vrednost pred operacijo, oznaka "$^+$" pa označuje rezultat operacije.

Obstajajo trije različni prefiksni operatorji: *optimistični* ($\uparrow$), *pesimistični* ($\downarrow$) in *uravnotežen* ($\leftrightarrow$). Rezultati operacij $\omega^+_{i,k}$ so določeni v posebni tabeli [13], v splošnem pa veljajo sledeče zakonitosti (2.16):

$$\uparrow_i (1, \omega_{j,k}) = 1,\ \downarrow_i (-1, \omega_{j,k}) = -1,\ \leftrightarrow_i (0, \omega_{j,k}) = 0,\ op_i(-, \omega_{j,k}) = - \tag{2.16}$$

Model omenja tri različne tipe skupnosti – *homogena*, *polno povezana* in *gruča zaupanja*. Najboljšo sliko dinamičnega zaupanja skozi čas dajejo operacije na polno povezani skupnosti. Rezultat operacij na vektorjih zaupanja dajejo stopnjo ugleda posamezne entitete in so lahko osnova za odločanje o interakcijah.

Obstajajo tudi drugačni pristopi pri modeliranju kvalitativnih vrednosti stopnje zaupanja. Ponavadi gre za uporabo različnih matematičnih struktur. Model podan v [59, 60] poleg kvalitativne metrike vpeljuje npr. tudi posebne algebrajske strukture – *rešetke* (mreže) (angl. lattice), ki so potrebne za samo metriko. Cilj je definirati model, ki se obnaša enako kot človeška percepcija zaupanja – *zdravorazumsko* (angl. common sense). Zaupanje je predstavljeno kot interval. Na začetku je prisotna popolna negotovost (angl. uncertainty), ker ni informacij o entiteti. Ta informacija je predstavljena kot interval $[0, 1]$ in določa maksimalno negotovost.

Sčasoma se zaradi pridobljenih informacij o entiteti negotovost (tudi interval) zmanjšuje. Zaupanje je potemtakem sestavljeno iz dveh delov [61]: (a) interval zaupanja, ki predstavlja negotovost in (b) informacija o zaupanju, ki predstavlja gotovost. Domena vrednosti zaupanja ima dve vrsti urejenosti: *urejenost zaupanja* $\preceq$ (angl. trust ordering) in *informacijska urejenost* $\sqsubseteq$ (angl. information ordering). Urejenost $a \preceq b$ pomeni, da vrednost b predstavlja več zaupanja kot vrednost a; urejenost $d \sqsubseteq e$ pa pomeni, da vsebuje vrednost e več informacije (je bolj natančna) kot vrednost d. Algebrajska struktura z obema vrstama urejenosti omogoča izpeljavo različnih matematičnih lastnosti [60].

Poglavje 3

Modeliranje zaupanja in ugleda

3.1 Zasnova modela

V prejšnjem poglavju je bil podan pregled obstoječih definicij zaupanja in ugleda, sistemov za njihovo obvladovanje ter pregled nekaterih modelov za izračun stopnje zaupanja in ugleda. Določene lastnosti so tudi izhodišče za formalizacijo našega modela zaupanja. V okviru tega poglavja bomo zgradili formalen model zaupanja, ki vsebuje tudi socialne koncepte pojmovanja zaupanja. Model predvideva, da ima izvorna entiteta kognitivne sposobnosti (racionalni uporabnik) in lahko izvaja socialne interakcije. Potrebno je rešiti več nalog: definirati osnovne pojme, določiti ustrezne dejavnike in dimenzije zaupanja, izpostaviti relacije zaupanja s poudarkom na tranzitivnosti, določiti jezik za opis politik zaupanja in njihovo verifikacijo. Potemtakem je potreben formalizem, ki omogoča zajem dinamičnih relacij zaupanja med entitetami z možnostjo formalnega sklepanja ter obvladovanje interakcij med tehnologijo in človeškim socialnim obnašanjem.

Model predstavlja generično ogrodje za obvladovanje zaupanja in ugleda v (porazdeljenem) računalniškem okolju s poudarkom na storitveno usmerjenih arhitekturah (temelji na predlaganih modelih v [62, 63]). Model predvideva uporabo kvalitativnih in kvantitativnih principov (glej poglavje 2, razdelek 2.6.2 in 2.6.1) za izračun stopnje zaupanja in ugleda. Model tudi ne dela eksplicitne razlike med OZ in OU (glej poglavje 2, razdelek 2.4.1), saj vsebuje elemente zaupanja in ugleda.

3.1.1 Dejavniki zaupanja

Predpogoj za formalizacijo modela je identifikacija vseh potrebnih dejavnikov, ki so posredno ali neposredno povezani s pojmom zaupanja. Zaupanje je kompleksen fenomen, ki izvira iz področja psihologije in sociologije, zato je potrebno faktorje iskati tudi na teh področjih. Iz predhodnih definicij zaupanja (glej poglavje 2, razdelek 2.3) lahko rečemo, da je zaupanje povezano s prepričanjem v poštenost, zanesljivost, sposobnost in pripravljenost entitete, ki se ji zaupa. Različne entitete imajo lahko različne vrste zaupanja v isto ciljno entiteto, kar nakazuje, da je zaupanje subjektivno.

Piaget [64] omenja, da je zaupanje manifestacija sklepanja in proces preudarnosti in ima svoj izvor v psihologiji. V obzir je potrebno vzeti določena dejstva in ugotovitve, ki so domena razvojne psihologije. Subjektov spomin npr. je v stalnem pogonu in ni statičen, njegov razum se začne zavedati samega sebe samo v primeru, ko je v kontaktu s fizičnim predmetom ali drugim subjektom. Subjekt verjame določenim dejstvom, kljub temu, da ne pozna pravih razlogov za to.

Podobno tudi Castelfranchi in Falcone predstavljata socialno-kognitivni model [19], kjer zaupanje ni samo subjektivna izkušnja, ampak kompleksni mentalni odnos do ciljne entitete b. Samo kognitivna izvorna entiteta a lahko zaupa entiteti b. Entiteta a pooblasti entiteto b za izvedbo specifične naloge. Odnos zaupanja temelji na vrednotenju entitete b (zmožnost, sposobnost, samozavest, vztrajnost), njenega obnašanja, situacije in verjetnih rezultatov. Vrednotenja in pričakovanja se lahko označijo kot prepričanja, ki temeljijo na določenih dejstvih in informacijskih virih. Model vsebuje 7 različnih prepričanj: zmožnost, razpoložljivost, odvisnost, izpolnitev, pripravljenost, vztrajnost in samozavest. Model eksplicitno navaja notranje in zunanje dejavnike – atribute zaupanja in posledično loči *notranje* in *zunanje* zaupanje. Notranji dejavniki temeljijo na prepričanjih v pripravljenost, vztrajnost, zmožnost, medtem ko zunanji predvsem na priložnostih in pogojih okolja. Odločitev glede zaupanja ciljni entiteti vsebuje stopnjo tveganja, ki ni nikoli določena z gotovostjo – je negotovost. Poglavitna komponenta odločitve zaupanja je sprejem in izpostavljanje tveganju. Zaupanje je *racionalno*, ko stopnja zaupanja ne presega stopnje zaznanega tveganja; v nasprotnem primeru je zaupanje *iracionalno* (zaupanje brez prepričanj – "slepo" zaupanje).

Podoben socialni pristop k zaupanju predstavljajo Zhu in drugi [12], kjer definirajo formalno ontologijo za prikaz relacij socialnega zaupanja med udeleženci znotraj storitveno usmerjenih arhitektur. Zaupanje je označeno kot prepričanje, socialni odnos in relacija. Podana je analiza ključnih dejavnikov pri formiranju relacij zaupanja med agenti in relacije med posameznimi faktorji. Zaupanje sestavlja pet dimenzij: samozavest, vztrajnost, poštenost, motivacija in ugled.

Mui in drugi [58] govorijo o konceptu vzajemnosti (angl. reciprocity) znotraj zaupanja in ugleda v socialnih mrežah. Koncept je predvsem predmet raziskovanja razvojnih biologov. Vzajemnost predstavlja osnovo za socialno sodelovanje – izhaja iz osnovnega pravila "bodi dober do tistih, ki so dobri s tabo". Pravilo vzajemnosti (angl. reciprocity norm) je pogosto v večkratnih interakcijah z istimi entitetami; posamezniki se odzivajo na pozitivne akcije pozitivno in na negativne akcije negativno. Obstajajo različne študije vzajemnih strategij, predvsem na področju teorij iger (npr. igra *zaporniške dileme* – strategija "milo za drago" (angl. tit-for-tat) [55]) (glej poglavje 2, razdelek 2.6.1.5). Mui [58] definira vzajemnost kot medsebojno izmenjavo dejanj (usluga ali maščevanje). Njegov model predvideva, da povečanje ugleda pomeni povečanje zaupanja, povečanje zaupanja poveča verjetnost za pozitivno vzajemnost nasprotne entitete. Pozitivna vzajemnost hkrati pomeni tudi povečanje ugleda (povratna zanka).

Glede na predstavljene socialne koncepte pri pojmovanju zaupanja, je potrebno identificirati

ključne dejavnike, ki so z zaupanjem povezani posredno ali neposredno[(1)]. Ti dejavniki so v osnovi sledeči [62, 63]:

- *Racionalnost in iracionalnost.* Potrebno je upoštevati, da se kognitivna entiteta (agent) lahko pod določenimi pogoji obnaša racionalno, pod drugimi pa iracionalno. Ta dejavnik govori o tem, da pri modeliranju zaupanja ne smemo upoštevati entitet kot racionalnih igralcev v ekonomskem smislu teorije iger (glej poglavje 2, razdelek 2.6.1.5).

  Racionalno oz. iracionalno obnašanje agentov je odvisno od različnih dejavnikov, ki temeljijo na različnih prepričanjih, dejstvih, izkušnjah, vplivih okolja, itd. Te dejavnike lahko razvrstimo na notranje (pogojeni z lastnostjo agenta) in zunanje (pogojeni z okoljem); podobno razvrstitev dejavnikov zaupanja srečamo tudi pri socialno-kognitivnem modelu zaupanja, ki sta ga predstavila Castelfranchi in Falcone [19].

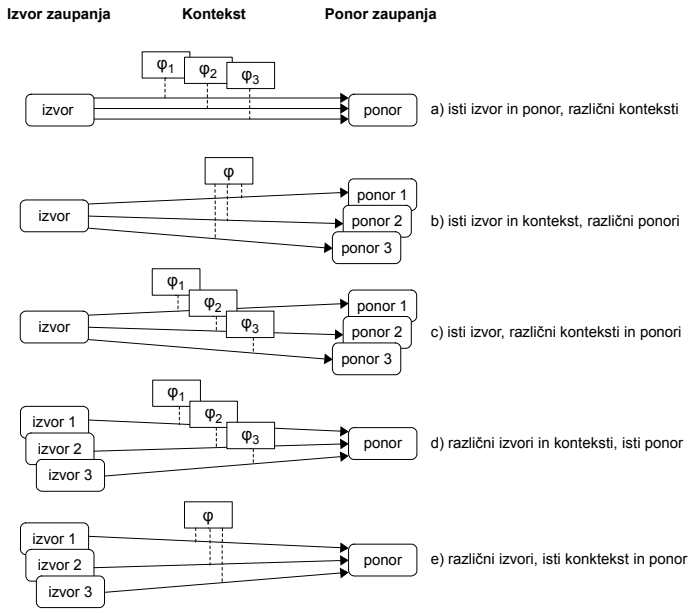
- *Interakcije z okoljem.* Zaupanje predstavlja osnovo za dejanja oz. interakcije agenta z okoljem v specifičnem kontekstu. Obenem pa tudi okolje povratno vpliva na formiranje zaupanja agenta za izbrani kontekst. Tu gre predvsem za akcije, mnenja drugih agentov in dejstva iz okolice, kar lahko povzroči spremembo obnašanja in mnenja o zaupanju pri izbranem agentu. Potemtakem lahko rečemo, da zaupanje omogoča interakcije (akcije) z okoljem, istočasno pa okolje oz. pridobljena izkušnja interakcij vpliva nazaj na oblikovanje zaupanja (t. i. povratna zanka).

- *Časovna dinamika.* Interakcije (transakcije) med entitetami se dogajajo znotraj prostora in časa. Zaupanje med entitetama oz. agentoma a in b v določenem časovnem trenutku je lahko precej drugačno kot isto zaupanje po nekaj interakcijah med a in b. Torej se zaupanje dinamično spreminja s časom; velja, da so novejše interakcije pomembnejše (imajo večjo težo) od starejših. Zaupanje je dinamični fenomen, ki se razvija v času in ima zgodovino [19]. Čas lahko modeliramo s strogo urejeno diskretno množico T časovnih vrednosti $t_i \in T$ ($i \in \mathbb{N}$, $t_i < t_{i+1}$). V splošnem vrednost $t_i \in T$ določa trenutek, ko pride do neke interakcije (t. i. časovna značka); množica $T_{a,b} \subseteq T$ vsebuje samo časovne vrednosti za vse pretekle interakcije med entitetama a in b.

- *Kontekst.* Relacija zaupanja med dvema entitetama se vedno nanaša na določen namen, cilj, področje (domeno); npr. "biti verodostojen" v primeru zaupanja v kriptografski ključ, "zagotavljanje zanesljive storitve" v primeru ponujanja storitev, "biti dober mehanik" v primeru popravila avtomobila. Vsaka relacija zaupanja med izvorom in ponorom ima določen kontekst φ; slika 3.1 prikazuje možne tipe odvisnosti.

  Kontekst definira obseg (področje, domeno) interakcij med entitetami. Kontekst lahko označimo kot množico opazovanih dogodkov (dejstev, situacij, področij), ki se zgodijo v

(1) Kashyap [65] je npr. v klasifikaciji dejavnikov zaupanja izpostavil, da let-ti vplivajo na formiranje zaupanja bodisi *neposredno*, *posredno* ali *tranzitivno*.



Slika 3.1: Možne odvisnosti med kontekstom, izvorom in ponorom.

okviru interakcije. To množico opišemo z ustrezno strukturo (t. i. zaporedje dogodkov), ki jo označimo kot φ in določa opis konteksta interakcij. Te strukture so elementi diskretne množice Φ, ki vsebuje vse možne kontekste oz. posamezne elemente $\varphi \in \Phi$.

- *Pretekle izkušnje (znanje).* Trenutno zaupanje med entitetama a in b je odvisno od izkušenj (znanja), ki jih ima entiteta a o b-ju; te izkušnje so rezultat preteklih interakcij. V primeru, da izkušnje ne obstajajo, se stopnja zaupanja lahko določi preko ugleda ciljne entitete v celotni skupnosti. Celotna izkušnja iz preteklosti oz. znanje se označuje kot $\lambda_{a,b}$ in se lahko smatra za kvantiteto vseh izkušenj, ki jih je entiteta a dobila s preteklimi interakcijami z izbrano ciljno entiteto b.

- *Tveganje.* Zaupanje je osnova za odločitve o prihodnjih interakcijah, te odločitve pa so povezane s tveganjem, ki jih lahko prinese pravilna/napačna odločitev. Vsaka odločitev je povezana s sprejemom in izpostavljanjem tveganju. Tveganje je večinoma izpuščeno iz modelov zaupanja. D. W. Manchala [66] je npr. razvil model na spremenljivkah zaupanja, kot sta strošek in zgodovina transakcij, vendar ne navaja eksplicitnih odvisnosti, ampak samo odločitveno matriko tveganje-zaupanje. V našem primeru se naslonimo na model, ki sta ga razvila A. Jøsang in S. Presti [17]. Model se osredotoča na transakcije (lahko jih enačimo z interakcijami) z dvema možnima izidoma – *uspešne* in *neuspešne*. Naj I označuje investicijo (denarno) v transakcijo. Vsaka transakcija ima ustrezen faktor – *faktor dobička* $G_s \in [0, \infty]$ za uspešne transakcije in *faktor izgube* $G_f \in [-1, 0]$ za neuspešne transakcije. Intervali vrednosti za faktorje se lahko interpretirajo v smislu, da je dobiček na investiciji I lahko poljubno velik (t. i. $G_s = \infty$), izguba pa je lahko največ enaka sami investiciji

(t. i. $G_f = -1$). Če je torej I investicija v transakcijo, potem je rezultat $I \cdot G_s$ v primeru uspešne transakcije in izguba $I \cdot G_f$ v primeru neuspešne transakcije.

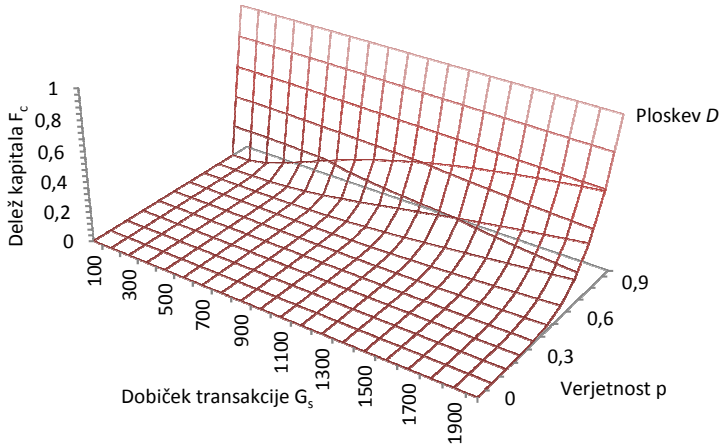
Ob predpostavki, da je p verjetnost uspeha, je *pričakovana korist* (angl. expected gain) enaka $EG = pG_s + (1-p)G_f = p(G_s - G_f) + G_f$. Racionalna entiteta (agent) vstopi v transakcijo, če je $I \cdot EG$ pozitiven. Parametri G_s, G_f in p torej določajo, kdaj je pričakovana korist EG pozitivna oz. negativna. V primeru, da je transakcija neuspela in faktor izgube $G_f = -1$, lahko iz ploskve ugotovimo točke, ko se $EG = p(G_s + 1) - 1$ prevesi v pozitivno vrednost.

Racionalni agent bo investiral v vsako transakcijo, vse dokler bo pričakovani dobiček pozitiven. Večja je verjetnost uspešne transakcije, večji delež svojega celotnega kapitala C je agent pripravljen tvegati. Naj bo $F_c \in [0, 1]$ *delež kapitala* C, ki ga je agent pripravljen investirati v transakcijo. Celotni vložek je tako enak $I = F_c \cdot C$. V splošnem se F_c povečuje s povečevanjem G_s ob konstantni verjetnosti p in podobno se F_c povečuje s povečevanjem p ob konstantnem G_s. F_c potemtakem lahko izrazimo kot funkcijo dveh spremenljivk – G_s in p. Odnos agenta do tveganja je določen s sledečo funkcijo (3.1) [17]:

$$F_c(p, G_s) = p^{\frac{\mu}{G_s}} \tag{3.1}$$

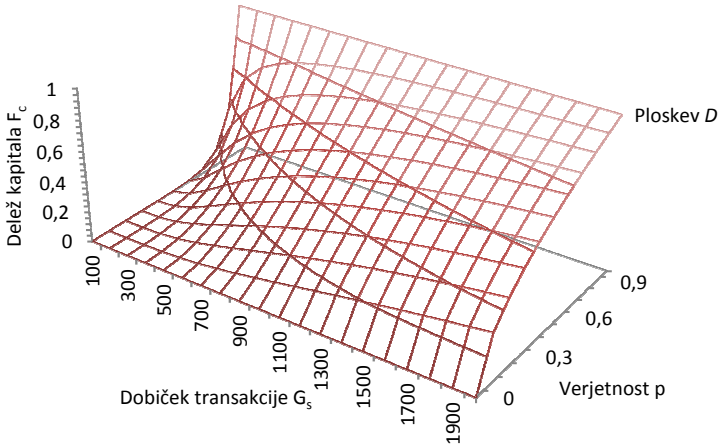
kjer je $\mu \in [1, \infty]$ faktor, ki določa v vpliv faktorja dobička G_s na delež kapitala, ki ga je agent pripravljen tvegati.

Nagnjenost k tveganju predstavlja *odločitvena ploskev* D s slike 3.2 ($\mu = 10000$) oz. slike 3.3 ($\mu = 1000$). Večja vrednost μ predstavlja *nenaklonjenost* k tveganju (zmanjšuje volumen pod ploskvijo D), medtem ko nižja vrednost μ predstavlja *nagnjenost* k tveganju (povečuje vrednost volumna pod ploskvijo D).



Slika 3.2: Nagnjenost k tveganju kot odločitvena ploskev ($\mu = 10000$).

Odnos do tveganja (naklonjenost, nenaklonjenost) je specifičen za vsakega agenta; posledično ima vsak agent svoj tip odločitvene ploskve (odvisnost od subjektivnih



Slika 3.3: Nagnjenost k tveganju kot odločitvena ploskev ($\mu = 1000$).

lastnosti, npr. preference, izkušnje, (i)racionalno obnašanje, različni konteksti transakcij). Posamezna transakcija je predstavljena kot 3-dimenzionalna točka na sliki 3.2 oz. 3.3 s koordinatami (G_s, p, F_c). Odločitvena ploskev D predstavlja nagnjenost k tveganju – agent bo po definiciji sprejel transakcijo, ki ima koordinate pod ploskvijo, in zavrnil transakcijo s koordinatami nad ploskvijo [17].

Predvidevamo, da je izid transakcije odvisen od ciljnega agenta b; predpostavka je, da verjetnost p pravzaprav predstavlja gotovost, da bo b ustvaril pozitiven izid transakcije. Torej p posredno predstavlja stopnjo zaupanja v b. Za vse kombinacije vrednosti G_s, p in F_c, ki so pod ploskvijo, izvorni agent a zaupa ciljnemu agentu b; v nasprotnem primeru, ko so vrednosti nad ploskvijo, pa b ni zaupanja vreden za izbrano transakcijo. Stopnja zaupanja do agenta b je odvisna od razdalje med trenutno situacijo in odločitveno ploskvijo; npr. v primeru, ko je G_s blizu 0 in F_c blizu 1, izvorni agent ne bo zaupal agentu b, kljub temu, da je vrednost p blizu 1. A. Jøsang [11, 17] enači verjetnost p kot *zaupanje v zanesljivost* (angl. reliability trust), hkrati pa definira *odločitveno zaupanje* $\mathcal{T}_D \in [-1, 1]$, ki je normalizirana razlika med p in prirejeno verjetnostjo p_D na odločitveni ploskvi D. Zaupanje $\mathcal{T}_D$ se interpretira glede na razdaljo $\delta = p - p_D$ ob izbranem faktorju dobička transakcije G_s (3.2).

$$\mathcal{T}_D(G_s) = \begin{cases} \frac{p-p_D}{p_D}, & \delta < 0 \quad (a \text{ ne zaupa } b\text{-ju za transakcijo}) \\ 0, & \delta = 0 \quad (a \text{ je neodločen}) \\ \frac{p-p_D}{1-p_D}, & \delta > 0 \quad (a \text{ zaupa } b\text{-ju za transakcijo}) \end{cases} \tag{3.2}$$

- *Korist.* Entiteta si lahko od interakcije (transakcije), ki jo ima s ciljno entiteto, obeta določeno korist. Korist se nanaša tudi na tveganje; racionalna entiteta stopi v interakcijo le, če je korist pozitivna; v nasprotnem primeru ni interakcije. Funkcijo, ki določa korist entitete a za interakcijo, označimo z u_a (princip teorije iger – glej poglavje 2, razdelek 2.6.1).

- *Vzajemnost (recipročnost).* Vzajemnost se nanaša na interakcije, kjer se entitete odzivajo na pozitivne akcije pozitivno in na negativne akcije negativno. Gre za koncept vzajemnosti, ki je predvsem predmet študije razvojnih biologov. Trivers [67] opisuje model *vzajemnega altruizma*, ki pogojuje evolucijo sodelovanja. Model prikazuje podrobnosti psihološkega sistema, ki regulira človeški altruizem in vzajemnost; npr. vpliv prijateljstva, (ne)naklonjenosti, jeze, (ne)zaupanja, krivde, (ne)poštenosti, itd. Človek se lahko obnaša altruistično ali pa tudi ne; obnašanje je odvisno od ravnotežja dejavnikov v socialnem okolju.

  Pojem vzajemnosti različnih vrst oz. strategij je prisoten na področju teorije iger; predvsem na področju igranja t. i. *Zaporniške dileme* [55] (kooperativno sodelovanje), kjer izstopa strategija *milo za drago* (angl. tit-for-tat – TFT). Obstajajo različne različice TFT strategij (GTFT, CTFT, TF2T), kot tudi množica drugih (PAVLOV, GRIM, ALLC, ALLD) [57, 55]. Igralci gredo v sodelovanje večinoma samo v primeru, ko pričakujejo vzajemnost od svojih nasprotnikov, ali pa v primeru, ko se igralec javno zaveže za sodelovanje. Zaupanje določa, v kakšni meri je posameznik pripravljen sodelovati v vzajemnih interakcijah [68]. Upoštevanje principa vzajemnosti v socialni mreži spodbuja pridobivanje večjega ugleda in posledično zaupanja pri izvajanju vzajemnih interakcij.

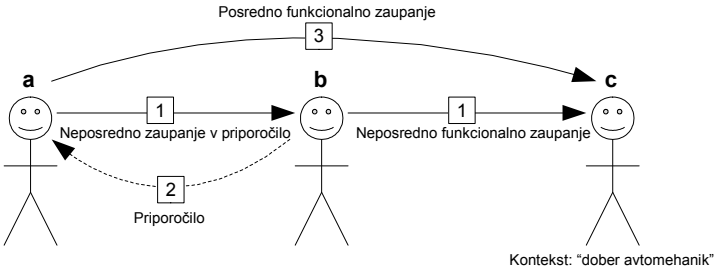
  L. Mui [58] v svojem modelu določa vzajemnost kot medsebojno izmenjavo dejanj (akcij). Predvideva dva tipa – *neposredno* (izmenjava akcij med dvema agentoma) in *posredno* (izmenjava akcij med dvema agentoma preko vmesnih agentov) vzajemnost. Faktor vzajemnosti $\gamma_{a,b} \in [0, 1]$ določa delež vzajemnih bilateralnih interakcij (akcij) med agentoma (entitetama) a in b; delež je določen kot celotno število pozitivnih (sodelovanje-sodelovanje) in negativnih (goljufija-goljufija) akcij glede na vse interakcije med agentoma a in b. Podobno se izraža tudi socialna vzajemnost kot delež vseh pozitivnih in negativnih akcij, glede na vse interakcije znotraj socialne mreže. Pozitivna vzajemnost pomeni povečanje ugleda in posledično stopnje zaupanja; le-ta pa poveča verjetnost za pozitivno vzajemnost (povratna zanka).

- *Stopnja zaupanja.* Stopnja zaupanja določa mero oz. vrednost, s katero izvorna entiteta zaupa ciljni entiteti. Ta vrednost je iz ustrezne množice, ki vsebuje diskretne ali zvezne vrednosti znotraj določenega intervala (glej poglavje o kvalitativnih in kvantitativnih modelih zaupanja – 2.6.2 in 2.6.1). To množico označimo kot domeno vrednosti $\mathcal{T}$, ki vsebuje relacijo popolne urejenosti. Stopnja zaupanja je rezultat računskega postopka (algoritma) znotraj modela zaupanja. Ker model zaupanja vsebuje tudi obvladovanje ugleda, je potrebno izpostaviti tudi množico vrednosti za ocene (angl. feedback), ki jih entitete podajo kot izkušnjo po končanih interakcijah. To popolno urejeno množico označimo kot Ω in lahko sovpada z množico $\mathcal{T}$ ($\Omega = \mathcal{T}$).

- *Neposredno in posredno zaupanje.* Relacije zaupanja se delijo na neposredne (direktne) in posredne (indirektne) relacije. Neposredno zaupanje se interpretira kot zaupanje "iz

prve roke" (neposredna izkušnja), posredno pa kot zaupanje, ki je izpeljano iz priporočil (nasvetov) drugih entitet [11, 69]. Posredne relacije zaupanja imajo različno dolžino l verige priporočil, kjer je $l > 0$, medtem ko pri neposrednih relacijah velja $l = 0$ [70].

Potrebno je razlikovati med zaupanjem v priporočilo (nasvet) in zaupanjem v samo sposobnost ciljne entitete (v okviru istega konteksta). Tako ločimo *zaupanje v priporočilo* (angl. recommendation trust) in *funkcionalno zaupanje* (angl. functional trust) [11]. S temi vrstami zaupanja lahko pod določenimi pogoji modeliramo *tranzitivnost* zaupanja. Tranzitivnost zaupanja pomeni, da če npr. entiteta a zaupa entiteti b, ta pa entiteti c, potem lahko a preko b-jevega priporočila zaupa entiteti c. Primer prikazuje slika 3.4, kjer je npr. kontekst zaupanja "dober avtomehanik". Posredno funkcionalno zaupanje, ki ga ima a do c, je izpeljano in ni rezultat neposredne izkušnje. Zaupanje v priporočilo vedno predvideva obstoj funkcionalnega zaupanja na zadnji relaciji tranzitivne verige [69, 71].



Slika 3.4: Princip tranzitivnega zaupanja.

3.1.2 Formalizacija

V tem razdelku formalno definiramo model zaupanja preko osnovnih gradnikov. V formalni model vpeljemo prej omenjene dejavnike (dimenzije) zaupanja (glej poglavje 3.1.1), ki jih razdelimo na *notranje* (subjektivne) in *zunanje* (okolje).

Glede na predstavljene dejavnike – dimenzije, vidimo, da je zaupanje kompleksna dinamična funkcija večih spremenljivk oz. dimenzij. Velja poudariti, da je ta funkcija subjektivna za vsakega agenta, saj je odvisna od mnogih psiholoških in socialnih dejavnikov – lahko jo imenujemo tudi *mentalna funkcija* [19]. Funkcijo je matematično težko modelirati, ker vsebuje veliko spremenljivk in veliko povratnih zank. V nadaljevanju sledi definicija posameznih gradnikov (od osnovnih do kompleksnih) modela zaupanja [63, 72].

Definicija 3.1.2.1 (Relacija zaupanja)**.** Zaupanje je relacija med entitetama (agentoma) a in b iz množice $\mathcal{A}$, ki se izraža kot dvomestna relacija $R \subseteq \mathcal{A} \times \mathcal{A}$. Relacija med a in b se zapiše kot aRb oz. $(a, b) \in R$.

Za relacijo zaupanja R je potrebno omeniti sledeče lastnosti:

- *Refleksivnost*: v splošnem relacija R ni refleksivna, ker entiteta ne zaupa sama sebi v vseh primerih oz. kontekstih (oseba a npr. ne zaupa sama sebi za izvedbo kirurške operacije, ampak zaupa b-ju, ki je kirurg); torej $\exists a \in \mathcal{A}, \neg(aRa)$.
- *Simetričnost*: relacija R večinoma ni simetrična, kajti če entiteta a zaupa entiteti b v določenem kontekstu, ni nujno, da velja tudi obratno; torej $\exists a, b \in \mathcal{A}, aRb \not\Rightarrow bRa$.
- *Tranzitivnost*: v splošnem relacija R ni tranzitivna; če prva entiteta zaupa drugi, druga pa tretji, ni nujno, da tudi prva entiteta zaupa tretji; torej $\exists a, b, c \in \mathcal{A}, aRb \wedge bRc \not\Rightarrow aRc$. Tranzitivnost velja samo pod določenimi pogoji [21] preko priporočil z neposrednimi in posrednimi relacijami zaupanja.

Definicija 3.1.2.2 (Graf zaupanja)**.** Množico entitet $\mathcal{A}$ z relacijo zaupanja $R \subseteq \mathcal{A} \times \mathcal{A}$, lahko predstavimo z usmerjenim grafom zaupanja $\mathbf{G}(\mathcal{A}, R)$, kjer elementi množice $\mathcal{A}$ predstavljajo vozlišča, elementi relacije R pa usmerjene povezave. Če $(a, b) \in R$ oz. aRb, to v grafu $\mathbf{G}$ predstavlja usmerjeno povezavo od vozlišča a do vozlišča b.

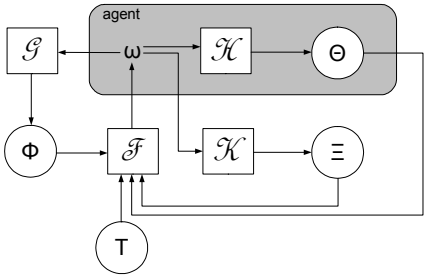
Števnost množice $\mathcal{A}$ je označena z n, števnost R pa z r. Velikost grafa $\mathbf{G}$ je enaka $n+r$. Naj bo $a \in \mathcal{A}$, potem lahko z $Indeg(a)$ označimo število vhodnih povezav do vozlišča a, z $Outdeg(a)$ pa število izhodnih povezav iz a. Povezavo med njima predstavlja sledeča enačba (3.3) [73]:

$$\sum_{a \in \mathcal{A}} Indeg(a) = \sum_{a \in \mathcal{A}} Outdeg(a) = r \tag{3.3}$$

Naj ω predstavlja oceno zaupanja agenta oz. entitete. Od dejavnikov zaupanja izpostavimo časovno dimenzijo T, kontekst Φ, notranje subjektivne dejavnike Θ (i/racionalnost agenta, pretekle izkušnje, vzajemnost, osebnostne lastnosti agenta), zunanje dejavnike okolja Ξ (mnenje drugih agentov, tveganje, korist). Naj bo $\mathscr{F}$ abstraktna dinamična subjektivna funkcija, ki določa trenutno zaupanje agenta (njegovo mnenje oz. oceno): $\mathscr{F} : T \times \Phi \times \Theta \times \Xi \mapsto \Omega$, kjer je Ω množica vrednosti ocen zaupanja. Naj bo $t \in T$ časovni trenutek interakcije, kontekst $\varphi \in \Phi$ (predstavlja dejstva, dogodke, situacije, področje znotraj interakcije), $\widetilde{\theta}$ zaporedje subjektivnih dejavnikov $\theta_1, \ldots, \theta_m$ ($\theta_i \in \Theta, 1 \leq i \leq m$) ter $\widetilde{\xi}$ zaporedje zunanjih dejavnikov $\xi_1, \ldots, \xi_n$ ($\xi_i \in \Xi, 1 \leq i \leq n$). Potemtakem se ocena zaupanja izraža kot rezultat abstraktne dinamične funkcije (3.4):

$$\omega = \mathscr{F}(t, \varphi, \widetilde{\theta}, \widetilde{\xi}) \tag{3.4}$$

Obstajajo tudi povratne odvisnosti med oceno zaupanja ω in posameznimi dejavniki. Odvisnost med kontekstom oz. njegovimi akcijami je določena s funkcijo $\mathscr{G}$ tako, da $\varphi = \mathscr{G}(\omega)$. Odvisnost subjektivnih dejavnikov je določena s funkcijo $\mathscr{H}$, kjer je $\widetilde{\theta} = \mathscr{H}(\omega)$, odvisnost zunanjih dejavnikov pa s funkcijo $\mathscr{K}$, kjer je $\widetilde{\xi} = \mathscr{K}(\omega)$. Medsebojne odvisnosti med oceno zaupanja in posameznimi dimenzijami oz. dejavniki prikazuje slika 3.5. Funkcija $\mathscr{H}$ je sestavljena iz množice funkcij za vsak subjektivni dejavnik θ_i: $\mathscr{H} = \{\mathscr{H}_1, \ldots, \mathscr{H}_m\}$, kjer je $\theta_i = \mathscr{H}_i(\omega), 1 \leq i \leq m$. Tudi funkcija $\mathscr{K}$ je sestavljena iz množice funkcij za vsak zunanji dejavnik ξ_i: $\mathscr{K} = \{\mathscr{K}_1, \ldots, \mathscr{K}_n\}$, kjer je $\xi_i = \mathscr{K}_i(\omega), 1 \leq i \leq n$.



Slika 3.5: Modeliranje zaupanja.

Definicija 3.1.2.3 (Ocena zaupanja)**.** Element $(a, b) \in R$ ima subjektivno oceno zaupanja z možnimi vrednostmi iz množice ocen Ω. Ocena zaupanja je označena kot vrednost $\omega \in \Omega$ in je rezultat abstraktne dinamične funkcije $\mathscr{F}$ kognitivne entitete a v odnosu do ciljne entitete b. Ocena zaupanja ω se vedno nanaša na določen časovni trenutek $t \in T_{a,b}$ in kontekst $\varphi \in \Phi$. Oznaka $\omega_{a,b}(t, \varphi)$ določa subjektivno oceno funkcionalnega zaupanja entitete a do entitete b ob času t v okviru konteksta φ.

Definicija 3.1.2.4 (Množica ocen in stopenj)**.** Množica stopenj $\mathcal{T}$ in množica ocen zaupanja Ω sta *popolno (linearno) urejeni* množici z dvočleno relacijo $\leq$, ki je *antisimetrična*, *tranzitivna* in *strogo sovisna*, torej za $\forall x, y, z \in \mathcal{T}$ oz. Ω velja [74]:

- če $x \leq y$ in $y \leq x$, potem sledi $x = y$ (antisimetričnost),
- če $x \leq y$ in $y \leq z$, potem sledi $x \leq z$ (tranzitivnost),
- velja $x \leq y$ ali $y \leq x$ (stroga sovisnost).

Popolna urejenost pomeni, da je vsak par elementov iz $\mathcal{T}$ oz. Ω med sabo primerljiv. Iz pogoja za strogo sovisnost sledi tudi lastnost *refleksivnosti* (t. i. $x \leq x$). Sledi, da je popolna urejenost tudi *delna urejenost* (dvočlena relacija $\leq$, ki je refleksivna, antisimetrična in tranzitivna). Potemtakem je popolna urejenost lahko definirana kot delna urejenost z lastnostjo stroge sovisnosti. Ilustrativno to v našem primeru pomeni: če $\tau_1 \leq \tau_2$ ($\tau_1, \tau_2 \in \mathcal{T}$) oz. $\omega_1 \leq \omega_2$ ($\omega_1, \omega_2 \in \Omega$), potem τ_2 oz. ω_2 pomeni več zaupanja kot τ_1 oz. ω_1. Struktura popolne urejenosti $(\mathcal{T}, \leq)$ oz. $(\Omega, \leq)$ ima določene lastnosti in jo lahko opišemo s pojmom *mreže* (t. i. *rešetke*) (angl. lattice) [74, 75].

Definicija 3.1.2.5 (Mreža)**.** Urejenost $(L, \leq)$ (L predstavlja množico $\mathcal{T}$ oz. Ω) je *mreža*, kjer za vsak par elementov $\forall x, y \in L$ velja:

- množica $\{x, y\}$ ima *najmanjšo zgornjo mejo*[(2)] *(supremum)* $\sup(\{x, y\}) = \top$,
- množica $\{x, y\}$ ima *najvišjo spodnjo mejo*[(3)] *(infimum)* $\inf(\{x, y\}) = \bot$.

Časovna dimenzija omogoča spremljanje interakcij in posledično zaupanja skozi čas. Vsaka interakcija med entitetami mora nositi tudi informacijo o času.

(2) *Least Upper Bound (LUB).*
(3) *Greatest Lower Bound (GLB).*

Definicija 3.1.2.6 (Časovna dinamika)**.** Čas modeliramo kot strogo urejeno diskretno množico $T_{a,b}$ časovnih vrednosti $t_i \in T_{a,b}$ ($i \in [1, |T_{a,b}|]$, $t_i < t_{i+1}$), ki določajo trenutek interakcij med entitetama $a, b \in \mathcal{A}$. Indeks i določa zaporedno številko interakcije. Množica časovnih vrednosti vseh interakcij med entitetami v socialni mreži je določena kot $T = \bigcup_{a,b \in \mathcal{A}} T_{a,b}$.

Dimenzija konteksta Φ vsebuje množico posameznih situacij (področij). Posamezen kontekst $\varphi \in \Phi$ je definiran kot dvojica $\varphi = (Name, ES)$, kjer $Name$ določa ime (opis) konteksta, ES pa je posebna *dogodkovna struktura* (angl. event structure) [76, 77], ki določa interakcije na podlagi dogodkov – ti so lahko med seboj *odvisni* ali pa so *konfliktni* (se ne morejo zgoditi istočasno). Dogodkovna struktura je matematično ogrodje za opis kontekstno odvisnih zgodovin interakcij med entitetami in omogoča mehanizme sklepanja z različnimi temporalnimi logikami (npr. verifikacija politik zaupanja glede na opazovano zgodovino interakcij). Sledi definicija [76]:

Definicija 3.1.2.7 (Dogodkovna struktura)**.** Dogodkovna struktura je trojica $(E, \preceq, \bowtie)$, kjer je E množica dogodkov $e \in E$, $\preceq$ in $\bowtie$ pa sta dvomestni relaciji na množici E. Relacija $\preceq$ je *relacija odvisnosti* in delno ureja množico E. Relacija $\bowtie \subset E \times E$ je *konfliktna relacija*, je simetrična in irefleksivna z lastnostmi:

- $\{e' \in E \mid e' \preceq e\}$ je končna,
- če $e \bowtie e'$ in $e' \preceq e''$, potem $e \bowtie e''$

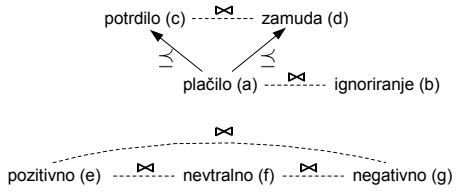
za vse $e, e', e'' \in E$. Pravimo, da sta dva dogodka *neodvisna*, če nista v nobeni od omenjenih relacij.

Kot primer lahko navedemo scenarij s spletnega mesta eBay, kjer obstajajo sledeči dogodki: (a) kupec plača izdelek z licitacije[(4)], (b) kupec ignorira licitacijo, (c) kupec prejeme potrdilo plačila, (d) kupec ne prejme potrdila plačila v določenem času, ter kupec poda (e) pozitivno, (f) nevtralno ali (g) negativno oceno o transakciji. Nekateri dogodki so lahko v konfliktu (pojav enega dogodka onemogoči pojav drugega); če kupec npr. poda pozitivno oceno (e), ne more podati tudi negativne (g). Nekateri dogodki pa so odvisni od drugih in se lahko zgodijo samo v primeru, če so se prej zgodili drugi dogodki; kupec npr. ne more dobiti potrdila o plačilu (c), če prej ni izvedel plačila (a). V primeru, da so dogodki neodvisni, se lahko zgodijo samostojno; npr. dogodki za podajanje ocen (e), (f), (g) in dogodek o prejemu potrdila (c) so neodvisni. Slika 3.6 prikazuje primer dogodkovne strukture za eBay protokol, gledano s strani kupca. Sledi definicija pojma konfiguracije, ki določa, kateri dogodki so možni znotraj dogodkovne strukture [76, 77].

Definicija 3.1.2.8 (Konfiguracija)**.** Naj bo $ES = (E, \preceq, \bowtie)$ dogodkovna struktura. Podmnožica dogodkov $\mathcal{X} \subseteq E$ je *konfiguracija*, če ima sledeče dve lastnosti:

1. *Konfliktno prosta*: za poljubna $e, e' \in \mathcal{X} : \neg(e \bowtie e')$ (t. i. $(e, e') \notin \bowtie$).

(4) V primeru, da je kupec ponudil najvišjo ceno na licitaciji oz. dražbi.



Slika 3.6: Primer dogodkovne strukture za scenarij eBay.

2. *Odvisnostno zaprta*: za poljubna $e \in \mathcal{X}, e' \in E : e' \preceq e \Rightarrow e' \in \mathcal{X}$.

$\mathcal{C}_{ES}$ vsebuje množico konfiguracij strukture ES, $\mathcal{C}^0_{ES} \subseteq \mathcal{C}_{ES}$ pa množico končnih konfiguracij. $\mathcal{C}_{ES}$ je *maksimalna*, če je maksimalna v delni urejenosti $(\mathcal{C}_{ES}, \subseteq)$.

Končna konfiguracija vsebuje informacije o posamezni interakciji, medtem ko maksimalna konfiguracija vsebuje celotno informacijo o interakciji. V primeru eBay (slika 3.6) je možna konfiguracija $\emptyset$, {`plačilo (a)`, `pozitivno (e)`}, maksimalna pa {`plačilo (a)`, `potrdilo (c)`, `pozitivno (e)`}. Pojem interakcije razširimo z oceno zaupanja $\omega \in \Omega$, ki jo entiteta a poda kot svojo izkušnjo s ciljno entiteto b.

Definicija 3.1.2.9 (Interakcija)**.** Interakcija med entitetama $a, b \in \mathcal{A}$ v določenem času $t \in T_{a,b}$ in kontekstu $\varphi \in \Phi$, je definirana kot dvojica:

$$x_{a,b}(t, \varphi) = (\mathcal{X}_{a,b}(t), \omega_{a,b}(t, \varphi)).$$

$\mathcal{X}_{a,b}(t) \in \mathcal{C}^0_{\varphi.ES}$ je končna konfiguracija v okviru konteksta φ, $\omega_{a,b}(t, \varphi) \in \Omega$ pa ocena zaupanja; torej je interakcija $x_{a,b}(t, \varphi) \in \mathcal{C}^0_{\varphi.ES} \times \Omega$.

S konfiguracijami lahko modeliramo zgodovino interakcij, ki služijo za odločitve in sklepanja glede prihodnjih interakcij preko interakcijskih politik zaupanja.

Definicija 3.1.2.10 (Zgodovina interakcij)**.** Naj bo ES dogodkovna struktura v okviru konteksta $\varphi \in \Phi$, potem je *zgodovina interakcij* sekvenca interakcij s končnimi konfiguracijami:

$$h_{a,b}(\varphi) = x_{a,b}(t_1, \varphi) x_{a,b}(t_2, \varphi) \ldots x_{a,b}(t_n, \varphi),\ x_{a,b}(t_i, \varphi) \in \mathcal{C}^0_{\varphi.ES} \times \Omega,\ t_i \in T_{a,b}.$$

Model zaupanja, ki ga formaliziramo, predvideva, da je entiteta a (npr. odjemalec storitve), sposobna podati oceno o ciljni entiteti b (npr. ponudniku storitev) po vsaki interakciji (transakciji). Preden a izvede interakcijo z b-jem, izračuna[5] stopnjo zaupanja v entiteto b glede na zgodovino interakcij. Obstaja tudi druga možnost: entiteta b ponuja npr. določeno storitev ali vir S. Ko želi entiteta a dostopati do S, pošlje zahtevo b-ju, ki izračuna svojo stopnjo zaupanja v a. Če je stopnja ustrezno visoka, je dostop dovoljen, drugače pa se zavrne. Prag zaupanja za dostop do S, je določen v politiki[6] zaupanja entitete b. Posledično naj bi vsaka

[5] Stopnjo izračuna sistem za obvladovanje zaupanja glede na zgodovino ocen.
[6] Se lahko smatra kot *interakcijska* politika.

entiteta imela svojo interakcijsko politiko, tako v primeru, da ponuja storitve/vire ali pa da do njih dostopa. Jezik za opis politike je deklarativen in lahko temelji na različnih variantah *temporalne logike* [76, 77, 78, 79]. V našem primeru je dimenzija konteksta definirana na osnovi dogodkovne strukture, ki omogoča konstrukcijo ustreznega jezika za opis interakcijskih politik.

Relacije zaupanja R znotraj omrežja entitet, so predstavljene kot graf interakcij (zaupanja) $\mathbf{G}$ (definicija 3.1.2.2). Graf predstavlja usmerjeni graf z entitetami kot vozlišči in povezavami kot interakcijami med njimi (definicija 3.1.2.10). Povezave so usmerjene in predstavljajo interakcije $x_{a,b}$ z ocenami zaupanja $\omega_{a,b}$; če povezava v grafu $\mathbf{G}$ označuje interakcijo med entitetama a in b v času $t \in T_{a,b}$ in kontekstu $\varphi \in \Phi$, je usmerjena od vozlišča a proti vozlišču b z oznako $x_{a,b}(t, \varphi)$, ki določa dvojico $(\mathcal{X}_{a,b}(t), \omega_{a,b}(t, \varphi))$. Omenjeni graf $\mathbf{G}$ lahko ekvivalentno predstavimo z matriko.

Definicija 3.1.2.11 (Matrika interakcij)**.** Interakcije znotraj omrežja z $n = |\mathcal{A}|$ entitetami[7] so predstavljene z matriko interakcij (zaupanja) $\mathbb{M}$ z elementi $x_{i,j} = (\mathcal{X}_{i,j}, \omega_{i,j}), i, j \in [1, n]$, $\omega_{i,j} \in \Omega$, $\mathcal{X}_{i,j} \in \mathcal{C}^0_{\varphi.ES}$. Elementi določajo relacijo zaupanja R i-te entitete do j-te entitete. V primeru, da relacija (še) ne obstaja, je označena kot $(\emptyset, -\infty)$[8]. Matrika predstavlja zaupanje v mreži entitet ob času $t \in T = \bigcup_{i,j \in |\mathcal{A}|} T_{i,j}$ in kontekstu $\varphi \in \Phi$ in je označena kot $\mathbb{M}(t, \varphi)$:

$$\mathbb{M}(t, \varphi) = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,n} \end{bmatrix} \tag{3.5}$$

Interakcije med entitetami so predstavljene kot elementi v hiperprostoru večih dimenzij v opazovanem času in kontekstu. Ti elementi so matrike interakcij $\mathbb{M}(t, \varphi), t \in T, \varphi \in \Phi$.

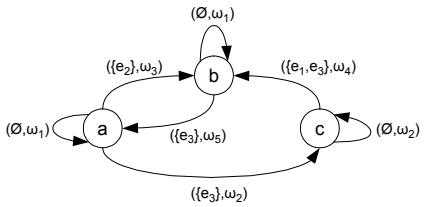
Izpostaviti velja, da je konfiguracija $\mathcal{X}_{a,b} \in \mathcal{C}^0_{ES}$ znotraj interakcije $x_{i,j}$ lahko tudi prazna množica. Diagonalni elementi $x_{i,i}$ so primer interakcij s praznimi množicami (v večini primerov entiteta zaupa sama sebi implicitno – brez posledice interakcije). Kljub temu, da relacija zaupanja R ni refleksivna, ponavadi lahko velja (odvisno od konteksta), da je ocena zaupanja entitete same vase maksimalna, torej $\omega_{i,i} = \sup(\Omega \cup \{-\infty\}) = \top$ oz. $x_{i,i} = (\emptyset, \top)$. Slika 3.7 prikazuje interakcije med tremi entitetami v določenem trenutku t in kontekstu $\varphi = (Name_1, ES_1)$, kjer je množica ocen $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4, \omega_5\}$, $\varphi.ES_1 = \{e_1, e_2, e_3\}$, množica konfiguracij pa $\mathcal{C}^0_{\varphi.ES_1} = \{\emptyset, \{e_2\}, \{e_3\}, \{e_1, e_3\}\}$. Pripadajoča matrika je sledeča (3.6):

$$\mathbb{M}(t, \varphi) = \begin{bmatrix} (\emptyset, \omega_1) & (\{e_2\}, \omega_3) & (\{e_3\}, \omega_2) \\ (\{e_3\}, \omega_5) & (\emptyset, \omega_1) & (\emptyset, -\infty) \\ (\emptyset, -\infty) & (\{e_1, e_3\}, \omega_4) & (\emptyset, \omega_2) \end{bmatrix} \tag{3.6}$$

Glede na definicijo 3.1.2.1, je relacija zaupanja (posledica interakcije) določena kot dvomestna relacija $R \subseteq \mathcal{A} \times \mathcal{A}$. Vrstice znotraj matrike interakcij predstavljajo subjektivno zaupanje

(7) V splošnem označujemo entitete s črkami $a, b, c, \cdots \in \mathcal{A}$ (npr. $\omega_{a,b}, x_{a,b}$), lahko pa jih označimo tudi z zaporednimi indeksi $i, j, k, \cdots \in [1, |\mathcal{A}|]$ znotraj matrike interakcij $\mathbb{M}$ (npr. $\omega_{i,j}, x_{i,j}$, kjer i pomeni indeks vrstice, j pa indeks stolpca).

(8) Če $x_{i,j} = (\emptyset, -\infty)$, to ne pomeni tudi $x_{j,i} = (\emptyset, -\infty)$.



Slika 3.7: Interakcije kot usmerjen graf.

izbrane entitete do drugih entitet; vrstice se lahko izrazijo kot *slika* entitete a, zapisano kot $a.R = \{b \in \mathcal{A} \mid (a,b) \in R\}$. Stolpci matrike predstavljajo zaupanje celotne socialne skupnosti do izbrane entitete; ti stolpci se imenujejo tudi *vektorji zaupanja* [13]. Definirani so kot pred-slika entitete b, zapisano kot $R.b = \{a \in \mathcal{A} \mid (a,b) \in R\}$.

Dimenzija preteklih izkušenj (znanje), ki jih ima izbrana entiteta a o ciljni entiteti b, se lahko izraža kot agregirana vrednost zaupanja skozi celotno zgodovino interakcij $h_{a,b}$ med a in b v okviru določenega konteksta. V splošnem entitete (agenti) spreminjajo obnašanje skozi čas, zato je smiselno upoštevati novejše interakcije z večjo težo kot starejše. V ta namen uporabimo ustrezni eksponentni utežni faktor interakcije $e^{-\alpha\Delta t}$, kjer Δt razlika med časom (trenutkom) opazovanja t' in trenutkom t pretekle interakcije $x_{a,b}(t,\varphi)$ ter $\alpha \in [0,\infty)$. V primeru, da je $\alpha = 0$, imajo vse interakcije enako težo. Podoben primer uteževanja ocen interakcij je predstavil tudi K. J. Lin [26, 22] v porazdeljenem modelu zaupanja in ugleda.

Agregirana stopnja zaupanja se lahko izračuna kot utežena vsota vseh ocen v zgodovini interakcij h. Operacija utežene vsote je možna na množici $\Re \subseteq \mathbb{R}^n, n \geq 1$, zato je potrebna ustrezna injektivna preslikava iz množice ocen[9] Ω v $\Re$ in nazaj.

Definicija 3.1.2.12 (Pretekla izkušnja – znanje)**.** Preteklo izkušnjo, ki jo ima i-ta entiteta do j-te entitete znotraj konteksta φ, se lahko posredno izraža kot utežena vsota ocen $\sigma_{i,j} \in \Re$ v zgodovini interakcij $h_{i,j}(\varphi)$ (glej definicijo 3.1.2.10) na sledeči način:

$$\sigma_{i,j}(t',\varphi) = \begin{cases} \frac{\sum_{t \leq t'} e^{-\alpha(t'-t)} f_\Omega(x_{i,j}(t,\varphi).\omega))}{m} & \text{če } x_{i,j}(t,\varphi) \neq (\emptyset, -\infty) \\ -\infty & \text{drugače} \end{cases} \tag{3.7}$$

kjer je $t' \in T$ čas opazovanja, $e^{-\alpha(t'-t)}$ je eksponentni utežni faktor z $\alpha \in [0,\infty)$, $f_\Omega : \Omega \cup \{-\infty\} \mapsto \Re$ je injektivna preslikava, $i,j \in [1,|\mathcal{A}|]$, m je število upoštevanih interakcij v zgodovini $h_{t \leq t'}(\varphi)$. Pretekla izkušnja $\lambda_{i,j}(\varphi) \in \Omega \cup \{-\infty\}$ se izračuna kot $\lambda_{i,j}(\varphi) = f'_\Omega(\sigma_{i,j}(t',\varphi))$, kjer je $f'_\Omega : \Re \mapsto \Omega \cup \{-\infty\}$ surjektivna preslikava.

Glede na to, da lahko izračunamo $\lambda_{i,j}$ za vse entitete znotraj množice $\mathcal{A}$, lahko definiramo novo *agregirano matriko* zaupanja $\mathbb{M}'$ z elementi $\lambda_{i,j}$ brez informacij o posameznih interakcijah. Matrika (3.8) predstavlja pretekle izkušnje med entitetami v socialni mreži znotraj določenega

[9] Predvsem v primeru kvalitativnih vrednosti.

konteksta, kjer je časovna dimenzija fiksirana na čas opazovanja t'.

$$\mathbb{M}'(t',\varphi) = \begin{bmatrix} \lambda_{1,1} & \lambda_{1,2} & \cdots & \lambda_{1,n} \\ \lambda_{2,1} & \lambda_{2,2} & \cdots & \lambda_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{n,1} & \lambda_{n,2} & \cdots & \lambda_{n,n} \end{bmatrix} \text{kjer } \lambda_{i,j} = f'_{\Omega}(\sigma_{i,j}(t',\varphi)). \tag{3.8}$$

Operacije na matrikah $\mathbb{M}$ niso enake kot v običajni linearni algebri. Namesto njih se uporabljajo *operacije zaupanja* (angl. trust operations). Operacija zaupanja je dvomestna in določena kot preslikava $\text{op} : \Omega \cup \{-\infty\} \times \Omega \cup \{-\infty\} \mapsto \mathcal{T} \cup \{-\infty\}$.

Definicija 3.1.2.13 (Operacija zaupanja)**.** Operacija zaupanja je definirana kot $\tau_{i,k}(\varphi) = \lambda_{i,k}(\varphi)\,\text{op}\,\lambda_{j,k}(\varphi)$, kjer $\tau_{i,j} \in \mathcal{T} \cup \{-\infty\}$, $\lambda_{i,j} \in \Omega \cup \{-\infty\}$, $i,j,k \in [1,|\mathcal{A}|]$, $\text{op} \in \mathcal{O}$ je operator, ki modelira zaupanje i-te entitete, medtem ko $\lambda_{i,k}(\varphi)$ in $\lambda_{j,k}(\varphi)$ določata preteklo izkušnjo zaupanja i-te in j-te entitete proti k-ti entiteti v kontekstu φ.

Navedene definicije (3.1.2.1–3.1.2.13) omogočajo, da lahko določimo abstraktni model zaupanja. Ta je splošen in lahko vsebuje tako kvalitativno kot tudi kvantitativno metriko. Abstraktni model vsebuje predstavitev mreže entitet, množico vrednosti za ocenjevanje in izračun stopnje zaupanja, nabor možnih operatorjev zaupanja, računski algoritem za izračun stopnje zaupanja vsake entitete ter množico interakcijskih politik, ki določajo pravila za dostop do storitev in virov.

Definicija 3.1.2.14 (Abstraktni model zaupanja)**.** Abstraktni model zaupanja je določen kot n-terica $ATM = (\mathbf{G}, \Omega, \mathcal{T}, \mathcal{F}, \mathcal{O}, \Gamma, \Pi)$, kjer je $\mathbf{G}$ usmerjeni graf $(\mathcal{A}, R)$ z $\mathcal{A}$ kot množico entitet (agentov) in R kot relacijo zaupanja med njimi. Ω je množica ocen zaupanja (t. i. parametrične vrednosti), ki jih podajo entitete po vsaki interakciji. $\mathcal{T}$ je množica stopenj zaupanja (t. i. ordinalne vrednosti), ki so rezultat operacij zaupanja. $\mathcal{F} = \{f_{\Omega}, f'_{\Omega}, f_{\mathcal{T}}, f'_{\mathcal{T}}\}$ je množica injektivnih in surjektivnih preslikav iz množic $\Omega \cup \{-\infty\}$ in $\mathcal{T} \cup \{-\infty\}$ v množico $\Re \subseteq \mathbb{R}^n (n \geq 1)$ in obratno. $\mathcal{O}$ je množica operatorjev za izvajanje operacij zaupanja na grafu $\mathbf{G}$. Γ je algoritem za izračun stopnje zaupanja $\tau \in \mathcal{T}$ za vsako entiteto znotraj grafa $\mathbf{G}$. Π je množica interakcijskih politik zaupanja z definiranimi pravili za dostop do storitev ali virov.

Model ATM predstavlja generično ogrodje za različne računske modele zaupanja (kvantitativni, kvalitativni). Možno ga je razširiti tudi z dodatnimi dejavniki oz. dimenzijami zaupanja. Npr. vpeljava odločitev o prihodnjih interakcijah na osnovi tveganja in stopnje zaupanja preko modela ATM, zahteva transformacijo stopenj zaupanja $\tau \in \mathcal{T}$ v verjetnost $p \in [0,1]$. Za vsako interakcijo oz. transakcijo, je potrebno izračunati odločitveno ploskev $F_c(G_s, p) = p^{\frac{\mu}{G_s}}$ (glej poglavje 3.1.1) na podlagi G_s (faktor dobička transakcije) in konstante μ (subjektiven odnos do tveganja), katerih vrednosti sta specifični za vsako entiteto.

3.1.3 Tranzitivnost

V mreži entitet $\mathcal{A}$, ki je predstavljena z usmerjenih grafom $\mathbf{G}$, se pojavi problem, kako izračunati stopnjo zaupanja med entitetama, ki nimata neposredne relacije zaupanja (t. i. ne obstaja usmerjena povezava med izbranima vozliščema). Obstajajo različne metode na podlagi različnih uteženih povprečij (normalizacija) [24, 80], iskanjem in kombiniranjem poti v grafu [81]. V našem primeru se osredotočimo na metodo tranzitivnosti zaupanja preko tranzitivnih verig v grafu $\mathbf{G}$. V osnovi zaupanje ni tranzitivno, kar dokazujejo tudi različne študije [82]. Z uporabo sistema priporočil, lahko dosežemo tranzitivnost zaupanja ob eksplicitnih pogojih [71, 69, 21].

Že predhodno (glej razdelek 3.1.1) smo določili, da je potrebno za tranzitivnost zaupanja ločiti med *funkcionalnim zaupanjem* in *zaupanjem v priporočilo*. Poleg tega je lahko vsako od teh zaupanj *posredno* ali *neposredno*. V našem modelu zaupanja ATM v osnovi upoštevamo samo funkcionalno zaupanje s stopnjo $\tau_{a,b}(t,\varphi) \in \mathcal{T} \cup \{-\infty\}$ (rezultat operacij zaupanja oz. algoritma Γ) oz. ocene le-tega $\omega_{a,b}(t,\varphi) \in \Omega \cup \{-\infty\}$. Za tranzitivno zaupanje je potrebno vpeljati tudi *zaupanje v priporočilo* z enako strukturo kot funkcionalno zaupanje (matrike interakcij, matrika izkušenj), vendar z medsebojno neodvisnostjo. Potreben pogoj je tudi, da je kontekst obeh zaupanj (funkcionalno, zaupanje v priporočilo) enak.

Definicija 3.1.3.1 (Zaupanje v priporočilo)**.** Zaupanje v priporočilo ima enako strukturo kot funkcionalno zaupanje; to pomeni vpeljavo elementov[10]: ocena zaupanja $\omega^r_{a,b} \in \Omega \cup \{-\infty\}$, interakcije $x^r_{a,b} = (\mathcal{X}^r_{a,b}, \omega^r_{a,b})$, matrike interakcij $\mathbb{M}^r$, pretekle izkušnje $\lambda^r_{a,b} \in \Omega \cup \{-\infty\}$, agregirano matriko izkušenj $\mathbb{M}'^r$ ter operacije zaupanja $\tau^r_{a,b} \in \mathcal{T} \cup \{-\infty\}$.

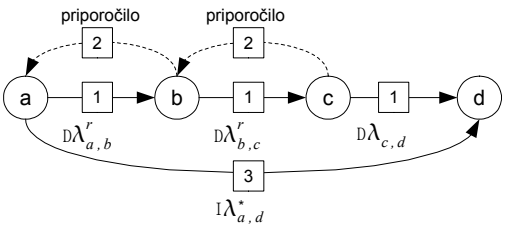
Preko sistema priporočil je mogoče izračunati izkušnjo (t. i. agregat) funkcionalnega zaupanja med izvorno in ciljno entiteto, čeprav entiteti nista bili nikoli v interakciji. Naj $\lambda_{i,j}$ označuje izkušnjo funkcionalnega zaupanja, $\lambda^r_{i,j}$ pa izkušnjo zaupanja v priporočilo. V nadaljevanju bomo zaradi preglednosti namesto besede "izkušnja zaupanja"[11] (določena z vrednostjo $\lambda_{i,j}$) uporabili samo besedo "zaupanje", čeprav pomenska razlika ostaja – izkušnja zaupanja predstavlja agregat ocen zaupanja (glej definicijo 3.1.2.12). Predpona "D" naj označuje izkušnjo direktnega (neposrednega) zaupanja, predpona "I" pa izkušnjo indirektnega (posrednega) zaupanja. Naj bodo $a, b, c, d \in \mathcal{A}$ s sledečima agregiranima matrikama izkušenj:

$$\mathbb{M}'(\varphi) = \begin{bmatrix} \top & \infty & -\infty & -\infty \\ -\infty & \top & \infty & -\infty \\ -\infty & -\infty & \top & \lambda_{c,d} \\ -\infty & -\infty & -\infty & \top \end{bmatrix} \quad \mathbb{M}'^r(\varphi) = \begin{bmatrix} \top & \lambda^r_{a,b} & -\infty & -\infty \\ -\infty & \top & \lambda^r_{b,c} & -\infty \\ -\infty & -\infty & \top & -\infty \\ -\infty & -\infty & -\infty & \top \end{bmatrix} \tag{3.9}$$

Tranzitivno verigo prikazuje slika 3.8 z izpeljavo izkušnje posrednega funkcionalnega zaupanja med entitetama a in d. Vrednost izkušnje posrednega funkcionalnega zaupanja je označena z

[10] Podvojeni elementi z indeksom r.
[11] Izkušnja funkcionalnega zaupanja, izkušnja zaupanja v priporočilo.

$\lambda^*_{a,d}$ (dodana zvezdica). Pri posrednih relacijah zaupanja lahko nadalje ločimo različne dolžine



Slika 3.8: Tranzitivna veriga zaupanja.

verige zaupanj v priporočila (angl. recommendation hops) $l > 0$ [70, 69]:

- $l = 1$: izvorna entiteta zaupa drugi entiteti, da ji lahko neposredno priporoči svoje funkcionalno zaupanje v ciljno entiteto (primer s slike 3.4).
- $l \geq 2$: izvorna entiteta neposredno zaupa drugi entiteti, da ji lahko priporoči tretjo entiteto, ki ji lahko priporoči svoje funkcionalno zaupanje v ciljno entiteto ($l = 2$). V splošnem ($l = n$) – izvorna entiteta zaupa drugi entiteti, da ji lahko priporoči tretjo entiteto, ki zopet lahko priporoči četrto entiteto, ta priporoči peto, ..., $n - 1$ entiteta priporoči n-to, ta pa lahko priporoči svoje funkcionalno zaupanje v ciljno entiteto. Opazimo lahko rekurzivno strukturo pri daljših tranzitivnih verigah.

Za izpeljavo posrednega funkcionalnega zaupanja morajo obstajati določeni pogoji, ki so podani v nadaljevanju.

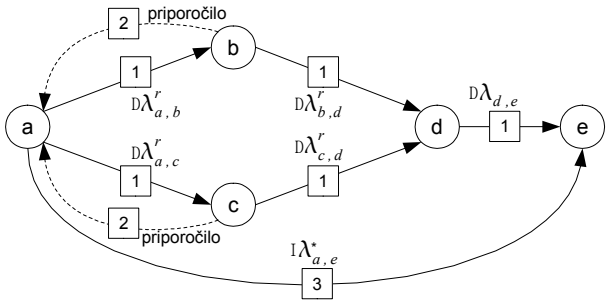
Kriterij 1 (Izpeljava funkcionalnega zaupanja)**.** Izpeljava funkcionalnega zaupanja preko zaupanja v priporočilo predvideva, da zadnja povezava (relacija) v verigi zaupanja predstavlja funkcionalno zaupanje, vse prejšnje povezave pa predstavljajo zaupanje v priporočilo.

Kriterij 2 (Enakost konteksta)**.** Tranzitivna veriga je veljavna samo v primeru, ko je kontekst funkcionalnega zaupanja enak kontekstu zaupanja v priporočilo.

Entiteta ima lahko istočasno funkcionalno zaupanje in zaupanje v priporočilo v isto ciljno entiteto. Velja, da se stopnja zaupanja zmanjšuje čez tranzitivno verigo. Obstaja velika razlika, ali entiteta priporoči zaupanje na podlagi neposredne izkušnje s ciljno entiteto (izkušnja interakcije), ali priporoči posredno zaupanje, ki je že rezultat priporočil drugih entitet.

Kriterij 3 (Zaupanje v priporočilo)**.** Entiteta v tranzitivni verigi lahko priporoči le neposredno zaupanje, ki je bodisi zaupanje v priporočilo ali pa funkcionalno zaupanje.

Izpeljano funkcionalno zaupanje na sliki 3.8 med a in d (omenjeni kriteriji so izpolnjeni) se izraža kot $\mathrm{I}\lambda^*_{a,d} = \mathrm{D}\lambda^r_{a,b} \circ \mathrm{D}\lambda^r_{b,c} \circ \mathrm{D}\lambda_{c,d}$, pri čemer je "$\circ$" *vezni* operator v tranzitivni verigi, ki združuje posamezne vrednosti zaupanja v verigi priporočil. V primeru, da nastopa od izvora do ponora več verig, je potrebno upoštevati vsako od njih. Primer take verige prikazuje slika



Slika 3.9: Paralelna veriga zaupanja.

3.9. Izpeljano funkcionalno zaupanje med a in e se izraža kot (zaradi preglednosti brez predpon "D" in "I") $\lambda^*_{a,e} = ((\lambda^r_{a,b} \circ \lambda^r_{b,d}) \diamond (\lambda^r_{a,c} \circ \lambda^r_{c,d})) \circ \lambda_{d,e}$, kjer je "$\diamond$" *agregacijski* operator, ki rabi za združevanje vrednosti zaupanja posameznih verig od izvorne do ciljne entitete.

3.1.3.1 Pravila za izračun

Za izpeljavo tranzitivnih relacij zaupanja, je potrebno določiti ustrezna pravila za modeliranje tranzitivnosti. Izhodišče predstavlja model [29, 30] za kombiniranje relacij zaupanja z ustreznimi vrednostmi. Oznaka $a \xrightarrow{f,\lambda} b$ določa funkcionalno zaupanje z vrednostjo λ, ki ga ima entiteta a v b; medtem ko $a \xrightarrow{r,\lambda} b$ določa zaupanje v priporočilo z vrednostjo λ, ki ga ima entiteta a v entiteto b. Sledijo pravila (tabela 3.1), ki omogočajo združevanje vrednosti izkušenj zaupanja glede na tranzitivno verigo (uporaba veznega operatorja "$\circ$") ali združevanje vrednosti izkušenj zaupanja pri paralelnih verigah (uporaba agregacijskega operatorja "$\diamond$"). Pravili (1) in (2) določata vrsto

$a \xrightarrow{f,\lambda} b$	(1) *Funkcionalno zaupanje*
$a \xrightarrow{r,\lambda} b$	(2) *Zaupanje v priporočilo*
$\dfrac{a \xrightarrow{r,\lambda_1} b \quad b \xrightarrow{r,\lambda_2} c}{a \xrightarrow{r,\lambda_1 \circ \lambda_2} c}$	(3) *Tranzitivnost*
$\dfrac{a \xrightarrow{r,\lambda_1} b \quad b \xrightarrow{f,\lambda_2} c}{a \xrightarrow{f,\lambda_1 \circ \lambda_2} c}$	(4) *Funkcionalno zaupanje preko priporočila*
$\dfrac{a \xrightarrow{f,\lambda_1} b \quad a \xrightarrow{f,\lambda_2} b}{a \xrightarrow{f,\lambda_1 \diamond \lambda_2} b}$	(5) *Agregacija*

Tabela 3.1: Pravila za izračun vrednosti izkušnje zaupanja preko tranzitivnosti.

zaupanja, ki velja med entitetama – funkcionalno zaupanje ali zaupanje v priporočilo. Pravilo (3) opisuje tranzitivnost v verigi priporočil; če a npr. zaupa v priporočilo b-ja za določen kontekst z vrednostjo λ_1 in b analogno zaupa c-ju z vrednostjo λ_2, potem lahko a direktno zaupa v priporočilo c-ja z vrednostjo $\lambda_1 \circ \lambda_2$. Pravilo (4) opisuje primer, ko a zaupa v priporočilo b-ja z vrednostjo λ_1 ter b funkcionalno zaupa c-ju z vrednostjo λ_2, potem a funkcionalno zaupa[12] c-ju z vrednostjo $\lambda_1 \circ \lambda_2$. Zadnje pravilo (5) določa skupno vrednost zaupanja v primeru, da

[12] Izpeljana izkušnja funkcionalnega zaupanja preko priporočila.

obstaja več tranzitivnih poti zaupanja do iste ciljne entitete; v primeru dveh poti je vrednost zaupanja enaka $\lambda_1 \diamond \lambda_2$.

Pri uporabi predstavljenih pravil za izpeljavo vrednosti izkušnje zaupanja znotraj tranzitivnih verig, morata vezni in agregacijski operator zadostiti določenim kriterijem. Vezni operator $\circ$ za združevanje poti znotraj tranzitivne verige mora biti *asociativen*, kjer ni pomembno zaporedje poti znotraj tranzitivne verige. Agregacijski operator $\diamond$ mora biti *asociativen*, poleg tega še *komutativen*, kjer vrstni red agregacije verig ni pomemben. Pomembna lastnost je tudi *distributivnost* $\circ$ čez $\diamond$. Dodatna željena lastnost je *idempotentnost* (t. i. $\forall x \in \Omega : x \diamond x = x$). Omenjene lastnosti privedejo do ustrezne algebrajske strukture – *polkolobar* (angl. semiring). Teorija polkolobarjev je uporabna pri iskanju stopnje zaupanja, kjer entitete niso bile v neposredni interakciji [83]; to se prevede na problem iskanja poti v usmerjenem grafu, kjer so vozlišča entitete, povezave pa relacije zaupanja (v našem primeru je to graf $\mathbf{G}$). Pred definicijo pojma polkolobarja je potrebno definirati tudi pojme *polgrupe* in *monoida* [74].

- *Polgrupa* je algebrajska struktura $(S, *)$ z množico S in dvomestno operacijo $*$, kjer velja:
  a) $\forall x, y \in S : x * y \in S$ (zakon zaprtosti);
  b) $\forall x, y, z \in S : (x * y) * z = x * (y * z)$ (zakon o asociativnosti).

- *Monoid* je algebrajska struktura $(S, *)$, kjer velja:
  a) $(S, *)$ je polgrupa;
  b) $\exists \mathfrak{e} \in S, \forall x \in S : \mathfrak{e} * x = x * \mathfrak{e} = x$ (nevtralni element – enota $\mathfrak{e}$).

V našem primeru je **polkolobar** trojica $(\Omega, \diamond, \circ)$, kjer $\forall x, y, z \in \Omega$ velja [74]:

a) $(\Omega, \diamond)$ je komutativen monoid z enoto $\mathfrak{e} \in \Omega$:
   1. $x \diamond y = y \diamond x$,
   2. $\mathfrak{e} \diamond x = x \diamond \mathfrak{e} = x$.

b) $(\Omega, \circ)$ je monoid z enoto $\mathfrak{e} \in \Omega$ in absorbcijskim elementom $\mathfrak{o} \in \Omega$:
   1. $\mathfrak{e} \circ x = x \circ \mathfrak{e} = x$,
   2. $\mathfrak{o} \circ x = x \circ \mathfrak{o} = \mathfrak{o}$.

c) Operacija $\circ$ je distributivna glede na $\diamond$:
   1. $x \circ (y \diamond z) = (x \circ y) \diamond (x \circ z)$,
   2. $(x \diamond y) \circ z = (x \circ z) \diamond (y \circ z)$.

Glede na naravo operatorjev $\circ$ (znotraj verige) in $\diamond$ (med verigami), je potrebno izpostaviti sledeč problem: v primeru, da uporabljena algebrajska struktura $(\Omega, \diamond, \circ)$ ni polkolobar oz. če vezni operator $\circ$ ni distributiven čez agregacijski operator $\diamond$ (minimalen pogoj), lahko pride do napačnega izračuna izpeljanega funkcionalnega zaupanja. Problem nastane v primeru, če se ista neposredna relacija zaupanja med določenima entitetama znotraj različnih verig pojavi večkrat. Ta problem je eksplicitno izpostavljen tudi v Jøsangovi algebri [34], ker operator za *konsenz*

ni distributiven čez operator za *priporočilo*. V grafu s slike 3.9 lahko npr. nastopa relacija $(d, e) \in R$ z vrednostjo $\lambda_{d,e}$ dvakrat:

a) Če je $(\Omega, \diamond, \circ)$ polkolobar, potem velja enakost (t. i. pravilo distributivnosti):

$$((\lambda^r_{a,b} \circ \lambda^r_{b,d}) \diamond (\lambda^r_{a,c} \circ \lambda^r_{c,d})) \circ \lambda_{d,e} = (\lambda^r_{a,b} \circ \lambda^r_{b,d} \circ \lambda_{d,e}) \diamond (\lambda^r_{a,c} \circ \lambda^r_{c,d} \circ \lambda_{d,e}) \tag{3.10}$$

b) V nasprotnem primeru enakost ne velja:

$$((\lambda^r_{a,b} \circ \lambda^r_{b,d}) \diamond (\lambda^r_{a,c} \circ \lambda^r_{c,d})) \circ \lambda_{d,e} \neq (\lambda^r_{a,b} \circ \lambda^r_{b,d} \circ \lambda_{d,e}) \diamond (\lambda^r_{a,c} \circ \lambda^r_{c,d} \circ \lambda_{d,e}) \tag{3.11}$$

To lahko ponazorimo s primerom polkolobarja $(\Omega, \max, \min)$, kjer je množica $\Omega = \{-1, -0.5, 0, 0.5, 1\}$, enota komutativnega monoida $(\Omega, \max)$ je -1, monoid $(\Omega, \min)$ ima enoto 1 in absorbcijski element -1. Velja tudi, da je operacija *min* distributivna čez operacijo *max* (3.12) in obratno (3.13):

$$\begin{aligned} \min(x, \max(y, z)) &= \max(\min(x, y), \min(x, z)) \\ \min(\max(x, y), z) &= \max(\min(x, z), \min(y, z)) \end{aligned} \tag{3.12}$$

$$\begin{aligned} \max(x, \min(y, z)) &= \min(\max(x, y), \max(x, z)) \\ \max(\min(x, y), z) &= \min(\max(x, z), \max(y, z)) \end{aligned} \tag{3.13}$$

Naj bodo posamezne izkušnje zaupanja enake $\lambda^r_{a,b} = 0.5, \lambda^r_{b,d} = -0.5, \lambda^r_{a,c} = -0.5, \lambda^r_{c,d} = -1, \lambda_{d,e} = 0.5$, potem je leva stran izraza (3.10) enaka:

$$\min(\max(\min(0.5, -0.5), \min(-0.5, -1)), 0.5) = \min(\max(-0.5, -1), 0.5) = -0.5,$$

desna stran izraza (3.10) pa:

$$\max(\min(\min(0.5, -0.5), 0.5), \min(\min(-0.5, -1), 0.5)) = \max(-0.5, -1) = -0.5.$$

V obeh primerih sta vrednosti enaki, kar pomeni, da sta leva in desna stran izraza (3.10) ekvivalentni. V primeru, da nimamo opravka s polkolobarjem (distributivnost operacij ne velja), je potrebno upoštevati vsako relacijo (povezavo) samo enkrat – graf s tranzitivnimi verigami od izvora do ponora lahko vsebuje isto povezavo največ enkrat (t. i. *kanonični izraz*).

Kriterij 4 (Kanonični izraz)**.** V primeru, da v algebrajski strukturi $(\Omega, \diamond, \circ)$ vezni operator $\circ$ ni distributiven čez agregacijski operator $\diamond$, lahko ista relacija zaupanja v izrazu za izračun izpeljane vrednosti zaupanja nastopa največ enkrat.

Obstajajo različne metode za izpeljavo kanoničnih izrazov; pogosta je poenostavitev grafa zaupanja z različnimi tehnikami kompozicij zaporednih in vzporednih povezav (npr. tehnika *Directed Series-Parallel Graphs – DSPG* [84]). Poenostavitev grafa pomeni odpravo zank (ciklov) in odvisnosti med izvorno in ciljno entiteto.

3.1.3.2 Distributivne mreže

V definiciji 3.1.2.4 smo definirali popolno urejenost množice Ω. V definiciji 3.1.2.5 smo jo opisali kot *mrežo* preko strukture urejenosti $(\Omega, \leq)$. Mrežo lahko ekvivalentno predstavimo tudi kot algebrajsko strukturo $(\Omega, \sqcup, \sqcap)$, kjer sta $\sqcup$ in $\sqcap$ dvomestni (dvočleni) operaciji [74] (3.14):

$$x, y \in \Omega : x \sqcup y = \sup(\{x, y\}), x \sqcap y = \inf(\{x, y\}) \tag{3.14}$$

Za algebrajsko strukturo mreže $(\Omega, \sqcup, \sqcap)$ veljajo različne lastnosti [74, 75].

Definicija 3.1.3.2.1 (Mreža)**.** Algebrajska struktura $(\Omega, \sqcup, \sqcap)$ je mreža, če je Ω neprazna množica, $\sqcup$ in $\sqcap$ pa dvomestni operaciji, kjer $\forall x, y, z \in \Omega$ velja:

- $x \sqcup y = y \sqcup x$, $x \sqcap y = y \sqcap x$ (komutativnost),
- $x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z$, $x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$ (asociativnost),
- $x \sqcup (x \sqcap y) = x$, $x \sqcap (x \sqcup y) = x$ (absorpcija),
- $x \sqcup x = x$, $x \sqcap x = x$ (idempotentnost).

Relacijo $\leq$ popolne urejenosti $(\Omega, \leq)$ lahko sedaj definiramo preko algebrajske strukture mreže $(\Omega, \sqcup, \sqcap)$ [74, 75]: $x \leq y \Leftrightarrow x \sqcup y = y \Leftrightarrow x \sqcap y = x$. Ker nastopata operaciji $\sqcup$ in $\sqcap$ povsem enakopravno, velja omeniti *princip dualnosti* [74], kjer lahko operaciji med sabo zamenjamo; torej je $(\Omega, \sqcap, \sqcup)$ dualna $(\Omega, \sqcup, \sqcap)$, vendar je urejenost $x \leq y$ zamenjana z nasprotno $y \leq x$: $y \leq x \Leftrightarrow y \sqcup x = x \Leftrightarrow y \sqcap x = y$.

Omejimo se samo na algebrajsko strukturo *omejene mreže* (angl. bounded lattice) [75]: to je mreža $(\Omega, \sqcup, \sqcap)$, ki ima enoto $\perp$ za operacijo $\sqcup$ (t. i. $x \sqcup \perp = \perp \sqcup x = x$) in enoto $\top$ za operacijo $\sqcap$ (t. i. $x \sqcap \top = \top \sqcap x = x$). Glede na lastnosti komutativnosti, asociativnosti ter obstoja enote, lahko mrežo obravnavamo kot dva komutativna monoida[13] $(\Omega, \sqcup)$ in $(\Omega, \sqcap)$. Torej $\forall x, y \in \Omega$ velja:

a) $(\Omega, \sqcup)$ je komutativni monoid z *enoto* $\inf(\Omega) = \perp$ in *absorbcijskim elementom* $\sup(\Omega) = \top$:
   1) $x \sqcup y = y \sqcup x = \sup(\{x, y\}) = \sup(\{y, x\})$,
   2) $x \sqcup \perp = \perp \sqcup x = \sup(\{x, \perp\}) = \sup(\{\perp, x\}) = x$,
   3) $x \sqcup \top = \top \sqcup x = \sup(\{x, \top\}) = \sup(\{\top, x\}) = \top$.

b) $(\Omega, \sqcap)$ je komutativni monoid z *enoto* $\sup(\Omega) = \top$ in *absorbcijskim elementom* $\inf(\Omega) = \perp$:
   1) $x \sqcap y = y \sqcap x = \inf(\{x, y\}) = \inf(\{y, x\})$,
   2) $x \sqcap \top = \top \sqcap x = \inf(\{x, \top\}) = \inf(\{\top, x\}) = x$,
   3) $x \sqcap \perp = \perp \sqcap x = \inf(\{x, \perp\}) = \inf(\{\perp, x\}) = \perp$.

Hitro lahko opazimo dualnost med $(\Omega, \sqcup)$ in $(\Omega, \sqcap)$ (enoti, absorbcijska elementa). Glede na prikazane lastnosti, manjka algebrajski strukturi $(\Omega, \sqcup, \sqcap)$ (oz. dualni $(\Omega, \sqcap, \sqcup)$) do polkolobarja (glede na podane kriterije v prejšnjem razdelku 3.1.3.1) samo lastnost distributivnosti operacije $\sqcap$ glede na $\sqcup$ (in obratno pri dualnosti). V primeru, da pokažemo omenjeno lastnost – da je $(\Omega, \sqcup, \sqcap)$ distributivna, jo lahko uporabimo kot predvideno algebrajsko strukturo $(\Omega, \diamond, \circ)$.

[13] Sta *polmreži* (angl. semilattice) [75].

Definicija 3.1.3.2.2 (Distributivna mreža)**.** Mreža $(\Omega, \sqcup, \sqcap)$ je *distributivna*, če za $\forall x, y, z \in \Omega$ velja sledeča lastnost: $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$; za dualno mrežo $(\Omega, \sqcap, \sqcup)$ je lastnost obrnjena: $x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z)$.

Izrek. Popolna urejenost $(\Omega, \leq)$ je distributivna mreža.

Dokaz. Pokazati moramo enakost leve in desne strani izraza (3.15):

$$\forall x, y, z \in \Omega : x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z). \tag{3.15}$$

Ker je $(\Omega, \leq)$ popolna urejenost iz definicije 3.1.2.4 sledi, da za poljubna elementa $x, y \in \Omega$ velja bodisi $x \leq y$ ali $y \leq x$ (stroga sovisnost). Predpostavimo, da $y \leq z$, kar je ekvivalentno $y \sqcap z = y$. Potemtakem iz $y \leq z$ sledi $(x \sqcap y) \sqcap (x \sqcap z) = x \sqcap y$ kar ekvivalentno pomeni $x \sqcap y \leq x \sqcap z$. Desna stran izraza (3.15) je potem enaka $(x \sqcap y) \sqcup (x \sqcap z) = x \sqcap z$, leva stran pa je enaka $x \sqcap (y \sqcup z) = x \sqcap z$. Leva stran je torej enaka desni, kar smo hoteli pokazati.

□

Algebrajska struktura $(\Omega, \sqcup, \sqcap)$ je torej polkolobar in celo več – je *idempotentni polkolobar* zaradi idempotentnosti operatorjev $\sqcap$ oz. $\sqcup$ (definicija 3.1.3.2.1). Posledično pomeni, da lahko za algebrajsko strukturo $(\Omega, \diamond, \circ)$ uporabimo kar $(\Omega, \sqcup, \sqcap)$ (oz. dualno $(\Omega, \sqcap, \sqcup)$) – to posledično omogoča pravilen izračun izpeljanega funkcionalnega zaupanja na principu tranzitivnosti. Hitro lahko vidimo, da je $(\Omega, \max, \min)$ (oz. dualna $(\Omega, \min, \max)$) distributivna mreža (to strukturo smo npr. uporabili v primeru prejšnjega razdelka 3.1.3.1).

3.1.3.3 Iskanje tranzitivnih verig

Vrednost zaupanja skozi tranzitivno verigo se praviloma zmanjšuje, torej $\lambda_1 \circ \lambda_2 \leq \lambda_1, \lambda_2$ (seveda odvisno od definicije operatorja $\circ$; če je $\circ$ npr. enak $\sqcup$, trditev ne drži), kjer relacija $\leq$ določa popolno urejenost množice Ω. Tranzitivna veriga oz. pot iz vozlišča (entitete) v_0 do v_k v grafu $\mathbf{G}$ je označena kot $v_0 \xrightarrow{*} v_k$ in predstavlja zaporedje povezav v obliki $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$. Dolžina verige (poti) je število povezav v zaporedju; pot iz v v w je *ne-ničelna* (označeno kot $v \xrightarrow{+} w$), če je dolžina pozitivna.

Za izračun vrednosti zaupanja med izvorno entiteto a in ciljno entiteto b, ki nimata neposredne relacije zaupanja, obstaja več pristopov [83], kot npr. (i) iskanje najkrajše tranzitivne poti med a in b ter (ii) iskanje tranzitivne poti z največjo vrednostjo zaupanja (glede na vezni in agregacijski operator); t. i. iskanje zaporedja vozlišč v grafu $\mathbf{G}(\mathcal{A}, R)$, ki ima največjo agregirano vrednost zaupanja med vsemi potmi, ki se začnejo v a in končajo v b.

V splošnem se iskanje tranzitivnih verig v grafu $\mathbf{G}(\mathcal{A}, R)$ posploši v iskanje *tranzitivne ovojnice* R^+ relacije R. Tranzitivna ovojnica omogoča odgovoriti na vprašanje, ali je vozlišče b dosegljivo iz vozlišča a preko ne-ničelne poti [73].

Definicija 3.1.3.2 (Tranzitivna ovojnica)**.** Relacija R^+ je tranzitivna ovojnica dvomestne relacije R, $R \subseteq R^+$, definirana s predpisom $R^+ = \bigcup_{m>0} R^m$, kjer $R^0 = I$, $R^{m+1} = R^m * R = R * R^m$, $I = \{(v, v) \mid v \in \mathcal{A}\}$ je relacija identitete. Torej aR^+b obstaja natanko tedaj, če obstaja končno zaporedje elementov iz R, kjer $a = x_0$:

$$(x_0, x_1)(x_1, x_2) \ldots (x_{m-1}, x_m)(x_m, b), \text{ kjer } m > 0.$$

Tranzitivna ovojnica grafa zaupanja $\mathbf{G}(\mathcal{A}, R)$ je graf $\mathbf{G}^+(\mathcal{A}, R^+)$, kjer R^+ vsebuje element (a, b) natanko tedaj, če $\mathbf{G}$ vsebuje ne-ničelno pot $a \xrightarrow{+} b$. Velikost tranzitivne ovojnice je označena kot r^+. Množica vseh *naslednikov* vozlišča (entitete) a je določena kot $Succ(a) = \{b \mid (a, b) \in R^+\}$ (t. i. množica vseh vozlišč dostopnih iz a preko ne-ničelnih poti). Množica vseh *predhodnikov* vozlišča (entitete) b je določena kot $Pred(b) = \{a \mid (a, b) \in R^+\}$ (t. i. množica vseh vozlišč, iz katerih je b dosegljiv preko ne-ničelnih poti).

Za iskanje tranzitivnih ovojnic obstaja velik nabor bolj ali manj učinkovitih postopkov in algoritmov (teorija grafov) [73]; npr. *Warshallov algoritem*, algoritmi na principu iskanja *močno povezanih komponent v grafu*, algoritmi na principu *množenja matrik povezanosti*, *hibridni algoritmi*, itd. Naš cilj ni poiskati najbolj učinkovit algoritem, temveč pokazati, kako reševati problem tranzitivnosti zaupanja v grafu $\mathbf{G}(\mathcal{A}, R)$.

V našem primeru imamo dva grafa $\mathbf{G}$ in $\mathbf{G}^r$, ki predstavljata izkušnje funkcionalnega zaupanja in izkušnje zaupanja v priporočilo entitet, ki sta predstavljena z agregiranima matrikama izkušenj $\mathbb{M}'(\varphi)$ in $\mathbb{M}'^r(\varphi)$ v izbranem kontekstu $\varphi \in \Phi$ – lahko ju označimo kot *matriki povezanosti* (angl. adjacency matrix), ki so običajen način predstavitve grafov [73] (drugi bolj učinkovit način predstavitve so *povezani seznami* (angl. adjacency list), predvsem v primeru *redkih* (angl. sparse) grafov – primer, ko v grafu $\mathbf{G}(\mathcal{A}, R)$ velja $|R| \ll |\mathcal{A}|^2$). Za izračun tranzitivne ovojnice grafa $\mathbf{G}^r$ oz. matrike $\mathbb{M}'^r$, uporabimo *Warshallow algoritem* [85], ki vrne novo matriko $\mathbb{M}'^{r+}$ z informacijami o dosegljivosti vozlišč (entitet).

```
integer n = |A|;
boolean M'^r+ = new boolean[n, n];
for k = 1 to n
  for i = 1 to n
    if M'^r[i, k] ≠ −∞ then
      for j = 1 to n
        M'^r+[i, j] = (M'^r[i, j] ≠ −∞) or (M'^r[k, j] ≠ −∞);
```

Algoritem 3.1: Izračun tranzitivne ovojnice matrike $\mathbb{M}'^r$.

Postopek oz. algoritem za izračun vrednosti zaupanja med izvorno entiteto a in ciljno entiteto b, ki nimata relacije funkcionalnega zaupanja (t. i. $\lambda_{a,b} = -\infty$), lahko definiramo s koraki, ki jih prikazuje algoritem 3.2. Koraka 1. in 2. se na prvi pogled ne zdita smiselna, vendar se tranzitivna

ovojnica $\mathbb{M}'^{r+}$ lahko izračunava sproti po vsaki interakciji med entitetami (in posledično množica $Succ(x)$). Tako je vedno na voljo informacija o dosegljivosti entitet.

1. Izračunaj tranzitivno ovojnico $\mathbb{M}'^{r+}$.
2. Za izvorno vozlišče $a \in \mathcal{A}$ preberi množico vseh naslednikov $Succ(a)$.
3. Zadnja relacija v verigi zaupanja mora predstavljati funkcionalno zaupanje – preveri ali obstaja: $\exists x \in Succ(a) : \mathbb{M}'[x, b] \neq -\infty, x \neq b$. V primeru, da obstaja tak x, potem med a in b obstaja vsaj ena tranzitivna pot.
4. V primeru, da pot (veriga) med a in b obstaja, najdi vse poti in za vsako pot izračunaj vrednost izpeljanega funkcionalnega zaupanja.

Algoritem 3.2: Izračun izpeljanega funkcionalnega zaupanja.

V nadaljevanju sledi definicija algoritma za iskanje vseh tranzitivnih poti med izvornim in ciljnim vozliščem. Nobena od najdenih poti ne vsebuje ciklov. Algoritem uporablja metodo *iskanja v globino* (angl. *Depth First Search* – DFS) [85] za sprehod po grafu od izvornega do ciljnega vozlišča. Relacije zaupanja oz. elementi v matrikah $\mathbb{M}'(\varphi)$ in $\mathbb{M}'^{r}(\varphi)$, so predstavljene v računalniški obliki kot tip (struktura) `TrustRelation` z ustreznimi atributi (izvor, ponor, kontekst, vrsta zaupanja). Algoritem 3.3 prikazuje psevdokodo, ki najde vse tranzitivne poti med izvorom in ponorom na podlagi relacij funkcionalnega zaupanja in zaupanja v priporočilo.

Po izvedbi algoritma vsebuje vektor `paths` vse tranzitivne poti med izvorom in ponorom. Za izračun izpeljanega funkcionalnega zaupanja bi bilo idealno, če bi upoštevali vse možne poti, ki so brez ciklov in medsebojnih odvisnosti. V našem primeru uporabimo hevristično pravilo [69] "`TrustLevel(path) + relation.trustLevel` $\geq$ `THRESHOLD`", ki zavrže poti z vrednostjo zaupanja, ki je nižja od vnaprej določenega praga. S tem se število poti lahko precej zniža in poenostavi analiza grafa zaupanja.

Pri izračunu izpeljanega funkcionalnega zaupanja je potrebno upoštevati vse poti znotraj vektorja `path` – uporaba veznega operatorja $\circ$ (znotraj poti) in agregacijskega operatorja $\diamond$ (med posameznimi potmi). V primeru, da je struktura $(\Omega, \diamond, \circ)$ polkolobar ali celo distributivna mreža, je izračun praviloma enostaven; v nasprotnem primeru je potrebno tranzitivne poti spremeniti v ustrezne kanonične izraze (glej kriterij 4).

V nadaljevanju sledi primer izračuna vrednosti zaupanja preko tranzitivnosti. Naj bo $(\Omega, \max, \min)$ polkolobar (distributivna mreža), kjer je množica $\Omega = \{0, 0.5, 1\}$, enota komutativnega monoida $(\Omega, \max)$ je 0, monoid $(\Omega, \min)$ ima enoto 1 in absorbcijski element

Struktura za predstavitev relacije zaupanja:

```
type TrustRelation {
  Entity source;
  Entity target;
  Context context;
  TrustType trustType;
  TrustLevel trustLevel;
}
```

Psevdokoda za deklaracijo ustreznih globalnih spremenljivk, konstant:

```
Vector path = new Vector of type TrustRelation;
Vector paths = new Vector of type path;
const THRESHOLD = λ_min ∈ Ω;
```

Psevdokoda algoritma:

```
void FindPaths(Entity source, Entity target, Context ctx) {
  select relations from M'(ctx) ∪ M'^r(ctx) where
    (relations.source==source) and (relations.context==ctx);
  foreach relation in relations {
    if (relation.target==target) and
       (relation.trustType=='functional') and
       TrustLevel(path) + relation.trustLevel ≥ THRESHOLD {
      paths.add(path + relation);
    } else if (relation.target ≠ target) and
       (relation.trustType=='recommendation') and
       (relation not in path) and
       TrustLevel(path) + relation.trustLevel ≥ THRESHOLD {
      path.add(relation);
      FindPaths(relation.target, target, ctx);
      path.remove(relation);
    }
  }
}
```

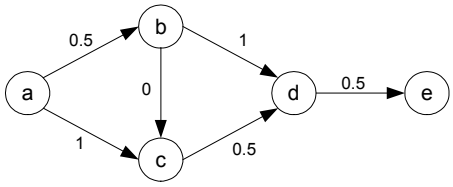
Algoritem 3.3: Iskanje tranzitivnih poti.

0. Naj bodo ustrezne matrike[14] enake:

$$\mathbb{M}'(\varphi) = \begin{bmatrix} \top & \infty & -\infty & -\infty & -\infty \\ -\infty & \top & \infty & -\infty & -\infty \\ -\infty & -\infty & \top & -\infty & -\infty \\ -\infty & -\infty & -\infty & \top & \lambda_{d,e} \\ -\infty & -\infty & -\infty & -\infty & \top \end{bmatrix} \quad \mathbb{M}'^r(\varphi) = \begin{bmatrix} \top & \lambda^r_{a,b} & \lambda^r_{a,c} & -\infty & -\infty \\ -\infty & \top & \lambda^r_{b,c} & \lambda^r_{b,d} & -\infty \\ -\infty & -\infty & \top & \lambda^r_{c,d} & -\infty \\ -\infty & -\infty & -\infty & \top & -\infty \\ -\infty & -\infty & -\infty & -\infty & \top \end{bmatrix}$$

Posamezne vrednosti oz. izkušnje zaupanja naj bodo: $\lambda_{d,e} = 0.5, \lambda^r_{a,b} = 0.5, \lambda^r_{a,c} = 1, \lambda^r_{b,c} = 0, \lambda^r_{b,d} = 1, \lambda^r_{c,d} = 0.5$. Za lažjo predstavitev je na sliki 3.10 prikazan pripadajoč graf zaupanja **G**. Izračunati hočemo izpeljano vrednost funkcionalnega zaupanja med entitetama a in e. Uporabimo korake algoritma 3.2:

[14] Matrike so redke, za bolj kompaktno predstavitev bi lahko uporabili povezane sezname.



Slika 3.10: Graf zaupanja na podlagi matrik $\mathbb{M}'(\varphi)$ in $\mathbb{M}'^r(\varphi)$.

1. Izračun tranzitivne ovojnice $\mathbb{M}'^{r+}$:

$$\mathbb{M}'^{r+}(\varphi) = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2. Nasledniki vozlišča a so $Succ(a) = \{b, c, d\}$.

3. Med a in e obstaja vsaj ena tranzitivna pot, ker obstaja vozlišče d, ki ustreza pogoju: $\exists x \in Succ(a) : \mathbb{M}'[x, e] \neq -\infty, x \neq e$.

4. Algoritem 3.3 (brez hevristike oz. $\lambda_{min} = 0$) najde vse 3 tranzitivne poti med a in d, ki jih shrani v vektor `paths`: $(a, b)(b, d)(d, e)$, $(a, c)(c, d)(d, e)$ in $(a, b)(b, c)(c, d)(d, e)$.

Pri izračunu izpeljanega funkcionalnega zaupanja $\lambda^*_{a,e}$, je potrebno upoštevati vse poti med izvorom in ponorom. Ker imamo opravka s polkolobarjem $(\Omega, \max, \min)$ oz. pravilom distributivnosti (minimalni pogoj), kombinacij posameznih poti ni potrebno preoblikovati v kanonične izraze. Izpeljana izkušnja funkcionalnega zaupanje $\lambda^*_{a,e}$ je enaka:

$$\begin{aligned} \lambda^*_{a,e} &= (\lambda^r_{a,b} \circ \lambda^r_{b,d} \circ \lambda_{d,e}) \diamond (\lambda^r_{a,c} \circ \lambda^r_{c,d} \circ \lambda_{d,e}) \diamond (\lambda^r_{a,b} \circ \lambda^r_{b,c} \circ \lambda^r_{c,d} \circ \lambda_{d,e}) = \\ &= \max(\min(\lambda^r_{a,b}, \lambda^r_{b,d}, \lambda_{d,e}), \min(\lambda^r_{a,c}, \lambda^r_{c,d}, \lambda_{d,e}), \min(\lambda^r_{a,b}, \lambda^r_{b,c}, \lambda^r_{c,d}, \lambda_{d,e})) = \\ &= \max(\min(0.5, 1, 0.5), \min(1, 0.5, 0.5), \min(0.5, 0, 0.5, 0.5)) = \\ &= \max(0.5, 0.5, 0) = 0.5. \end{aligned}$$

3.1.4 Kvalitativni model

Predstavljeni model zaupanja $ATM = (\mathbf{G}, \Omega, \mathcal{T}, \mathcal{F}, \mathcal{O}, \Gamma, \Pi)$ iz definicije 3.1.2.14 predstavlja splošen abstraktni model, ki ga bomo v tem razdelku opisali s kvalitativne perspektive. Za uporabnike je bolj običajno in intuitivno, da razmišljajo o zaupanju v kvalitativnih pojmih iz omejene diskretne množice. To dokazujejo tudi nekateri eksperimenti; vir [13] npr. dokazuje, da je uporabnikom za zaznavanje zaupanja lahko bližja kvalitativna metodologija kot pa kvantitativna (npr. verjetnostni modeli). V nadaljevanju sledijo definicije posameznih komponent oz. dimenzij abstraktnega modela zaupanja ATM [63, 72].

Definicija 3.1.4.1 (Množica ocen Ω in množica stopenj $\mathcal{T}$)**.** Množici Ω in $\mathcal{T}$ sta ekvivalentni in vsebujeta pet kvalitativnih elementov: *zaupanja nevreden* (angl. *distrusted* (D)), *delno zaupanja nevreden* (angl. *partially distrusted* (PD)), *neodločen* (angl. *undecided* (U)), *delno zaupanja vreden* (angl. *partially trusted* (PT)), *zaupanja vreden* (angl. *trusted* (T)). Množici sta tako enaki $\Omega = \mathcal{T} = \{\text{D,PD,U,PT,T}\}$ z relacijo popolne urejenosti: $\text{D} \leq \text{PD} \leq \text{U} \leq \text{PT} \leq \text{T}$.

Definicija 3.1.4.2 (Množica preslikav $\mathcal{F}$)**.** $\mathcal{F} = \{f_\Omega, f'_\Omega, f_\mathcal{T}, f'_\mathcal{T}\}$ je množica injektivnih in surjektivnih preslikav. Injektivna preslikava $f_\Omega : \Omega \cup \{-\infty\} \mapsto [-1, 1] \cup \{-\infty\}$ je definirana kot:

$$\begin{aligned} f_\Omega(\text{D}) &= -1, \\ f_\Omega(\text{PD}) &= -0.5, \\ f_\Omega(\text{U}) &= 0, \\ f_\Omega(\text{PT}) &= 0.5, \\ f_\Omega(\text{T}) &= 1. \end{aligned}$$

Surjektivna preslikava $f'_\Omega : [-1, 1] \cup \{-\infty\} \mapsto \Omega \cup \{-\infty\}$ je definirana kot:

$$f'_\Omega(\sigma) = \omega, \text{kjer } \arg\min_{\omega \in \Omega} |\sigma - f_\Omega(\omega)|.$$

Ker je $\Omega = \mathcal{T}$, velja $f_\Omega = f_\mathcal{T}$ in $f'_\Omega = f'_\mathcal{T}$. Velja tudi $f_\Omega(-\infty) = f'_\Omega(-\infty) = f_\mathcal{T}(-\infty) = f'_\mathcal{T}(-\infty) = -\infty$.

Definicija 3.1.4.3 (Operatorji zaupanja $\mathcal{O}$)**.** Množica operatorjev zaupanja je določena kot $\mathcal{O} = \{\curlywedge, \curlyvee, \sim\}$, kjer $\curlywedge$ določa *optimistični* (angl. optimistic) operator, $\curlyvee$ *pesimistični* (angl. pessimistic) operator in $\sim$ *uravnotežen* (angl. balanced) operator. Definicija operatorjev je sledeča – kvalitativna algebra (simbol "$*$" predstavlja poljubno vrednost iz množice Ω):

$$\begin{aligned} &\lambda_{i,k} \curlywedge \lambda_{j,k} = \max(\lambda_{i,k}, \lambda_{j,k}), \\ &\lambda_{i,k} \curlyvee \lambda_{j,k} = \min(\lambda_{i,k}, \lambda_{j,k}), \\ &\lambda_{i,k} \text{ op } -\infty = \lambda_{i,k}, \\ &-\infty \text{ op } * = -\infty, \\ &\lambda_{i,k} \sim \text{U} = \text{U} \sim \lambda_{j,k} = \text{U}, \\ &\lambda_{i,k} \sim \lambda_{j,k} = \min(\lambda_{i,k}, \lambda_{j,k}) \Leftrightarrow f_\Omega(\lambda_{i,k}) > 0, f_\Omega(\lambda_{j,k}) > 0, \\ &\lambda_{i,k} \sim \lambda_{j,k} = \text{U} \Leftrightarrow f_\Omega(\lambda_{i,k}) > 0, f_\Omega(\lambda_{j,k}) < 0, \\ &\lambda_{i,k} \sim \lambda_{j,k} = \text{U} \Leftrightarrow f_\Omega(\lambda_{i,k}) < 0, f_\Omega(\lambda_{j,k}) > 0, \\ &\lambda_{i,k} \sim \lambda_{j,k} = \max(\lambda_{i,k}, \lambda_{j,k}) \Leftrightarrow f_\Omega(\lambda_{i,k}) < 0, f_\Omega(\lambda_{j,k}) < 0. \end{aligned}$$

Prioriteta operatorjev je enakovredna; za specifičen vrstni red je potrebno uporabiti oklepaje. To je zlasti pomembno pri uporabi operatorjev na t. i. vektorjih zaupanja. Izračun stopnje zaupanja τ na izbranem vektorju zaupanja agregirane matrike izkušenj $\mathbb{M}'(t', \varphi)$, je dinamična funkcija, ki vključuje izkušnje i-tega agenta in izkušnje celotne skupnosti do ciljnega j-tega agenta. Izračunana stopnja zaupanja $\tau_{i,j}(\varphi) \in \mathcal{T}$ na določenem vektorju zaupanja $\lambda_{i,j=const}$, je označen kot $tvec^{\mathrm{op}}_{i,j}(\varphi)$ z definicijo (3.16):

$$tvec^{\mathrm{op}}_{i,j}(\varphi) = (\ldots((\lambda_{i,j} \text{ op } \lambda_{1,j}) \text{ op } \lambda_{2,j})\ldots \text{ op } \lambda_{n-1,j}) \text{ op } \lambda_{n,j},\ \lambda_{i,j} \neq -\infty \tag{3.16}$$

V primeru, ko ne obstaja pretekla izkušnja med i-tim in j-tim agentom (t. i. $\lambda_{i,j} = -\infty$), lahko uporabimo princip tranzitivnosti (glej razdelek 3.1.3) – vendar samo v primeru, da obstaja tudi matrika izkušenj zaupanja v priporočilo $\mathbb{M}'^{r}(\varphi)$. Uporabimo postopek oz. algoritem 3.2 in v primeru, da obstaja ne-ničelna tranzitivna pot med i-tim in j-tim agentom $i \xrightarrow{+} j$, algoritem izračuna izpeljano izkušnjo funkcionalnega zaupanja $\lambda^{*}_{i,j}$. Vendar moramo za pravilen izračun določiti ustrezno algebrajsko strukturo (polkolobar) $(\Omega, \diamond, \circ)$ z ustreznimi operatorji iz množice $\mathcal{O}$. Glede na definicije operatorjev, obstaja več možnih polkolobarjev; npr. $(\Omega, \curlywedge, \curlyvee)$ in $(\Omega, \curlyvee, \curlywedge)$[15]:

(1) $(\Omega, \curlywedge, \curlyvee)$: enota komutativnega monoida $(\Omega, \curlywedge)$ je D (t. i. $\mathrm{D} \curlywedge \lambda = \lambda \curlywedge \mathrm{D} = \lambda$), monoid $(\Omega, \curlyvee)$ pa ima enoto T (t. i. $\mathrm{T} \curlyvee \lambda = \lambda \curlyvee \mathrm{T} = \lambda$) in absorbcijski element D (t. i. $\mathrm{D} \curlyvee \lambda = \lambda \curlyvee \mathrm{D} = \mathrm{D}$).

- Operacija $\curlyvee$ je distributivna čez $\curlywedge$: $x \curlyvee (y \curlywedge z) = (x \curlyvee y) \curlywedge (x \curlyvee z)$ in $(x \curlywedge y) \curlyvee z = (x \curlyvee z) \curlywedge (y \curlyvee z)$.

(2) $(\Omega, \curlyvee, \curlywedge)$: enota komutativnega monoida $(\Omega, \curlyvee)$ je T (t. i. $\mathrm{T} \curlyvee \lambda = \lambda \curlyvee \mathrm{T} = \lambda$), monoid $(\Omega, \curlywedge)$ pa ima enoto D (t. i. $\mathrm{D} \curlywedge \lambda = \lambda \curlywedge \mathrm{D} = \lambda$) in absorbcijski element T (t. i. $\mathrm{T} \curlywedge \lambda = \lambda \curlywedge \mathrm{T} = \mathrm{T}$).

- Operacija $\curlywedge$ je distributivna čez $\curlyvee$: $x \curlywedge (y \curlyvee z) = (x \curlywedge y) \curlyvee (x \curlywedge z)$ in $(x \curlyvee y) \curlywedge z = (x \curlywedge z) \curlyvee (y \curlywedge z)$.

Če pogledamo posamezne algebrajske strukture $(\Omega, \mathrm{op}), \mathrm{op} \in \mathcal{O}$, lahko ugotovimo naslednje:

- $(\Omega, \sim)$ je komutativna polgrupa (ni monoid), ker ne obstaja enota, obstaja pa absorbcijski element U;

- $(\Omega, \curlywedge)$ in $(\Omega, \curlyvee)$ sta komutativna monoida z ustrezno enoto in absorbcijskim elementom;

- operator $\sim$ je distributiven čez $\curlywedge$ in $\curlyvee$: $x \sim (y \text{ op } z) = (x \sim y) \text{ op } (x \sim z)$ in $(x \text{ op } y) \sim z) = (x \sim z) \text{ op } (y \sim z)$, kjer $\mathrm{op} \in \{\curlywedge, \curlyvee\}$;

[15] $(\Omega, \curlyvee, \curlywedge)$ in $(\Omega, \curlywedge, \curlyvee)$ sta tudi dualni distributivni mreži.

- operatorja $\curlywedge$ in $\curlyvee$ sta distributivna čez $\sim$: x op $(y \sim z) = (x$ op $y) \sim (x$ op $z)$ in $(x \sim y)$ op $z) = (x$ op $z) \sim (y$ op $z)$, kjer op $\in \{\curlywedge, \curlyvee\}$.

Glede na navedene ugotovitve, lahko za uporabo principa tranzitivnosti uporabimo različne algebrajske strukture $(\Omega, \mathrm{op}_1, \mathrm{op}_2)$, kjer je op_1 $\mathrm{op}_2 \in \mathcal{O}$, kljub temu, da niso vse strukture polkolobarji (velja pa pravilo distributivnosti). Z izpeljano izkušnjo funkcionalnega zaupanja $\lambda^*_{i,j}$, lahko izračunamo stopnjo zaupanja $\tau^*_{i,j} \in \mathcal{T}$ na enak način kot v izrazu (3.16), le da tukaj vrednost označimo s $tvec^{\mathrm{op}\,*}_{i,j}(\varphi)$, ki ima definicijo (3.17):

$$tvec^{\mathrm{op}\,*}_{i,j}(\varphi) = (\ldots((\lambda^*_{i,j} \text{ op } \lambda_{1,j}) \text{ op } \lambda_{2,j})\ldots \text{ op } \lambda_{n-1,j}) \text{ op } \lambda_{n,j},\ \lambda^*_{i,j} \neq -\infty \tag{3.17}$$

V primeru, da ne obstaja ne-ničelna tranzitivna pot med i-im in j-im agentom (t. i. $i \overset{+}{\nrightarrow} j$), stopnje funkcionalnega zaupanja ni mogoče določiti; v tem primeru se izračuna *ugled* j-tega agenta. Ugled j-tega agenta v kontekstu φ je označen kot $rep^{\mathrm{op}}_j(\varphi)$ z definicijo (3.18):

$$rep^{\mathrm{op}}_j(\varphi) = (\ldots(\ldots((\lambda_{1,j} \text{ op } \lambda_{2,j}) \text{ op } \lambda_{3,j})\ldots \text{ op } \lambda_{k,j})\ldots \text{ op } \lambda_{n-1,j}) \text{ op } \lambda_{n,j},\ k \neq i. \tag{3.18}$$

Sedaj lahko povzamemo pravila za izračun stopnje zaupanja; le-ta predstavlja vitalno informacijo za odločitev, ali naj agent a stopi v interakcijo z agentom b znotraj specifičnega konteksta φ.

Definicija 3.1.4.4 (Algoritem Γ)**.** Izračunana stopnja zaupanja agenta a do agenta b za izbrani operator op $\in \mathcal{O}$ v kontekstu $\varphi \in \Phi$, je označena kot $trust^{\mathrm{op}}_{a,b}(\varphi)$ z definicijo (3.19):

$$trust^{\mathrm{op}}_{a,b}(\varphi) = \begin{cases} tvec^{\mathrm{op}}_{a,b}(\varphi), & \lambda_{a,b}(\varphi) \neq -\infty \\ tvec^{\mathrm{op}\,*}_{a,b}(\varphi), & \lambda_{a,b}(\varphi) = -\infty,\ a \overset{+}{\rightarrow} b \\ rep^{\mathrm{op}}_b(\varphi), & \lambda_{a,b}(\varphi) = -\infty,\ a \overset{+}{\nrightarrow} b \end{cases} \tag{3.19}$$

Zanimivo vprašanje je tudi, kako izračunati *celotno* stopnjo zaupanja ne glede na izbrani kontekst φ. To ima smisel samo v primeru, ko nas zanima t. i. združena (angl. overall) stopnja zaupanja oz. ugleda izbranega ciljnega agenta znotraj socialne mreže. Ta stopnja podaja neke vrste *globalno sliko* – t. i. "indikator obnašanja" izbranega agenta. Seveda je ta stopnja zaupanja smotrna samo v primeru, ko izvorni agent sprejema zaupanje različnih kontekstov z enako težo. Ena od možnosti je uporaba ustreznega kvalitativnega operatorja na vsaki kontekstno odvisni stopnji zaupanja. V ta namen definiramo celotno stopnjo zaupanja $totalTrust^{\mathrm{op}}_{a,b}$, kjer $\varphi_i \in \Phi, m = |\Phi|$ (3.20):

$$totalTrust^{\mathrm{op}}_{a,b} = trust^{\mathrm{op}}_{a,b}(\varphi_1) \text{ op } trust^{\mathrm{op}}_{a,b}(\varphi_2)\ldots \text{op } trust^{\mathrm{op}}_{a,b}(\varphi_m). \tag{3.20}$$

Primer izračuna[16]

Tukaj podamo primer izračuna stopnje zaupanja glede na definiran algoritem Γ (definicija

[16] Dodatne lastnosti kvalitativnega modela prikazujejo ustrezne simulacije – glej dodatek A.

3.1.4.4). Naj bo množica entitet $\mathcal{A} = \{a, b, c\}$, prisotna sta dva različna konteksta interakcij $\Phi = \{\varphi_1, \varphi_2\}$, kjer je $\varphi_i = (Name_i, ES_i)$. Naj bo $ES_1 = \{e_1, e_2, e_3\}$, $ES_2 = \{g_1, g_2\}$, množice končnih konfiguracij pa $\mathcal{C}^0_{\varphi_1.ES_1} = \{\emptyset, \{e_2\}, \{e_3\}, \{e_1, e_3\}\}$ in $\mathcal{C}^0_{\varphi_2.ES_2} = \{\emptyset, \{g_1\}, \{g_2\}\}$. Zabeležene interakcije med entitetami prikazuje tabela 3.2. Iz tabele lahko izluščimo matrike

izvor	ponor	$x_{i,j}$	$f_\Omega(x_{i,j}.\omega)$	čas interakcije t_i	Φ
a	b	$(\{e_2\}, \mathrm{D})$	-1	03.09.2008	φ_1
a	c	$(\{e_2\}, \mathrm{U})$	0	03.09.2008	φ_1
b	c	$(\{e_3\}, \mathrm{PD})$	-0.5	03.09.2008	φ_1
a	b	$(\{e_1, e_3\}, \mathrm{PT})$	0.5	10.09.2008	φ_1
a	c	$(\{e_3\}, \mathrm{T})$	1	10.09.2008	φ_1
a	b	$(\{g_1\}, \mathrm{PD})$	-0.5	04.09.2008	φ_2
a	c	$(\{g_1\}, \mathrm{U})$	0	04.09.2008	φ_2
b	c	$(\{g_1\}, \mathrm{T})$	1	04.09.2008	φ_2
a	c	$(\{g_2\}, \mathrm{T})$	1	10.09.2008	φ_2

Tabela 3.2: Interakcije med entitetami.

interakcij $\mathbb{M}(t, \varphi)$, ki so sledeče:

$$\mathbb{M}(03.09.2008, \varphi_1) = \begin{bmatrix} (\emptyset, -\infty) & (\{e_2\}, \mathrm{D}) & (\{e_2\}, \mathrm{U}) \\ (\emptyset, -\infty) & (\emptyset, -\infty) & (\{e_3\}, \mathrm{PD}) \\ (\emptyset, -\infty) & (\emptyset, -\infty) & (\emptyset, -\infty) \end{bmatrix} \quad \mathbb{M}(10.09.2008, \varphi_1) = \begin{bmatrix} (\emptyset, -\infty) & (\{e_1, e_3\}, \mathrm{PT}) & (\{e_3\}, \mathrm{T}) \\ (\emptyset, -\infty) & (\emptyset, -\infty) & (\emptyset, -\infty) \\ (\emptyset, -\infty) & (\emptyset, -\infty) & (\emptyset, -\infty) \end{bmatrix}$$

$$\mathbb{M}(04.09.2008, \varphi_2) = \begin{bmatrix} (\emptyset, -\infty) & (\{g_1\}, \mathrm{PD}) & (\{g_1\}, \mathrm{U}) \\ (\emptyset, -\infty) & (\emptyset, -\infty) & (\{g_1\}, \mathrm{T}) \\ (\emptyset, -\infty) & (\emptyset, -\infty) & (\emptyset, -\infty) \end{bmatrix} \quad \mathbb{M}(10.09.2008, \varphi_2) = \begin{bmatrix} (\emptyset, -\infty) & (\emptyset, -\infty) & (\{g_2\}, \mathrm{T}) \\ (\emptyset, -\infty) & (\emptyset, -\infty) & (\emptyset, -\infty) \\ (\emptyset, -\infty) & (\emptyset, -\infty) & (\emptyset, -\infty) \end{bmatrix}$$

Sedaj je potrebno izračunati pretekle izkušnje $\sigma_{i,j}(t', \varphi)$ (glej definicijo 3.1.2.12) med entitetami za oba podana konteksta. Naj bo čas opazovanja enak za oba konteksta – $t' = 11.09.2008$; razlika med časom opazovanja t' in časi preteklih interakcij naj se izraža v številu dni. Utežni faktor za posamezno interakcijo naj bo enak $\alpha = 0.1$. Vrednosti preteklih izkušenj so sledeče:

(a) Kontekst φ_1:

$$\sigma_{a,b}(11.09.2008, \varphi_1) = \frac{e^{-0.1(11.09.2008-10.09.2008)} f_\Omega(\mathrm{PT}) + e^{-0.1(11.09.2008-03.09.2008)} f_\Omega(\mathrm{D})}{2} = \\ = \frac{e^{-0.1}0.5 + e^{-0.8}(-1)}{2} = 0.00308$$

$$\sigma_{a,c}(11.09.2008, \varphi_1) = \frac{e^{-0.1(11.09.2008-10.09.2008)} f_\Omega(\mathrm{T}) + e^{-0.1(11.09.2008-03.09.2008)} f_\Omega(\mathrm{U})}{2} = \\ = \frac{e^{-0.1}1 + e^{-0.8}0}{2} = 0.45241$$

$$\sigma_{b,c}(11.09.2008, \varphi_1) = e^{-0.1(11.09.2008-03.09.2008)} f_\Omega(\mathrm{PD}) = e^{-0.8}(-0.5) = -0.22466$$

(b) Kontekst φ_2:

$$\sigma_{a,b}(11.09.2008, \varphi_2) = e^{-0.1(11.09.2008-04.09.2008)} f_\Omega(\text{PD}) = e^{-0.7}(-0.5) = 0.24829$$

$$\sigma_{a,c}(11.09.2008, \varphi_2) = \frac{e^{-0.1(11.09.2008-10.09.2008)} f_\Omega(\text{T}) + e^{-0.1(11.09.2008-04.09.2008)} f_\Omega(\text{U})}{2} = \\ = \frac{e^{-0.1}1 + e^{-0.7}0}{2} = 0.45241$$

$$\sigma_{b,c}(11.09.2008, \varphi_2) = e^{-0.1(11.09.2008-04.09.2008)} f_\Omega(\text{T}) = e^{-0.7}1 = 0.49658$$

Sedaj je potrebno določiti pretekle izkušnje $\lambda_{i,j}(\varphi) \in \Omega$ preko surjektivne preslikave $f'_\Omega(\sigma) = \omega$, kjer $\arg\min_{\omega\in\Omega}|\sigma - f_\Omega(\omega)|$ (definicija 3.1.4.2); tako dobimo $\lambda_{a,b}(\varphi_1) = \text{U}$, $\lambda_{a,c}(\varphi_1) = \text{PT}$, $\lambda_{b,c}(\varphi_1) = \text{U}$, $\lambda_{a,b}(\varphi_2) = \text{U}$, $\lambda_{a,c}(\varphi_2) = \text{PT}$ in $\lambda_{b,c}(\varphi_2) = \text{PT}$. Kot naslednji korak lahko določimo matrike izkušenj $\mathbb{M}'(t', \varphi)$:

$$\mathbb{M}'(11.09.2008, \varphi_1) = \begin{bmatrix} -\infty & \text{U} & \text{PT} \\ -\infty & -\infty & \text{U} \\ -\infty & -\infty & -\infty \end{bmatrix} \quad \mathbb{M}'(11.09.2008, \varphi_2) = \begin{bmatrix} -\infty & \text{U} & \text{PT} \\ -\infty & -\infty & \text{PT} \\ -\infty & -\infty & -\infty \end{bmatrix}$$

Naslednji korak je izračun stopnje zaupanja $trust^{\text{op}}_{i,j}(\varphi)$ iz matrik izkušenj $\mathbb{M}'(\varphi)$. Uporabimo optimistični operator zaupanja op $= \curlywedge \in \mathcal{O}$.

(a) Kontekst φ_1:

- $trust^{\curlywedge}_{a,b}(\varphi_1) = tvec^{\curlywedge}_{a,b}(\varphi_1) = (((\lambda_{a,b} \curlywedge \lambda_{a,b}) \curlywedge \lambda_{b,b}) \curlywedge \lambda_{c,b}) = (((\text{U} \curlywedge \text{U}) \curlywedge -\infty) \curlywedge -\infty) = \text{U}$
- $trust^{\curlywedge}_{a,c}(\varphi_1) = tvec^{\curlywedge}_{a,c}(\varphi_1) = (((\lambda_{a,c} \curlywedge \lambda_{a,c}) \curlywedge \lambda_{b,c}) \curlywedge \lambda_{c,c}) = (((\text{PT} \curlywedge \text{PT}) \curlywedge \text{U}) \curlywedge -\infty) = \text{PT}$
- $trust^{\curlywedge}_{b,c}(\varphi_1) = tvec^{\curlywedge}_{b,c}(\varphi_1) = (((\lambda_{b,c} \curlywedge \lambda_{a,c}) \curlywedge \lambda_{b,c}) \curlywedge \lambda_{c,c}) = (((\text{U} \curlywedge \text{PT}) \curlywedge \text{U}) \curlywedge -\infty) = \text{PT}$
- $trust^{\curlywedge}_{b,a}(\varphi_1) = rep^{\curlywedge}_a = -\infty$
- $trust^{\curlywedge}_{c,a}(\varphi_1) = rep^{\curlywedge}_a = -\infty$
- $trust^{\curlywedge}_{c,b}(\varphi_1) = rep^{\curlywedge}_b = \text{U}$

(b) Kontekst φ_2:

- $trust^{\curlywedge}_{a,b}(\varphi_2) = tvec^{\curlywedge}_{a,b}(\varphi_2) = (((\lambda_{a,b} \curlywedge \lambda_{a,b}) \curlywedge \lambda_{b,b}) \curlywedge \lambda_{c,b}) = (((\text{U} \curlywedge \text{U}) \curlywedge -\infty) \curlywedge -\infty) = \text{U}$
- $trust^{\curlywedge}_{a,c}(\varphi_2) = tvec^{\curlywedge}_{a,c}(\varphi_2) = (((\lambda_{a,c} \curlywedge \lambda_{a,c}) \curlywedge \lambda_{b,c}) \curlywedge \lambda_{c,c}) = (((\text{PT} \curlywedge \text{PT}) \curlywedge \text{PT}) \curlywedge -\infty) =$ PT
- $trust^{\curlywedge}_{b,c}(\varphi_2) = tvec^{\curlywedge}_{b,c}(\varphi_2) = (((\lambda_{b,c} \curlywedge \lambda_{a,c}) \curlywedge \lambda_{b,c}) \curlywedge \lambda_{c,c}) = (((\text{U} \curlywedge \text{PT}) \curlywedge \text{U}) \curlywedge -\infty) = \text{PT}$
- $trust^{\curlywedge}_{b,a}(\varphi_2) = rep^{\curlywedge}_a = -\infty$
- $trust^{\curlywedge}_{c,a}(\varphi_2) = rep^{\curlywedge}_a = -\infty$
- $trust^{\curlywedge}_{c,b}(\varphi_2) = rep^{\curlywedge}_b = \text{U}$

Posledično lahko izračunamo tudi celotno stopnjo zaupanja npr. med entiteto a in b – to zaupanje podaja t. i. globalno sliko entitete b:

$$totalTrust^{\curlywedge}_{a,b} = trust^{\curlywedge}_{a,b}(\varphi_1) \curlywedge trust^{\curlywedge}_{a,b}(\varphi_2) = \text{U} \curlywedge \text{U} = \text{U}.$$

3.1.5 Kvantitativni model

Kvantitativni model temelji že na opisanem modelu prepričanja (glej poglavje 2, razdelek 2.6.1). Osnovo predstavlja Jøsangova subjektivna logika [47, 34], kjer se stopnja zaupanja v okviru konteksta izraža kot mnenje, sestavljeno iz stopnje zaupanja, nezaupanja in negotovosti. Ta model je za uporabnike manj intuitiven kot kvalitativni model, ker zahteva poznavanje osnov logike in verjetnosti. V nadaljevanju sledijo definicije posameznih komponent abstraktnega modela ATM.

Definicija 3.1.5.1 (Množica ocen Ω in množica stopenj $\mathcal{T}$)**.** Množici Ω in $\mathcal{T}$ sta ekvivalentni in vsebujeta mnenja $\omega = (b, d, u, a)$, kjer b predstavlja stopnjo *zaupanja* (angl. belief), d stopnjo *nezaupanja* (angl. disbelief) in u stopnjo *negotovosti* (angl. uncertainty). Parameter a je relativna atomarnost[17] in določa, kako negotovost vpliva na matematično upanje $E(\omega) = b+au$. Definicija množic je enaka $\Omega = \mathcal{T} = \{(b, d, u, a) \mid b, d, u, a \in [0, 1],\ b + d + u = 1\}$. Množici sta popolno urejeni (glej definicijo 3.1.2.4), kjer sta mnenji $\omega_1 = (b_1, d_1, u_1, a_1)$ in $\omega_2 = (b_2, d_2, u_2, a_2)$ urejeni po sledeči prioriteti:

1) Mnenje z manjšim matematičnih upanjem ima manjšo težo: $\omega_1 \leq \omega_2$, če $E(\omega_1) \leq E(\omega_2)$.
2) Mnenje z večjo negotovostjo ima manjšo težo: $\omega_1 \leq \omega_2$, če $u_2 \leq u_1$.
3) Mnenje z večjo relativno atomarnostjo ima manjšo težo: $\omega_1 \leq \omega_2$, če $a_2 \leq a_1$.

Definicija 3.1.5.2 (Množica preslikav $\mathcal{F}$)**.** $\mathcal{F} = \{f_\Omega, f'_\Omega, f_\mathcal{T}, f'_\mathcal{T}\}$ je množica identičnih bijektivnih preslikav. Ker je $\Omega = \mathcal{T} \subseteq [0, 1]^3$, velja $f_\Omega = f_\mathcal{T} = f'_\Omega = f'_\mathcal{T}$, kjer $f_\Omega : \Omega \cup \{-\infty\} \mapsto \Omega \cup \{-\infty\}$ in $f_\Omega(\omega) = \omega$ ter $f_\Omega(-\infty) = -\infty$.

Subjektivna logika [47] vsebuje tako logične operatorje (npr. konjunkcija, disjunkcija, negacija), kot tudi izpeljane operatorje (npr. konsenz, priporočilo).

Definicija 3.1.5.3 (Operatorji zaupanja $\mathcal{O}$)**.** Množica operatorjev zaupanja je določena kot $\mathcal{O} = \{\wedge, \vee, \neg, \oplus, \otimes\}$, kjer je $\wedge$ *konjunkcija*, $\vee$ *disjunkcija*, $\neg$ *negacija*, $\oplus$ *konsenz* in $\otimes$ *priporočilo*.

Definicija nekaterih operatorjev je podana že v poglavju 2 (razdelek 2.6.1), tu samo izpostavimo spremenjeno notacijo za mnenje, kot ga določa Jøsangov model [47, 35], kjer ω_p^A pomeni mnenje agenta A o neki trditvi p. V našem primeru je mnenje zapisano kot $\omega_{i,j}(t, \varphi) = (b_{i,j}(t, \varphi), d_{i,j}(t, \varphi), u_{i,j}(t, \varphi), a_{i,j}(t, \varphi))$ in določa subjektivno mnenje i-te do j-te entitete v času $t \in T$ in kontekstu $\varphi \in \Phi$ $(i, j \in [1, |\mathcal{A}|])$. V našem modelu uporabimo agregirane vrednosti mnenj $\lambda_{i,j}(\varphi) = (b_{i,j}(\varphi), d_{i,j}(\varphi), u_{i,j}(\varphi), a_{i,j}(\varphi))$ (pretekle izkušnje[18] – glej definicijo 3.1.2.12), ki so odvisne samo od konteksta. Za naš model zaupanja tako uporabimo sledeče operatorje:

[17] V splošnem je $a = 0.5$ in se lahko izpusti iz zapisa za mnenje $\omega : (b, d, u) = (b, d, u, 0.5)$.
[18] Za uteženo vsoto je potrebno, da je utežni faktor $\alpha = 0$.

- *konjunkcija*: $\lambda_{i,j}(\varphi_1, \varphi_2) = \lambda_{i,j}(\varphi_1) \wedge \lambda_{i,j}(\varphi_2)$ za združevanje mnenj iste entitete za različne kontekste (operator je komutativen in asociativen);

- *konsenz*: $\lambda_{[i,j],k}(\varphi) = \lambda_{i,k}(\varphi) \oplus \lambda_{j,k}(\varphi)$ za združevanje neodvisnih mnenj entitet znotraj istega konteksta (operator je komutativen in asociativen) – t. i. agregacijski operator $\diamond$ za združevanje tranzitivnih poti;

- *priporočilo*: $\lambda^*_{i,j}(\varphi) = \lambda^r_{i,j}(\varphi) \otimes \lambda_{i,j}(\varphi)$ za združevanje mnenj v tranzitivni verigi (operator ni komutativen, je asociativen) – t. i. vezni operator $\circ$ za združevanje mnenj v tranzitivni verigi.

Noben od omenjenih operatorjev nima lastnosti idempotentnosti, zato je uporaba operatorja na istih mnenjih brez pomena [34]. Poleg tega vsak operator predvideva statistično neodvisna mnenja. Prioriteta operatorjev je enakovredna, za ustrezen vrstni red je potrebno uporabiti oklepaje.

Podobno kot pri kvalitativnem modelu (glej razdelek 3.1.4) lahko definiramo stopnjo zaupanja na izbranem vektorju zaupanja. Ta vključuje mnenje i-te entitete in mnenje celotne skupnosti do ciljne j-te entitete. Izračunano mnenje $\tau_{i,j}(\varphi) \in \mathcal{T}$ na izbranem vektorju zaupanja $\lambda_{i,j=const}$, je označeno kot $tvec_{i,j}(\varphi)$ z definicijo (3.21):

$$tvec_{i,j}(\varphi) = (\ldots((\lambda_{i,j} \oplus \lambda_{1,j}) \oplus \lambda_{2,j}) \ldots \oplus \lambda_{n-1,j}) \oplus \lambda_{n,j}, \ \lambda_{i,j} \neq -\infty, \ i \neq j. \tag{3.21}$$

V primeru, ko ne obstaja pretekla izkušnja med i-tim in j-tim agentom (t. i. $\lambda_{i,j} = -\infty$), lahko uporabimo princip tranzitivnosti iz razdelka 3.1.3, vendar samo v primeru, da obstaja matrika izkušenj zaupanja v priporočilo $\mathbb{M}'^r(\varphi)$. Če obstaja ne-ničelna pot $i \xrightarrow{+} j$, lahko uporabimo algoritem 3.2, ki izračuna izpeljano funkcionalno zaupanje $\lambda^*_{i,j}$. Vendar moramo za pravilen izračun določiti ustrezno algebrajsko strukturo (polkolobar) $(\Omega, \diamond, \circ)$ z ustreznimi operatorji iz množice $\mathcal{O}$. V poštev prideta samo operatorja za priporočilo $\otimes$ (t. i. vezni operator $\circ$) in konsenz $\oplus$ (t. i. agregacijski operator $\diamond$). Razlog je v dejstvu, da to nista običajna logična operatorja in se semantično nanašata na točno določene nosilce mnenj oz. prepričanj (t. i. izvorne entitete) [21]. Pri običajnih logičnih operatorjih je nosilec mnenja vedno isti (konjunkcija npr. združuje mnenja iste entitete za različne kontekste). Za izbrana operatorja $\otimes$ in $\oplus$ moramo torej preveriti algebrajsko strukturo $(\Omega, \oplus, \otimes)$.

- $(\Omega, \oplus)$ je komutativen monoid z enoto $\mathfrak{e} \in \Omega$:

  (1) $\forall x, y \in \Omega, x \neq y$ velja $x \oplus y = y \oplus x$;
  (2) enota $\mathfrak{e} = (0, 0, 1)$, kar potrdimo z reševanjem linearnega sistema enačb – če označimo $a = (b_a, d_a, u_a)$ in $\mathfrak{e} = (b_e, d_e, u_e)$, mora veljati $\mathfrak{e} \oplus a = a \oplus \mathfrak{e} = a$. Iz definicije za

operator konsenza [47], rešimo sistem sledečih enačb:

$$\frac{b_a u_e + b_e u_a}{u_a + u_e - u_a u_e} = b_a$$
$$\frac{d_a u_e + d_e u_a}{u_a + u_e - u_a u_e} = d_a$$
$$\frac{u_a u_e}{u_a + u_e - u_a u_e} = u_a$$

Dobimo rešitev $b_e = 0, d_e = 0, u_e = 1$ oz. $\mathfrak{e} = (0, 0, 1)$.

- $(\Omega, \otimes)$ je samo polgrupa, ker pa operacija $\otimes$ ni komutativna (t. i. $x \otimes y \neq y \otimes x$), ne obstaja enota $\mathfrak{e} \in \Omega : \mathfrak{e} \otimes x = x \otimes \mathfrak{e} = x$, zato ni monoid, prav tako ne obstaja absorbcijski element $\mathfrak{o} \in \Omega : \mathfrak{o} \otimes x = x \otimes \mathfrak{o} = \mathfrak{o}$.

- Operacija $\otimes$ ni distributivna čez $\oplus$; izračun izpeljanega funkcionalnega zaupanja $\lambda^*_{i,j}$ ni enostaven kot v primeru kvalitativnega modela, ker je potrebno posamezne tranzitivne verige preoblikovati v kanonične izraze – vsa mnenja morajo biti statistično neodvisna [34] (glej kriterij 4, razdelek 3.1.3).

Z izpeljano izkušnjo funkcionalnega zaupanja $\lambda^*_{i,j}$, lahko izračunamo stopnjo zaupanja $\tau^*_{i,j}$ na izbranem vektorju zaupanja na podoben način kot v izrazu (3.21), le da vrednost označimo s $tvec^*_{i,j}$, ki ima sledečo definicijo (3.22):

$$tvec^*_{i,j}(\varphi) = (\ldots((\lambda^*_{i,j} \oplus \lambda_{1,j}) \oplus \lambda_{2,j})\ldots \oplus \lambda_{n-1,j}) \oplus \lambda_{n,j},\ \lambda^*_{i,j} \neq -\infty,\ i \neq j \tag{3.22}$$

V primeru, da ne obstaja ne-ničelna tranzitivna pot med i-tim in j-tim agentom (t. i. $i \overset{+}{\nrightarrow} j$), stopnje funkcionalnega zaupanja ni mogoče določiti; v tem primeru se izračuna ugled j-tega agenta. Ugled j-tega agenta v kontekstu φ, je označen kot $rep_j(\varphi)$ z definicijo (3.23):

$$rep_j(\varphi) = (\ldots(\ldots((\lambda_{1,j} \oplus \lambda_{2,j}) \oplus \lambda_{3,j})\ldots \oplus \lambda_{k,j})\ldots \oplus \lambda_{n-1,j}) \oplus \lambda_{n,j},\ k \neq i \neq j \tag{3.23}$$

Sedaj lahko povzamemo opisana pravila za izračun stopnje zaupanja, ki jo ima agent a do agenta b – algoritem Γ.

Definicija 3.1.5.4 (Algoritem Γ)**.** Izračunana stopnja zaupanja agenta a do agenta b v kontekstu $\varphi \in \Phi$, je označena kot $trust_{a,b}(\varphi)$ z definicijo (3.24):

$$trust_{a,b}(\varphi) = \begin{cases} tvec_{a,b}(\varphi), & \lambda_{a,b}(\varphi) \neq -\infty \\ tvec^*_{a,b}(\varphi), & \lambda_{a,b}(\varphi) = -\infty,\ a \overset{+}{\rightarrow} b \\ rep_b(\varphi), & \lambda_{a,b}(\varphi) = -\infty,\ a \overset{+}{\nrightarrow} b \end{cases} \tag{3.24}$$

Podobno kot v kvalitativnem modelu, lahko izračunamo celotno stopnjo zaupanja ne glede na izbrani kontekst φ (t. i. “indikator obnašanja” izbranega agenta). V ta namen uporabimo

operator za konjunkcijo[19] $\wedge$ za vsak kontekst, kjer definiramo celotno stopnjo zaupanja $totalTrust_{a,b}$, kjer $\varphi_i \in \Phi, m = |\Phi|$ (3.25):

$$totalTrust_{a,b} = trust_{a,b}(\varphi_1) \wedge trust_{a,b}(\varphi_2) \ldots \wedge trust_{a,b}(\varphi_m) = \bigwedge_{\varphi_i \in \Phi} trust_{a,b}(\varphi_i) \tag{3.25}$$

3.1.6 Interakcijske politike

V predhodnih razdelkih definicije kvalitativnega in kvantitativnega modela smo izpustili opis oz. definicijo množice interakcijskih politik Π abstraktnega modela ATM. Element $\pi \in \Pi$ predstavlja interakcijsko politiko zaupanja, ki vsebuje pravila za dostop do storitev ali virov. Odločitve o prihodnjih interakcijah so odvisne od pravil politike in preteklih interakcij posameznega agenta. Če torej agent a dobi dovoljenje za dostop do vira r ob času t, potem je preteklo obnašanje a-ja vse do trenutka t v skladu s pravili (zahtevami) politike π_r.

V našem modelu je dimenzija konteksta $\varphi \in \Phi$ predstavljena kot dvojica $(Name, ES)$, kjer $Name$ določa ime (opis) konteksta, ES pa predstavlja dogodkovno strukturo – množico dogodkov (glej definicijo 3.1.2.6). Posamezna interakcija med entitetama a in b ob določenem času t in kontekstu φ, je določena kot $x_{a,b} = (\mathcal{X}_{a,b}(t), \omega_{a,b}(t, \varphi))$ (glej definicijo 3.1.2.9), kjer je $\mathcal{X}_{a,b}(t) \in \mathcal{C}^0_{\varphi.ES}$ končna konfiguracija dogodkov (glej definicijo 3.1.2.8). To je osnova za modeliranje zgodovine interakcij (opisuje obnašanje entitet), ki je sekvenca interakcij s končnimi konfiguracijami:

$$h_{a,b}(\varphi) = x_{a,b}(t_1, \varphi) x_{a,b}(t_2, \varphi) \ldots x_{a,b}(t_n, \varphi),\ x_{a,b}(t_i) \in \mathcal{C}^0_{\varphi.ES} \times \Omega, t_i \in T_{a,b}$$

(glej definicijo 3.1.2.10) in služi za sklepanje o prihodnjih interakcijah na podlagi definiranih pravil politike. Ta pravila (zahteve) so opisana z različnimi formalnimi deklarativnimi jeziki; v našem primeru se bomo omejili na linearno različico temporalne logike (angl. Linear Temporal Logic – LTL) za preteklost (angl. Pure Past LTL – PPLTL) [76, 86]. Obnašanje agenta je predstavljeno z linearnim zaporedjem množic dogodkov – zgodovino interakcij, kar predstavlja ustrezen model za PPLTL.

Modeli LTL so v osnovi *Kripke-ovi modeli* [87, 76], ki so zelo podobni naši strukturi interakcijske zgodovine h. V osnovi LTL vsebuje samo časovne operatorje za prihodnost; dodani časovni operatorji za preteklost omogočajo bolj zgoščeno in naravno specifikacijo modelov kot LTL [88]. Izrazni moči LTL in PPLTL sta enaki, prav tako je problem preverjanja modelov (angl. model checking) PSPACE-poln problem [89]. Znotraj tega razdelka bomo opisali osnovne koncepte PPLTL za opis interakcijskih politik $\pi \in \Pi$ in postopek za preverjanje modelov.

3.1.6.1 Sintaksa PPLTL

Sintaksa jezika za PPLTL je v našem primeru prirejena in parametrična glede na dogodkovno strukturo $ES = (E, \preceq, \bowtie)$. Sintaksa je sestavljena iz atomarnih trditev $e, e', \ldots$ za dogodke

[19] Lahko bi uporabili tudi operator za disjunkcijo.

$e \in E$, običajnih logičnih operatorjev $\neg, \wedge, \vee, \rightarrow$ ter temporalnih operatorjev $\mathsf{G}^{-1}, \mathsf{F}^{-1}, \mathsf{X}^{-1}, \mathsf{S}$. Sintakso označimo kot $\mathcal{L}_{ES}$ [76] in je podana v Backus-Naur obliki (BNF) (3.26):

$$\pi ::= \texttt{true}|\texttt{false}|e|\Diamond e|\pi_1 \wedge \pi_2|\pi_1 \vee \pi_2|\neg\pi|\pi \rightarrow \pi|\mathsf{G}^{-1}\pi|\mathsf{F}^{-1}\pi|\mathsf{X}^{-1}\pi|\pi_0\mathsf{S}\pi_1 \tag{3.26}$$

Izraza e in $\Diamond e$ sta *atomarni izjavi* oz. *formuli*. Formula e je resnična (angl. true) v izbrani pretekli interakciji, če obstaja dogodek e znotraj interakcije. Formula $\Diamond e$ (e je *možen*) je resnična, če se dogodek e lahko še pojavi v izbrani interakciji. Temporalni operator G^{-1} pomeni "*vedno v preteklosti*" (angl. always in the past), operator F^{-1} "*enkrat v preteklosti*" (angl. at some point in the past), operator X^{-1} "*zadnjič*" (angl. last time) in operator S "*odtlej (od takrat)*" (angl. since). Tabela 3.3 prikazuje pravila delovanja omenjenih operatorjev ob različnih časovnih intervalih t_i. Nekatere od logičnih in temporalnih operatorjev lahko izrazimo z drugimi. To

Čas	t_0	t_1	t_2	t_3	t_{Now}	Opis
π_0	1	1	0	0	-	
π_1	1	0	1	0	-	
$\mathsf{G}^{-1}\pi_0$	-	1	1	0	0	π_0 velja v vseh preteklih interakcijah
$\mathsf{F}^{-1}\pi_0$	-	1	1	1	1	π_0 velja v eni od preteklih interakcij
$\mathsf{X}^{-1}\pi_0$	-	1	1	0	0	π_0 velja v zadnji interakciji
$\pi_0\mathsf{S}\pi_1$	-	1	1	0	0	π_1 velja v eni od preteklih interakcij, π_0 velja za vse interakcije od takrat naprej

Tabela 3.3: Interpretacija temporalnih operatorjev.

pomeni, da sintakso $\mathcal{L}_{ES}$, lahko izrazimo z minimalnim naborom operatorjev. Veljajo sledeče zveze za določen $e \in E$ (3.27):

$$\begin{aligned}
\texttt{true} &\equiv e \vee \neg e \\
\texttt{false} &\equiv \neg\texttt{true} \\
\pi_0 \vee \pi_1 &\equiv \neg(\neg\pi_0 \wedge \neg\pi_1) \\
\pi_0 \rightarrow \pi_1 &\equiv \neg\pi_0 \vee \pi_1 \\
\mathsf{F}^{-1}(\pi) &\equiv \texttt{true}\ \mathsf{S}\ \pi \\
\mathsf{G}^{-1}(\pi) &\equiv \neg\mathsf{F}^{-1}(\neg\pi)
\end{aligned} \tag{3.27}$$

Sintakso jezika razširimo s predikatom $\texttt{Trust}(\tau)$, ki določa minimalno stopnjo zaupanja $\tau \in \mathcal{T}$. Stopnja se izračuna preko opisanih algoritmov v prejšnjih razdelkih (3.1.4 in 3.1.5). Sintakso z minimalnim naborom označimo kot $\mathcal{L}_{ES}^{min}$ in je podana kot BNF (3.28):

$$\pi ::= e \mid \Diamond e \mid \pi_1 \wedge \pi_2 \mid \neg\pi \mid \mathsf{X}^{-1}\pi \mid \pi_0\mathsf{S}\pi_1 \mid \texttt{Trust}(\tau) \tag{3.28}$$

3.1.6.2 Semantika

Struktura za sintakso $\mathcal{L}_{ES}^{min}$, kjer je $ES = (E, \preceq, \bowtie)$ dogodkovna struktura, je ne-prazna zgodovina interakcij znotraj ES v kontekstu φ, $h(\varphi) \in \mathcal{C}_{\varphi.ES}^{0*} \times \Omega$, kjer je $\mathcal{C}_{\varphi.ES}^{0*}$ *Kleenova*

ovojnica [85] vseh možnih končnih konfiguracij (3.29):

$$\mathcal{C}^{0*}_{\varphi.ES} = \bigcup_{\mathcal{X}_i \in \mathcal{C}^{0}_{\varphi.ES}} \mathcal{X}_i \tag{3.29}$$

Velja spomniti, da se zgodovina interakcij vedno nanaša na izvorno entiteto a in ciljno entiteto b, med katerima so potekale interakcije. Definiramo relacijo *ustreznosti* $\models$ med strukturami in politikami: $h \models \pi$ pomeni, da zgodovina h[20] ustreza (zadošča) zahtevam politike π. Relacija je definirana od konca proti začetku zgodovine interakcij h; torej $h \models \pi$ natanko tedaj, če $(h, |h|) \models \pi$ (izraz (h, i) označuje interakcijo v zgodovini $h(\varphi)$ ob času $t_i : x(t_i, \varphi)$). Naj bo $h = x_1 x_2 \ldots x_N \in \mathcal{C}^{0*}_{\varphi.ES} \times \Omega$, kjer $x_i \equiv x(t_i, \varphi), N > 0$. V nadaljevanju za oznako konfiguracije $x_i.\mathcal{X}$ znotraj interakcije x_i zaradi preglednosti uporabljamo kar skrajšano oznako $\mathcal{X}_i$. Sledi definicija pravil za $(h, 1) \models \pi$ s strukturalno indukcijo znotraj π (3.30).

$$\begin{array}{lll}
(h,1) \models e & \text{iff} & e \in \mathcal{X}_1 \\
(h,1) \models \Diamond e & \text{iff} & \neg(e \bowtie e'),\ e' \in \mathcal{X}_1 \\
(h,1) \models \pi_0 \wedge \pi_1 & \text{iff} & (h,1) \models \pi_0 \wedge (h,1) \models \pi_1 \\
(h,1) \models \neg\pi & \text{iff} & (h,1) \not\models \pi \\
(h,1) \models \mathsf{X}^{-1}\pi & \text{iff} & \texttt{false} \\
(h,1) \models \pi_0 \mathsf{S} \pi_1 & \text{iff} & (h,1) \models \pi_1
\end{array} \tag{3.30}$$

Če predpostavimo $1 < i \leq N$ in $(h, i-1) \models \pi$, potem lahko definiramo $(h, i) \models \pi$ preko strukturalne indukcije znotraj π (3.31).

$$\begin{array}{lll}
(h,i) \models e & \text{iff} & e \in \mathcal{X}_i \\
(h,i) \models \Diamond e & \text{iff} & \neg(e \bowtie e'),\ e' \in \mathcal{X}_i \\
(h,i) \models \pi_0 \wedge \pi_1 & \text{iff} & (h,i) \models \pi_0, (h,i) \models \pi_1 \\
(h,i) \models \neg\pi & \text{iff} & (h,i) \not\models \pi \\
(h,i) \models \mathsf{X}^{-1}\pi & \text{iff} & (h,i-1) \models \pi \\
(h,i) \models \pi_0 \mathsf{S} \pi_1 & \text{iff} & \big((h,i) \models \pi_1\big) \ \vee\ \big([(h,i-1) \models \pi_0 \mathsf{S} \pi_1] \ \wedge\ [(h,i) \models \pi_0]\big)
\end{array} \tag{3.31}$$

Prikazana semantika velja samo za ne-prazno zgodovino interakcij h. To pomeni, da preverjanje politike π ni možno, če ni bilo predhodnih interakcij. V tem primeru je odločitev odvisna od samega agenta. Semantiko lahko razširimo tudi za prazno zgodovino ϵ: $\epsilon \models \pi$ iff $\emptyset \models \pi$.

Sedaj lahko prikažemo primer enostavne interakcijske politike v opisanem jeziku PPLTL. Že ob definiciji dogodkovne strukture (glej razdelek 3.1.2, definicija 3.1.2.7), smo navedli primer scenarija spletnega mesta eBay s perspektive kupca. Dogodkovno strukturo z ustreznimi možnimi dogodki prikazuje slika 3.6. Sledi primer politike kupca za odločitev o vstopu na licitacijo predmeta prodajalca. Pravilo določa, da je prodajalec v preteklosti vedno poslal kupljeni predmet kupcu pravočasno in je stopnja zaupanja med kupcem in prodajalcem vsaj $\tau_0 \in \mathcal{T}$:

$$\pi^{lic} \equiv \neg \mathsf{F}^{-1}(\texttt{zamuda}) \wedge \texttt{Trust}(\tau_0)$$

[20] Za zgodovino interakcij $h(\varphi)$ lahko pišemo samo h, pomen je enak.

3.1.6.3 Verifikacija

Problem verifikacije (preverjanja) dane politike glede na zgodovino interakcij, predstavlja *problem preverjanja modela* (angl. model-checking problem); če imamo $h \in \mathcal{C}^{0*}_{\varphi.ES} \times \Omega$ in $\pi \in \Pi$, ali velja $h \models \pi$? Obstajajo različne metode preverjanja modelov (npr. končni avtomati) v LTL s poudarkom na preteklosti [88, 90].

V našem primeru se osredotočimo na dinamično tehniko preverjanja [76] z nekaterimi spremembami za naš model zaupanja. Vsaka odločitev agenta oz. entitete o prihodnji interakciji je povezana s preverjanjem ustreznosti pravil politike s preteklim obnašanjem. Operacijo preverjanja označimo kot `Check()`, ki na podlagi dane zgodovine interakcij h in politike π vrne, ali velja $h \models \pi$. Ideja za preverjanje temelji na učinkovitem algoritmu Havelund in Rosuja [91] za preverjanje LTL v preteklosti. Zaradi rekurzivne semantike jezika PPLTL, je preverjanje $(h, m) \models \pi$ možno, če:

(1) velja $(h, m-1) \models \pi_j$ za vse pod-formule π_j formule π;
(2) velja $(h, m) \models \pi_i$ za vse pod-formule π_i formule π.

Veljavnost atomarnih pod-formul π znotraj (h, m), se lahko izračuna neposredno iz konfiguracije $\mathcal{X}_m$, ki jo vsebuje m-ta interakcija x_m v zgodovini h. Če je npr. $\pi_3 = \mathsf{X}^{-1}\pi_4 \wedge e$, potem $(h, m) \models \pi_3$ natanko tedaj, ko $(h, m-1) \models \pi_4$ in $e \in \mathcal{X}_m$. Informacija za odločitev, ali velja $(h, m) \models \pi$, se lahko shrani v dva ločena seznama (polji) B_{last} in B_{cur} z logičnimi vrednostmi `true/false`. Obe polji sta indeksirani z indeksi pod-formul π: $B_{last}[j]$ je `true`, ko $(h, m-1) \models \pi_j$ ter $B_{cur}[i]$ je `true`, ko $(h, m) \models \pi_i$. S podanim poljem B_{last} in trenutno konfiguracijo $\mathcal{X}_m$, lahko zgradimo polje B_{cur}, začenši z atomarnimi formulami (imajo največji indeks) proti sestavljenim tako, da na koncu pridemo do indeksa 0, kjer vrednost $B_{cur}[0]$ pomeni veljavnosti $(h, m) \models \pi$.

Algoritem mora razbiti formulo politike π na seznam pod-formul; metoda se imenuje `Subformula(`π`)`. Naj obstaja $n+1$ pod-formul formule π in naj bo $\pi_0 = \pi$. Seznam je enak $\pi_0, \pi_1, \pi_2, \ldots, \pi_n$ in če je π_i pod-formula π_j, potem je $i > j$ (indeks pod-formule je večji). Za preverjanje $(h, k) \models \pi_0 = \pi$ pogledamo vrednost pri indeksu 0 polja $B_k[0]$. Algoritem predvideva obstoj struktur – razredov za opis abstraktnega jezika politike oz. ustrezne operatorje: `Formula`, `EventFormula`, `PossibleFormula`, `Conjunction`, `Negation`, `LastTime`, `Since`, `Trust`. Glede na podano semantiko obstajajo tri metode, ki preverijo konfiguracijo $\mathcal{X}_k$ z vsemi pod-formulami π in rezultat shranijo v polje B_{cur} z vrednostmi `true/false`:

a) `BaseSem(Interaction x,Formula` π`, boolean[]` B_{cur}`)`: vzame interakcijo x in formulo π ter napolni polje $B_{cur}[i]$ glede na veljavnost $x.\mathcal{X} \models \pi_i$;

b) `InductiveSem(boolean[]` B_{last}`, Interaction x, Formula` π`, boolean[]` B_{cur}`)`: podobno kot metoda `BaseSem`, le da implementira induktivno – rekurzivno semantiko;

c) `CommonSem(Interaction x, Formula` π_i`, boolean[]` B_{cur}`, Formula` π`)`: vsebuje logiko za preverjanje pod-formul v `BaseSem` in `InductiveSem`. Če je pod-formula π_i

predikat `Trust(`τ`)`, potem se za veljavnost $x.\mathcal{X} \models \pi_i$ izvede t. i. implementacija algoritma Γ za izračun zaupanja (glej razdelek 3.1.4 oz. 3.1.5) kot metoda na spletni storitvi (več o tem v sledečih poglavjih). Veljavnost velja v primeru, ko metoda vrne stopnjo zaupanja, ki je večja ali enaka od praga $\tau \in \mathcal{T}$.

Psevdokoda omenjenih metod je osnovana na [76] in prirejena za naš model zaupanja ATM; prikazana je kot algoritem 3.4 in je osnova za metodo preverjanja `Check()`. Algoritem 3.5 vsebuje nekatere potrebne strukture[21], globalne spremenljivke, inicializacijo zgodovine interakcij (metoda `Init()`) ter preverjanje skladnosti (metoda `Check()`). Vsaka interakcija x_i znotraj zgodovine h (vektor `h`) ima pripadajoče polje B_i (vektor polj `B_arrays`), katerega dolžina je enaka številu pod-formul formule π. Zgodovina interakcij je implicitno podana v matrikah zaupanja $\mathbb{M}(t, \varphi)$, kjer so shranjene posamezne interakcije med entitetami $x_{a,b}(t, \varphi)$. Te interakcije v fazi inicializacije shranimo v vektor `h` in preverimo veljavnost $(h, i) \models \pi$ za vsako interakcijo x_i. Zgodovina interakcij se vedno nanaša na izvorno in ciljno entiteto (npr. $a, b \in \mathcal{A}$); predpostavka je, da politika π pripada izvorni entiteti a, ki dostopa do b-ja ali obratno – b dostopa do a-ja (oz. njenih virov in storitev).

Časovno kompleksnost [92] algoritma označimo z oznako velikega O-ja, ki je definiran kot [93, 85] (3.32):

$$f(n) = O(g(n)) \Leftrightarrow (\exists c, n_0(c))(\forall n)(n \geq n_0 \Rightarrow f(n) \leq cg(n)) \tag{3.32}$$

V našem primeru je kompleksnost inicializacije $O(|h| \cdot |\pi|)$, kompleksnost metode `Check()` je po inicializaciji $O(1)$; predpostavljamo tudi, da je kompleksnost za izračun pod-formul `Subformula(`π`)` konstantna. Torej, če se zgodovina interakcij ne spreminja pogosto (npr. klic inicializacije samo ob določenih intervalih, možnost predpomnenja), je preverjanje skladnosti s politiko π precej učinkovita. Seveda delež kompleksnosti prispeva tudi implementacija algoritma Γ za izračun stopnje zaupanja (glej razdelek 3.1.4 oz. 3.1.5) – v primeru, da v politiki π nastopa predikat `Trust(`τ`)`.

[21] Samo struktura `Interaction`.

```
void BaseSem(Interaction x,Fomula π,boolean[] B_cur) {
  integer i = Subformula(π).Size-1;
  Formula π_i;
  while (i ≥ 0) {
    π_i = p.FormulaByIndex(i);
    if (π_i is LastTime) {
      B_cur[i] = false;
    } else if (π_i is Since) {
      Since s = (Since)π_i;
      Formula π_2 = s.SecondArg();
      B_cur[i]=B_cur[π.IndexOf(π_2)];
    } else
      B_cur[i] = CommonSem(x,π_i,B_cur,π);
    i--;
  }
}

void InductiveSem(boolean[] B_last,Interaction x,Fomula π,boolean[] B_cur) {
  integer i = Subformula(π).Size-1;
  Formula π_i;
  while (i ≥ 0) {
    π_i = π.FormulaByIndex(i);
    if (π_i is LastTime) {
      Formula f = ((LastTime)π_i).Inner();
      B_cur[i]=B_last[π.IndexOf(f)];
    } else if (π_i is Since) {
      Since s = (Since)π_i;
      Formula π_1 = s.FirstArg();
      Formula π_2 = s.SecondArg();
      B_cur[i]=B_cur[π.IndexOf(π_2)] Or (B_last[i] And B_cur[π.IndexOf(π_1)]);
    } else
      B_cur[i] = CommonSem(x,π_i,B_cur,π);
    i--;
  }
}

boolean CommonSem(Interaction x,Fomula π_i,boolean[] B_cur,Formula π) {
  if (π_i is EventFormula) {
    Event e = ((EventFormula)π_i).Event();
    return x.X.Contains(e);
  } else if (π_i is PossibleFormula) {
    Event e = ((PossibleFormula)π_i).Event();
    return x.X.IsPossible(e);
  } else if (π_i is Conjunction) {
    Conjunction c = (Conjunction)π_i;
    Formula π_1 = c.FirstArg();
    Formula π_2 = c.SecondArg();
    return B_cur[π.IndexOf(π_1)] And B_cur[π.IndexOf(π_2)];
  } else if (π_i is Negation) {
    Formula f = ((Negation)π_i).Inner();
    return Not B_cur[π.IndexOf(f)];
  } else if (π_i is Trust) {
    Trust t = (Trust)π_i;
    TrustLevel level = ServiceProxy.Trust(a, b, x.context);
    return level ≥ t.level;
  } else throw new Exception("Wrong formula syntax:  " + π_i.ToString());
}
```

Algoritem 3.4: Metode za implementacijo semantike PPLTL.

Struktura za predstavitev interakcije:

```
type Interaction {
  Configuration X;
  TrustLevel omega;
  Context context;
  DateTime time;
  Integer timeIndex;
}
```

Ustrezne globalne spremenljivke:

```
Vector h = new Vector of type Interaction;
Vector B_arrays = new Vector of type Boolean Array;
Formula π;
EventStructure ES;
Entity a; /* izvorna entiteta, lastnik politike */
Entity b; /* ciljna entiteta */
boolean[] B_0; /* polje v primeru prazne zgodovine */
```

Inicializacija spremenljivk, zgodovine interakcij:

```
void Init(Entity a, Entity b, Context φ, Formula π) {
  this.a = a;
  this.b = b;
  this.ES = φ.ES;
  this.π = π;
  /* polje B_0 vsebuje vrednosti ∅ ⊨ π_i */
  B_0 = new boolean[Subformula(π).Size];
  BaseSem(ES.EmptyConfiguration(), π, B_0);
  /* branje zgodovine interakcij */
  foreach (Interaction x_{a,b} in ⋃_{t∈T_{a,b}} M(t,φ)[a,b]) {
    boolean[] B_new = new boolean[Subformula(π).Size];
    if (Not B_arrays.IsEmpty()) {
      boolean[] B_last = (boolean[])B_arrays.LastElement();
      InductiveSem(B_last, x_{a,b}, π, B_new);
    } else {
      BaseSem(x_{a,b}, π, B_new);
    }
    h.Add(x_{a,b});
    B_arrays.Add(B_new);
  }
}
```

Preverjanje skladnosti:

```
boolean Check() {
  if (B_arrays.IsEmpty()) return B_0[0];
  return ((boolean[])B_arrays.LastElement())[0];
}
```

Algoritem 3.5: Preverjanje skladnosti s politko π.

Poglavje 4

Storitveno usmerjene arhitekture

4.1 Splošno o SOA

Storitvene usmerjene arhitekture (SOAs) predstavljajo pomemben koncept pri elektronskem poslovanju (e-poslovanju) v porazdeljenih okoljih. Pri e-poslovanju se interakcije (transakcije) izvajajo med entitetami, ki so ločene v prostoru in praviloma še niso bile v neposrednem kontaktu. Odločitve za medsebojno poslovanje temeljijo na oceni tveganja. Posledica je, da je zaupanje poleg varnosti temeljni pojem za učinkovito izvajanje in odločanje pri poslovnih transakcijah v okoljih e-poslovanja – torej tudi v SOA. V prejšnjem poglavju (poglavje 3), smo predstavili abstrakten model zaupanja $ATM = (\mathbf{G}, \Omega, \mathcal{T}, \mathcal{F}, \mathcal{O}, \Gamma, \Pi)$ s formalno definicijo vseh njegovih komponent v kvalitativni in kvantitativni perspektivi. Zanimivost modela je, da ne podpira samo kognitivnih agentov (t. i. *inteligentne* entitete), ampak tudi tehnološke komponente ali storitve. To narekuje, da je model uporaben tudi v okoljih SOA.

Današnji informacijski sistemi so v smislu e-poslovanja zelo kompleksni; predvsem so v ospredju vprašanja, ki se tičejo porazdeljenih aplikacij, integracije z obstoječimi sistemi, različne platforme, protokoli in naprave. SOA naj bi v večji meri reševala omenjena vprašanja. Sicer koncept SOA ni nov, uporabljen je bil že v komponentnih tehnologijah, kot so npr. DCOM[(1)]/COM+, CORBA[(2)] in RMI[(3)], kjer gre v osnovi za oddaljeno klicanje procedur (angl. Remote Procedure Call – RPC). SOA v splošnem pomeni zbirko storitev (t. i. komponent sistema), ki komunicirajo med seboj preko izmenjave sporočil. Komunikacija lahko pomeni prenašanje enostavnih podatkov, ali pa usklajevanje dveh ali več storitev, ki izvajajo neko aktivnost. Ključni poudarek SOA je na definiciji načina integracije avtonomnih obstoječih ali na novo razvitih aplikacij (imenovanih storitev) v celovit sistem s posebnim poudarkom na modelu komunikacije, podpori različnih transportnih mehanizmov, zagotavljanju varnosti, podpora transakcijam, zanesljivosti, koordinaciji in kompoziciji [4].

(1) Distributed Component Object Model.
(2) Common Object Request Broker Architecture.
(3) Remote Method Invocation.

4.2 Storitve

Storitve so osnovni avtonomni gradniki SOA. Opravljajo natanko določeno operacijo, za katero imajo natančno definiran vmesnik, ki je natančen opis sporočil za izmenjavo. Do vmesnika se lahko dostopa preko omrežja z uporabo standardnih protokolov in podatkovnih tipov. Storitve v SOA naj bi imele različne lastnosti: (a) avtonomnost, (b) meje storitev so eksplicitne, (c) delijo se na atomarne in kompozitne, (d) izpostavljajo operacije z izmenjavo sporočil po načelu šibke sklopljenosti (angl. loosely coupled), (e) sporočila so natančno definirana, (f) izmenjujejo samo sporočila, ne pa obnašanja [3, 4].

Realizacija SOA je povezana z uporabo ustreznih tehnologij. Trenutno so *spletne storitve* (angl. web services) [94] najprimernejša tehnologija za implementacijo SOA. Omogočajo povezljivost med informacijskimi sistemi ne glede na programski jezik, platformo ali operacijski sistem. Temeljijo na naboru različnih standardov, ki stremijo k odprtosti in povezljivosti (angl. interoperability). Enostavna tehnična definicija spletnih storitev pravi, da so spletne storitve komponente, ki komunicirajo preko protokola SOAP [95]. Spletne storitve so avtonomne programske enote s sledečimi lastnostmi:

- Nudijo funkcionalnost preko odprtega standardnega spletnega protokola – protokol SOAP (angl. Simple Object Access Protocol) [96].
- Njihovi podatki so predstavljeni v predpisanem standardu (ustrezen tipski sistem – so tipizirana), za predstavitev se uporabljajo sheme XML – standard XSD (XML Schema Definition) [97].
- Zunanjemu svetu so predstavljene preko standardnega opisnega jezika, ki opisuje vmesnike in njihovo obnašanje; opisni jezik temelji na XML in se imenuje WSDL (Web Services Description Language) [98].
- Na voljo so končnim uporabnikom preko posebnega mehanizma za odkrivanje (lociranje); mehanizem se imenuje UDDI (Universal Discovery Description and Intergration) [99].
- Med seboj izmenjujejo samo sporočila (podatke), ne pa tudi obnašanja (t. i. razredov oz. metod). Storitve namreč same vzdržujejo svoje stanje, če ga potrebujejo.
- Obdelajo sporočila po načelih *šibke sklopljenosti* (angl. loose coupling) – vmesniki storitev niso vezani na vmesnike ostalih komponent, ki komunicirajo s storitvami.

Naš cilj je preučitev obstoječe situacije znotraj SOA ter integracija mehanizmov zaupanja v obstoječe standarde. Pregleden pogled spletnih storitev, njenih standardov in protokolov, podaja vir [100]. Velja omeniti, da osnovne specifikacije (XML, SOAP, WSDL, UDDI) same po sebi ne zadoščajo, ker v celoti ne obravnavajo področij, ki so poglavitna za učinkovito e-poslovanje (varnost, transakcije, zanesljivost). Zato je potrebno uporabiti sklad oz. sklope specifikacij WS-* organizacij W3C(4) in OASIS(5), ki se nanašajo na različna področja za večjo povezljivost

(4) The World Wide Web Consortium.
(5) Organization for the Advancement of Structured Information Standards.

(interoperabilnost) spletnih storitev (podroben pregled in opis specifikacij je podan v [100, 3, 101]):

a) *XML*:
   - XML 1.0,
   - imenski prostori XML,
   - XML Infoset.

b) *Metapodatki (Metadata)*:
   - WSDL,
   - UDDI,
   - WS-Policy,
   - WS-PolicyAssertions,
   - WS-PolicyAttachments,
   - WS-MetadataExchange.

c) *Izmenjava sporočil (Messaging)*:
   - SOAP,
   - WS-Addressing,
   - MTOM,
   - WS-Eventing.

d) *Varnost (Security)*:
   - WS-Security,
   - WS-SecureConversation,
   - WS-SecurityPolicy,
   - WS-Trust,
   - WS-Federation.

e) *Zanesljivost (Reliable Messaging)*:
   - WS-ReliableMessaging.

f) *Transakcije (Transactions)*:
   - WS-AtomicTransaction,
   - WS-BusinessActivity,
   - WS-Coordination.

V našem primeru je v središču predvsem področje varnosti (t. i. trdni varnostni mehanizmi) [102], kjer obstoječi standardi (WS-Security, WS-Trust, WS-Federation) rešujejo problem varne izmenjave SOAP sporočil med samostojnimi spletnimi storitvami. Podpora zaupanja kot mehkega varnostnega (t. i. socio-kognitivnega) mehanizma ni obravnavan v nobenem od obstoječih SOA standardov. Naš cilj je vpeljava teh mehanizmov v obstoječe standarde (npr. WS-Trust). Zanimivo je tudi področje metapodatkov, predvsem standardi za opis zahtev in zmožnosti spletnih storitev preko ustreznih politik (npr. WS-Policy*). To pomeni, da lahko tudi interakcijske politike $\pi \in \Pi$ (glej razdelek 3.1.6) modela ATM preslikamo v omenjene

standarde. To omogoča tudi dejansko podpora SOA aplikacij z našim modelom zaupanja kot psihološko-sociološkim mehanizmom [63, 62].

4.2.1 Kompozicija storitev

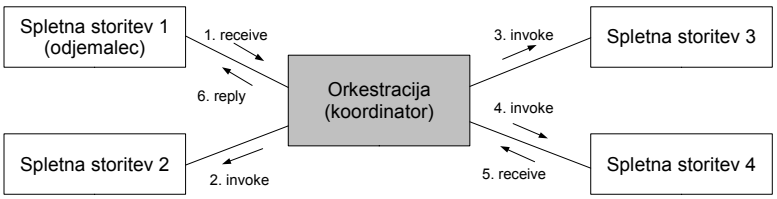
Izpostavljanje funkcionalnosti posamezne spletne storitve za e-poslovanje ni dovolj; potrebno je sestaviti funkcionalnost več samostojnih storitev v ustreznem vrstnem redu – tako pridemo do definicije poslovnih procesov (kompozitov) oz. ustrezne procesne logike. Ideja kompozicije je zajem poslovne logike in vmesnikov obnašanja storitve v logične enote – procesne modele. Ti modeli so lahko izraženi na različnih nivojih abstrakcije vse do nivoja izvršljivih modelov. Izpostavljanje funkcionalnosti v obliki ustrezno načrtovanih storitev je eden od pomembnih vidikov realizacije SOA in ga imenujemo pristop *od spodaj navzgor* (ang. bottom-up). Je pogoj za drugi pristop – kompozicijo storitev ali procesnim vidikom realizacije SOA in se imenuje pristop *od zgoraj navzdol* (angl. top-down). S kompozicijo storitev SOA integrira funkcionalnosti oz. storitve v nove kontekste izvajanja – poslovne procese. Ta vidik realizacije SOA predstavlja torej neke vrste "lepilo" med storitvami na eni in poslovnimi procesi na drugi strani.

Združevanje oz. kompozicija storitev je možna na dva načina: z *orkestracijo* (angl. orchestration) in s *koreografijo* (angl. choreography) [103]:

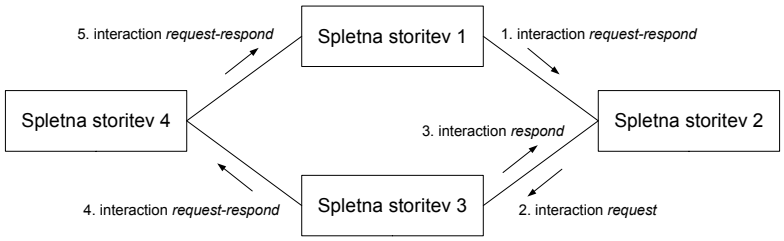
- Orkestracija se nanaša na poslovni proces, ki je lahko sestavljen iz notranjih (znotraj organizacije) in zunanjih (zunaj organizacije) spletnih storitev. Poslovni proces je opisan na nivoju izmenjave sporočil in vrstnega reda izvajanja. Poslovni proces nadzoruje centralni koordinator (agent, proces), ki skrbi za nadzor in koordinacijo vseh udeleženih spletnih storitev (slika 4.1).

- Koreografija se nanaša na opis interakcij oz. sodelovanja med spletnimi storitvami. Ni centralnega koordinatorja, temveč vsaka spletna storitev ve, katero operacijo mora izvršiti in s katero storitvijo se mora povezati. Vse udeležene spletne storitve se obravnavajo enako (slika 4.2).

- Orkestracija in koreografija se nanašata na različne perspektive. Orkestracija je usmerjena na notranje obnašanje poslovnega procesa (t. i. *procesni model*). Koreografija je usmerjena na globalno (zunanjo) perspektivo opazovanja interakcij (t. i. *interakcijski model*).



Slika 4.1: Kompozicija spletnih storitev z orkestracijo.



Slika 4.2: Kompozicija spletnih storitev s koreografijo.

Omeniti velja še dve razliki poslovnih procesov glede na orkestracijo oz. koreografijo. *Izvršljivi* procesi omogočajo specifikacijo vseh podrobnosti poslovnih procesov. Temeljijo na vzorcu orkestracije in se lahko izvajajo na orkestracijskem strežniku. Na drugi strani so abstraktni procesi, ki omogočajo samo specifikacijo izmenjave javnih sporočil med udeleženimi stranmi. Temeljijo na vzorcu koreografije in niso izvršljivi.

Obstajajo različne tehnologije oz. standardi, ki podpirajo omenjene vzorce kompozicije spletnih storitev. Jezik BPEL (Business Process Execution Language) [104, 105, 106, 107] sledi vzorcu orkestracije, koreografija pa je podprta s strani jezikov WS-CDL (Web Services Choreography Description Language) [108, 109, 105] in WSCI (Web Service Choreography Interface) [110]. Velja poudariti, da koreografija v industriji nima take podpore, ki bi bila primerljiva z BPEL; le-ta predstavlja industrijski standard. Zato je tudi znotraj tega dela večji poudarek na orkestraciji. Kratek pregled obeh jezikov (struktura, sintaksa) je podan v dodatku: dodatek B (jezik BPEL), dodatek C (jezik WS-CDL).

4.2.2 BPEL

Jezik BPEL[(6)] temelji na sintaksi XML, ki uporablja sklad tehnologij spletnih storitev (SOAP, WSDL, UDDI, WS-ReliableMessaging [111], WS-Addressing [112], WS-Coordination [113] in WS-AtomicTransaction [102]). Ker proces BPEL komunicira z zunanjimi spletnimi storitvami, je močno odvisen od opisov storitev z jezikom WSDL.

Z BPEL sestavimo spletne storitve v poslovne procese. BPEL je namenjen definiciji procesnega toka (t. i. *workflow*) in koordinaciji. Poslovni proces, definiran z uporabo BPEL, določa procesno logiko vpletenih storitev in načine sodelovanja. Znotraj BPEL-a je logika interakcij med določeno storitvijo in njenim okoljem opisana s kompozicijo komunikacijskih akcij (npr. *send*, *receive*, ...). Te akcije so ustrezno povezane s kontrolnimi odvisnostmi znotraj procesnega toka; npr. zaporedno, paralelno, pogojno izvajanje. Procese, definirane v BPEL, lahko izvajamo na orkestracijskih strežnikih. Jezik BPEL poleg procesnega toka in koordinacije definira tudi semantiko, ki vključuje zaporedje izvajanja, transformacije podatkov, obravnavo napak ter kompenzacijske in korelacijske mehanizme.

V osnovi je poslovni proces BPEL graf aktivnosti, zato ga lahko izrazimo tudi preko *diagramov aktivnosti* jezika UML (angl. Unified Modeling Language) [114]. Proces BPEL

(6) Nastal je iz jezikov WSFL (IBM) in XLANG (Microsoft).

je sestavljen iz korakov, ki jih sestavljajo posamezne aktivnosti (osnovne in strukturirane – za podrobnosti glej dodatek B). Osnovne aktivnosti so namenjene nalogam kot:

- proženje drugih spletnih storitev preko `<invoke>`;
- čakanje odjemalca, da proži poslovni proces s pošiljanjem sporočila preko `<receive>`;
- generiranje odgovora za sinhrone operacije preko `<reply>`;
- prirejanje vrednosti spremenljivkam preko `<assign>`;
- čakanje za določen čas preko `<wait>`;
- končanje celotnega procesa preko `<terminate>`.

Strukturirane aktivnosti omogočajo združevanje osnovnih in izvedbo sestavljenih nalog:

- definicija urejenega zaporedja aktivnosti preko `<sequence>`;
- definicija vzporednih izvajajočih aktivnosti preko `<flow>`;
- pogojno vejanje aktivnosti preko `<switch>`;
- iteracije oz. zanke preko `<while>`;
- izbira med več alternativnimi možnostmi preko `<pick>`.

Znotraj procesa BPEL je možno definirati tudi povezave z udeleženimi spletnimi storitvami (t. i. partner links) preko `<partnerLink>` in deklarirati spremenljivke preko `<variable>`.

4.2.3 Formalna predstavitev

Orkestracija in jezik BPEL kot pomemben del SOA mora podpirati aktivnosti, ki se tičejo komunikacije z drugimi storitvami, kot tudi aktivnosti za kontrolo delovnega toka poslovnega procesa. BPEL v ta namen nudi osnovne in strukturirane aktivnosti (glej dodatek B). Osnovne aktivnosti se lahko pojmujejo kot primitivne komponente, ki komunicirajo z zunanjim svetom in samim procesom. Strukturne aktivnosti nadzirajo celotni procesni tok z vrstnim redom aktivnosti. To omogoča modeliranje kompleksnih poslovnih procesov, ki jih je težko formalno analizirati, kajti semantika BPEL ni vedno jasno izražena, saj specifikacija BPEL nima formalne osnove.

Kljub temu nudi BPEL različne mehanizme za ravnanje z različnimi situacijami. Izjemne situacije obravnavajo mehanizmi za obvladovanje napak (izjem), dogodkov in kompenzacije. Mehanizem za napake je aktiven med izvajanjem aktivnosti, mehanizem za kompenzacijo pa ob končanju aktivnosti. Pojem kompenzacije je povezan s transakcijami med udeleženci poslovnega procesa. Lastnosti ACID (angl. Atomicity, Consistency, Isolation, Durability) v smislu klasičnih transakcij niso dovolj, ker imamo opravka z dolgo trajajočimi transakcijami[(7)] (angl. long running transactions). Kompenzacija pomeni razveljavitev že izvedenih akcij (glej dodatek B.1.1.5) in je ena od možnih metod za implementacijo WS-BusinessActitivity (temelji na WS-AtomicTransaction in WS-Coordination).

[(7)] To vrsto transakcij obravnava standard WS-BusinessActivity (WS-BA) [115].

Jezik BPEL ne vsebuje formalne semantike in glede na številne vidike, ki jih izpostavlja, je zelo težko formalno analizirati celotno obnašanje poslovnega procesa. Mnogi raziskovalci si prizadevajo postaviti formalne osnove za določena področja znotraj BPEL:

- dinamika aktivnosti, sočasnost, časovna analiza procesa, dosegljivost aktivnosti, specifikacija obnašanja, odkrivanje mrtvih poti in zank [116, 117, 118, 119];
- korelacijski mehanizmi za opis sočasnega obnašanja procesnih instanc [120, 121];
- opis relacij in semantike procesnih elementov z ontologijo [122];
- modeliranje delovnega toka, formalna semantika kompozicijskih operatorjev, odkrivanje nekonsistentnosti, časovna analiza lastnosti procesa [123, 124];
- mehanizmi za obvladovanje napak, dogodkov in kompenzacije [125].

Seznam omenjenih področij je še daljši, vendar naš namen ni raziskovati vsake od njih. V splošnem jih lahko grupiramo v sklope, ki uporabljajo ustrezne formalne metode[(8)]:

- Različni koncepti *Petrijevih mrež* (angl. Petri Nets – PN) [123, 119, 117] za preverjanje statičnih lastnosti in verifikacijo kompozitov spletnih storitev (končnost, mrtve zanke, dosegljivost stanj, redukcija kompozitov).

- *Časovni* (angl. Timed Automata – TA) in *razširjeni končni avtomati* (angl. Extended Finite Automata – EFA) [124, 116, 118] za preverjanje končnosti, odkrivanja mrtvih zank, dosegljivosti stanj, časovna analiza izvajanja kompozitnih storitev.

- *Abstraktne* in *procesne algebre* (npr. Calculus of Communicating Systems – CCS, π-calculus) [121, 120, 125, 126] za opis procesnih modelov in interakcij znotraj kompozita storitev, opis različnih lastnosti mehanizmov za obvladovanje napak, dogodkov, sočasnost.

Obširen pregled nekaterih vidikov znotraj BPEL (obvladovanje napak, dogodkov, kompenzacije, sočasnost dostopov, sekvence, paralelnost, sinhronizacija) z ustreznimi formalnimi metodami (π-calculus, abstraktna procesna algebra, Petrijeve mreže), je podan v [105]. V našem primeru bomo uporabili predvsem metode, ki omogočajo analizo in ustrezno verifikacijo integracije mehanizma zaupanja oz. modela ATM v kompozit storitev. V naslednjih razdelkih bomo uporabili koncept Petrijevih mrež za izračun stopnje zaupanja kompozita spletnih storitev (t. i. procesa BPEL). Petrijeve mreže so formalna metoda, ki je intuitivno blizu sistemu *delovnih tokov* (angl. workflow), kar procesi BPEL dejansko so. Poleg tega obstaja tudi množica orodij za formalno analizo sistemov in procesov na podlagi Petrijevih mrež.

4.2.4 Petrijeve mreže

Jedro jezika BPEL sestavljajo enostavni komunikacijski primitivi, ki se lahko kombinirajo z ustreznimi konstrukti za definicijo zaporedij, paralelnosti, sinhronizacije, sočasnosti itd. Analiza

[(8)] Nabor formalnih metod za analizo BPEL je širok, mi se omejimo samo na nekatere.

teh dinamičnih področij je težko izvedljiva brez formalizacije procesnega toka BPEL. Konstrukti znotraj BPEL, predvsem tisti, ki se tičejo kontrole procesnega toka, so podobni konstruktom znotraj jezikov za definicijo delovnih tokov. Na področju modeliranja delovnih tokov, so določeni avtorji pokazali (npr. Murata [127]), da Petrijeve mreže (angl. Petri Net – PN) nudijo osnovo za statično analizo in verifikacijo. Analiza se lahko uporabi na procesih BPEL na ta način, da se prevedejo v PN in uporabijo že obstoječe tehnike in orodja za analizo PN.

Kot je bilo omenjeno v prejšnjem razdelku 4.2.3, lahko z uporabo PN analiziramo veliko področij delovnih procesov: odkrivanje mrtvih zank in nedosegljivih aktivnosti, dosegljivost stanj, pokritost poti, redukcija procesnih mrež. Obstajajo različni pristopi preslikave konstruktov jezika BPEL v PN (PN z barvnimi žetoni [117], PN algebra [123], običajne Petrijeve mreže s prostori/stanji (angl. place/transition) [119]), ki predstavljajo vmesno stopnjo za uporabo različnih tehnik in orodij za analizo in verifikacijo (npr. *WofBPEL* [119], *pose++* [117]).

Petrijeve mreže, imenovane tudi *mreže prostor/stanje* (angl. P/T nets), so grafična in matematična predstavitev diskretnih porazdeljenih sistemov (vključujejo sočasnost, asinhronost, nedeterminizem). V osnovi so PN usmerjeni in povezani grafi, kjer je vsako vozlišče bodisi *prostor* (angl. place) ali *prehod* (angl. transition). Usmerjena povezava povezuje prostor s prehodom. *Žetoni* (angl. tokens) se rabijo za simulacijo dinamičnih in sočasnih aktivnosti znotraj sistema. Žetoni se nahajajo v prostorih; prehod je omogočen natanko tedaj, ko je v vsakem povezanem prostoru na omenjeni prehod vsaj en žeton. Vsak omogočen prehod se lahko *proži* in ob tem odstrani en žeton iz vsakega vhodnega prostora ter pošlje po en žeton v vsak izhodni prostor. V nadaljevanju sledijo osnovne definicije [127, 128, 129].

Definicija 4.2.4.1 (Petrijeva mreža PN)**.** Končna Petrijeva mreža je definirana kot $PN = (P, T, F, W, M_0, \ell)$ kjer:

- $P = \{p_1, p_2, \ldots, p_m\}$ je končna množica prostorov;
- $T = \{t_1, t_2, \ldots, t_n\}$ je končna množica prehodov[(9)], ki predstavljajo posamezne aktivnosti znotraj procesa BPEL;
- $F \subseteq (P \times T) \cup (T \times P)$ je množica usmerjenih povezav, ki predstavljajo kontrolni tok znotraj procesa;
- $W : F \mapsto \mathbb{N}$ je utežna funkcija[(10)] in določa število potrebnih žetonov med p in t oz. t in p;
- $M_0 : P \mapsto \mathbb{N}$ je začetna oznaka PN;
- $\ell : T \mapsto Act$ je označevalna funkcija, kjer je Act končna množica imen aktivnosti oz. akcij (vsebuje tudi element τ, ki je oznaka za *nevidno* oz. *tiho* (angl. silent) akcijo).

(9) Zaradi uveljavljene notacije Petrijevih mrež, oznaka množice prehodov T izjemoma sovpada z oznako časovne dimenzije T abstraktnega modela ATM, vendar samo v okviru tega poglavja oz. razdelka 4.2.

(10) V večini naših primerov je $W(p, t) = W(t, p) = 1$ (1 žeton).

Oznaka mreže $M : P \mapsto \mathbb{N}$ določa $M(p)$ žetonov v prostoru $p \in P$; ta oznaka se lahko izrazi tudi z vektorjem $[M(p_1), \ldots, M(p_m)]$. Definirajmo $\Delta(t) = [W(t, p_1) - W(p_1, t), \ldots, W(t, p_m) - W(p_m, t)]$ za vsak prehod $t \in T$ in $\Delta(u) = \sum_{i=1}^{n} \Delta(t_i)$ za zaporedje $u = t_1 t_2 \ldots t_n$. Sedaj lahko formalno definiramo dinamično obnašanje PN preko proženja prehodov [127, 129].

Definicija 4.2.4.2 (Omogočenost prehoda $t \in T$)**.** Prehod t je omogočen na oznaki M natanko tedaj, ko $\forall p \in P : M(p) \geq W(p, t)$. Ob proženju prehoda t pride do spremembe oznake mreže: $M' = M + \Delta(t)$.

Množico dosegljivosti (angl. reachability set) oznake mreže M označimo kot $\mathcal{S}(M) = \{M' \mid M' = M + \Delta(t)\}$. Množica dosegljivosti Petrijeve mreže PN je enaka $\mathcal{S}(PN) = \mathcal{S}(M_0)$. Velja omeniti nekaj lastnosti PN, ki so odvisne od začetne oznake M_0 [127]:

- *Dosegljivost* (angl. reachability): oznaka M_n je dosegljiva iz začetne oznake M_0, če je $M_n \in \mathcal{S}(M_0)$ – obstaja zaporedje prehodov iz M_0 v M_n. Problem preverjanja $M_n \in \mathcal{S}(M_0)$ je rešljiv, toda je eksponentne zahtevnosti [127].

- *Omejenost* (angl. boundedness): mreža je k-omejena, če je število žetonov v vseh prostorih manjše od nekega končnega števila k za vsako oznako M dosegljivo iz M_0; t. i. $\forall p \in P, \forall M \in \mathcal{S}(M_0) : M(p) \leq k$. Če je $k = 1$ pravimo, da je PN *varna* (angl. safe).

- *Pokritost* (angl. coverability): oznaka M je pokrita, če obstaja oznaka $M' \in \mathcal{S}(M_0)$, tako da velja $M'(p) \geq M(p)$ za vsak prostor $p \in P$.

Spremembo oznake mreže M lahko opišemo tudi v smislu avtomatov oz. *sistema označenih prehodov* (angl. Labelled Transition System – LTS). LTS tako vsebuje množico *stanj* (v našem primeru množica oznak prostorov M), množico *akcij* in *relacijo prehoda* $\rightarrow$, ki označuje spremembo oznak prostorov.

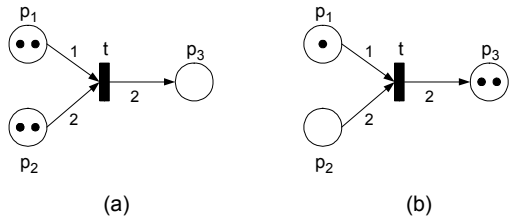
Definicija 4.2.4.3 (Sistem označenih prehodov LTS za PN)**.** Sistem označenih prehodov (graf prehodov) LTS je trojica $(\mathcal{S}(PN), Act, \{\xrightarrow{\alpha} \mid \alpha \in Act\})$, kjer je [130]:

- $\mathcal{S}(PN)$ množica možnih stanj oz. oznak mreže PN z M_0 kot začetnim stanjem;
- Act množica imen akcij oz. aktivnosti, vključno z elementom τ (tiha, nevidna akcija);
- $\xrightarrow{\alpha} \subseteq \mathcal{S}(PN) \times \mathcal{S}(PN)$ relacija prehodov za vsako ime akcije $\alpha \in Act$ (uporabljamo notacijo $s \xrightarrow{\alpha} s'$ namesto $(s, s') \in \xrightarrow{\alpha}$).

LTS je končen, če sta množici stanj $\mathcal{S}(PN)$ in imen akcij Act končni.

Spremembo oznake mreže $M' = M + \Delta(t)$ lahko sedaj zapišemo kot $M \xrightarrow{t} M'$. Za poljubno akcijo $\alpha \in Act, M \xrightarrow{\alpha} M'$ označuje zapis oblike $M \xrightarrow{t} M'$ določen prehod $t \in T$, tako da velja $\ell(t) = \alpha$. To pomeni, da obstaja homomorfizem med končno sekvenco prehodov in končno sekvenco imen akcij. Zapis oblike $M \longrightarrow M'$ pomeni, da obstaja zaporedje u prehodov, tako da velja $M \xrightarrow{u} M'$.

Slika 4.3 prikazuje primer mreže s tremi prostori $P = \{p_1, p_2, p_3\}$, enim prehodom $T = \{t\}$, $W(p_2, t) = W(t, p_3) = 2$, $W(p_1, t) = 1$, $Act = \{t, \tau\}$, $\ell(t) = t$, $\mathcal{S}(PN) = \{M_0, M'\}$. Oznaka mreže pred proženjem (a) je enaka $M_0 = [2, 2, 0]$, po proženju (b) pa $M' = [1, 0, 2]$. Zapisano kot LTS: $M_0 \xrightarrow{t} M'$.



Slika 4.3: Primer proženja prehoda t: (a) pred in (b) po proženju.

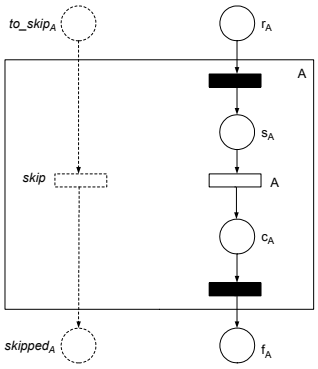
4.2.5 Preslikava BPEL v PN

Znotraj tega razdelka bomo predstavili osnovne koncepte preslikave konstruktov BPEL (osnovne in strukturirane aktivnosti – predstavljajo jedro BPEL) v PN; deloma se bomo naslonili na princip preslikav v [125, 119]. Označeni prehodi preko funkcije ℓ se uporabljajo za modeliranje osnovnih aktivnosti (npr. `receive`, `invoke`) in dogodkov. Oznake prehodov so nekakšni meta-podatki, povezani z aktivnostmi in dogodki (npr. imena tipov sporočil, operacije). Neoznačeni prehodi so τ-prehodi in predstavljajo interne akcije, ki jih ni mogoče opazovati s strani zunanjega opazovalca.

Najprej se osredotočimo na preslikavo osnovne aktivnosti, ki jo označimo z oznako A. Preslikana PN je sestavljena iz dveh delov: (a) normalno procesiranje aktivnosti A (prikaz z neprekinjene črto) in (b) preskok aktivnosti A (prikaz s črtkano črto).

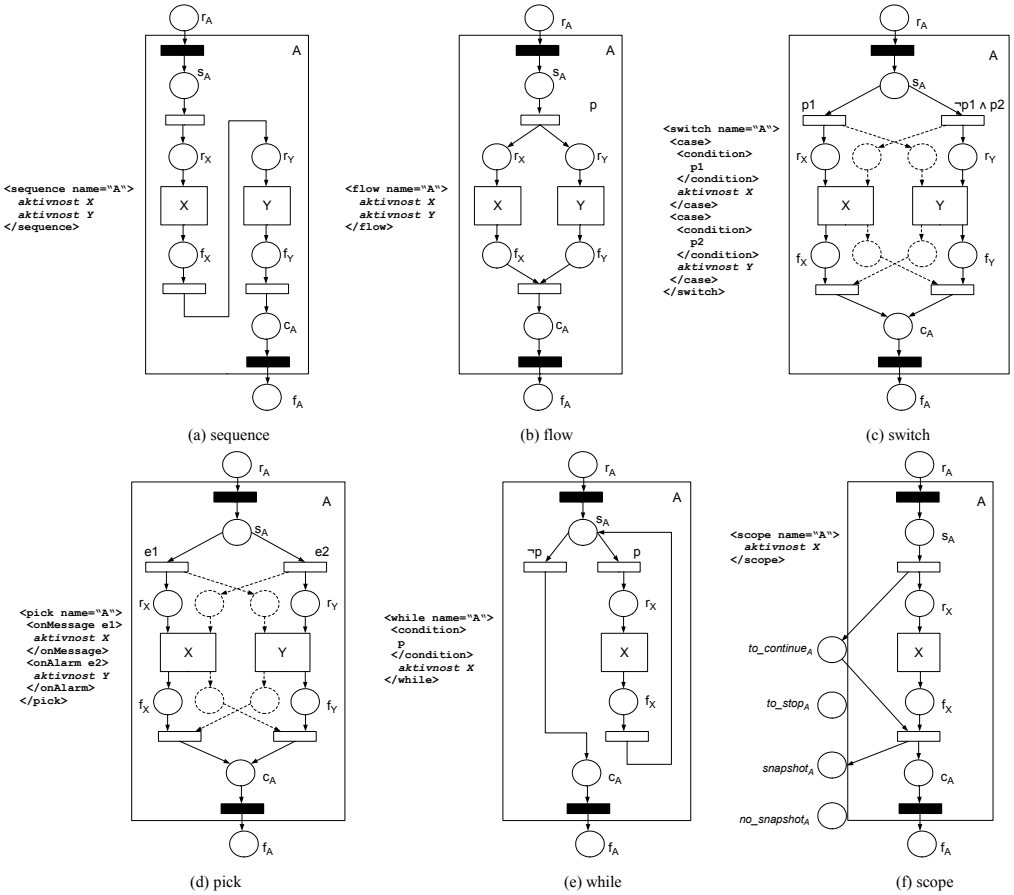
V primeru normalnega procesiranja prostor r_A (t. i. *ready*) določa začetno stanje kot pogoj za začetek A pred preverjanjem vseh kontrolnih povezav, ki imajo ponor znotraj aktivnosti A. Prostor f_A (t. i. *finished*) določa končno stanje, ko se konča aktivnost A in ugotovijo statusi kontrolnih povezav, ki imajo izvor v A. Abstraktni prehod z oznako A predstavlja akcijo za izvedbo. To je način modeliranja aktivnosti, kjer se jedro vsake aktivnosti smatra kot atomarna (nedeljiva) akcija. Prehod A ima vhodni prostor s_A (t. i. *started*) za stanje, ko se aktivnost A začne in izhodni prostor c_A (t. i. *completed*) za stanje, ko se A konča. Uporabljena sta tudi dva tiha τ-prehoda za modeliranje internih akcij (preverjanje pogojev za aktivnosti). Pot za preskok (angl. skip) aktivnosti se rabi za preslikavo kontrolnih povezav. Uporabljena sta prostora *to_skip* (začetek) in *skipped* (konec). Na sliki 4.4 je prikazan primer preslikane osnovne aktivnosti A.

Preslikava strukturiranih aktivnosti temelji na enakem pristopu kot za osnovne aktivnosti. Slika 4.5 prikazuje preslikavo strukturiranih aktivnosti v PN s poudarkom na normalnem izvajanju (t. i. brez napak). Pot za preskok aktivnosti ni prikazana, če ni uporabljena. Zaradi poenostavitve sta prikazani do največ dve pod-aktivnosti izbrane strukturirane aktivnosti, kar



Slika 4.4: Osnovna aktivnost kot PN.

pa se lahko posploši na poljubno število pod-aktivnosti. Preslikave nekaterih strukturiranih



Slika 4.5: Strukturirane aktivnosti kot PN.

aktivnosti so intuitivne (*sequence*, *flow*), medtem ko ostale zahtevajo kratko obrazložitev.

Aktivnost *switch* (slika 4.5(c)) vsebuje množico odločitvenih vej, prva se izbere tista, kjer je izpolnjen logični pogoj. Ostale veje se potem avtomatsko preskočijo. Preverjanje logičnih pogojev se izvede na τ-prehodu (pogoj p_1 – izbere se prva veja ali $\neg p_1 \wedge p_2$ – izbere se druga veja).

Aktivnost *pick* izbere ustrezno vejo na podlagi pojavitve dogodkov. Ti so dveh vrst – (a) sporočilni dogodki (`onMessage`), ki se zgodijo ob prispetju določenega sporočila in (b) alarmni dogodki (`onAlarm`), ki se zgodijo ob sistemskih zamudah. Na sliki 4.5(d) je aktivnost *pick* modelirana podobno kot *switch* z razliko dveh prehodov e_1 in e_2, ki predstavljata ustrezne dogodke.

Aktivnost *while* predstavlja mehanizem za zanke. Na sliki 4.5(e) ima aktivnost A pod-aktivnost X, ki se izvede tolikokrat, dokler je pogoj p veljaven. Zanka se konča, ko je pogoj neveljaven $\neg p$.

Aktivnost tipa *scope* določa ločene hierarhične enote, v katere je lahko razdeljen proces BPEL. Omogoča obvladovanje napak in dogodkov. Ima primarno aktivnost, ki predstavlja normalno izvajanje. Slika 4.5(f) vsebuje posebne prostore, ki predstavljajo izjemne situacije: (a) `to_continue` označuje izvajanje območja brez napak, (b) `to_stop` označuje pojavitev napake ter končanje vseh izvajajočih in gnezdenih aktivnosti, (c) `snapshot` označuje posnetek uspešno končanega območja in (d) `no_snapshot` označuje odsotnost posnetega stanja.

Preslikava nekaterih osnovnih in strukturiranih aktivnosti v PN je seveda premalo za preslikavo vseh konstruktov BPEL. Prikazan je samo koncept in namen tega dela ni analiza in preslikava BPEL v PN. Podrobni detajli preslikave vseh ostalih konstruktov BPEL so podani v [119].

4.2.6 Analiza PN

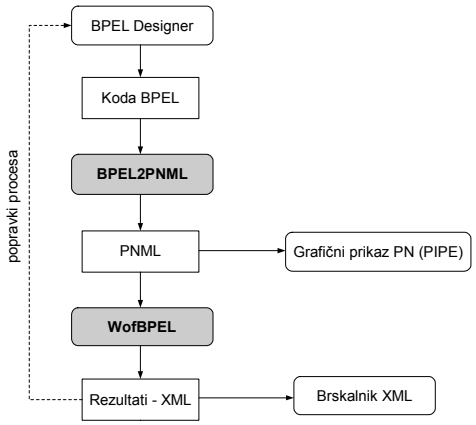
Rezultat preslikav iz jezika BPEL v PN (delno predstavljeno v prejšnjem razdelku 4.2.5), se lahko uporabi za formalno analizo in verifikacijo procesov BPEL z uporabo obstoječih PN metod in tehnik. V ta namen že obstajajo izdelana računalniška orodja, kot npr. *BPEL2PNML* [119], *WofBPEL* [131] in *PIPE2*(11) [132], ki jih lahko uporabimo skupaj.

Proces BPEL lahko napišemo ročno ali pa ga generiramo iz ustreznih orodij (npr. Oracle BPEL Designer, Microsoft BizTalk, ActiveBPEL Designer). Orodje BPEL2PNML vzame kot vhod kodo BPEL ter jo preslika (transformira) v sintakso označevalnega jezika *PNML* (angl. Petri Net Markup Language – PNML). Preslikava temelji na konceptu preslikav, prikazanih v prejšnjem razdelku. Izhod PNML se potem lahko uporabi kot vhod v orodje WofBPEL, ki na podlagi izbranih opcij izvede različne metode analize in rezultate zapiše v izhodno datoteko XML. Poleg tega se lahko datoteka v obliki PNML uporabi za grafično predstavitev preko orodja PIPE2. Slika 4.6 prikazuje vloge posameznih orodij pri analizi procesa BPEL.

Orodje WofBPEL lahko izvede sledeče tri vrste analize (več v [119]):

- Odkrivanje nedosegljivih aktivnosti znotraj procesa BPEL. Analiza poteka z uporabo dveh metod – *ohlapna pravilnost* (angl. relaxed soundness) [133] in *prehodi invariant* (angl. transition invariants) [134]. Prva metoda je računsko precej bolj zahtevna od druge, zato je za kompleksne procese bolj uporabna druga metoda [119].

(11) Platform Independent Petri net Editor.

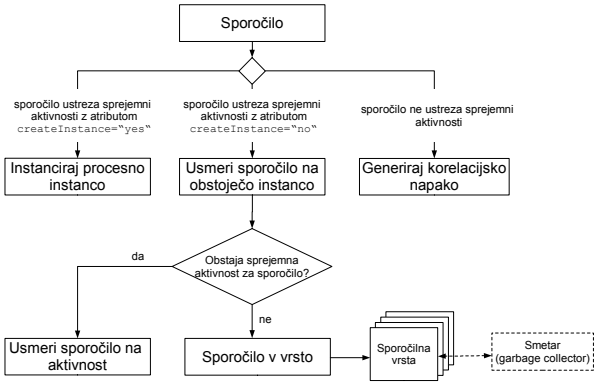


Slika 4.6: Koraki analize procesa BPEL (BPEL2PNML, WofBPEL, PIPE).

- Odkrivanje kršitev pri nekaterih omejitvah specifikacije BPEL – npr. dve sočasni aktivnosti ne smeta preko aktivnosti *receive* uporabiti istega tipa sporočila (določen preko kombinacije vrednosti elementov `partnerLink`, `portType` in `operation`). Analiza teh lastnosti je možna preko preverjanja celotnega prostora stanj, če obstajata vsaj dva sočasna prehoda, ki predstavljata isti tip sporočila.

- Analiza dosegljivosti za ugotavljanje, kateri tipi sporočil se lahko uporabijo v prihodnjih stanjih izvajanja procesa BPEL. Rezultati te analize se lahko uporabijo znotraj strežnika BPEL za učinkovito upravljanje z viri. Sporočila, ki se ne uporabijo v prihodnjih stanjih izvajanja procesa, se lahko zavržejo iz ustreznih sporočilnih vrst in tako sprostijo vire. Slika 4.7 prikazuje korake procesiranja sporočila znotraj strežnika BPEL. Strežnik usmerja vsako prejeto sporočilo do ustrezne instance procesa BPEL z uporabo korelacijskega mehanizma [121, 105]. Če ima sporočilo korelacijsko množico in vsebuje žetone z enakimi vrednostmi, ko to določa sprejemna aktivnost v procesu, potem sporočilo ustreza aktivnosti. Sprejemna aktivnost je lahko začetna aktivnost v procesu, kar je označeno z atributom `createInstance="yes"`. V tem primeru aktivnost instancira nov proces in uporabi sporočilo. V primeru, da sporočilo ustreza sprejemni aktivnosti z atributom `createInstance="no"`, se usmeri na obstoječo procesno instanco. Strežnik preveri, če znotraj instance obstaja kaka sprejemna aktivnost (npr. *receive*, *pick*, mehanizem obvladovanje dogodkov), ki čaka na sporočilo. V tem primeru se sporočilo usmeri na izbrano aktivnost, drugače pa se shrani v sporočilno vrsto za kasnejšo uporabo.

Orodje WofBPEL omogoča tudi tehniko *redukcije* Petrijevih mrež, ki jih generira orodje BPEL2PNML. Predvsem gre za tehniko redukcij, ki jih je predlagal Murata [127]; gre za odstranjevanje nepotrebnih tihih τ-prehodov in odvečnih prostorov. Na ta način se velikost PN lahko občutno zmanjša. V prilogi D je prikazan primer preslikave enostavnega procesa BPEL s strukturno aktivnostjo `while` v PN z uporabo orodja BPEL2PNML.



Slika 4.7: Mehanizem procesiranja sporočil v strežniku BPEL.

4.2.7 Stopnja zaupanja kompozita storitev

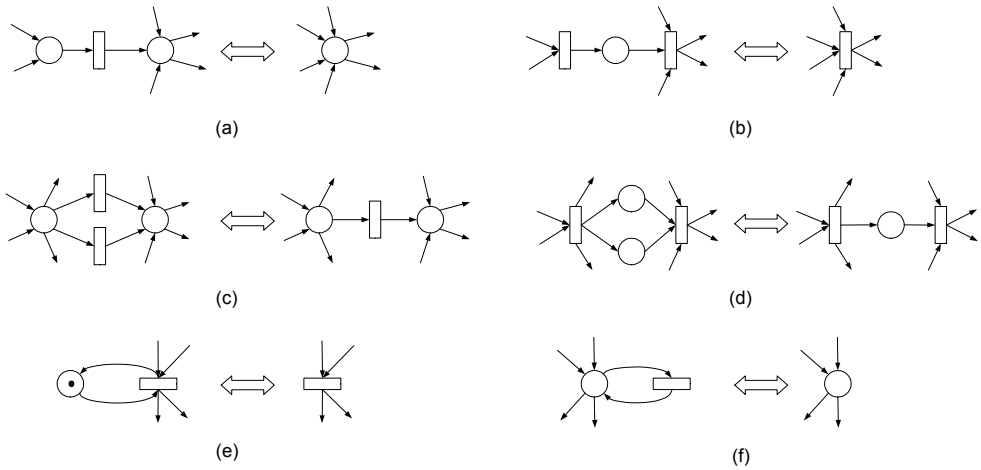
V poglavju 3 (razdelek 3.1.4 in 3.1.5) smo definirali algoritem Γ modela ATM za izračun stopnje zaupanja za izbrani kontekst med poljubno izvorno entiteto a in ciljno entiteto b ($a, b \in \mathcal{A}$) z oznako $trust_{a,b}(\varphi)$. Če je torej ciljna entiteta b storitev v smislu SOA, potem lahko entiteta a pred proženjem b, izračuna stopnjo zaupanja in na podlagi tega (ne)proži storitev b (oz. stopi v interakcijo s ponudnikom, ki ponuja storitev b).

V primeru, da je storitev b atomarna, njeno stopnjo zaupanja neposredno določa algoritem Γ oz. $trust_{a,b}(\varphi)$. Težava nastane v primeru, če b ni atomarna storitev, ampak kompozit večih storitev (glej razdelek 4.2.1), dobljen z orkestracijo ali koreografijo. V našem primeru se omejimo na kompozite – poslovne procese BPEL. Skušamo torej izračunati stopnjo zaupanja poslovnega procesa p_{BPEL}, ki ga predstavlja storitev $b \in \mathcal{A}$. Algoritem za izračun označimo kot $CompositeTrust_{a,b}(\varphi)$.

Proces BPEL lahko predstavimo s Petrijevo mrežo (glej razdelek 4.2.5), kar omogoča analizo in verifikacijo procesa. Predstavitev procesa s PN uporabimo tudi pri izračunu stopnje zaupanja kompozita (procesa). Pri PN, ki je rezultat preslikave procesa, se osredotočimo samo na kontrolni tok (preslikane strukturirane aktivnosti) in na prehode, ki predstavljajo proženje zunanjih spletnih storitev (aktivnost `invoke`). Za poenostavitev in zmanjšanje PN se poslužimo redukcijskih pravil [127], predvsem tistih, ki zmanjšajo število prostorov.

Definicija 4.2.7.1 (Reducirana PN)**.** Reducirana končna Petrijeva mreža $PN' = (P', T', F', W, M_0, \ell')$ nastane iz končne Petrijeve mreže $PN = (P, T, F, W, M_0, \ell)$ z uporabo redukcijskih pravil [127]. Velja $|P'| \leq |P|, |T'| \leq |T|, |F'| \leq |F|$. Redukcijska pravila (slika 4.8) so sledeča: (a) združevanje zaporedja prostorov, (b) združevanje zaporedja prehodov, (c) združevanje vzporednih prehodov, (d) združevanje vzporednih prostorov, (e) odprava zank na prostore in (f) odprava zank na prehode.

V našem primeru upoštevamo pravila, ki zmanjšajo število prostorov, medtem ko ohranjajo število prehodov (pravila (d) in (e)). Postavi se vprašanje, ali sta PN in reducirana PN'



Slika 4.8: Redukcijska pravila PN.

enakovredni (ekvivalentni) v smislu obnašanja? Petrijeva mreža je v osnovi proces s svojimi lastnostmi in obnašanjem. Iščemo torej ekvivalenco med obnašanjem originalnega procesa in obnašanjem njegove reducirane oblike. Najprej je potrebno definirati pojem ekvivalence v matematičnem smislu.

Definicija 4.2.7.2 (Ekvivalenca)**.** Pojem ekvivalence je povezan s pojmom *ekvivalenčne relacije.* Naj bo X množica, potem je ekvivalenčna relacija na množici X dvomestna relacija $\mathcal{R} \subseteq X \times X$, ki ima sledeče lasnosti[(12)]:

1) $\mathcal{R}$ je *refleksivna*: $\forall x \in X : x\mathcal{R}x$;
2) $\mathcal{R}$ je *simetrična*: $\forall x, y \in X : x\mathcal{R}y \Rightarrow y\mathcal{R}x$;
3) $\mathcal{R}$ je *tranzitivna*: $\forall x, y, z \in X : x\mathcal{R}y \wedge y\mathcal{R}z \Rightarrow x\mathcal{R}z$.

Obstaja več različnih metod za dokazovanje ekvivalence med procesi [128, 130] – npr. *sledna ekvivalenca* (angl. trace equivalence) in *bisimulacija* (angl. bisimulation). Za dokazovanje ekvivalence sta obe metodi *odločljivi* (angl. decidable) [135, 129, 128]:

- *Sledna ekvivalenca* temelji na dokazovanju enakosti *sledi* (angl. trace) procesov. Naj bo sled procesa P zaporedje $\alpha_1 \ldots \alpha_k \in Act, k \geq 0$ tako da obstaja zaporedje prehodov $P = P_0 \xrightarrow{\alpha_1} P_1 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_{k-1}} P_{k-1} \xrightarrow{\alpha_k} P_k$, za ustrezne $P_1, \ldots, P_k$. Oznaka $Traces(P)$ pomeni množico vseh sledi oz. končnih zaporedij interakcij procesa P. Če sta torej dva procesa P in Q ekvivalentna glede obnašanja, potem je $Traces(P) = Traces(Q)$ [130].

- *Bisimulacija* je dvomestna relacija med dvema LTS, ki se enako obnašata oz. en sistem simulira obnašanje drugega in obratno. Iz definicij 4.2.4.1 (PN) in 4.2.4.3 (LTS) lahko Petrijevo mrežo predstavimo kot LTS. To omogoča uporabo metode bisimulacije za iskanje ekvivalence obnašanja med PN in reducirano PN'. Sledi definicija bisimulacije [129].

(12) Te lastnosti so bile omenjene že pri *relaciji zaupanja* (poglavje 3, definicija 3.1.2.1).

Definicija 4.2.7.3 (Močna bisimulacija)**.** Naj bosta PN_1 in PN_2 dve Petrijevi mreži. Dvomestna relacija $\mathcal{B}$ med PN_1 in PN_2 je *močna bisimulacija* natanko takrat, ko za vse pare oznak $M_1\mathcal{B}M_2$ in za vsa imena akcij $\alpha \in Act$ velja:

- če $M_1 \xrightarrow{\alpha} M_1'$ potem obstaja $M_2 \xrightarrow{\alpha} M_2'$ za nek M_2', tako da velja $M_1'\mathcal{B}M_2'$;
- če $M_2 \xrightarrow{\alpha} M_2'$ potem obstaja $M_1 \xrightarrow{\alpha} M_1'$ za nek M_1', tako da velja $M_1'\mathcal{B}M_2'$.

Oznaka M_1 je *močno bisimilarna* oznaki M_2, zapisano kot $M_1 \sim M_2$, če obstaja $M_1\mathcal{B}M_2$ za neko močno bisimulacijo $\mathcal{B}$.

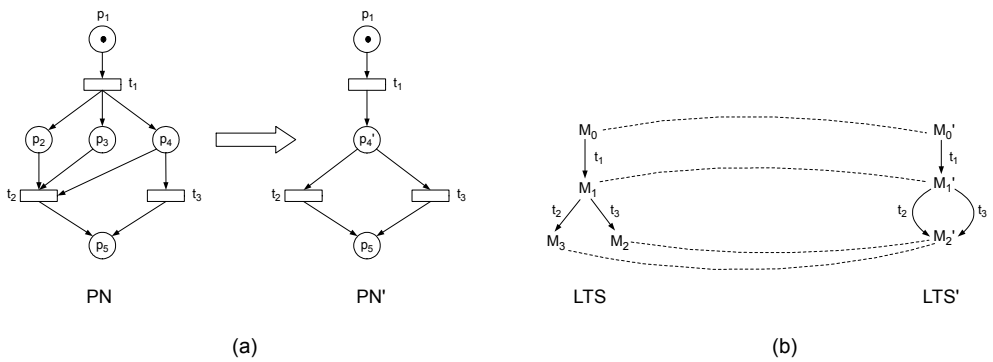
Definicija 4.2.7.4 (Šibka bisimulacija)**.** Naj bo podan simbol $\overset{\alpha}{\Longrightarrow}$ s pomenom $\overset{\alpha}{\Longrightarrow} = (\xrightarrow{\tau})^* \xrightarrow{\alpha} (\xrightarrow{\tau})^*$ za vsako ime akcije $\alpha \in Act\backslash\{\tau\}$, kjer oznaka $(\xrightarrow{\tau})^*$ pomeni refleksivno in tranzitivno ovojnico relacije $\xrightarrow{\tau}$. Relacija $\mathcal{B}$ med oznakami PN_1 in PN_2 je *šibka bisimulacija* natanko takrat, ko za vse pare oznak $M_1\mathcal{B}M_2$ in za vsa imena akcij $\alpha \in Act$ velja:

- če $M_1 \xrightarrow{\alpha} M_1'$ potem obstaja $M_2 \overset{\alpha}{\Longrightarrow} M_2'$ za nek M_2', tako da velja $M_1'\mathcal{B}M_2'$;
- če $M_2 \xrightarrow{\alpha} M_2'$ potem obstaja $M_1 \overset{\alpha}{\Longrightarrow} M_1'$ za nek M_1', tako da velja $M_1'\mathcal{B}M_2'$.

Oznaka M_1 je *šibko bisimilarna* oznaki M_2, zapisano kot $M_1 \approx M_2$, če obstaja $M_1\mathcal{B}M_2$ za neko šibko bisimulacijo $\mathcal{B}$.

Relaciji bisimulacije $\sim$ in $\approx$ sta ekvivalenčni relaciji [130]. Dve Petrijevi mreži PN_1 in PN_2 sta močno (šibko) bisimilarni, če sta njuni začetni oznaki $M_0^{PN_1}$ in $M_0^{PN_2}$ močno (šibko) bisimilarni [130].

Za primer lahko pokažemo ekvivalenco med dvema Petrijevima mrežama: $PN = \{P, T, F, W, M_0, \ell\}$ in reducirano obliko $PN' = \{P', T, F', W, M_0', \ell\}$. Slika 4.9 (a) prikazuje redukcijo mreže z uporabo pravila združevanja vzporednih prostorov (slika 4.8 (d)). PN ima množico prostorov $P = \{p_1, p_2, p_3, p_4, p_5\}$, prehode $T = \{t_1, t_2, t_3\}$, začetno oznako $M_0 = [1, 0, 0, 0, 0]$. Reducirana mreža PN' ima manjše število prostorov $P' = \{p_1, p_4', p_5\}$ (p_4' nastane z združevanjem p_2, p_3 in p_4) z začetno oznako mreže $M_0' = [1, 0, 0]$. Slika 4.9 (b)



Slika 4.9: Redukcija iz PN v PN' (a) in bisimulacija (b).

prikazuje sistem označenih prehodov za PN in PN', kjer so ustrezne množice oznak mreže enake $\mathcal{S}(PN) = \{M_0, M_1, M_2, M_3\}$ ($M_1 = [0, 1, 1, 1, 0]$, $M_2 = [0, 1, 1, 0, 1]$, $M_3 = [0, 0, 0, 0, 1]$)

in $\mathcal{S}(PN') = \{M'_0, M'_1, M'_2\}$ ($M'_1 = [0,1,0]$, $M'_2 = [0,0,1]$). Množica imen akcij je enaka kar oznakam prehodov $Act = \{t_1, t_2, t_3, \tau\}$, relacije prehodov pa lahko vidimo direktno iz slike: $\xrightarrow{t_1} = \{(M_0, M_1), (M'_0, M'_1)\}$, $\xrightarrow{t_2} = \{(M_1, M_3), (M'_1, M'_2)\}$, $\xrightarrow{t_3} = \{(M_1, M_2), (M'_1, M'_2)\}$. Dokazali bomo, da sta $M_0 \sim M'_0$, kar pomeni, da sta PN in PN' močno bisimilarni (posledično tudi šibko bisimularni – $M_0 \approx M'_0$). V ta namen skušamo definirati relacijo močne bisimulacije $\mathcal{B}$, da lahko pokažemo $M_0 \mathcal{B} M'_0$ oz. $(M_0, M'_0) \in \mathcal{B}$. Naj bo

$$\mathcal{B} = \{(M_0, M'_0), (M_1, M'_1), (M_2, M'_2), (M_3, M'_2)\}.$$

Za lažjo predstavo je relacija $\mathcal{B}$ prikazana s črtkanimi črtami na sliki 4.9 (b). Pokazati moramo, da je $\mathcal{B}$ močna bisimulacija oz. da ustreza zahtevam definicije 4.2.7.3. Za vsak element oz. par iz množice $\mathcal{B}$ moramo raziskati vse možne prehode iz prvega stanja (oznake mreže) in najti ujemanje z ustreznimi prehodi iz drugega stanja.

a) Par (M_0, M'_0).
Prehodi iz M_0: $M_0 \xrightarrow{t_1} M_1$ se ujema z $M'_0 \xrightarrow{t_1} M'_1$ in velja $(M_1, M'_1) \in \mathcal{B}$.
Prehodi iz M'_0: $M'_0 \xrightarrow{t_1} M'_1$ se ujema z $M_0 \xrightarrow{t_1} M_1$ in velja $(M_1, M'_1) \in \mathcal{B}$.

b) Par (M_1, M'_1).
Prehodi iz M_1:
$M_1 \xrightarrow{t_2} M_3$ se ujema z $M'_1 \xrightarrow{t_2} M'_2$ in velja $(M_3, M'_2) \in \mathcal{B}$;
$M_1 \xrightarrow{t_3} M_2$ se ujema z $M'_1 \xrightarrow{t_3} M'_2$ in velja $(M_2, M'_2) \in \mathcal{B}$.
Prehodi iz M'_1:
$M'_1 \xrightarrow{t_2} M'_2$ se ujema z $M_1 \xrightarrow{t_2} M_3$ in velja $(M_3, M'_2) \in \mathcal{B}$;
$M'_1 \xrightarrow{t_3} M'_2$ se ujema z $M_1 \xrightarrow{t_3} M_2$ in velja $(M_2, M'_2) \in \mathcal{B}$.

c) Par (M_2, M'_2) in (M_3, M'_2) – oznake M_2, M_3 kot tudi M'_2 ne morejo narediti nobenega prehoda, zato sta lahko oba para v množici $\mathcal{B}$.

Dokazali smo, da je $\mathcal{B}$ močna bisimulacija in ker je $(M_0, M'_0) \in \mathcal{B}$ velja $M_0 \sim M'_0$ (PN in PN' sta močno bisimularni).

4.2.7.1 Izračun stopnje zaupanja kompozita

Videli smo, da lahko poslovni proces BPEL preslikamo v Petrijevo mrežo PN po pravilih iz razdelka 4.2.5. Preko redukcijskih pravil (združevanje prostorov) jo lahko poenostavimo v mrežo PN', ki je po obnašanju ekvivalentna PN – uporaba metode bisimulacije. Reducirana mreža PN' služi kot osnova za definicijo algoritma za izračun stopnje zaupanja poslovnega procesa BPEL.

Pri izračunu upoštevamo tiste prehode $t_i \in T, i > 0$ oz. imena akcij $\ell(t_i) = invoke_i \in Act$ mreže PN', kjer gre za proženje spletnih storitev (aktivnost `invoke`). Prožena storitev b je enolično določena preko treh vrednosti atributov znotraj `invoke`: `partnerLink`, `portType` in

`operation`. Za storitev b lahko vedno izračunamo stopnjo zaupanja v ustreznem kontekstu φ preko $trust_{a,b}(\varphi)$ (glej razdelek 3.1.4 in 3.1.5).

Ostali prehodi predstavljajo kontrolno strukturo (t. i. topologijo) PN', ki se uporabi za različna pravila pri izračunu stopnje zaupanja kompozita. Tu gre za osnovne strukturirane aktivnosti BPEL, ki omogočajo kontrolni tok: `sequence`, `flow`, `switch`, `while` in `pick` (glej razdelek 4.2.5, slika 4.5). Kontrolni tok za zaporedje (`sequence`) je analogija za tranzitivne verige, tok za paralelno izvajanje (`flow`) pa za združevanje posameznih tranzitivnih verig (glej razdelek 3.1.3). Podoben princip kot pri računanju izpeljane ocene funkcionalnega zaupanja (glej razdelek 3.1.3), lahko uporabimo tudi pri problemu izračuna stopnje zaupanja kompozita.

Za izhodišče rabimo običajno definicijo Petrijeve mreže (glej razdelek 4.2.4, definicija 4.2.4.1). Definirana je tudi množica imen akcij Act z ustrezno preslikavo $\ell(t) : T \mapsto Act$. Glede na začetno oznako M_0 in kriterija za omogočenost prehodov (definicija 4.2.4.2), dobimo množico dosegljivosti PN': $\mathcal{S}(PN') = \mathcal{S}(M_0)$. Končno oznako mreže označimo kot $M_{end} \in \mathcal{S}(M_0)$. Oznaka $transitionTrust_{a,t_i}(\varphi)$ določa stopnjo zaupanja v akcijo (storitev), ki jo predstavlja prehod $t_i \in T$ (4.1):

$$transitionTrust_{a,t_i}(\varphi) = \begin{cases} \sup(\mathcal{T}), & \ell(t_i) \neq invoke_j \\ trust_{a,t_i}(\varphi), & \ell(t_i) = invoke_j \end{cases} \tag{4.1}$$

PN' transformiramo v *drevo pokritosti* (angl. coverability tree) preko algoritma, ki ga je predstavil Murata [127]. Drevo predstavlja vse možne oznake, ki so dosegljive iz začetne oznake[13] $M \in \mathcal{S}(M_0)$. Vozlišča drevesa predstavljajo oznake M, ki so dosegljive iz M_0 (predstavlja korensko vozlišče), povezave med vozlišči pa so prehodi $t_i \in T$. Algoritem je pravzaprav analogija predstavitvi PN v sistemu označenih prehodov LTS. Konstrukcijo drevesa pokritosti prikazuje algoritem 4.1 [127].

V primeru, da PN' ni omejena[14], je drevo pokritosti neskončno. Da ohranimo drevo končno, uvedemo poseben simbol ϖ, ki predstavlja "neskončnost" (velja $\forall n \in \mathbb{N} : \varpi > n, \varpi \pm n = \varpi$) – določa število žetonov v prostoru $p \in P$: $M(p) = \varpi$ (več v [127]). Nekatere lastnosti PN so vidne neposredno iz generiranega drevesa pokritosti $\mathscr{T}$ [127]:

(a) mreža PN je omejena in posledično končna, če simbol ϖ ni prisoten na nobenem vozlišču $\mathscr{T}$ kot del oznake M;
(b) PN je varna, če vozliča $\mathscr{T}$ vsebujejo samo vrednosti 0 ali 1;
(c) prehod t je mrtev, če se ne pojavi kot nobena povezava v drevesu $\mathscr{T}$;
(d) če je vozlišče z oznako M dosegljivo iz M_0, t. i. $M \in \mathcal{S}(M_0)$, potem obstaja vozlišče z oznako M', ki pokriva M (t. i. $M' \geq M$).

Definirajmo primer PN, ki določa nek poslovni proces BPEL. PN je določena s sledečimi množicami: $P = \{p_1, \ldots, p_8\}$, $T = \{t_1, \ldots, t_8\}$, $M_0 = [1, 0, 0, 0, 0, 0, 0, 0]$, $Act =$

[13] Oznaka M je vektor, kjer element i predstavlja število žetonov v prostoru $p_i \in P$.
[14] Če je PN omejena, se drevo pokritosti imenuje *drevo dosegljivosti*.

```
procedure CoverabilityTree(Marking M0) {
  Node node = new Node();
  node.Marking = M0;
  node.Tag = "new";
  Node rootNode = node;

  while (node.Tag == "new") {
    foreach (Marking M = S(M0)) {;
      Node child = new Node(M);
      if (rootNode.Path(child).Contains(M)) {
        child.Tag = "old";
      } else if (not child.HasTransitions()) {
        child.Tag = "leaf"
      } else {
        foreach (Transition t in child.Transitions) {
          Marking M' = t.FireFrom(M);
          foreach (Node node in rootNode.Path(M)) {
            Marking M" = node.Marking;
            if (M' ≥ M" and M' ≠ M") {
              foreach (Place p in M.Places) {
                if (M'.Places[p] > M".Places[p]) M'.Places[p] = ϖ;
              }
            }
          }
          Node temp = new Node(M');
          temp.Tag = "new";
          temp.Label = t.Label;
          child.AddChild(temp);
        }
      }
    }
  }
  return node;
}
```

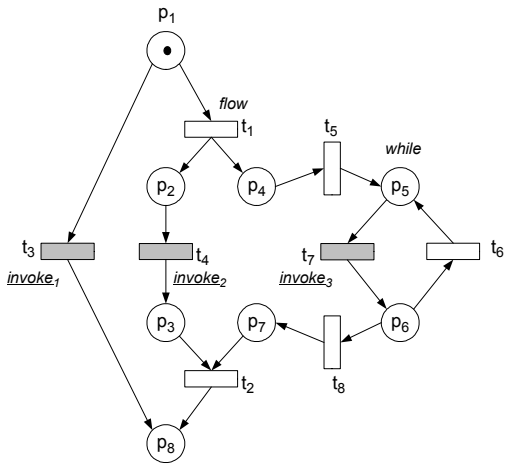
Algoritem 4.1: Konstrukcija drevesa pokritosti $\mathscr{T}$.

$\{t_1, t_2, invoke_1, invoke_2, t_5, t_6, invoke_3, t_8\}$ (torej prehodi t_3, t_4, t_7 predstavljajo proženje spletne storitve). Ogrodje procesa BPEL, ki se preslika v omenjeno PN, prikazuje primer 4.1 (dokument XML mora biti skladen s shemo XSD za specifikacijo BPEL [136]). Slika 4.10 prikazuje PN, ki predstavlja proces BPEL. Z algoritmom 4.1 preslikamo PN v drevo pokritosti s končno oznako $M_{end} = M_8 = [0, 0, 0, 0, 0, 0, 0, 1]$. Drevo pokritosti prikazuje slika 4.11. Opazimo lahko več lastnosti pripadajoče PN: (i) PN je omejena in končna, (ii) PN je varna in (iii) PN ne vsebuje mrtvih prehodov.

Vsaka povezava v drevesu pokritosti ima določeno stopnjo zaupanja, ki jo podaja izraz (4.1) $transitionTrust_{a,t_i}(\varphi), t_i \in T$. Prehod t_i, ki predstavlja osnovno aktivnost `invoke`, določa spletno storitev. Ta je enolično določena preko trojice `partnerLink`, `portName` in `operation`. Zaradi poenostavitve označimo spletno storitev kar s prehodom t_i. Pri določanju stopnje zaupanja kompozita je potrebno upoštevati vse povezave v zaporedju med začetno oznako M_0

```
<process name="SampleProcess" targetNamespace=...>
  <partnerLinks>...</partnerLinks>
  <variables>...<variables>
  <sequence>
    <receive name="..." partnerLink="..." portType="..." operation="..." />
    <!-- invoke 1 -->
    <invoke name="t3" partnerLink="..." portName="..." operation="..." />
    <flow>
      <!-- invoke 2 -->
      <invoke name="t4" partnerLink="..." portName="..." operation="..." />
      <while name="C">
        <condition>....</condition>
        <!-- invoke 3 -->
        <invoke name="t7" partnerLink="..." portName="..." operation="..." />
      </while
    </flow>
    <reply partnerLink="..." portType="..." operation="..." />
  </sequence>
</process>
```

Primer 4.1: Ogrodje procesa BPEL za preslikavo v PN.



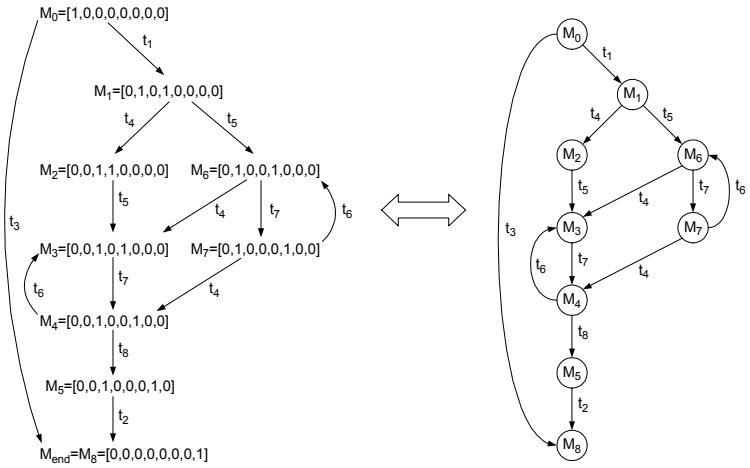
Slika 4.10: Proces BPEL kot PN.

in končno oznako M_{end} znotraj drevesa $\mathscr{T}$. Stopnja zaupanja kompozita je minimalna vrednost stopnje zaupanja med vsemi zaporedji povezav.

Drevo pokritosti je v splošnem usmerjen graf, ki lahko vsebuje cikle. Pri izračunu stopnje zaupanja kompozita je potrebno upoštevati dva primera: (a) drevo pokritosti je usmerjen *acikličen* graf (angl. directed acyclic graph – DAG) in (b) drevo pokritosti je usmerjen *cikličen* graf (angl. directed cyclic graph).

4.2.7.2 (a) Usmerjeni acikličen graf

V tem primeru je drevo pokritosti brez ciklov (zank). Značilnost drevesa oz. usmerjenega acikličnega grafa DAG je, da ga je možno *topološko urediti*.



Slika 4.11: Drevo pokritosti Petrijeve mreže.

Topološka urejenost je linearna urejenost vseh vozlišč usmerjenega grafa DAG [73, 137]. Če obstaja povezava iz vozlišča i do vozlišča j, potem je vozlišče i pred j znotraj topološke urejenosti. Sledi formalna definicija [137].

Definicija 4.2.7.5 (Topološka urejenost)**.** Naj bo podan usmerjeni aciklični graf $DAG = (\mathcal{V}, \mathcal{E})$ z množico vozlišč $\mathcal{V}$ in povezav $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Topološka urejenost vozlišč $\mathcal{V}$ je zaporedje $(v_1, v_2, \ldots, v_{|\mathcal{V}|})$, kjer vsako vozlišče nastopa samo enkrat. Če sta v_i in v_j v zaporedju ter obstaja povezava $v_i \rightarrow v_j$ oz. $(v_i, v_j) \in \mathcal{E}$, potem je $i < j$.

Topološka urejenost za DAG nima enolične rešitve, obstaja namreč lahko več rešitev; vsak DAG ima vsaj eno topološko urejenost [73]. Velja poudariti, da topološko urejanje ni možno v usmerjenih grafih s cikli. V našem primeru bomo za topološko urejanje oznak M drevesa $\mathscr{T}$ uporabili *Kahnov algoritem* [138], ki ima linearno časovno zahtevnost $O(|\mathcal{V}| + |\mathcal{E}|)$, kjer je $\mathcal{V} \equiv \mathcal{S}(M_0)$ in $\mathcal{E} \subseteq \mathcal{S}(M_0) \times \mathcal{S}(M_0)$. Algoritem 4.2 vrne topološko urejenost danega grafa DAG (v našem primeru drevo pokritosti $\mathscr{T}$) kot seznam $L = \{M_0, \ldots, M_{end}\}$; v primeru, da graf vsebuje cikle, je seznam L prazen.

Vsaka povezava znotraj $\mathscr{T}$ predstavlja prehod $t \in T$ s svojo stopnjo zaupanja $transitionTrust_{a,t}(\varphi)$. Ko je drevo topološko urejeno, je potrebno upoštevati tudi stopnjo zaupanja vozlišč – oznak $M \in \mathcal{S}(M_0)$; privzeta vrednost je enaka najmanjši zgornji meji (supremum)[15] množice vrednosti stopenj zaupanja $\sup(\mathcal{T}) = \top$ (glej poglavje 3, definicija 3.1.2.5).

Algoritem za izračun stopnje zaupanja kompozita mora izračunati minimalno vrednost zaupanja med zaporedji proženj prehodov od M_0 do M_{end}. Algoritem je okvirno sestavljen iz dveh delov: (1) topološko urejanje drevesa $\mathscr{T}$ in (2) sprehod po drevesu glede na topološko urejenost vozlišč oz. oznak M. Korak (2) izračuna minimalno vrednost stopnje zaupanja kompozita v okviru izbranega operatorja op $\in \mathcal{O}$ modela ATM. Postopek izračuna prikazuje

[15] Element, ki predstavlja največjo stopnjo zaupanja.

```
List TopologicalSort(DAG graph) {
  List L ← Prazen seznam za shranjevanje urejenih vozlišč
  Queue Q ← Vsa vozlišča brez vhodnih povezav
  while (Q ni prazen) do
    odstrani vozlišče n iz Q
    vstavi n v L
    foreach (vozlišče m s povezavo e iz n v m) do
      odstrani e iz graph
      if (m nima več drugih vhodnih povezav) then
         vstavi m v Q
  if (graph ima povezave) then
    L = ∅
  end if
  return L
}
```

Algoritem 4.2: Topološko sortiranje DAG.

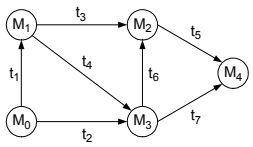
algoritem 4.3. Metoda `ComputeCompositeTrust` sprejme več argumentov: (a) drevo pokritosti

```
procedure ComputeCompositeTrust(DAG 𝒯, Entity a, Operator op, Context φ) {
  M_0 = root node of 𝒯;
  foreach M ∈ 𝒮(M_0) {
    CompositeTrust_{a,M}(φ) = sup(𝒯);
  }
  /* (1) topološko sortiranje */
  List L = TopologicalSort(𝒯);
  /* (2) sprehod po drevesu */
  while (L is not empty) {
    M_x = L.Head;
    L.DeleteHead();
    foreach t ∈ T : M_x --t--> M_y ∧ M_y ∈ L {
      var trust_x = CompositeTrust_{a,M_x}(φ) op transitionTrust_{a,t}(φ);
      CompositeTrust_{a,M_y}(φ) = min(trust_x, CompositeTrust_{a,M_y}(φ));
    }
  }
  return CompositeTrust_{a,M_end}(φ);
}
```

Algoritem 4.3: Izračun stopnje zaupanja kompozita ($\mathscr{T}$ brez ciklov).

$\mathscr{T}$, (b) izvorna entiteta $a \in \mathcal{A}$, (c) operacija zaupanja $op \in \mathcal{O}$ ter (d) kontekst zaupanja $\varphi \in \Phi$. Stopnja zaupanja celotnega kompozita je enaka vrednosti $CompositeTrust_{a,M_{end}}(\varphi)$, ki predstavlja minimalno stopnjo zaupanja med oznakami M_0 in M_{end}. Časovna zahtevnost algoritma (skupaj s topološkim sortiranjem) je polinomska: $O(|\mathcal{V}| + |\mathcal{E}| + |\mathcal{V}||\mathcal{E}|)$.

Za ilustracijo sledi primer izračuna stopnje zaupanja kompozita, ki ga predstavlja drevo pokritosti oz. DAG na sliki 4.12 (drevo na sliki 4.11 ni mogoče uporabiti, ker vsebuje cikle). Vzamemo množico vrednosti stopenj zaupanja $\mathcal{T}$ kvalitativnega modela ATM (glej poglavje 3, razdelek 3.1.4) z uravnoteženim operatorjem op $= \sim$. Naj prehodi t_2, t_4, t_5, t_7 predstavljajo



Slika 4.12: Primer acikličnega usmerjenega grafa.

aktivnost `invoke` z ustreznimi stopnjami zaupanja $trust_{a,t_i}(\varphi) \in \mathcal{T}$: $trust_{a,t_2}(\varphi) = \text{PD}$, $trust_{a,t_4}(\varphi) = \text{T}$, $trust_{a,t_5}(\varphi) = \text{U}$ in $trust_{a,t_7}(\varphi) = \text{PT}$. Potek algoritma 4.3 je sledeč:

a) $CompositeTrust_{a,M_i}(\varphi) = \text{T}, 0 \leq i \leq 4$;

b) topološka urejenost je enaka $L = \{M_0, M_1, M_3, M_2, M_4\}$;

c) $M_x = M_0, L = \{M_1, M_3, M_2, M_4\}$

- $t = t_1, M_y = M_1$:
  $trust_x = CompositeTrust_{a,M_0}(\varphi) \sim transitionTrust_{a,t_1}(\varphi) = \text{T} \sim \text{T} = \text{T}$;
  $CompositeTrust_{a,M_1}(\varphi) = \min(\text{T}, \text{T}) = \text{T}$;
- $t = t_2, M_y = M_3$:
  $trust_x = CompositeTrust_{a,M_0}(\varphi) \sim transitionTrust_{a,t_2}(\varphi) = \text{T} \sim \text{PD} = \text{U}$;
  $CompositeTrust_{a,M_3}(\varphi) = \min(\text{U}, \text{T}) = \text{U}$;

d) $M_x = M_1, L = \{M_3, M_2, M_4\}$

- $t = t_3, M_y = M_2$:
  $trust_x = CompositeTrust_{a,M_1}(\varphi) \sim transitionTrust_{a,t_3}(\varphi) = \text{T} \sim \text{T} = \text{T}$;
  $CompositeTrust_{a,M_2}(\varphi) = \min(\text{T}, \text{T}) = \text{T}$;
- $t = t_4, M_y = M_3$:
  $trust_x = CompositeTrust_{a,M_1}(\varphi) \sim transitionTrust_{a,t_4}(\varphi) = \text{T} \sim \text{T} = \text{T}$;
  $CompositeTrust_{a,M_3}(\varphi) = \min(\text{T}, \text{U}) = \text{U}$;

e) $M_x = M_3, L = \{M_2, M_4\}$

- $t = t_6, M_y = M_2$:
  $trust_x = CompositeTrust_{a,M_3}(\varphi) \sim transitionTrust_{a,t_6}(\varphi) = \text{U} \sim \text{T} = \text{U}$;
  $CompositeTrust_{a,M_2}(\varphi) = \min(\text{U}, \text{T}) = \text{U}$;
- $t = t_7, M_y = M_4$:
  $trust_x = CompositeTrust_{a,M_3}(\varphi) \sim transitionTrust_{a,t_7}(\varphi) = \text{U} \sim \text{PT} = \text{U}$;
  $CompositeTrust_{a,M_4}(\varphi) = \min(\text{U}, \text{T}) = \text{U}$;

f) $M_x = M_2, L = \{M_4\}$

- $t = t_5, M_y = M_4$:
  $trust_x = CompositeTrust_{a,M_2}(\varphi) \sim transitionTrust_{a,t_5}(\varphi) = \text{U} \sim \text{U} = \text{U}$;
  $CompositeTrust_{a,M_4}(\varphi) = \min(\text{U}, \text{U}) = \text{U}$;

g) $M_x = M_4 = M_{end}, L = \emptyset$, ni več izhodnih povezav;

h) izračunana in vrnjena vrednost: $CompositeTrust_{a,M_4}(\varphi) = \text{U}$.

4.2.7.3 (b) Usmerjeni cikličen graf

V primeru, da vsebuje drevo pokritosti cikle (zanke), je izračun stopnje zaupanja kompozita težja naloga. Eden od možnih načinov je odprava ciklov v grafu in uporaba že obstoječega algoritma 4.3.

Obstaja več metod za odpravo ciklov v umerjenih grafih, predvsem kot vmesni korak pri iskanju tranzitivnih ovojnic znotraj grafov [73, 139, 137]. Ena od metod je uporaba *močno povezanih komponent* (angl. Strongly Connected Component – SCC) znotraj usmerjenega grafa [139].

Definicija 4.2.7.6 (Močno povezane komponente SCC)**.** Naj bo podan graf $G = (\mathcal{V}, \mathcal{E})$. Vozlišči $u, v \in \mathcal{E}$ sta močno povezani, če obstaja pot od u do v in pot od v do u. Množice vseh takih vozlišč znotraj G, se imenujejo močne komponente – SCC. Močna komponenta, ki vsebuje vozlišče v, je označena kot $Comp(v)$. Močna komponenta, ki vsebuje samo eno vozlišče, se imenuje *trivialna komponenta.*

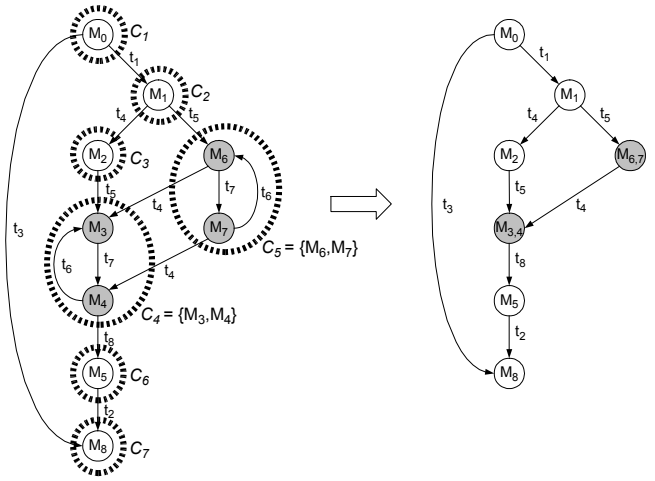
Vsako vozlišče v znotraj močne komponente ima enako množico naslednikov $Succ(v)$ [73], kar npr. omogoča učinkovite algoritme za iskanje tranzitivnih ovojnic. Obstajajo učinkoviti algoritmi za iskanje močnih komponent grafov; npr. *Tarjanov algoritem* [140, 73], ki ima linearno časovno zahtevnost $O(|\mathcal{V}| + |\mathcal{E}|)$. Ko najdemo SCC, lahko formiramo *graf komponent* G^{SCC}, ki je brez ciklov (je DAG).

Definicija 4.2.7.7 (Graf komponent)**.** Naj ima graf G k močno povezanih komponent $C_1, C_2, \ldots, C_k$. Graf komponent je $G^{SCC} = (\mathcal{V}^{SCC}, \mathcal{E}^{SCC})$. Množica vozlišč $\mathcal{V}^{SCC} = \{v_1, v_2, \ldots, v_k\}$ vsebuje v_i za vsako močno komponento C_i. Množica $\mathcal{E}^{SCC}$ vsebuje povezavo (v_i, v_j), če obstaja usmerjena povezava (x, y) za nek $x \in C_i$ in nek $y \in C_j$.

Če uporabimo Tarjanov algoritem [140] na drevesu pokritosti $\mathscr{T}$ s slike 4.11, dobimo močne komponente in posledično graf komponent, kot je prikazano na sliki 4.13. Vse močne komponente, razen $C_4 = \{M_3, M_4\}$ in $C_5 = \{M_6, M_7\}$, so trivialne ($C_1 = \{M_0\}$, $C_2 = \{M_1\}$, $C_3 = \{M_2\}$, $C_6 = \{M_5\}$, $C_7 = \{M_8\}$). Graf komponent G^{SCC} vsebuje manj vozlišč in povezav kot drevo pokritosti $\mathscr{T}$, vendar je sedaj brez ciklov. Direktna uporaba algoritma 4.3 na grafu G^{SCC} je neustrezna, ker ni informacij o stopnjah zaupanja manjkajočih povezav oz. prehodov $t_i \in T$.

Vsako ne-trivialno močno komponento C_i znotraj drevesa pokritosti transformiramo v dve novi vozlišči M_{C_i} in M'_{C_i} in usmerjeno povezavo t_{C_i} med njima. Množica $\mathcal{M}^{C_i} \subseteq \mathcal{S}(M_0)$ označuje vozlišča oz. oznake M_i, ki jih vsebuje močna komponenta C_i, množica $T^{C_i} \subseteq T$ pa označuje povezave oz. prehode t_i med vozlišči množice $\mathcal{M}^{C_i}$ (tvorijo cikel znotraj C_i).

Vozlišči M_{C_i} in M'_{C_i} nastaneta s združevanjem (t. i. *kompresijo*) vozlišč znotraj množice $\mathcal{M}^{C_i}$, pri čemer $|\mathcal{M}^{C_i}| \geq 2$. Pri konstrukciji novega DAG z M_{C_i} in M'_{C_i}, je potrebno upoštevati tudi vhodne in izhodne povezave po sledečem pravilu:



Slika 4.13: Pretvorba drevesa $\mathscr{T}$ v graf komponent G^{SCC}.

(a) Vhodna povezava $t \notin T^{C_i}$ je povezana na neko vozlišče $M \in \mathcal{M}^{C_i}$:
   - vozlišče M ima izhodno povezavo do drugega vozlišča $M' \in \mathcal{M}^{C_i} \Rightarrow$ vhodna povezava t je povezana na vozlišče M_{C_i};
   - vozlišče M nima izhodne povezave do drugega vozlišča $M' \in \mathcal{M}^{C_i} \Rightarrow$ vhodna povezava t je povezana na vozlišče M'_{C_i};

b) Izhodna povezava $t \notin T^{C_i}$ iz močne komponente C_i, ki ima izvor v nekem vozlišču $M' \in \mathcal{M}^{C_i}$, ima v novem DAG izvor v vozlišču M'_{C_i}. V primeru, da obstaja k izhodnih povezav $t_1, \ldots, t_k \notin T^{C_i}$ iz močne komponente C_i v drugo močno komponento C_j, jih nadomestimo z eno povezavo $t_{out} \notin T^{C_i}$ z izvorom v vozlišču M'_{C_i}. Povezava t_{out} ima minimalno stopnjo zaupanja med vsemi izhodnimi povezavami $t_i, i = 1, \ldots, k$ (4.2):

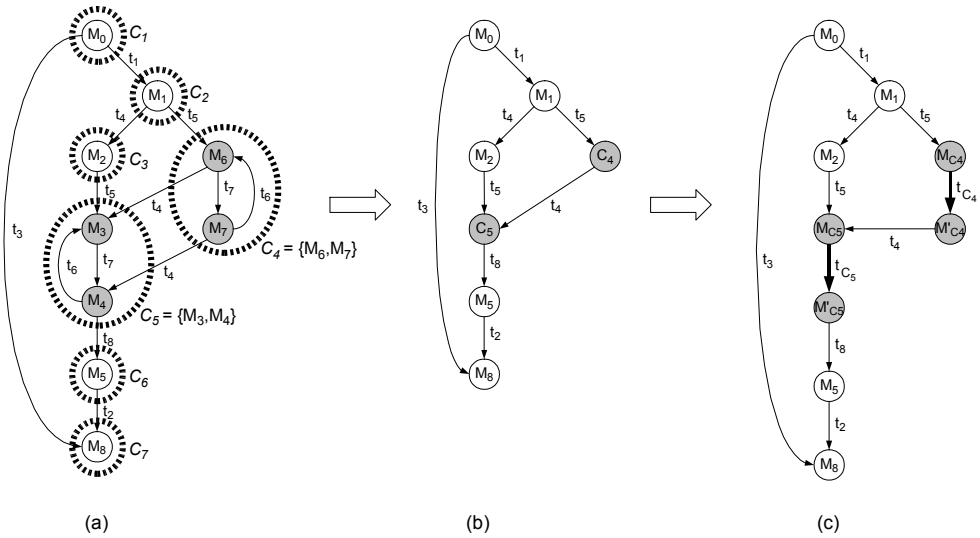
$$transitionTrust_{a,t_{out}}(\varphi) = \underset{t_i, i=1,\ldots,k}{\arg\min}\ transitionTrust_{a,t_i}(\varphi) \tag{4.2}$$

Za vsako netrivialno komponento C_i je potrebno določiti tudi stopnjo zaupanja povezave t_{C_i} med vozliščema M_{C_i} in M'_{C_i}; ta povezava nadomešča vse povezave $t_i \in T^{C_i}$ (sestavljajo cikel) znotraj močne komponente C_i. Zopet je potrebno vzeti minimalno stopnjo zaupanja med vsemi povezavami (4.3):

$$transitionTrust_{a,t_{C_i}}(\varphi) = \underset{t_i \in T^{C_i}}{\arg\min}\ transitionTrust_{a,t_i}(\varphi) \tag{4.3}$$

Sedaj lahko uporabimo predstavljena pravila pri našem drevesu pokritosti $\mathscr{T}$ oz. izpeljanem grafu G^{SCC} komponent s slike 4.13. G^{SCC} pretvorimo naprej v DAG[16], na katerem uporabimo algoritem 4.3 in tako izračunamo stopnjo zaupanja kompozita. Slika 4.14 prikazuje celotno transformacijo drevesa pokritosti z vsebovanimi cikli v usmerjen acikličen graf DAG, kjer je možna aplikacija algoritma. Netrivialni močni komponenti sta C_4 in C_5 s pripadajočimi množicami vozlišč ter povezav: $\mathcal{M}^{C_4} = \{M_6, M_7\}$, $\mathcal{M}^{C_5} = \{M_3, M_4\}$, $T^{C_4} = T^{C_5} = \{t_6, t_7\}$.

[16] Sicer je že G^{SCC} acikličen usmerjen graf, vendar neustrezen za algoritem 4.3.



Slika 4.14: Drevo pokritosti $\mathscr{T}$ (a), graf komponent G^{SCC} (b), končni DAG (c).

Vhodna povezava za C_4 je t_5, ki je povezana na M_6. Ker je to vozlišče povezano z M_7, je v novem grafu DAG t_5 povezana na M_{C_4}. Prav tako je v primeru močne komponente C_5 – vhodna povezava t_4 je povezana na M_3, to vozlišče pa z M_4. V novem grafu je t_4 povezana na M_{C_5}. Izhodna povezava iz C_4 v C_5 je t_4, v novem grafu DAG povezuje M'_{C_4} in M_{C_5} (slika 4.14 (c)). Stopnji zaupanja povezav t_{C_4} in t_{C_5} sta enaki, ker množici T^{C_4} in T^{C_5} sovpadata. Stopnja zaupanja je glede na formulo 4.3 enaka:

$$transitionTrust_{a,t_{C_4}}(\varphi) = \min(transitionTrust_{a,t_6}(\varphi), transitionTrust_{a,t_7}(\varphi)).$$

Sedaj lahko izračunamo stopnjo zaupanja kompozita, ki ga predstavlja Petrijeva mreža (slika 4.10) oz. pripadajoče drevo pokritosti $\mathscr{T}$ (slika 4.11). Množica vrednosti stopenj zaupanja $\mathcal{T}$ je vzeta iz kvantitativnega modela ATM (glej poglavje 3, razdelek 3.1.5) z operatorjem npr. za konjunkcijo op $= \wedge$. Prehodi t_3, t_4 in t_7 predstavljene PN (slika 4.10), so aktivnosti tipa `invoke` (prožijo spletne storitve) s sledečimi stopnjami zaupanja $trust_{a,t_i}(\varphi) \in \mathcal{T}$:

$$trust_{a,t_3}(\varphi) = (0.6, 0.3, 0.1),$$
$$trust_{a,t_4}(\varphi) = (0.8, 0.2, 0),$$
$$trust_{a,t_7}(\varphi) = (0.3, 0.5, 0.2).$$

Ostali prehodi $t_i, i \in \{1, 2, 5, 6, 8\}$ so kontrolni s stopnjo zaupanja (glej formulo 4.1):

$$transitionTrust_{a,t_i}(\varphi) = \sup(\mathcal{T}) = (1, 0, 0).$$

Algoritem 4.3 uporabimo na acikličnem grafu s slike 4.14 (c), vendar je pred tem potrebno izračunati stopnje zaupanja za povezavi oz. prehoda t_{C_4} in t_{C_5} (sta enaki):

$$transitionTrust_{a,t_{C_4}}(\varphi) = \min((1, 0, 0), (0.3, 0.5, 0.2)) = (0.3, 0.5, 0.2)^{(17)}.$$

(17) Ker je $E((1, 0, 0,)) = 1$ in $E((0.3, 0.5, 0.2)) = 0.4$ sledi $(0.3, 0.5, 0.2) \leq (1, 0, 0)$.

Potek algoritma 4.3 je sledeč[18]:

a) $CompositeTrust_{a,M_i}(\varphi) = (1,0,0)$, $M_i \in \{M_0, M_1, M_2, M_5, M_8, M_{C_4}, M'_{C_4}, M_{C_5}, M'_{C_5}\}$
b) topološka urejenost $L = \{M_0, M_1, M_2, M_{C_4}, M'_{C_4}, M_{C_5}, M'_{C_5}, M_5, M_8\}$;

c) $M_x = M_0, L = \{M_1, M_2, M_{C_4}, M'_{C_4}, M_{C_5}, M'_{C_5}, M_5, M_8\}$
- $t = t_3, M_y = M_8$:
  $trust_x = CTrust_{a,M_0}(\varphi) \wedge tTrust_{a,t_3}(\varphi) = (1,0,0) \wedge (0.6, 0.3, 0.1) = (0.6, 0.3, 0.1)$;
  $CTrust_{a,M_8}(\varphi) = \min((0.6, 0.3, 0.1), (1,0,0)) = (0.6, 0.3, 0.1)$;
- $t = t_1, M_y = M_1$:
  $trust_x = CTrust_{a,M_0}(\varphi) \wedge tTrust_{a,t_1}(\varphi) = (1,0,0) \wedge (1,0,0) = (1,0,0)$;
  $CTrust_{a,M_1}(\varphi) = \min((1,0,0), (1,0,0)) = (1,0,0)$;

d) $M_x = M_1, L = \{M_2, M_{C_4}, M'_{C_4}, M_{C_5}, M'_{C_5}, M_5, M_8\}$
- $t = t_4, M_y = M_2$:
  $trust_x = CTrust_{a,M_1}(\varphi) \wedge tTrust_{a,t_4}(\varphi) = (1,0,0) \wedge (0.8, 0.2, 0) = (0.8, 0.2, 0)$;
  $CTrust_{a,M_2}(\varphi) = \min((0.8, 0.2, 0), (1,0,0)) = (0.8, 0.2, 0)$;
- $t = t_5, M_y = M_{C_4}$:
  $trust_x = CTrust_{a,M_1}(\varphi) \wedge tTrust_{a,t_5}(\varphi) = (1,0,0) \wedge (1,0,0) = (1,0,0)$;
  $CTrust_{a,M_{C_4}}(\varphi) = \min((1,0,0), (1,0,0)) = (1,0,0)$;

e) $M_x = M_2, L = \{M_{C_4}, M'_{C_4}, M_{C_5}, M'_{C_5}, M_5, M_8\}$
- $t = t_5, M_y = M_{C_5}$:
  $trust_x = CTrust_{a,M_2}(\varphi) \wedge tTrust_{a,t_5}(\varphi) = (0.8, 0.2, 0) \wedge (1,0,0) = (0.8, 0.2, 0)$;
  $CTrust_{a,M_{C_5}}(\varphi) = \min((0.8, 0.2, 0), (1,0,0)) = (0.8, 0.2, 0)$;

f) $M_x = M_{C_4}, L = \{M'_{C_4}, M_{C_5}, M'_{C_5}, M_5, M_8\}$
- $t = t_{C_4}, M_y = M'_{C_4}$:
  $trust_x = CTrust_{a,M_{C_4}}(\varphi) \wedge tTrust_{a,t_{C_4}}(\varphi) = (1,0,0) \wedge (0.3, 0.5, 0.2) =$
  $= (0.3, 0.5, 0.2)$;
  $CTrust_{a,M'_{C_4}}(\varphi) = \min((0.3, 0.5, 0.2), (1,0,0)) = (0.3, 0.5, 0.2)$;

g) $M_x = M'_{C_4}, L = \{M_{C_5}, M'_{C_5}, M_5, M_8\}$
- $t = t_4, M_y = M_{C_5}$:
  $trust_x = CTrust_{a,M'_{C_4}}(\varphi) \wedge tTrust_{a,t_4}(\varphi) = (0.3, 0.5, 0.2) \wedge (0.8, 0.2, 0) =$
  $= (0.24, 0.6, 0.16)$;
  $CTrust_{a,M_{C_5}}(\varphi) = \min((0.24, 0.6, 0.16), (0.8, 0.2, 0)) = (0.24, 0.6, 0.16)$;

h) $M_x = M_{C_5}, L = \{M'_{C_5}, M_5, M_8\}$
- $t = t_{C_5}, M_y = M'_{C_5}$:
  $trust_x = CTrust_{a,M_{C_5}}(\varphi) \wedge tTrust_{a,t_{C_5}}(\varphi) = (0.24, 0.6, 0.16) \wedge (0.3, 0.5, 0.2) =$
  $= (0.072, 0.8, 0.128)$;
  $CTrust_{a,M'_{C_5}}(\varphi) = \min((0.072, 0.8, 0.128), (1,0,0)) = (0.072, 0.8, 0.128)$;

[18] Zaradi preglednosti označujemo $CompositeTrust_{a,t}(\varphi)$ z okrajšavo $CTrust_{a,t}(\varphi)$, $transitionTrust_{a,t}(\varphi)$ pa z $tTrust_{a,t}(\varphi)$.

i) $M_x = M'_{C_5}, L = \{M_5, M_8\}$

- $t = t_8, M_y = M_5$:
  $trust_x = CTrust_{a,M'_{C_5}}(\varphi) \wedge tTrust_{a,t_8}(\varphi) = (0.072, 0.8, 0.128) \wedge (1, 0, 0) =$
  $= (0.072, 0.8, 0.128)$;
  $CTrust_{a,M_5}(\varphi) = \min((0.072, 0.8, 0.128), (1, 0, 0)) = (0.072, 0.8, 0.128)$;

j) $M_x = M_5, L = \{M_8\}$

- $t = t_2, M_y = M_8$:
  $trust_x = CTrust_{a,M_5}(\varphi) \wedge tTrust_{a,t_2}(\varphi) = (0.072, 0.8, 0.128) \wedge (1, 0, 0) =$
  $= (0.072, 0.8, 0.128)$;
  $CTrust_{a,M_8}(\varphi) = \min((0.072, 0.8, 0.128), (0.6, 0.3, 0.1)) = (0.072, 0.8, 0.128)$;

k) $M_x = M_8 = M_{end}, L = \emptyset$, ni več izhodnih povezav;

l) izračunana in vrnjena vrednost: $CompositeTrust_{a,M_8}(\varphi) = (0.072, 0.8, 0.128)$.

Sedaj lahko združimo posamezne postopke za izračun stopnje zaupanja kompozita storitev, da dobimo algoritem, ki velja za splošen proces BPEL. Algoritem 4.4 prikazuje posamezne korake pri računanju stopnje procesa BPEL; nekateri od njih so opisni, drugi pa predstavljajo ustrezne algoritme, definirane znotraj tega razdelka. Proces BPEL se preslika v PN preko orodja BPEL2PNML ter analizira preko orodja WofBPEL (glej razdelek 4.2.5 in 4.2.6). V primeru, da analiza oz. formalna verifikacija ni uspešna (process BPEL vsebuje napake), tudi izračun stopnje kompozita ni možen.

```
procedure CompositeTrust(BPEL p_BPEL, Entity a, Operator op, Context φ) {
  PetriNet PN = BPEL2PNML(p_BPEL);
  if (AnalyseWofBPEL(PN) == OK) {
     /* redukcijska pravila */
     PetriNet PN' = MakeReduction(PN);
     /* bisimularnost */
     while not (PN' ∼ PN or PN' ≈ PN)
       PN' = MakeReduction(PN);
     DAG 𝒯 = CoverabilityTree(PN');
     /* topološko sortiranje */
     List L = TopologicalSort(𝒯);
     if (L == ∅) {
       /* prisotni cikli */
       Graph G^SCC = Tarjan(𝒯);
       /* kompresija močnih komponent v nov DAG */
       /* za vsako netrivialno močno komponento C_i določi: M_Ci, M_Ci', t_Ci */
       𝒯 = Compress(G^SCC);
     }
     return ComputeCompositeTrust(𝒯, a, op, φ);
  } else {
    throw error ("Proces BPEL ima napake");
  }
}
```

Algoritem 4.4: Izračun stopnje zaupanja za proces BPEL.

Poglavje 5

Sistem za obvladovanje zaupanje in ugleda

5.1 Splošno

V predhodnih poglavjih (poglavji 3 in 4) smo definirali ustrezne gradnike (formalni model zaupanja ATM, povezava ATM s kompozicijo storitev oz. poslovnimi procesi BPEL), katere bomo združili v ustrezno aplikativno rešitev za obvladovanje zaupanja in ugleda na infrastrukturi SOA. Ker so spletne storitev osnovne komponente v SOA (tvorijo t. i. storitveno omrežje), bo tudi rešitev izvedena kot samostojna spletna storitev. Pri uporabi modela zaupanja ATM lahko spletne storitve smatramo za programske agente oz. entitete, ki znajo izpolniti točno določeno nalogo in jih izvajajo (t. i. prožijo) udeleženci v procesu e-poslovanja. Tako prihaja do interakcij med *tehnologijo* (t. i. spletne storitve) in uporabniki (ljudje) z določenim socialnim obnašanjem. Mehki varnostni mehanizem ATM predstavlja osnovo za odločanje o prihodnjih interakcijah med entitetami v storitvenem omrežju, poleg tega pa omogoča tudi identifikacijo entitet z zlonamernim obnašanjem.

Obstajajo različni pristopi za integracijo zaupanja in ugleda v okolja SOA. Zhengping in ostali [141] predlagajo ogrodje zaupanja, ki omogoča načrtovanje zaupanja vrednih aplikacij SOA. Obsega nadzor obnašanja storitev pri izvajanju (za odkrivanje napak, izjem) in njihovo selekcijo glede na stopnjo zaupanja. Za izračun zaupanja se uporablja kvantitativni model na podlagi priporočil. Stopnja zaupanja storitve v SOA določa *kakovost storitve* (angl. quality of service – QoS). Chang in ostali [142, 143, 144] definirajo model z osnovnimi relacijami zaupanja in ugleda med različnimi entitetami (izvorna, ciljna, tretja). Zaupanje se oblikuje na podlagi priporočil tretjih (angl. third party) entitet. Entiteta v modelu lahko predstavlja agenta, izdelek ali storitev. Stopnja zaupanja entitete določa njeno kakovost. Model zaupanja v SOA, ki ga predlagajo Zhu in ostali [12], uporablja socio-kognitivni model zaupanja, ki sta ga predlagala Castelfranchi in Falcone [19]. Formalni model omogoča analizo in sklepanje o akcijah in zaupanju preko podanih socialnih pravil agentov.

Naš pristop integracije modela zaupanja v SOA ima določene sorodnosti z omenjenimi pristopi, izhajamo pa iz predlaganih modelov v [63, 62]. V nadaljevanju se osredotočimo predvsem na arhitekturo predlaganega sistema za obvladovanje zaupanja, izpostavitvi varnostnih

mehanizmov spletnih storitev (WS-Security, WS-Trust) [102] ter integracijo modela ATM v standard WS-Trust.

5.2 Infrastruktura SOA

Spletne storitve so osnovni gradniki in ključna tehnologija za porazdeljene aplikacije SOA. Infrastruktura SOA predvideva tri ključne elemente oz. entitete [3, 100]:

- *Ponudnik storitev* (angl. service provider): ponuja vmesnike do storitev, ki izvajajo množico poslovnih opravil oz. procesov.
- *Register storitev* (angl. service registry): vsebuje opise storitev ponudnikov; vsaka storitev je uvrščena v ustrezne kategorije z opisanimi atributi. Za lociranje storitev so na voljo iskalni mehanizmi.
- *Odjemalec storitev* (angl. service consumer): locira storitve v registru, se poveže na njihove *končne točke* in jih proži.

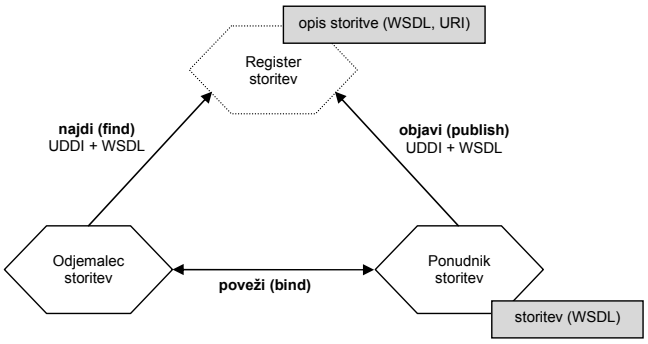
Omenjene entitete medsebojno izvajajo cikel operacij *objavi-najdi-poveži* (angl. publish-find-bind):

- *Objavi* (angl. publish): ponudnik storitev objavi svoje storitve znotraj registra storitev; objava poteka preko specifikacije UDDI [99].
- *Najdi* (angl. find): odjemalec storitev znotraj registra preko vmesnika UDDI najde (locira) specifično storitev, ki je opisana z jezikom WSDL [98] in vsebuje končno točko URI storitve na strani ponudnika.
- *Poveži* (angl. bind): odjemalec storitve se na podlagi najdenih informacij poveže na storitev in jo lahko neposredno proži.

Entitete in operacije predstavljajo t. i. *trikotnik* SOA, ki ga prikazuje slika 5.1. Velja poudariti, da v praksi rešitve SOA redko implementirajo celoten trikotnik SOA oz. cikel operacij *objavi-najdi-poveži*. Največkrat gre samo za odjemalca in ponudnika storitev (samo operacija *poveži*). To pomeni, da so storitve oz. končne točke v naprej določene na statičen način, kar ruši osnovne principe SOA (t. i. šibka sklopljenost). Omenjeni problem izpostavlja Michlmayr in ostali [145] ter predlagajo rešitev za dinamično povezovanje in proženje storitev.

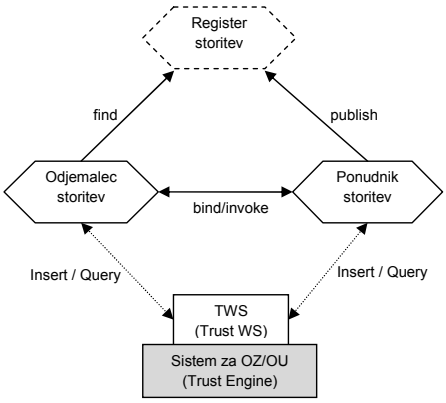
5.2.1 Spletna storitev zaupanja

Naša rešitev za OZ/OU na podlagi modela zaupanja ATM je izvedena kot samostojna *spletna storitev zaupanja* (angl. Trust Web Service – TWS) [63]. Storitev predstavlja dodaten logičen gradnik, ki ga je potrebno umestiti v trikotnik SOA.



Slika 5.1: Trikotnik SOA z entitetami in operacijami.

TWS predstavlja vmesnik do stroja za izračun zaupanja po modelu ATM. Vmesnik vsebuje sledeči operaciji – (a) *Insert* (omogoča dodajanje novih ocen zaupanje po končanih interakcijah) in (b) *Query* (omogoča izračun stopnje zaupanja). Novi cikel operacij trikotnika SOA je sedaj *publish-find-Query-bind-invoke-Insert*. Operaciji nista obvezni, prožita pa jih odjemalec in ponudnik pred bilateralno interakcijo. Razširjeni trikotnik SOA prikazuje slika 5.2. Dodan je element TWS kot vmesnik do sistema za OZ/OU (angl. trust engine – TE). TE



Slika 5.2: Trikotnik SOA s spletno storitvijo zaupanja (TWS).

omogoča shranjevanje ocen na trajni podatkovni vir ter izračun stopnje zaupanja entitet glede na algoritem Γ (glej poglavje 3, razdelek 3.1.4 in 3.1.5).

Izpostaviti je potrebno, da mora obstajati implicitno zaupanje med odjemalcem/ponudnikom storitev in TWS/registrom storitev. Implicitno zaupanje se lahko omogoči s trdnimi varnostnimi mehanizmi spletnih storitev.

5.2.2 Varnostni mehanizmi spletnih storitev

Zaradi odprte narave interakcij med spletnimi storitvami, je varnost poglavitni pogoj, ki ga je potrebno upoštevati pri realizaciji aplikacij SOA [102, 146]. Najprej je potrebno izpostaviti osnovne lastnosti, ki jih je potrebno zagotoviti v okviru varnosti informacij [5]:

- *Zaupnost* (angl. confidentiality): dostop do informacije ima le tisti posameznik ali skupina, ki ji je informacija namenjena.
- *Celovitost* (angl. integrity): informacija od svojega nastanka pa do trenutka, ko jo sprejemamo, ni bila spremenjena (uporaba mehanizma digitalnega podpisa).
- *Verodostojnost — overjanje* (angl. authenticity): zmožnost, da lahko identificiramo vir informacije in se prepričamo o tem, od kod informacija prihaja (uporaba mehanizma šifriranja/enkripcije informacij).
- *Nezanikanje* (angl. non-repudiation): to lastnost ima informacija, če tisti, ki jo je pripravil, pozneje ne more zanikati, da je res njen avtor.

Glede na to, da je protokol SOAP [96] osnovni način za komunikacijo med storitvami, je potrebna ustrezna varnost ravno tukaj. Varnost spodaj ležečih transportnih protokolov ni osrednji problem, saj obstaja precej standardov, ki se lahko uporabljajo tudi za varnost spletnih storitev [5]: *SSL* (Secure Socket Layer), novejši *TLS* (Transport Layer Security), *IPSec* (Internet Protocol Security). Osnovna omejitev teh mehanizmov je, da lahko delujejo samo med dvema končnima točkama (angl. point-point) po principu zahteva/odgovor, standardna implementacija pa je samo na protokolu HTTP (Hypertext Transfer Protocol), kar je precejšnja omejitev.

Bolj učinkovit način je uporaba sklopa varnostnih specifikacij WS-Security (WSS) [147, 102]. Določa razširjene varnostne mehanizme znotraj glave SOAP, kar omogoča pošiljanje *varnostnih žetonov* (angl. security tokens) za varno identifikacijo (avtorizacijo) in verodostojnost (overjanje) odjemalcev, poleg tega zagotavlja celovitost in zaupnost sporočil SOAP.

V splošnem je specifikacija WSS odprta, podpira vse znane varnostne modele (PKI, SSL, Kerberos), različne formate digitalnega podpisa in različne kriptografske algoritme. WSS funkcionalno razširja sporočila SOAP preko elementa `<wsse:Security>` kot ga prikazuje izsek sporočila SOAP na primeru 5.1.

```
<?xml version="1.0" encoding="utf-8"?>
<S12:Envelope xmlns:S12="..." xmlns:wsse="...">
  <S12:Header>
    <wsse:Security S12:role="..." S12:mustUnderstand="...">
      ...
    </wsse:Security>
    ...
  </S12:Header>
  <S12:Body>
   ...
  </S12:Body>
</S11:Envelope>
```

Primer 5.1: Sporočilo SOAP z dodano varnostno glavo.

Element ima dva opcijska atributa:

- `S12:role`: določa ustrezno vlogo SOAP 1.2 v obliki URI (Uniform Resource Identifier), na katero se usmerja glava SOAP. Vloga določa točko, ki je lahko vmesna ali končna [96].
- `S12:mustUnderstand`: določa, ali mora prejemnik sporočila SOAP obvezno ali opcijsko procesirati varnostno glavo. V primeru, da je atribut izpuščen, procesiranje glave ni obvezno.

Znotraj `<wsse:Security>` so definirani varnostni elementi, ki pripadajo različnim imenskim prostorom (specifikacija XML 1.0 + Namespaces [148]) specifikacije WSS [147] (tabela 5.1) v obliki URI.

Predpona	**Imenski prostor**
ds	http://www.w3.org/2000/09/xmldsig#
S11	http://schemas.xmlsoap.org/soap/envelope/
S12	http://www.w3.org/2003/05/soap-envelope
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wsse11	http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd
xenc	http://www.w3.org/2001/04/xmlenc#
wsc	http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512
wst	http://docs.oasis-open.org/ws-sx/ws-trust/200512

Tabela 5.1: Predpone in pripadajoči imenski prostori WSS.

Elementi znotraj `<wsse:Security>` zagotavljajo prej naštete lastnosti: (a) zaupnost se zagotavlja preko šifriranja (enkripcije) XML, (b) celovitost se zagotavlja preko digitalnega podpisa XML, in (c) verodostojnost preko varnostnih žetonov (angl. security tokens). Sledi osnovni pregled omenjenih varnostnih lastnosti sporočil SOAP.

5.2.2.1 Zaupnost

Za zaupnost sporočil se uporablja standard *XML Encryption* (XMLEnc) [149]. Omogoča šifriranje celotnega ali posameznih delov sporočila SOAP, kot tudi njegovih priponk (angl. attachments). Za šifriranje podatkov se lahko uporabi simetrični ali asimetrični algoritem. Pri simetričnem pošiljatelj in naslovnik za šifriranje/dešifriranje uporabljata enak ključ. Pri asimetričnem algoritmu se uporablja par privatni/javni ključ (certifikat X.509 [5]). Z javnim ključem se podatki šifrirajo, s privatnim pa dešifrirajo.

Glavni elementi, ki jih določa standard [149], so `EncryptedData`, `EncryptedKey` in `ReferenceList`. Primer 5.2 prikazuje sporočilo SOAP, ki vsebuje šifrirane podatke kot tudi šifriran simetrični ključ. Pri simetrični enkripciji se ključ zaradi večje varnosti pošilja šifriran z javnim ključem prejemnika sporočila. Ko prejemnik prejme sporočilo, ga lahko dešifrira s svojim

```
<S11:Envelope xmlns:S11="..."
     xmlns:wsse="..." xmlns:wsu="..." xmlns:ds="..." xmlns:xenc="...">
<S11:Header>
 <wsse:Security>
  <xenc:EncryptedKey>
   <xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
   <ds:KeyInfo>
    <ds:KeyName>
     CN=Key13, C=US
    </ds:KeyName>
   </ds:KeyInfo>
   <xenc:CipherData>
    <xenc:CipherValue>
      ir4DfG . . .
    </xenc:CipherValue>
   </xenc:CipherData>
   <xenc:ReferenceList>
    <xenc:DataReference URI="#EncryptedBody"/>
   </xenc:ReferenceList>
  </xenc:EncryptedKey>
 </wsse:Security>
</S11:Header>
<S11:Body>
 <xenc:EncryptedData wsu:Id="EncryptedBody">
  <xenc:EncryptionMethod
        Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
   <xenc:CipherData>
    <xenc:CipherValue>
      r5KipsDV . . .
    </xenc:CipherValue>
   </xenc:CipherData>
  </xenc:EncryptedData>
 </S11:Body>
</S11:Envelope>
```

Primer 5.2: Šifrirano sporočilo SOAP.

privatnim ključem, potem pa uporabi simetrični ključ za dejansko dešifriranje podatkov. V tem primeru se vsi poglavitni elementi nahajajo v telesu (angl. body) sporočila:

- `xenc:EncryptedData`: vsebuje informacije o šifriranju in šifrirane podatke (rezultat enkripcije) preko svojih podrejenih elementov, splošna struktura elementa je sledeča (znak "`?`" pomeni nič ali eno pojavitev, "`+`" pomeni eno ali več pojavitev, "`*`" pomeni nič ali več pojavitev elementa) [149]:

  ```
  <EncryptedData Id? Type? MimeType? Encoding?>
    <EncryptionMethod/>?
    <ds:KeyInfo>
      <EncryptedKey>?
      <AgreementMethod>?
      <ds:KeyName>?
      <ds:RetrievalMethod>?
  ```

```
        <ds:*>?
      </ds:KeyInfo>?
      <CipherData>
        <CipherValue>?
        <CipherReference URI?>?
      </CipherData>
      <EncryptionProperties>?
    </EncryptedData>
```

- `xenc:EncryptionMethod`: določa algoritem za šifriranje podatkov (simetrični ali asimetrični).
- `xenc:CipherData`: preko podrejenega elementa `xenc:CipherValue` vsebuje šifrirane podatke v formatu `base64`.

Informacija o ključu, ki je šifriran, se nahaja v glavi sporočila preko elementa `xenc:EncryptedKey`; šifriran je preko javnega ključa z algoritmom RSA (element `xenc:EncryptionMethod`). Ostali elementi so še:

- `xenc:KeyInfo`: informacija o ključu; ime ključa (element `xenc:KeyName`).
- `xenc:CipherData`: enak pomen kot pri `xenc:EncryptedData` elementu zgoraj, le da ta vsebuje šifrirano vrednost ključa.
- `xenc:ReferenceList`: podaja seznam referenc na dele sporočila SOAP (element `xenc:DataReference`), ki uporabljajo ta ključ oz. so šifrirani s tem ključem.

5.2.2.2 Celovitost (integriteta)

Celovitost sporočil SOAP je rešena preko standarda *XML Signature* (XMLDsig) [150], ki omogoča digitalni podpis [5, 8] dokumentov XML. Osnovni element je `Signature` (določa, kje se podpis začne in konča) s sledečo strukturo (znak "`?`" pomeni nič ali eno pojavitev, "`+`" pomeni eno ali več pojavitev, "`*`" pomeni nič ali več pojavitev elementa) [150]:

```
<Signature ID?>
   <SignedInfo>
     <CanonicalizationMethod/>
     <SignatureMethod/>
     (<Reference URI? >
       (<Transforms>)?
       <DigestMethod>
       <DigestValue>
     </Reference>)+
   </SignedInfo>
   <SignatureValue>
  (<KeyInfo>)?
  (<Object ID?>)*
</Signature>
```

Podrejeni elementi (`SignedInfo`, `SignatureValue`, `KeyInfo`) natančno opisujejo digitalni podpis. Če je podpisov znotraj enega dokumenta več, vsak element `Signature` vsebuje identifikacijsko vrednost preko atributa `wsu:Id`. Primer 5.3 prikazuje digitalni podpis znotraj sporočila SOAP.

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="...">
 <S11:Header>
  <wsse:Security>
   <wsse:BinarySecurityToken ValueType="wsse:X509v3"
         EncodingType="wsse:Base64Binary"
         wsu:Id="X509Cert">
     KkFPle ...
   </wsse:BinarySecurityToken>
   <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
     <ds:CanonicalizationMethod
         Algorithm="http://www.w3.org/2001/10/xml-exc-c14N"/>
     <ds:SignatureMethod
         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
     <ds:Reference URI="#MessageBody">
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>
        aOb4Luuk...
      </ds:DigestValue>
     </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
      A9qqIrtE3xZ...
    </ds:SignatureValue>
    <ds:KeyInfo>
     <wsse:SecurityTokenReference>
      <wsse:Reference URI="#X509Cert"/>
     </wsse:SecurityTokenReference>
    </ds:KeyInfo>
   </ds:Signature>
  </wsse:Security>
 </S11:Header>
 <S11:Body wsu:Id="MessageBody">
  ...
 </S11:Body>
</S11:Envelope>
```

Primer 5.3: Digitalni podpis znotraj sporočila SOAP.

Element `Signature` vsebuje več podrejenih elementov:

- `ds:SignedInfo`: informacija o podpisu, je najkompleksnejši element, ker vsebuje največ podrejenih elementov in referenc. Vsebuje ključno informacijo o podpisu.
- `ds:CanonicalizationMethod`: normalizacijski algoritem (npr. normiranje začetnih in končnih presledkov) za podatke, ki bodo digitalno podpisani. Pomembno je, da je dokument XML za digitalni podpis v predpisani standardni obliki.

- `ds:SignatureMethod`: algoritem za digitalni podpis.
- `ds:Reference`: kazalec URI na podatke znotraj dokumenta oz. na spletna sredstva, kjer so podatki za podpis. Sklic se lahko nanaša na poljubno datoteko, bodisi na disku, krajevnem omrežju ali spletu. Znotraj sporočila SOAP tipično kaže na podatke znotraj telesa in dele znotraj glave.
- `ds:DigestMethod`: algoritem za zgoščevanje (vsebuje kazalec na opis).
- `ds:DigestValue`: vrednost (rezultat) zgoščevalnega algoritma.
- `ds:SignatureValue`: vrednost (rezultat) je seznam bajtov, ki nastane s podpisom podatkov.
- `ds:KeyInfo`: določa javni ključ za validacijo digitalnega podpisa. Lahko je določen kot referenca na digitalno potrdilo (certifikat), ki vsebuje javni ključ (kot v primeru 5.3) ali pa je vsebovan znotraj podrejenih elementov `ds:KeyName` in `ds:KeyValue`.

5.2.2.3 Verodostojnost

Verodostojnost[1] (overjanje) sporočil se rešuje preko *varnostnih žetonov* (angl. security tokens). Varnostni žeton predstavlja zbirka ene ali več *trditev* (angl. claims) [147]. Overjanje sporočil SOAP je možno z različnimi tipi varnostnih žetonov, ki jih določajo ustrezni profili specifikacij v okviru WSS [147]. Profili so sledeči:

(1) *WSS UsernameToken Profile* [151]. Varnostni žeton je predstavljen z uporabniškim imenom in geslom. Definiran je z elementom `wsse:UsernameToken` znotraj `wsse:Security`; vsebuje uporabniško ime `wsse:Username` in geslo `wsse:Password`. Tip gesla je določen preko atributa `Type`, ki ima dve možni vrednosti v obliki URI "`imenski-prostor`[2]`#vrednost`":

   - `#PasswordText`: geslo je podano kot navaden (angl. plain) tekst;
   - `#PasswordDigest`: geslo je v kodirani obliki preko zgoščevalne (angl. hash) funkcije (SHA-1, MD5 [8]).

   Žeton, ki ga prikazuje sledeči izsek glave SOAP, vsebuje kombinacijo uporabniškega imena in gesla kot navaden tekst.

```
<wsse:UsernameToken wsu:Id="primer1">
  <wsse:Username>Janez</wsse:Username>
  <wsse:Password Type="..#PasswordText">MojeGeslo</wsse:Password>
</wsse:UsernameToken>
```

[1] Tu gre predvsem za odgovor na vprašanje: koga overjam oz. kdo je poslal sporočilo?
[2] `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0`.

Obstajata tudi dva opcijska elementa, ki otežkočata delo napadalca pri razbijanju vrednosti žetona – `wsse:Nonce` in `wsu:Created`. Prvi element *določa priložnostno skovanko (besedo)* (angl. nonce), ki jo pošiljatelj doda v vsak varnostni žeton. Drugi element določa časovni žig žetona. Če sta omenjena elementa prisotna, se uporabita za izračun vrednosti gesla po sledeči formuli (5.1) [151]:

$$PasswordDigest = \text{Base64}(\text{SHA-1}(nonce + created + password)) \tag{5.1}$$

(2) *WSS X.509 Certificate Token Profile* [152]. Varnostni žeton predstavlja digitalno potrdilo (certifikat) X.509 [33]. Definiran je preko elementa `wsse:BinarySecurityToken` znotraj `wsse:Security`. Žeton znotraj sporočila SOAP predstavlja javni ključ potrdila. Tip žetona za X.509 je podan preko atributa `ValueType` in ima več možnih vrednosti v obliki URI "`imenski-prostor`[3]`#vrednost`":

- `#X509v3`, `#X509v1`: digitalno potrdilo tipa X.509 v3 oz. X.509 v1;
- `#X509PKIPathv1`: urejen seznam digitalnih potrdil X.509, ki je predstavljen v obliki *PKIPath* [153];
- `#PKCS7`: seznam digitalnih potrdil X.509, kjer je lahko opcijsko podan tudi seznam preklicanih potrdil (CRL) v obliki ovojnice `PKCS#7` [154].

Način kodiranja znotraj sporočila je dočen z atributom `EncodingType`, ki ima lahko vrednost `#Base64Binary` in `#HexBinary`. Potrdilo samo po sebi točno določa uporabnika, sicer pa omogoča tudi digitalni podpis dela sporočila s privatnim ključem. Sledeči primer prikazuje definicijo žetona:

```
<wsse:BinarySecurityToken
      ValueType="..#X509v3" EncodingType="..#Base64Binary">
   KkFPleDjCCB ...
</wsse:BinarySecurityToken>
```

(3) *WSS Kerberos Token Profile* [155]. Varnostni žeton predstavlja vstopnico Kerberos [156]. Definiran je preko elementa `wsse:BinarySecurityToken` znotraj elementa `wsse:Security`. Tip vstopnice Kerberos določa atribut `ValueType`, ki ima več različnih možnih vrednosti v obliki URI "`imenski-prostor`[4]`#vrednost`" (obstaja več različic tipov glede na specifikacije `RFC`[5]):

- `#Kerberosv5_AP_REQ`: specifično sporočilo Kerberos V5, ki predstavlja zahtevo tipa `AP-REQ` (glej [156]);
- `#GSS_Kerberosv5_AP_REQ`: žeton mehanizma `GSS-API` s sporočilom tipa `KRB_AP_REQ` (glej [157]).

[3] `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token- profile-1.0`.
[4] `http://docs.oasis-open.org/wss/oasiswss-kerberos-token-profile-1.1`.
[5] IETF Request For Comments: 1510, 1964, 4120.

(4) *WSS SAML Token Profile* [158]. Varnostni žetoni predstavljajo trditve v jeziku *SAML* (angl. Security Assertion Markup Language) [159], ki je jezik, definiran na sintaksi XML (v uporabi so verzije 1.1 in 2.0, med katerimi so določene razlike). Jezik omogoča izmenjavo podatkov za overjanje in avtorizacijo med varnostnimi domenami. Trditve SAML so sestavljene iz izjav; npr. izjava v SAML 1.1 vsebuje element `saml`[(6)]`:Subject` z elementom `saml:SubjectConfirmation`, ki definira pravila za potrditev subjekta in trditev. Sledi primer sporočila SOAP, ki vsebuje trditev SAML 1.1:

```
<S12:Envelope xmlns:S12="...">
  <S12:Header>
    <wsse:Security xmlns:wsse="...">
      <saml:Assertion xmlns:saml="..."
            AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
            IssueInstant="2003-04-17T00:46:02Z"
            Issuer="www.opensaml.org"
            MajorVersion="1"
            MinorVersion="1">
        <saml:AuthenticationStatement>
         <saml:Subject>
           <saml:NameIdentifier
           NameQualifier="www.example.com"
           Format="urn:oasis:names:tc:SAML:1.1:nameidformat:X509SubjectName">
             uid=joe,ou=people,ou=saml-demo,o=baltimore.com
           </saml:NameIdentifier>
            <saml:SubjectConfirmation>
              <saml:ConfirmationMethod>
                 urn:oasis:names:tc:SAML:1.0:cm:bearer
              </saml:ConfirmationMethod>
            </saml:SubjectConfirmation>
         </saml:Subject>
        </saml:AuthenticationStatement>
      </saml:Assertion>
    </wsse:Security>
  </S12:Header>
  <S12:Body>
   ...
  </S12:Body>
</S12:Envelope>
```

(5) *WSS Rights Expression Language (REL) Token Profile* [160]. Varnostni žetoni predstavljajo dovoljenja v jeziku specifikacije ISO/IEC 21000-5 Rights Expressions [161]. Dovoljenja so podana preko elementa `r:license`. Primer prikazuje sporočilo SOAP z dovoljenjem REL:

(6) `urn:oasis:names:tc:SAML:1.0:assertion`.

```
<S12:Envelope xmlns:S12="...">
  <S12:Header>
    <wsse:Security xmlns:wsse="...">
      <r:license xmlns:r="urn:mpeg:mpeg21:2003:01-REL-R-NS">
        ...
      </r:license>
     ...
    </wsse:Security>
  </S12:Header>
  <S12:Body>
   ...
  </S12:Body>
</S12:Envelope>
```

Predstavljeni varnostni žetoni so lahko vstavljeni direktno v glavo sporočila SOAP ali pa so naslovljeni znotraj ločenih lokacij. V ta namen obstaja element `SecurityTokenReference` [147], ki vsebuje lokacijo URI, kjer se žeton nahaja.

5.2.2.4 Varna seja sporočil SOAP

Specifikacija WSS se osredotoča na enosmermo (angl. one-way) varno izmenjavo sporočil med končnimi točkami (t. i. storitve) po principu zahteva/odgovor (angl. request/response). V primeru, da storitve izmenjajo več sporočil, je potrebno uporabiti drugačen mehanizem.

Specifikacija *WS-SecureConversation* [162] določa varno komunikacijo (t. i. sejo) preko večih sporočil SOAP med končnimi točkami. Varna seja je predstavljena z *deljenim* (angl. shared) *varnostnim kontekstom*, ki vsebuje šifrirne ključe in druge varnostne informacije. Varnostni kontekst se deli v okviru seje med vsemi končnimi točkami, ki komunicirajo. Kontekst je določen preko elementa `wsc:SecurityContextToken` (SCT). Vsak kontekst ima enolično oznako (atribut `wsu:Id`) in ime (element `wsc:Identifier`), znotraj katerega so našteti eden ali več ključev. Sklicevanje na kontekst je preko elementa `wsse:SecureTokenReference` (enako kot na varnostni žeton). Velja poudariti, da WS-SecureConversation deluje v povezavi s specifikacijo WS-Trust [163], ki določa mehanizme za izdajanje različnih varnostnih žetonov (razdelek 5.2.2.3). Primer 5.4 prikazuje vzpostavljen varnostni kontekst med dvema končnima točkama, ki poznata sejni ključ za šifriranje telesa sporočila.

Obstajajo trije načini vzpostavitve deljenega varnostnega konteksta SCT med končnimi točkami, ki komunicirajo [162, 164]:

a) *Kontekst je dodeljen preko servisa žetonov*: iniciator pošlje zahtevo po žetonu preko elementa `wst:RequestSecurityToken` (`wst` predstavlja imenski prostor za specifikacijo WS-Trust), servis mu vrne odgovor z elementom `wst:RequestSecurityTokenResponse`, ki vsebuje varnostni kontekst (žeton) `wst:RequestedSecurityToken` in deljeni šifrirani simetrični ključ `wst:RequestedProofToken`.

```
<S11:Envelope xmlns:S11="..." xmlns:ds="..." xmlns:wsse="..."  xmlns:wsc="...">
  <S11:Header>
      ...
    <wsse:Security>
      <wsc:SecurityContextToken wsu:Id="MyID">
        <wsc:Identifier>uuid:652d2aaa-4857-4d8c-865c-f9549e5806f0</wsc:Identifier>
      </wsc:SecurityContextToken>
      <ds:Signature>
        ...
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#MyID"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S11:Header>
  <S11:Body wsu:Id="MsgBody">
    <tru:StockSymbol xmlns:tru="http://fabrikam123.com/payloads">
      QQQ
    </tru:StockSymbol>
  </S11:Body>
</S11:Envelope>
```

Primer 5.4: Deljeni varnostni kontekst.

b) *Kontekst kreira eden od udeležencev komunikacije*: iniciator kreira varnostni žeton in ga pošlje drugemu udeležencu preko sporočila SOAP z elementom `wst:RequestSecurityTokenResponse`. Ta kot v prejšnjem primeru vsebuje elementa `wst:RequestedSecurityToken` in `wst:RequestedProofToken`.

c) *Kontekst se kreira preko pogajanja med udeležencema*: iniciator pošlje sporočilo z elementom `wst:RequestSecurityToken` drugemu udeležencu, le-ta mu vrne odgovor z `wst:RequestSecurityTokenResponse`, ki kot v prejšnjih primerih vsebuje elementa `wst:RequestedSecurityToken` in `wst:RequestedProofToken`.

Varnostni kontekst vedno vsebuje deljeni simetrični ključ, ki se lahko uporabi za šifiriranje in digitalni podpis sporočil. Specifikacija zaradi večje varnosti priporoča uporabo izpeljanega ključa (element `wst:ComputedKey`), ki je unikaten glede na kontekst. Algoritem za izpeljavo ključev temelji na specifikaciji TLS [165].

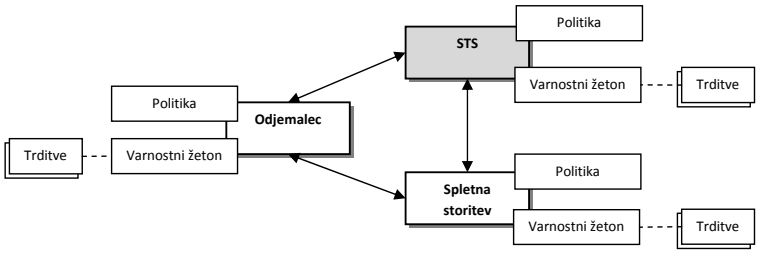
5.2.3 WS-Trust

Prejšnji razdelek 5.2.2 je okvirno predstavil t. i. *trde* varnostne mehanizme sklopa WSS. Kljub vsebovanosti besede *trust (zaupanje)* v imenu specifikacije WS-Trust [163], velja poudariti, da specifikacija določa varnostni mehanizem in ne sistem za obvladovanje zaupanja/ugleda spletnih storitev.

WS-Trust razširja mehanizme WSS z metodami za zahtevo, izdajanje in posredovanje varnostnih žetonov. Definirane so ustrezne operacije za *pridobivanje*, *izdajanje*, *obnovitev* in *validacijo* žetonov. Cilj WS-Trust je omogočiti izmenjavo zaupanja vrednih sporočil SOAP med končnimi točkami. Zaupanje je predstavljeno v smislu izmenjave in posredovanja varnostnih žetonov. Zaupanje torej ni prisotno kot mehki varnostni mehanizem. Naš cilj v nadaljevanju je razširitev WS-Trust z našim modelom zaupanja ATM.

Varnostni model WS-Trust temelji na procesu, kjer lahko spletna storitev zahteva, da vhodno sporočilo SOAP dokazuje (upravičuje) množico trditev (angl. claims) (npr. ključ, ime, pravica, ...) preko varnostnih žetonov. Če vhodno sporočilo ne upravičuje zahtevanih trditev, ga spletna storitev zavrne. Spletna storitev lahko zahteva trditve in ostale informacije preko svoje *politike*, kot jo določa specifikacija *WS-Policy* [166] in *WS-PolicyAttachment* [167]. Overjanje zahteve temelji na kombinaciji varnostnih in informacijskih trditev, ki jih upravičuje sporočilo. V primeru, da odjemalec nima potrebnih varnostnih žetonov, s katerimi bi lahko upravičil zahtevane trditve ciljne spletne storitve, lahko kontaktira *pooblaščene entitete* (angl. authorities) (kot so določene v politiki spletne storitve) in zahteva ustrezne varnostne žetone. Pooblaščena entiteta se označuje kot *storitev varnostnih žetonov* (angl. Security Token Service – STS) in izdaja varnostne žetone. STS je spletna storitev, ki lahko tudi zahteva svojo množico trditev za zahteve po varnostnih žetonih (preko politike). STS z izdajanjem varnostnih žetonov predstavlja osnovo za zaupanje v smislu trdega varnostnega mehanizma.

Slika 5.3 prikazuje varnostni model WS-Trust, ki spominja na modificiran trikotnik SOA s spletno storitvijo zaupanja TWS (slika 5.2). Model prikazuje, da je lahko tudi odjemalec spletna storitev. Puščice določajo možne komunikacijske poti; odjemalec lahko pridobi varnostni žeton preko STS ali pa na drug način. Ciljna spletna storitev ima svojo politiko z zahtevami; ko prejme sporočilo z varnostnimi žetoni, preveri ali žetoni (kot tudi celotno sporočilo SOAP) ustrezajo predpisani politiki (validacijo žetonov lahko izvede sama storitev ali pa jih pošlje na STS). Če je validacija uspešna, ciljna storitev sprejme in obdela sporočilo. Poudariti velja, da mora odjemalec kot tudi ciljna spletna storitev implicitno zaupati STS za izdajanje in validacijo varnostnih žetonov. STS lahko primerjamo s predlagano spletno storitvijo zaupanja TWS (slika 5.2).



Slika 5.3: Varnostni model zaupanja WS-Trust preko STS.

Model WS-Trust s trditvami, politikami in varnostnimi žetoni lahko podpira in vključuje bolj specifične varnostne modele. V kombinaciji z elementi WS-Security in WS-Policy omogoča

protokole za pogajanja in izmenjavo ključev, overjanja, digitalnih podpisov, ACL (Access Control List) na podlagi politik, delegiranje in posredovanje žetonov in posledično vzpostavitev kompleksnih relacij zaupanja (v smislu varnostnih mehanizmov). Pri teh relacijah omenimo, da WS-Trust loči dva tipa zaupanja [102]:

- *neposredno* (angl. direct) zaupanje: nastopa v primeru, da si dve entiteti izmenjata varnostne žetone brez vmesnih entitet (posrednikov);
- *posredovano* (angl. brokered) zaupanje: dve entiteti si izmenjata varnostne žetone preko posredovanja drugih neodvisnih entitet.

Z vključitvijo transportnih varnostnih mehanizmov (IPSec in SSL/TLS), omogoča WS-Trust še dodatne varnostne scenarije. Naj omenimo še specifikacijo *WS-Federation* [168], ki razširja WS-Trust z mehanizmi za združevanje različnih varnostnih domen preko posredovanja identitet, atributov in overjanja med sodeluječimi spletnimi storitvami. Poudarimo, da naš cilj ni študija teh protokolov, ampak uporaba in razširitev mehanizmov WS-Trust za predlagani model zaupanja ATM.

5.2.3.1 Operacije WS-Trust

Model WS-Trust temelji na operacijah nad različnimi vrstami varnostnih žetonov ob dostopu do ciljne spletne storitve. Te operacije so zahtevanje, izdajanje, obnavljanje in validacija varnostnih žetonov. Žetoni so lahko različnih vrst (tudi uporabniško definirani), identificirajo se preko oznak URI, ki so določeni v različnih profilih žetonov WSS [147] (glej razdelek 5.2.2.3). Operacije so definirane kot posebni elementi XML znotraj sporočila SOAP. Shema XML (XSD) [97] vseh elementov je podana v specifikaciji [169]. V našem primeru se bomo osredotočili zgolj na dve operaciji(7): (a) zahteva za nov žeton in (b) izdajanje žetonov.

(a) Zahteva za varnostni žeton

Zahteva je definirana preko elementa XML `wst:RequestSecurityToken` (RST), ki se nahaja v telesu sporočila SOAP. Osnovni atributi in pod-elementi so sledeči [163]:

- `/wst:RequestSecurityToken/@Context`: opcijski atribut v obliki URI, ki določa identifikator zahteve. Omogoča korelacijo med zahtevo in poznejšimi odgovori.
- `/wst:RequestSecurityToken/wst:TokenType`: določa tip varnostnega žetona v obliki URI (določen v profilih WSS).
- `/wst:RequestSecurityToken/wst:RequestType`: določa tip oz. funkcijo zahteve v obliki URI(8), ki se ujema z vrednostjo akcije URI (angl. Action URI) specifikacije *WS-Addressing* [112].
- `/wst:RequestSecurityToken/wst:SecondaryParameters`: določa opcijske parametre za storitev STS ob procesiranju zahteve.

(7) Operacije WS-Trust uporablja tudi specifikacija WS-SecureConversation.
(8) Vrednost je enaka `http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue`.

- `/wst:RequestSecurityToken/{any}`: razširitveni mehanizem, ki omogoča dodatne elemente in atribute, ki jih lahko uporabi STS ob procesiranju zahteve.

Obstajajo še dodatni opcijski atributi in pod-elementi, ki nosijo dodatne informacije in mehanizme o zahtevi RST (npr. časovna veljavnost RST, entropija ključev, pogajanje in izmenjava, delegiranje in posredovanje, vključevanje politik) [163, 169]. Primer 5.5 prikazuje zahtevo RST za digitalno potrdilo X.509.

```
<S12:Envelope xmlns:S12="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wst="...">
 <S12:Header>
  <wsse:Security>
   ....
  </wsse:Security>
 </S12:Header>
 <S12:Body>
  <wst:RequestSecurityToken Context="http://example.org/ctx">
   <wst:TokenType>
     http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3
   </wst:TokenType>
   <wst:RequestType>
     http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
   </wst:RequestType>
  </wst:RequestSecurityToken>
 </S12:Body>
</S12:Envelope>
```

Primer 5.5: Zahteva RST za certifikat X.509.

(b) Izdajanje varnostnega žetona

Odgovor na zahtevo RST se vrne preko `wst:RequestSecurityTokenResponse` (RSTR) elementa v telesu sporočila SOAP (lahko se nahaja tudi v glavi znotraj `wsse:Security`, če se žeton uporablja za zaščito sporočila); odgovor vsebuje zahtevani varnostni žeton. Osnovni atributi in pod-elementi so sledeči [163]:

- `/wst:RequestSecurityTokenResponse/@Context`: opcijski atribut v obliki URI, ki določa identiteto zahteve RST (če ima RST ta atribut, ga mora imeti tudi RSTR – enaka vrednost).
- `/wst:RequestSecurityTokenResponse/wst:TokenType`: določa tip izdanega varnostnega žetona.
- `/wst:RequestSecurityTokenResponse/wst:RequestedSecurityToken`: vsebuje vsebino zahtevanega žetona. Lahko je podana tudi referenca[9] v primeru, da se žeton nahaja v glavi SOAP (se rabi za zaščito sporočila).
- `/wst:RequestSecurityTokenResponse/{any}`: razširitveni mehanizem, ki omogoča dodatne elemente in atribute v odgovoru RSTR.

[9] `wsse:SecurityTokenReference`.

Podobno kot zahteva RST, ima tudi odgovor RSTR dodatne opcijske atribute in elemente (več v [163, 169]). Primer 5.6 prikazuje odgovor RSTR s potrdilom X.509.

```
<S12:Envelope xmlns:S12="..." xmlns:wsse="..." xmlns:wsu="..." xmlns:wst="...">
 <S12:Header>
  <wsse:Security>
   ....
  </wsse:Security>
 </S12:Header>
 <S12:Body>
  <wst:RequestSecurityTokenResponse Context="http://example.org/ctx">
   <wst:RequestedSecurityToken>
    <wsse:BinarySecurityToken ValueType="..#X509v3" EncodingType="..#Base64Binary">
      KkFPle ...
    </wsse:BinarySecurityToken>
   </wst:RequestedSecurityToken>
  </wst:RequestSecurityTokenResponse>
 </S12:Body>
</S12:Envelope>
```

Primer 5.6: Odgovor RSTR s certifikatom X.509.

5.2.4 Žetoni zaupanja

V razdelku 5.2.3 smo prikazali osnovne mehanizme WS-Trust preko operacij (RST, RSTR) z varnostnimi žetoni. Ti mehanizmi z ustreznimi razširitvami so izhodišče za naš primer integracije modela ATM v ogrodje WS-Trust. Poleg tega omogočajo tudi implicitno zaupanje med entitetami znotraj trikotnika SOA. Že iz definicije razširjenega trikotnika SOA (slika 5.2) ter primerjavo z varnostnim modelom WS-Trust (slika 5.3), lahko ugotovimo podobno funkcionalnost med entiteto oz. storitvijo TWS (storitev zaupanja) in STS – obe dobivata zahteve in vračata ustrezne informacije. V primeru STS so te informacije varnostni žetoni, katere lahko posplošimo tudi za primer TWS.

Uvedemo t. i. *žetone zaupanja* (angl. trust tokens), ki so podobni varnostnim žetonom. So elementi XML, ki nosijo informacije v domeni modela zaupanja ATM (npr. izvorna, ciljna entiteta, stopnja zaupanja). Kot smo že omenili, storitev TWS podpira dve operaciji (glej razdelek 5.2.1): *Query* (zahteva za izračun stopnje zaupanja $\tau_{i,j}(\varphi)$ glede na algoritem Γ) in *Insert* (zahteva za dodajanje nove informacije o končani interakciji $x_{i,j}(t, \varphi)$, ki vsebuje tudi oceno zaupanja (glej poglavje 3, definicija 3.1.2.9).

Obe operaciji lahko izvedemo preko operacij WS-Trust – RST (zahteva za žeton zaupanja) in RSTR (odgovor z žetonom). Ločimo torej dva primera:

(a) Operacija *Query*: definiramo zahtevo `TrustRequest` za RST in žeton zaupanja `TrustToken`, ki ga vrne RSTR.

(b) Operacija *Insert*: definiramo zahtevo `InsertRequest` za RST in žeton zaupanja `InsertToken` (t. i. statusni žeton).

Obe zahtevi `TrustRequest` in `InsertRequest` sta elementa XML, ki ju dodamo kot dodatne parametre v RST znotraj elementa `wst:SecondaryParameters`. Strukturo zahtev in žetonov lahko definiramo preko sheme XML (XSD) [97]. Pred nadaljevanjem je potrebno definirati ustrezne oznake URI za posamezen tip žetona zaupanja; to vrednost uporablja RST znotraj elementa `wst:TokenType`.

Definicija 5.2.4.1 (Tip žetona zaupanja)**.** Tip žetona zaupanja se identificira preko oznake URI in sicer:

- `TrustToken`: `http://tempuri.org/STS/TrustToken`,
- `InsertToken`: `http://tempuri.org/STS/InsertToken`.

V nadaljevanju sledi definicija posameznih zahtev in žetonov zaupanja preko shem XML.

5.2.4.1 TrustRequest

Zahteva `TrustRequest` mora vsebovati informacijo o: (a) izvorni in ciljni entiteti (unikatni identifikator preko URI, URL, e-naslov), (b) model zaupanja (npr. kvantitativni, kvalitativni), (c) operator zaupanja za izbrani model in (d) informacije o kontekstu (enolično ime). Ciljna entiteta `TargetEntity` ima opcijski atribut `IsBPEL`; v primeru, da ima vrednost `true` (če je izpuščen, ima vrednost `false`), entiteta predstavlja poslovni proces BPEL (podan je preko URI-ja oz. URL-ja):

```
...
<TargetEntity IsBPEL="true">
   http://tempuri.org/BPEL/PurchaseOrderProcess.xml
</TargetEntity>
...
```

V tem primeru sistem za OZ/OU (t. i. "trust engine") prebere definicijo procesa BPEL s podanega naslova URI in preko algoritma 4.4 (poglavje 4, razdelek 4.2.7) izračuna stopnjo zaupanja kompozita storitev.

Definicija 5.2.4.2 (Zahteva `TrustRequest`)**.** Struktura `TrustRequest` je določena preko sledeče sheme XML:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://tempuri.org/STS/TrustRequest.xsd"
           xmlns="http://tempuri.org/STS/TrustRequest.xsd">
  <xs:element name="TrustRequest">
    <xs:complexType>
```

```
      <xs:sequence>
        <xs:element name="SourceEntity" type="xs:string" minOccurs="0" />
        <xs:element name="TargetEntity">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:string">
                <xs:attribute name="IsBPEL" type="xs:boolean" use="optional" default="false" />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
        <xs:element name="TrustModel">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="QUANTITATIVE" />
              <xs:enumeration value="QUALITATIVE" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="Operator">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <!-- kvantitativni model -->
              <xs:enumeration value="DISJUNCTION" />
              <xs:enumeration value="CONJUNCTION" />
              <xs:enumeration value="CONSENSUS" />
              <xs:enumeration value="RECOMMENDATION" />
              <!-- kvalitativni model -->
              <xs:enumeration value="OPTIMISTIC" />
              <xs:enumeration value="PESSIMISTIC" />
              <xs:enumeration value="BALANCED" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="Context" type="xs:string" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Primer 5.7 prikazuje zahtevo RST (znotraj telesa SOAP) z vsebovanim elementom `tr:TrustRequest`, ker je `tr` ustrezen imenski prostor[10].

[10] V našem primeru `http://tempuri.org/STS/TrustRequest.xsd`.

```
<wst:RequestSecurityToken xmlns:wst="...">
  <wst:TokenType>
    http://tempuri.org/STS/TrustToken
  </wst:TokenType>
  <wst:RequestType>
    http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
  </wst:RequestType>
  <wst:SecondaryParameters>
    <tr:TrustRequest xmlns:tr="http://tempuri.org/STS/TrustToken.xsd">
      <tr:SourceEntity>source@example1.com</tr:SourceEntity>
      <tr:TargetEntity>target@example2.com</tr:TargetEntity>
      <tr:TrustModel>QUALITATIVE</tr:TrustModel>
      <tr:Operator>OPTIMISTIC</tr:Operator>
      <tr:Context>eBay nakup knjige</tr:Context>
    </tr:TrustRequest>
  </wst:SecondaryParameters>
</wst:RequestSecurityToken>
```

Primer 5.7: Zahteva RST za žeton zaupanja.

5.2.4.2 TrustToken

Ob prejemu zahteve RST storitev, STS kliče sistem za OZ/OU (t. i. "trust engine"), ki izračuna stopnjo zaupanja med izvorno in ciljno entiteto. V primeru, da element `tr:SourceEntity` v RST ni podan, se vrne stopnja ugleda. Tudi v primeru, da element `tr:Context` manjka, se vzame t. i. privzeti kontekst oz. se izračuna celotna stopnja zaupanja ($totalTrust$) glede na algoritem Γ.

Definicija 5.2.4.3 (Žeton zaupanja `TrustToken`)**.** Struktura žetona `TrustToken` je določena preko sledeče sheme XML:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://tempuri.org/STS/TrustToken.xsd"
           xmlns="http://tempuri.org/STS/TrustToken.xsd">
  <xs:element name="TrustToken">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="SourceEntity" type="xs:string" minOccurs="0"/>
        <xs:element name="TargetEntity" type="xs:string" />
        <xs:element name="Context" type="xs:string" minOccurs="0" />
        <xs:element name="TimeStamp" type="xs:dateTime" />
        <!-- vrne stopnjo zaupanja samo za en model -->
        <xs:choice>
          <xs:element name="QuantitativeValue" minOccurs="0">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Belief" type="xs:decimal" />
```

```
                <xs:element name="Disbelief" type="xs:decimal" />
                <xs:element name="Uncertainty" type="xs:decimal" />
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="QualitativeValue" minOccurs="0">
            <xs:simpleType>
              <xs:restriction base="xs:string">
                <xs:pattern value="P|PD|U|PT|T|-inf" />
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Primer 5.8 prikazuje odgovor RSTR (znotraj telesa SOAP) z vsebovanim žetonom zaupanja `tt:TrustToken`.

```
<wst:RequestSecurityTokenResponse xmlns:wst="...">
  <wst:TokenType>
    http://tempuri.org/STS/TrustToken
  </wst:TokenType>
  <wst:RequestedSecurityToken>
    <tt:TrustToken xmlns:tt="http://tempuri.org/STS/TrustToken.xsd">
      <tt:SourceEntity>source@example1.com</tt:SourceEntity>
      <tt:TargetEntity>target@example2.com</tt:TargetEntity>
      <tt:Context>eBay nakup knjige</tt:Context>
      <tt:TimeStamp>2007-03-07T00:00:00.0000000+01:00</tt:TimeStamp>
      <tt:QualitativeValue>PT</tt:QualitativeValue>
    </tt:TrustToken>
  </wst:RequestedSecurityToken>
</wst:RequestSecurityTokenResponse>
```

Primer 5.8: Odgovor RSTR z vsebovanim žetonom zaupanja.

5.2.4.3 InsertRequest

Dodajanje novih ocen zaupanja po končanih interakcijah se izvede preko zahteve `InsertRequest` znotraj RST. Zahteva mora vsebovati informacije o (a) izvorni in ciljni entiteti (so lahko tudi abstraktne), (b) informacija o kontekstu, (c) informacija o interakciji $x_{i,j}(t, \varphi)$ (vsebuje konfiguracijo (glej poglavje 3, definicija 3.1.2.8) s kvantitativno in/ali kvalitativno oceno zaupanja). Velja izpostaviti tip ocene zaupanja – lahko je funkcionalno zaupanje $\omega_{i,j}(t, \varphi)$ (privzeto) ali pa zaupanje v priporočilo $\omega^r_{i,j}(t, \varphi)$.

Definicija 5.2.4.4 (Zahteva `InsertRequest`)**.** Struktura zahteve `InsertRequest` je določena preko sledeče sheme XML:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://tempuri.org/STS/InsertRequest.xsd"
           xmlns="http://tempuri.org/STS/InsertRequest.xsd">
  <!-- podatki o entiteti -->
  <xs:complexType name="EntityType">
    <xs:sequence>
      <xs:element name="Identification" type="xs:string" />
      <xs:element name="IsAbstract" type="xs:boolean" />
      <xs:element name="Description" type="xs:string" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>

  <!-- kvantitativna domena -->
  <xs:complexType name="QuantitativeDomain">
    <xs:sequence>
      <xs:element name="Belief" type="xs:decimal" />
      <xs:element name="Disbelief" type="xs:decimal" />
      <xs:element name="Uncertainty" type="xs:decimal" />
    </xs:sequence>
  </xs:complexType>
  <!-- kvalitativna domena -->
  <xs:simpleType name="QualitativeDomain">
    <xs:restriction base="xs:string">
      <xs:pattern value="P|PD|U|PT|T|-inf" />
    </xs:restriction>
  </xs:simpleType>

  <!-- ocena zaupanja-->
  <xs:complexType name="TrustAssessmentType">
    <xs:sequence>
     <xs:element name="QuantitativeValue" type="QuantitativeDomain" minOccurs="0"/>
     <xs:element name="QualitativeValue" type="QualitativeDomain" minOccurs="0"/>
    </xs:sequence>
    <!-- atribut za tip ocene zaupanja -->
    <xs:attribute name="trustType" default="functional" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:pattern value="functional|recommendation|both" />
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>

  <xs:complexType name="InsertRequestType">
```

```
    <xs:sequence>
      <xs:element name="SourceEntity" type="EntityType" />
      <xs:element name="TargetEntity" type="EntityType" />
      <xs:element name="Context" type="xs:string" />
      <xs:element name="Interaction">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Configuration" type="xs:string" minOccurs="0" />
            <xs:element name="TrustAssessment" type="TrustAssessmentType" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <!-- korenski element -->
  <xs:element name="InsertRequest" type="InsertRequestType" />
</xs:schema>
```

Primer 5.9 prikazuje zahtevo RST (znotraj telesa SOAP) z vsebovanim elementom `ir:InsertRequest`, kjer je `ir` ustrezen imenski prostor. Izvorna entiteta poda kvantitativno in kvalitativno oceno zaupanja.

5.2.4.4 InsertToken

Ob prejemu zahteve RST z elementom `ir:InsertRequest`, storitev STS kliče sistem OZ/UZ, ki shrani podano oceno zaupanja na obstojni podatkovni vir (npr. relacijska podatkovna baza). Sistem vrne rezultat v obliki žetona `InsertToken`, ki nosi informacijo o tem, ali se je operacija (ne)uspešno izvedla.

Definicija 5.2.4.5 (Žeton `InsertToken`)**.** Struktura žetona `InsertToken` je določena preko sledeče sheme XML:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://tempuri.org/STS/InsertToken.xsd"
           xmlns="http://tempuri.org/STS/InsertToken.xsd">
  <xs:element name="InsertToken">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="UID" type="xs:string" />
        <xs:element name="TimeStamp" type="xs:dateTime" />
        <xs:element name="StatusCode" type="xs:byte" />
        <xs:element name="Description" type="xs:string" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<wst:RequestSecurityToken xmlns:wst="...">
  <wst:TokenType>
    http://tempuri.org/STS/InsertToken
  </wst:TokenType>
  <wst:RequestType>
    http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
  </wst:RequestType>
  <wst:SecondaryParameters>
    <!-- zahteva InsertRequest -->
    <ir:InsertRequest xmlns:ir="http://tempuri.org/STS/InsertTest.xsd">
      <ir:SourceEntity>
        <ir:Identification>entity1@emaple.org</ir:Identification>
        <ir:IsAbstract>False</ir:IsAbstract>
        <ir:Description>Customer..</ir:Description>
      </ir:SourceEntity>
      <ir:TargetEntity>
        <ir:Identification>entity2emaple.org</ir:Identification>
        <ir:IsAbstract>False</ir:IsAbstract>
        <ir:Description>Service provider..</ir:Description>
      </ir:TargetEntity>
      <ir:Context>eBay nakup knjige</ir:Context>
      <ir:Interaction>
        <ir:Configuration>
          event1;event2;event3
        </ir:Configuration>
        <ir:TrustAssessment trustType="functional">
          <ir:QuantitativeValue>
            <ir:Belief>0.6</ir:Belief>
            <ir:Disbelief>0.3</ir:Disbelief>
            <ir:Uncertainty>0.1</ir:Uncertainty>
          </ir:QuantitativeValue>
          <ir:QualitativeValue>PT</ir:QualitativeValue>
        </ir:TrustAssessment>
      </ir:Interaction>
    </ir:InsertRequest>
  </wst:SecondaryParameters>
</wst:RequestSecurityToken>
```

Primer 5.9: Zahteva RST za vpis ocene zaupanja.

Primer 5.10 prikazuje odgovor RSTR (znotraj telesa SOAP) z vsebovanim žetonom `it:InsertToken`, ki določa status vpisane ocene zaupanja (element `UID` je unikatni ključ[11] zapisa v podatkovnem viru).

5.2.5 Arhitektura

V prejšnjem razdelku smo definirali ustrezno strukturo sporočil (žetone zaupanja) znotraj ogrodja WS-Trust, ki posredno preko storitve STS omogočajo izvedbo dveh osnovnih operacij sistema za OZ/OU – *Query* in *Insert*. Omenjeni operaciji sta lahko definirani oz. implementirani na dva načina:

[11] V obliki GUID (Global Unique Identifier).

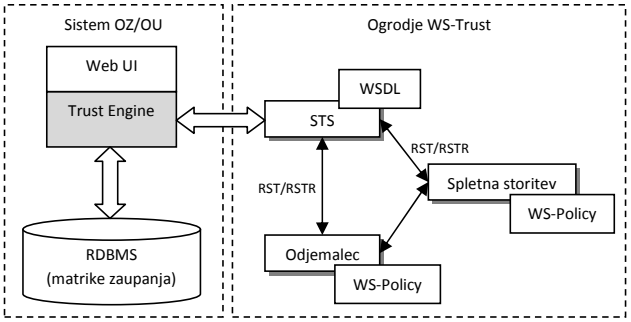
```
<wst:RequestSecurityTokenResponse xmlns:wst="...">
  <wst:TokenType>
    http://tempuri.org/STS/TrustToken
  </wst:TokenType>
  <wst:RequestedSecurityToken>
    <it:InsertToken xmlns:it="http://tempuri.org/STS/InsertToken.xsd">
      <it:UID>ff093030-b4b7-11dd-ad8b-0800200c9a66</it:UID>
      <it:TimeStamp>2008-03-09T00:00:00.0000000+01:00</it:TimeStamp>
      <it:StatusCode>1</it:StatusCode>
      <it:Description>Success</it:Description>
    </it:InsertToken>
  </wst:RequestedSecurityToken>
</wst:RequestSecurityTokenResponse>
```

Primer 5.10: Odgovor RSTR s statusnim žetonom.

(a) STS ima tudi vlogo TWS (glej razdelek 5.2.1), ki poleg varnostnih žetonov lahko preko zahtev RST izdaja tudi žetone zaupanja (`TrustToken`, `InsertToken`) z neposredno komunikacijo s sistemom za OZ/OU (STS torej predstavlja vmesnik do sistema);

(b) STS ob prejemu zahteve RST samo posredujo zahtevo storitvi TWS, le-ta pa predstavlja vmesnik do sistema za OZ/OU.

V našem primeru uporabimo možnost (a) – storitev STS predstavlja vmesnik do sistema za OZ/OU. Obstaja tudi možnost razširitve STS oz. njene definicije WSDL [170] z operacijama `Query(TrustRequest)` in `Insert(InsertRequest)`, vendar ta pristop odstopa od standarda WS-Trust. Slika 5.4 prikazuje vpeljavo sistema za OZ/OU v ogrodje WS-Trust. Kot opcija obstaja spletni uporabniški vmesnik (angl. web user interface – Web UI), ki omogoča interaktivno komunikacijo s sistemom.



Slika 5.4: Integracija sistema za OZ/OU v ogrodje WS-Trust.

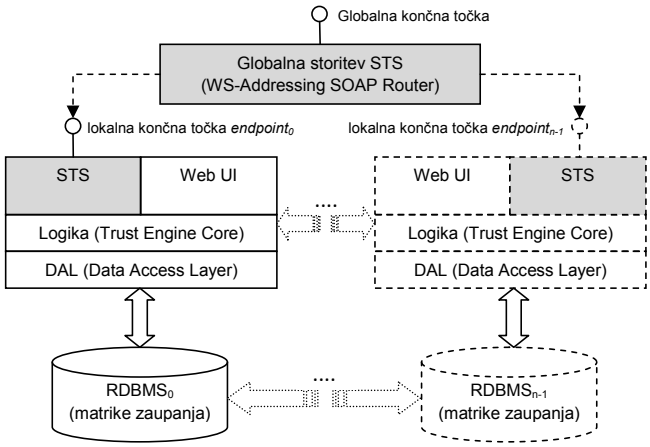
5.2.5.1 Implementacija

Predlagani arhitekturni koncept sistema za OZ/OU lahko implementiramo na različne načine. Pomembno je, da je sistem za OZ/OU z ustreznimi operacijami (algoritmi) in komunikacijskimi

protokoli učinkovit in skalabilen. Kot smo že omenili v poglavju 2 (razdelek 2.4.2), obstajata dva pristopa implementacije [9, 20]: centraliziran in porazdeljen. Centraliziran pristop je najbolj običajen na splošno komercialno uveljavljenih sistemih za obvladovanje ugleda (npr. eBay, Amazon), kjer obstaja centralna podatkovna baza za shranjevanje ocen. Problem tega pristopa je, da ni skalabilen in lahko odpove, ko število transakcij s sistemom preseže določeno mejo.

Porazdeljen pristop [22] je bolj kompleksen in se večinoma uporablja v omrežjih P2P [20]. Je bolj skalabilen in je precej neobčutljiv na napake ob velikem številu transakcij. Deluje na principu porazdeljenih podatkovnih baz (npr. vsako vozlišče oz. entiteta lahko hrani del ocen), zato je zelo pomemben učinkovit in hiter komunikacijski protokol za porazdeljene poizvedbe (operacija *Query*) in ažuriranje ocen (operacija *Insert*).

Naš glavni cilj ni iskanje učinkovitega arhitekturnega pristopa pri implementaciji sistema za OZ/OU, zato bomo podali samo okvirno konceptualno sliko sistema. Pravzaprav smo že razvili centraliziran prototip sistema *trustGuard* (glej dodatek E), ki je trenutno zgrajen na tehnologiji Microsoft WCF(12) [171], izboljšana verzija pa v JEE(13) [172]. Obvladovanje matrik interakcij oz. zaupanja $\mathbb{M}(t, \varphi)$, je modelirano preko relacijske podatkovne baze. Centraliziran pristop lahko razširimo na t. i. *delno-porazdeljen* (angl. semi-distrubuted) pristop z n sistemi za OZ/OU, ki tečejo paralelno. Vsak sistem uporablja lokalno podatkovno bazo, ki vsebuje samo izbrano ali replicirano množico vseh ocen (matrike zaupanja). Posamezen sistem je dosegljiv preko *lokalne končne točke* (naslovljiva entiteta ali vir, kamor se lahko pošiljajo sporočila). *Globalna spletna storitev STS* ima funkcijo usmerjanja (angl. routing) zahtev na ustrezne lokalne točke; usmerjanje zahtev SOAP je izvedeno preko specifikacije *WS-Addressing* [112]. Slika 5.5 prikazuje arhitekturni koncept sistema za OZ/OU.



Slika 5.5: Arhitekturni koncept sistema za OZ/OU.

WS-Addressing določa transportno neodvisne mehanizme za naslavljanje virov (končnih točk); določa pošiljatelje in sprejemnike sporočil. Vse informacije so shranjene v sporočilu samem

(12) Windows Communication Foundation.
(13) Java Enterprise Edition.

in so neodvisne od transportnega protokola. Obstajata dva mehanizma za naslavljanje: *reference končnih točk* (angl. endpoint references) in *informacijske glave* (angl. message information headers).

Algoritem za usmerjanje sporočil SOAP je lahko ustrezen *izenačevalnik obremenitev* (angl. load balancer). Ena od možnosti za usmerjanje, je uporaba *zgostitvenih funkcij* (angl. hash functions), ki preslika vsako zahtevo SOAP v ustrezno celo število $m \in \mathbb{N}$. Naj bo $g : SOAP_{Query,Insert} \mapsto \mathbb{N}$ ustrezna zgostitvena funkcija, ki preslika zahteve `TrustRequest` (operacija *Query*) in `InsertRequest` (operacija *Insert*) v neko naravno število m. Glede na to, da je vsaka entiteta določena z enolično oznako, se lahko ta vrednost uporabi kot argument zgostitvene funkcije in sicer:

(1) `TrustRequest`: argument je vrednost elementa `TargetEntity` (glej razdelek 5.2.4.1);
(2) `InsertRequest`: argument je vrednost elementa `Identification` znotraj elementa `TargetEntity` (glej razdelek 5.2.4.3).

Ustrezna lokalna končna točka $endpoint_m$, $m \in [0, n-1]$ (n je število operativnih lokalnih končnih točk) se izračuna preko sledečega izraza (5.2):

$$m = \begin{cases} g(\texttt{TargetEntity}.Value) \bmod n, & \text{operacija } Query \\ g(\texttt{TargetEntity.Identification}.Value) \bmod n, & \text{operacija } Insert \end{cases} \tag{5.2}$$

5.2.5.2 Scenarij interakcij

Znotraj tega razdelka podamo možen scenarij interakcij med posameznimi entitetami v predlaganem modelu znotraj ogrodja WS-Trust (slika 5.4). Vsaka entiteta (odjemalec, ponudnik) ima lahko svojo interakcijsko politiko $\pi \in \Pi$ modela ATM (glej poglavje 3, razdelek 3.1.6), da z njo določi pravila za dostop do svojih storitev (virov) oz. pravila za interakcije z drugimi entitetami. V svetu SOA obstajajo različni načini predstavitve zahtev oz. politik, v katere lahko preslikamo množico interakcijskih politik Π, ki so podane v formalnem jeziku PPLTL.

Predpostavimo, da odjemalec a najde ustrezno funkcionalno storitev b s končno točko (oz. ponudnika storitev) v registru storitev (bodisi preko UDDI ali kak drugačen repozitorij), s katero hoče stopiti v interakcijo oz. jo prožiti. Naj bo politika odjemalca a enaka $\pi^a \equiv \texttt{Trust}(\tau_1)$, storitve oz. ponudnika b pa $\pi^b \equiv \texttt{Trust}(\tau_2)$. Enostaven abstrakten protokol interakcije je potem sledeč:

(1) Glede na politiko se mora odjemalec a prepričati, ali je njegova stopnja zaupanja v ustreznem kontekstu do b večja ali enaka τ_0. Odjemalec generira zahtevo `TrustRequest` in pošlje zahtevo RST na storitev STS. Ta mu vrne rezultat kot preko RSTR žeton zaupanja `TrustToken`.

(2) V primeru, da je stopnja zaupanja ustrezna, se a poveže na končno točko storitve ponudnika b. Tudi b ima svojo politiko, ki določa minimalno stopnjo zaupanja b do

odjemalca a v ustreznem kontekstu. Tudi b kontaktira storitev STS in v primeru, da je stopnja zaupanja večja ali enaka τ_1, je dostop odjemalca a do storitve omogočen.

(3) Po končani interakciji odjemalec a in/ali ponudnik storitev b podata informacijo o interakciji z oceno izkušnje preko zahteve RST z elementom `InsertRequest`.

Omenjeni scenarij lahko posplošimo na različne interakcije (ne samo proženje spletnih storitev) med dvema entitetama oz. agentoma (npr. kupec in prodajalec na spletnem mestu eBay).

V primeru, da je interakcijska politika bolj kompleksna, opisani abstraktni protokol ni več ustrezen. Za uporabo interakcijskih politik znotraj množice Π je potrebno preslikati politike $\pi \in \Pi$ v jeziku PPLTL v katero izmed specifikacij za opis politik SOA, kar bomo prikazali v nadaljevanju.

5.2.5.3 Interakcijske politike

Znotraj tega razdelka okvirno predstavimo integracijo interakcijskih politik znotraj predlaganega sistema za OZ/OU. Pri tem uporabimo standardne specifikacije SOA (WS-*), saj je naš cilj razširitev teh standardov z mehanizmi zaupanja predlaganega modela ATM.

Obstajajo različne specifikacije za opis politik spletnih storitev; politika določa zmožnosti, zahteve in druge značilnosti entitet. Dostop oz. branje politik entitet oz. storitev je praviloma prvi korak, ki ga mora izvesti odjemalec storitev, da lahko dostopi do ciljne storitve. Branje politik je možno preko protokola *WS-MetadataExchange* [173]. Za spletne storitve obstajajo različne specifikacije za opis politik:

- Ogrodje *WS-Policy* [166] določa gramatiko v jeziku XML za izražanje zahtev in splošnih značilnosti entitet, ki nastopajo znotraj sistema spletnih storitev. WS-Policy določa politiko kot zbirko različnih alternativ (izbir), kjer je vsaka alternativa zbirka *trditev* politike (angl. policy assertion). Znotraj ogrodja spadajo še specifikacije WS-PolicyAssertions, WS-PolicyAttachment in WS-SecurityPolicy, ki določa nabor trditev za uporabo varnostnih mehanizmov (WSS, WS-Trust, WS-SecureConversation).

- Specifikacija *WSPL* (Web Services Policy Language) [174] je profil politik za OASIS XACML (eXtensible Access Control Markup Language) standard [175]. Politika je opisana kot zaporedje pravil v jeziku XML, kjer vsako pravilo vsebuje množico predikatov.

- Specifikacija *WSDL 2.0* [98] omogoča tudi vključevanje izrazov politik, vendar z manjšim obsegom možnosti kot pri WS-Policy ali WSPL.

Omenjene specifikacije so podobne, vendar trenutno je najbolj podprto ogrodje WS-Policy (je sprejet odprti standard), ki omogoča poljubne razširitve. Posledično je omenjena specifikacija izhodišče za integracijo interakcijskih politik množice Π modela ATM.

Znotraj WS-Policy je politika predstavljena kot zbirka *alternativ*; vsaka alternativa politike je zbirka *trditev*. Trditev politike določa zahtevo, zmožnost ali drugo lastnost obnašanja

entitete.[14] *Izraz* politike ima lahko *normalno* ali ekvivalentno *kompaktno* obliko. Zahteve politike lahko določajo varnostne zahteve (npr. shema overjanja, transportni protokol) ali pa druge značilnosti storitev (npr. QoS). Potemtakem lahko v zahteve preslikamo tudi izraze politike π, podane v jeziku PPLTL.

Politika WS-Policy je dokument XML, ki je sestavljen iz različnih elementov in atributov. Korenski element `wsp:Policy` določa ustrezne imenske prostore in določa izraz politike z možno strukturo (natančna shema WS-Policy dokumenta je podana preko sheme XML [176]), kjer znak "`?`" pomeni nič ali eno pojavitev, znak "`*`" pa nič ali več pojavitev elementa/atributa:

```
<wsp:Policy (Name="..")? (wsu:Id="..")?
            xmlns:wsp="http://www.w3.org/ns/ws-policy">
  <wsp:ExactlyOne>
    (<wsp:All>
      <Assertion1... ( wsp:Optional="xs:boolean" )? ...> ... </Assertion1>
      <Assertion2...> ... </Assertion2>
    </wsp:All>)*
  <wsp:ExactlyOne>
  (<wsp:PolicyReference URI="..." (Digest="...")? (DigestAlgorithm="...")? />)*
  ...
</wsp:Policy>
```

Pomen nekaterih elementov oz. atributov je sledeč (več podrobnosti v [166]):

- `/wsp:Policy/@Name` in `/wsp:Policy/@wsu:Id` sta opcijska atributa, ki določata ime oz. identifikator politike, katerih vrednosti se lahko uporabi pri sklicevanju iz drugih politik.

- Operatorji politike omogočajo združevanje posameznih trditev; poleg tega so operatorji lahko rekurzivno gnezdeni. Ti operatorji so:

  - `wsp:ExactlyOnce` določa, da se mora upoštevati natanko eden od podrejenih elementov; določa zbirko možnih alternativ politike;
  - `wsp:Policy` oz. `wsp:All`: določa, da se morajo upoštevati vsi podrejeni elementi; določa zbirko trditev alternative politike.

  Operatorji se lahko gnezdijo, prav tako veljajo različne lastnosti [166] – komutativnost, asociativnost, idempotentnost, distributivnost (podrobnosti v [166]).

- `/Assertion/@wsp:Optional` določa, ali je trditev opcijska; če je atribut izpuščen, je trditev obvezna. Obstaja pomenska ekvivalenca izrazov z opcijskimi trditvami [166]; npr. trditev
  `<Assertion wsp:Optional="true">...</Assertion>` je enaka kot:

[14] Entiteta je lahko končna točka, sporočilo, vir, operacija.

```
<wsp:ExactlyOne>
  <wsp:All> <Assertion..> ... </Assertion> </wsp:All>
  <wsp:All />
</wsp:ExactlyOne>
```

- `/wsp:PolicyReference/@URI` omogoča sklicevanje oz. vključitev obstoječe politike. Element oz. atribut se lahko nahaja povsod tam, kjer se lahko nahaja trditev politike.

Znotraj opisanih konstruktov se lahko nahajajo različne trditve, ki jih WS-Policy ne specificira. WS-SecurityPolicy [177] npr. določa množico elementov XML, ki so trditve za ustrezne varnostne zahteve specifikacije WS-Security, kot npr. vrsta overjanja (žetoni), algoritmi za digitalni podpis (celovitost), algoritmi za šifriranje (zaupnost), deli sporočila, ki so šifrirani in podobno. Za ilustracijo lahko podamo primer politike, ki določa dve alternativi: (a) šifriranje ali (b) digitalni podpis telesa SOAP (primer 5.11).

```
<wsp:Policy
   xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702"
   xmlns:wsp="http://www.w3.org/ns/ws-policy" >
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:EncryptedParts>
        <sp:Body/>
      </sp:EncryptedParts>
    </wsp:All>
    <wsp:All>
      <sp:SignedParts>
        <sp:Body/>
      </sp:SignedParts>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Primer 5.11: Varnostna politika za telo sporočila SOAP.

Na podoben način je potrebno definirati interakcijsko politiko zaupanja, ki smo jo formalno določili v poglavju 3, razdelek 3.1.6. Izhajamo iz sintakse PPLTL z minimalnim naborom $\mathcal{L}_{ES}^{min}$ (izraz 3.28) za opis politike $\pi \in \Pi$. Algoritem 3.5 (poglavje 3) za preverjanje skladnosti s politiko π predvideva obstoj struktur oz. razredov za opis abstraktnega jezika PPLTL in njegovih operatorjev. Preslikavo med sintakso in strukturami oz. elementi XML, prikazuje tabela 5.2.

Naslednji korak pri definiranju interakcijske politike zaupanja je izpeljava oz. opis ustrezne sheme XML (XSD) za elemente XML znotraj tabele 5.2.

Definicija 5.2.5.1 (Interakcijska politika zaupanja)**.** Interakcijska politika zaupanja $\pi \in \Pi$ je določena kot dokument XML s sledečo shemo XSD:

Izraz PPLTL	Element XML
e	<EventFormula>*event*</EventFormula>
$\Diamond e$	<PossibleFormula>*event*</PossibleFormula>
$\pi_1 \wedge \pi_2$	<Conjuction> <Arg1>$formula_1$</Arg1> <Arg1>$formula_2$</Arg1> <Conjunction>
$\neg\pi$	<Negation>*formula*</Negation>
$\mathsf{X}^{-1}\pi$	<LastTime>*formula*</LastTime>
$\pi_0 \mathsf{S} \pi_1$	<Since> <Arg1>$formula_1$</Arg1> <Arg2>$formula_2$</Arg2> </Since>
Trust(τ)	<TrustLevel>*value*</TrustLevel>

Tabela 5.2: Preslikava PPLTL v elemente XML.

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://tempuri.org/STS/TrustPolicy.xsd"
           xmlns="http://tempuri.org/STS/TrustPolicy.xsd">

  <!-- tip kvantitativne domene zaupanja -->
  <xs:complexType name="QuantitativeDomainType">
    <xs:sequence>
      <xs:element name="Belief" type="xs:decimal" />
      <xs:element name="Disbelief" type="xs:decimal" />
      <xs:element name="Uncertainty" type="xs:decimal" />
    </xs:sequence>
  </xs:complexType>

  <!-- tip kvalitativne domene zaupanja -->
  <xs:simpleType name="QualitativeDomainType">
    <xs:restriction base="xs:string">
      <xs:pattern value="P|PD|U|PT|T|-inf" />
    </xs:restriction>
  </xs:simpleType>

  <!-- tip za EventFormula -->
  <xs:simpleType name="EventFormulaType">
    <xs:restriction base="xs:string" />
  </xs:simpleType>

  <!-- tip za PossibleFormula -->
  <xs:simpleType name="PossibleFormulaType">
    <xs:restriction base="xs:string" />
  </xs:simpleType>
```

```
<!-- tip vrednosti stopnje zaupanja -->
<xs:complexType name="TrustLevelType">
  <xs:choice>
    <xs:element name="QuantitativeValue" type="QuantitativeDomainType" />
    <xs:element name="QualitativeValue" type="QualitativeDomainType" />
  </xs:choice>
</xs:complexType>

<!-- tip za element (operator) Conjunction -->
<xs:complexType name="ConjunctionType" mixed="false">
    <xs:sequence>
      <xs:element name="Arg1" type="FormulaContentType" />
      <xs:element name="Arg2" type="FormulaContentType" />
    </xs:sequence>
</xs:complexType>

<!-- tip za element (operator) Negation -->
<xs:complexType name="NegationType" mixed="false">
  <xs:complexContent>
    <xs:extension base="FormulaContentType" />
  </xs:complexContent>
</xs:complexType>

<!-- tip za element (operator) LastTime -->
<xs:complexType name="LastTimeType" mixed="false">
  <xs:complexContent>
    <xs:extension base="FormulaContentType" />
  </xs:complexContent>
</xs:complexType>

<!-- tip za element (operator) Since -->
<xs:complexType name="SinceType" mixed="false">
  <xs:sequence>
    <xs:element name="Arg1" type="FormulaContentType" />
    <xs:element name="Arg2" type="FormulaContentType" />
  </xs:sequence>
</xs:complexType>

<!-- možni tipi za znotraj elementa Formula -->
<xs:complexType name="FormulaContentType" mixed="false">
  <xs:sequence>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="Formula"  />
      <xs:element name="EventFormula" type="EventFormulaType" />
      <xs:element name="PossibleFormula" type="PossibleFormulaType" />
      <xs:element name="Conjunction" type="ConjunctionType"  />
      <xs:element name="Negation" type="NegationType" />
      <xs:element name="LastTime" type="LastTimeType" />
```

```
        <xs:element name="Since" type="SinceType" />
        <xs:element name="TrustLevel" type="TrustLevelType" />
      </xs:choice>
    </xs:sequence>
  </xs:complexType>

  <!-- korenski element Formula -->
    <xs:element name="Formula">
    <xs:complexType mixed="false">
      <xs:complexContent>
        <xs:extension base="FormulaContentType">
          <xs:attribute name="Context" type="xs:string" use="required" />
          <xs:attribute name="SourceEntity" type="xs:string" use="required" />
          <xs:attribute name="TargetEntity" type="xs:string" use="optional" />
          <xs:anyAttribute />
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Iz definicije 5.2.5.1 lahko opazimo rekurzivno strukturo politike, kar omogoča gnezdenje politik. Tako strukturo ima politika tudi v sintaksi PPLTL z naborom $\mathcal{L}_{ES}^{min}$. Elementu `Formula` smo dodali še tri atribute:

(a) `Context`: določa ime oz. identifikator konteksta, na katerega se formula nanaša;
(b) `SourceEntity`: določa lastnika politike (izvorna entiteta);
(c) `TargetEntity`: določa ciljno entiteto; v primeru, da je atribut izpuščen, velja za vse entitete, ki skušajo stopiti v interakcijo z izvorno entiteto.

Za ilustracijo lahko podamo preslikavo interakcijske politike π v jeziku PPLTL, ki smo jo definirali v poglavju 3 (razdelek 3.1.6), v politiko WS-Policy. Gre za primer spletnega mesta eBay (za podrobnosti glej poglavje 3, definicija 3.1.2.7) s perspektive kupca, ki nastopa kot izvorna entiteta. Politika kupca določa pravilo za vstop na licitacijo predmeta prodajalca. Pravilo določa, da je prodajalec v preteklosti vedno poslal kupljeni predmet kupcu pravočasno in je stopnja zaupanja med kupcem in prodajalcem vsaj $\text{PT} \in \mathcal{T}$ (kvalitativni model – glej razdelek poglavje 3, razdelek 3.1.4):

$$\pi^{lic} \equiv \neg \mathsf{F}^{-1}(\texttt{zamuda}) \wedge \texttt{Trust}(\text{PT})$$

Politiko preslikamo v dokument XML glede na shemo XSD iz definicije 5.2.5.1. Glede na minimalni nabor sintakse PPLTL, je potrebno operator F^{-1} in $\vee$ izraziti z drugimi operatorji nabora $\mathcal{L}_{ES}^{min}$. Iz enakosti 3.27 (poglavje 3) velja: $\mathsf{F}^{-1}(\pi) \equiv \texttt{true}\ \mathsf{S}\ \pi$, $\texttt{true} \equiv e \vee \neg e$ (e je dogodek znotraj strukture ES) in $\pi_0 \vee \pi_1 \equiv \neg(\neg\pi_0 \wedge \neg\pi_1)$. Politiko lahko sedaj zapišemo kot:

$$\pi^{lic} \equiv \neg\Big(\big(\neg(\neg\texttt{zamuda} \wedge \texttt{zamuda})\big)\ \mathsf{S}\ (\texttt{zamuda})\Big) \wedge \texttt{Trust}(\text{PT})$$

Politiko π^{lic} preslikamo v dokument XML, ki ga opišemo kot trditve v ogrodju WS-Policy z ustreznimi imenskimi prostori. Politiko prikazuje primer 5.12.

```
<wsp:Policy Name="http://example.org/policies/eBay"
            xmlns:tp="http://tempuri.org/STS/TrustPolicy.xsd"
            xmlns:wsp="http://www.w3.org/ns/ws-policy">
  <wsp:ExactlyOne>
    <wsp:All>
      <tp:Formula Context="eBay licitacija"
                  SourceEntity="customer@example.org"
                  TargetEntity="seller@ebay.example.org">
        <tp:Negation>
          <tp:Since>
            <tp:Arg1>
              <tp:Negation>
                <tp:Conjunction>
                  <tp:Arg1>
                    <tp:Negation>
                      <tp:EventFormula>zamuda</tp:EventFormula>
                    </tp:Negation>
                  </tp:Arg1>
                  <tp:Arg2>
                    <tp:EventFormula>zamuda</tp:EventFormula>
                  </tp:Arg2>
                </tp:Conjunction>
              </tp:Negation>
            </tp:Arg1>
            <tp:Arg2>
              <tp:EventFormula>zamuda</tp:EventFormula>
            </tp:Arg2>
          </tp:Since>
        </tp:Negation>
        <tp:TrustLevel>
          <tp:QualitativeValue>PT</tp:QualitativeValue>
        </tp:TrustLevel>
      </tp:Formula>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Primer 5.12: Interakcijska politika zaupanja znotraj WS-Policy.

5.2.5.4 Mehanizem za preverjanje skladnosti

V prejšnjem razdelku smo definirali preslikavo iz formalnega jezika za opis interakcijskih politik PPLTL v trditve oz. ustrezne elemente XML, ki jih lahko uporabimo v ogrodju WS-Policy. Z omenjenim ogrodjem je možno opisati kompleksne interakcijske politike posameznih entitet SOA (odjemalci, ponudniki, storitve), ki temelji na našem formalnem modelu zaupanja ATM.

Posledično je mogoče opisati bolj kompleksne scenarije oz. protokole interakcije, ki so pogojene z opisanimi politikami. Poleg tega lahko interakcijske politike dopolnimo tudi z

varnostnimi politikami oz. trditvami (specifikacija WS-SecurityPolicy [177]). To omogoča gradnjo kompleksnih scenarijev aplikacij SOA v smislu obvladovanja zaupanja in varnosti.

V poglavju 3, razdelek 3.1.6 smo definirali tudi ustrezen algoritem (algoritem 3.5) za preverjanje skladnosti z interakcijsko politiko π. Algoritem (metoda `Check()`) vrne logično vrednost, ki pove, ali na podlagi preteklih interakcij (interakcijska zgodovina h – glej definicijo 3.1.2.10) velja skladnost s politiko π: $h \models \pi$. Glede na to, da smo obstoječi sistem za OZ/OU definirali na ogrodju WS-Trust preko storitve STS (operaciji oz. sporočili RST/RSTR), lahko z vpeljavo dodatnih sporočil oz. žetonov sistem ustrezno funkcionalno razširimo. Naš cilj je razširitev sistema z mehanizmom oz. strojem (angl. policy engine) za preverjanje skladnosti za podano interakcijsko politiko preko konstruktov WS-Policy.

Vpeljemo nov tip žetona za sporočilo RST (glede na definicijo 5.2.4.1 obstajata že dva tipa: `TrustToken` in `InsertToken`), ki ga označimo kot `PolicyToken`.

Definicija 5.2.5.2 (Tip žetona za interakcijsko politiko)**.** Tip žetona za interakcijsko politiko π se identificira preko oznake URI kot:

- `http://tempuri.org/STS/PolicyToken`.

Ob sprejemu zahteve RST s tipom žetona `PolicyToken` storitev STS ve, da mora posredovati zahtevo stroju za politiko (je del sistema za OZ/OU), ki za eno ali več prejetih politik preveri skladnost ter vrne žeton z logično vrednostjo preko RSTR. Zahteva RST vsebuje (znotraj elementa `wst:SecondaryParameters`) eno ali več interakcijskih politik zaupanja $\pi \in \Pi$, opisanih preko sheme XSD (definicija 5.2.5.1) znotraj ogrodja WS-Policy. Primer 5.13 prikazuje zahtevo RST za preverjanje skladnosti interakcijskih politik (v primeru večih politik mora mehanizem preverjanja narediti presek politik).

Žeton politike ima enostavno strukturo; za podane politike vrne logično vrednost `true` ali `false`.

Definicija 5.2.5.3 (Žeton `PolicyToken`)**.** Struktura žetona `PolicyToken` je določena preko sledeče sheme XSD:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:wsp="http://www.w3.org/ns/ws-policy"
           xmlns:tp="http://tempuri.org/STS/TrustPolicy.xsd"
           targetNamespace="http://tempuri.org/STS/PolicyToken.xsd"
           xmlns="http://tempuri.org/STS/PolicyToken.xsd">
  <!-- lokacija sheme XSD za WS-Policy (wsp) -->
  <xs:import namespace="http://www.w3.org/ns/ws-policy"
             schemaLocation="http://www.w3.org/2007/02/ws-policy.xsd" />
  <!-- lokacija sheme XSD za TrustPolicy (tp) -->
  <xs:import namespace="http://tempuri.org/STS/TrustPolicy.xsd"
             schemaLocation="http://tempuri.org/STS/TrustPolicy.xsd" />
```

```
  <xs:element name="PolicyToken">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="TimeStamp" type="xs:dateTime" />
        <xs:element name="RequestedPolicies">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="wsp:Policy" maxOccurs="unbounded"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Result" type="xs:boolean" />
        <xs:element name="ErrorCode" type="xs:short" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Primer 5.14 prikazuje odgovor RSTR z vsebovanim žetonom politike, ki za podane politike vrne logično vrednost glede skladnosti.

Mehanizem (t. i. "policy engine") za preverjanje skladnosti politike π z interakcijsko zgodovino h, je del sistema za OZ/OU in je lahko implementiran kot *mehanizem za kontrolo dostopa* (angl. access control mechanism) v smislu varnostnega mehanizma [178]. Ta vsebuje

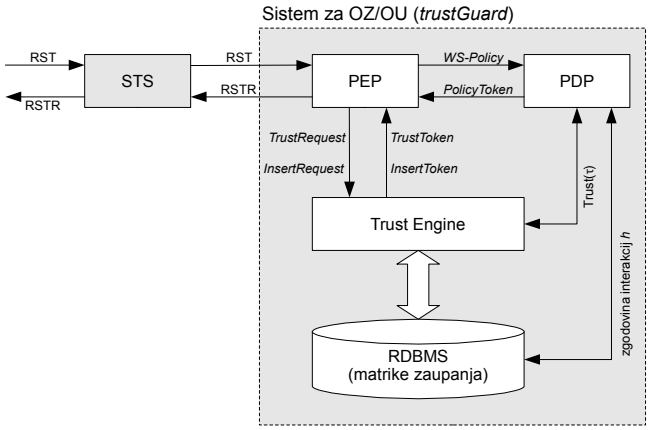
```
<wst:RequestSecurityToken xmlns:wst="...">
  <wst:TokenType>
    http://tempuri.org/STS/PolicyToken
  </wst:TokenType>
  <wst:RequestType>
    http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
  </wst:RequestType>
  <wst:SecondaryParameters>
    <wsp:Policy Name="policy_1"
                xmlns:tp="http://tempuri.org/STS/TrustPolicy.xsd"
                xmlns:wsp="http://www.w3.org/ns/ws-policy" ...>
      <!-- tp:assertions -->
    </wsp:Policy>
    ....
    <wsp:Policy Name="policy_n"
                xmlns:tp="http://tempuri.org/STS/TrustPolicy.xsd"
                xmlns:wsp="http://www.w3.org/ns/ws-policy" ...>
      <!-- tp:assertions -->
    </wsp:Policy>
  </wst:SecondaryParameters>
</wst:RequestSecurityToken>
```

Primer 5.13: Zahteva RST z interakcijskimi politikami.

```
<wst:RequestSecurityTokenResponse xmlns:wst="...">
  <wst:TokenType>
    http://tempuri.org/STS/PolicyToken
  </wst:TokenType>
  <wst:RequestedSecurityToken>
    <pt:PolicyToken xmlns:pt="http://tempuri.org/STS/PolicyToken.xsd">
      <pt:TimeStamp>2008-03-09T00:00:00.0000000+01:00</pt:TimeStamp>
      <pt:RequestedPolicies>
        <wsp:Policy>policy1</wsp:Policy>
        <wsp:Policy>policy2</wsp:Policy>
      </pt:RequestedPolicies>
      <pt:Result>true</pt:Result>
    </pt:PolicyToken>
  </wst:RequestedSecurityToken>
</wst:RequestSecurityTokenResponse>
```

Primer 5.14: Odgovor RSTR z žetonom politike.

točko uveljavljanja politike (angl. Policy Enforcement Point – PEP) in *odločitveno točko o politiki* (angl. Policy Decision Point – PDP). Točka PEP ima funkcijo prestrezanja zahtev – v našem primeru prestreže sporočilo SOAP z zahtevo RST, ki vsebuje tip žetona `PolicyToken`. Točka PEP zahtevo RST preda točki PDP, ki vrne rezultat oz. odločitev (uporaba algoritma 3.5), ali je podana politika skladna z interakcijsko zgodovino. Slika 5.6 prikazuje okvirno shemo logičnih delov znotraj sistema za OZ/OU. Storitev STS posreduje vse zahteve RST na točko PEP;



Slika 5.6: Mehanizem za preverjanje skladnosti politik kot del sistema za OZ/OU.

le-ta v primeru pošiljanja interakcijskih politik posreduje zahtevo na točko PDP. V nasprotnem primeru PEP zahteve transparentno posreduje na stroj zaupanja (t. i. "trust engine"), ki izračuna stopnjo zaupanja (zahteva `TrustRequest`) ali doda nove oceno oz. interakcijo (zahteva `InsertRequest`). Točka PDP preveri skladnost politike preko algoritma 3.5 (metoda `Check()`) in vrne rezultat.

Možno je, da točka PDP prejme istočasno več interakcijski politik – npr. politiko odjemalca in politiko storitve (ponudnika), do katere bi odjemalec rad dostopal. V tem primeru

mora odjemalec ugotoviti, ali izpolnjuje zahteve storitve; združiti (angl. merge) mora torej svojo in politiko storitve (odjemalec pravzaprav pošlje obe politike na točko PDP, ki opravi omenjeno funkcijo združevanja). Obstajajo različni algoritmi združevanja; npr. vir [174] opisuje združevanje politik v jeziku WSPL. Mi se omejimo na združevanje politik v ogrodju WS-Policy [166].

WS-Policy omogoča združevanje večih politik preko operacije *preseka* (angl. intersection). Če npr. odjemalec in ponudnik izražata zahteve preko svojih politik, operacija preseka določi *združljive* (t. i. kompatibilne) alternative v obeh politikah. Obstajata dve vrsti preseka [177]: (a) *strog* (angl. strict) in (b) *ohlapen* (angl. lax). V obeh primerih gre za preverjanja združljivosti trditev politike; strogi presek upošteva vse trditve, ohlapen pa samo trditve brez atributa `wsp:Ignorable="true"`. Dve trditvi sta združljivi, če sta *istega tipa*. V primeru, da katera od trditev vsebuje vgnezden izraz politike, sta trditvi združljivi natanko tedaj, ko imata obe trditvi vgnezden izraz in so tudi trditve v vgnezdenih izrazih medsebojno združljive.

Rezultat preseka politik je lahko ena ali več združljivih alternativ, ki vsebuje eno ali več trditev istega tipa. Entiteta mora izpolnjevati vse zahteve znotraj ene od teh izbranih alternativ preseka. Točka PDP v primeru večih podanih politik izvede najprej operacijo preseka ter na dobljeni množici trditev preveri skladnost z interakcijsko zgodovino h. V podrobnosti algoritma operacije preseka se tukaj ne spuščamo; osnovna izhodišča in smernice za implementacijo najdemo v [166, 174].

5.2.5.5 Protokol interakcij

Do sedaj smo definirali in predstavili vse glavne gradnike sistema za OZ/OU, ki temelji na formalnem modelu ATM. V razdelku 5.2.5.2 smo opisali osnovni scenarij oz. protokol med odjemalcem a in ponudnikom storitev b. Z vpeljavo interakcijskih politik zaupanja in mehanizma za preverjanje skladnosti lahko scenarij ustrezno razširimo. Predpostavimo, da ima vsaka entiteta svoj nabor zahtev, ki jih morajo izpolnjevati druge entitete, ki želijo stopiti v interakcijo z izbrano entiteto.

Branje oz. pridobivanje (interakcijskih) politik drugih entitet je možno preko specifikacije *WS-MetadataExchange* (WS-MEX) [173]. Specifikacija določa mehanizme za pridobivanje različnih metapodatkov – informacij o spletni storitvi (WS-Policy, definicija WSDL, sheme sporočil XSD); ti metapodatki so potrebni za komunikacijo (interakcijo) z izbrano spletno storitvijo. Pridobivanje metapodatkov poteka preko ustrezne zahteve GET znotraj elementa `wsa:Action`, končna točka pa je določena preko elementa `wsa:To` (WS-Addressing [112]). Primer 5.15 prikazuje zahtevo WS-MEX za branje politike WS-Policy s končne točke `http://services.example.org/purchaseOrder`. Odjemalec kot odgovor na zahtevo dobi politiko WS-Policy vsebovano v telesu sporočila SOAP.

Protokol interakcij med odjemalcem a in ponudnikom storitev b v infrastrukturi SOA lahko opišemo s sledečimi točkami:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
            xmlns:wsa="http://www.w3.org/2005/08/addressing"
            xmlns:mex="http://schemas.xmlsoap.org/ws/2004/09/mex" >
  <s12:Header>
    <wsa:To>http://services.example.org/purchaseOrder</wsa:To>
    <wsa:Action>
      http://schemas.xmlsoap.org/ws/2004/09/mex/GetMetadata/Request
    </wsa:Action>
    <wsa:MessageID>
      urn:uuid:73d7edfc-5c3c-5909-ba46-2480caee51e9
    </wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://client.example.org</wsa:Address>
    </wsa:ReplyTo>
  </s12:Header>
  <s12:Body>
    <mex:GetMetadata>
      <mex:Dialect>http://schemas.xmlsoap.org/ws/2004/09/policy</mex:Dialect>
      <mex:Identifier>
        http://services.example.org/purchaseOrder/policy
      </mex:Identifier>
    </mex:GetMetadata>
  </s12:Body>
</s12:Envelope>
```

Primer 5.15: Zahteva WS-MEX za branje politike storitve.

(1) Odjemalec v registru storitev (npr. UDDI) najde ustrezno spletno storitev s končno točko (oz. ponudnika, ki jo ponuja). Naj bo interakcijska politika odjemalca enaka π^a, politika storitve pa π^b.

(2) Odjemalec s končne točke b prebere politiko π^b (preko mehanizma WS-MEX).

(3) Odjemalec generira zahtevo RST za `PolicyToken` z dodanima π^a in π^b, ki jo pošlje na storitev STS.

(4) Točka PEP posreduje politiki π^a in π^b na točko PDP, ki izvede algoritem preverjanja skladnosti. Označimo rezultat preverjanja (logična vrednost) s $\texttt{Check}(\pi)$, potem je rezultat določen kot (5.3):

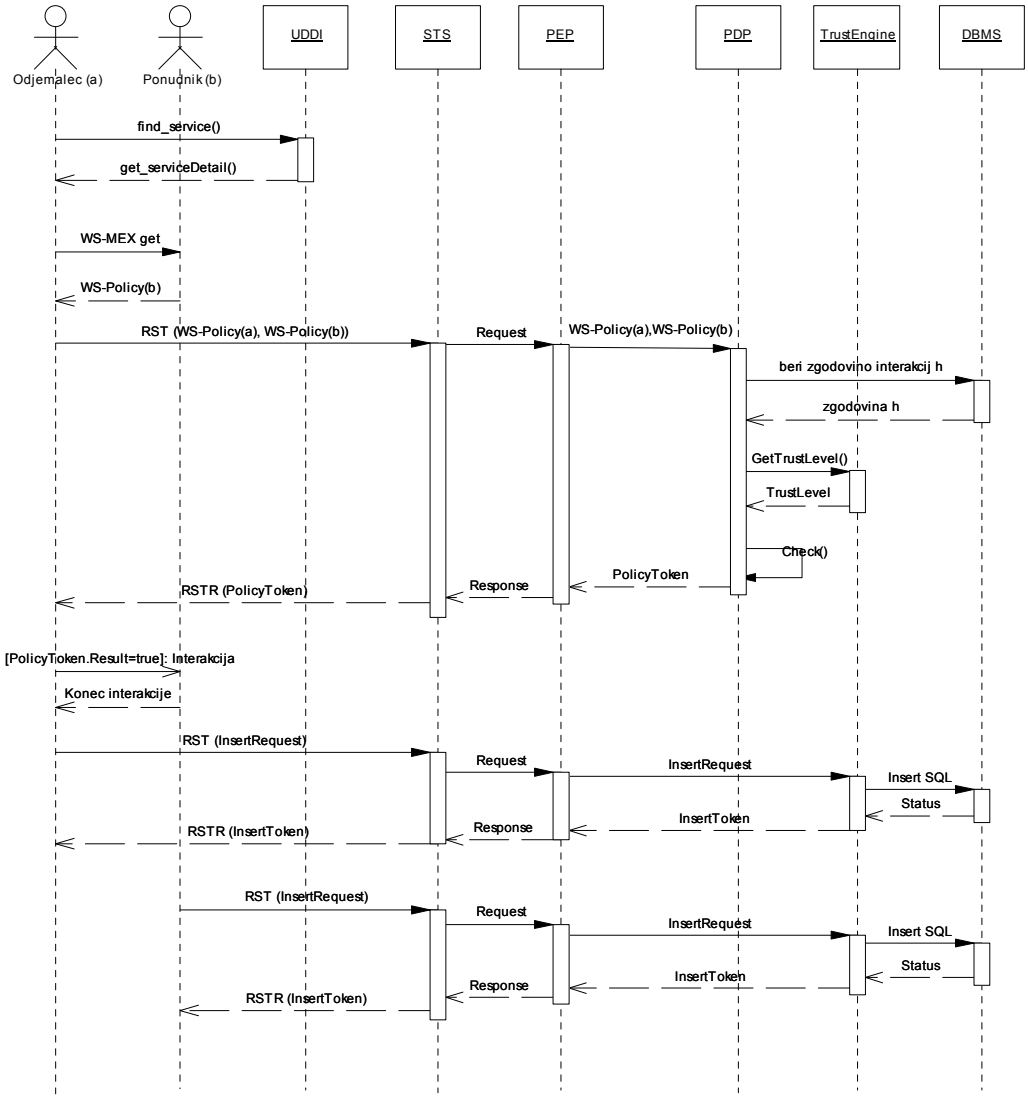
$$\texttt{Check}(\pi) = \begin{cases} \texttt{Check}(\pi^a) \wedge \texttt{Check}(\pi^b), & \pi^a \cap \pi^b = \emptyset \\ \texttt{Check}(\pi^a \cap \pi^b), & \pi^a \cap \pi^b \neq \emptyset \end{cases} \tag{5.3}$$

(5) V primeru, da dobi odjemalec a odgovor RSTR z žetonom `PolicyToken`, ki ima vrednost `true`, lahko stopi v interakcijo oz. proži storitev b.

(6) Po končani interakciji odjemalec a in/ali ponudnik storitev b podata informacijo o izvedeni interakciji z oceno izkušnje preko zahteve RST z elementom `InsertRequest`.

Omenjeni koraki pridejo v poštev za interakcijske politike zaupanja posameznih entitet; v primeru dodatnih varnostnih zahtev preko WS-SecurityPolicy [177] (npr. zaupnost, celovitost sporočil SOAP, izmenjava ključev), so potrebni dodatni koraki in komunikacija s storitvijo STS – v tem primeru lahko postane protokol bistveno bolj kompleksen.

Opisani protokol oz. scenarij interakcij lahko formalno opišemo tudi z *diagramom zaporedja* (angl. sequence diagram) jezika UML [114]. Diagram prikazuje interakcije na osnovi časovnega zaporedja – kako objekti (v našem primeru entitete) sodelujejo in izmenjujejo sporočila. Diagram interakcij prikazuje slika 5.7.



Slika 5.7: Diagram zaporedja interakcij.

Poglavje 6

Zaključek

V disertaciji smo obširno obravnavali področje obvladovanja zaupanja in ugleda v okoljih e-poslovanja. Ta okolja vsebujejo množico prostorsko ločenih entitet (agentov) iz različnih varnostnih domen, ki medsebojno sodelujejo oz. izvajajo transakcije. Medsebojno zaupanje in dobro obnašanje entitet je poleg informacijske varnosti lahko pogoj za učinkovito izvajanje in odločanje o medsebojnih transakcijah oz. interakcijah. Zaupanje je osnova za kooperativne akcije (interakcije) med agenti – bolj ko so občutljive, višje mora biti zaupanje.

Večina avtorjev obravnava zaupanje kot običajni varnostni mehanizem, v disertaciji pa ga obravnavamo kot *mehki* varnostni mehanizem. Običajni varnostni mehanizmi s pripadajočimi storitvami (zaupnost, celovitost, overjanje) varujejo lastnika informacij in informacijskih virov pred zlonamernimi uporabniki. V porazdeljenih okoljih e-poslovanja se pojavi tudi obratna situacija, kjer se morajo uporabniki varovati pred zlonamernimi ponudniki virov in storitev, ki ponujajo napačne ali zavajajoče informacije. Pri teh grožnjah običajni varnostni mehanizmi odpovejo, medtem ko socialni oz. mehki varnostni mehanizmi (npr. sistemi za obvladovanje zaupanja in ugleda) lahko varujejo uporabnike pred takimi grožnjami.

Disertacija obravnava zaupanje kot socio-kognitivni proces in ga formalizira kot več-dimenzijsko strukturo. Tako definira abstraktni model zaupanja kot n-terico $ATM = (\mathbf{G}, \Omega, \mathcal{T}, \mathcal{F}, \mathcal{O}, \Gamma, \Pi)$ s komponentami:

- $\mathbf{G}$ predstavlja dinamičen usmerjen graf entitet z medsebojnimi relacijami zaupanja,
- Ω je množica ocen zaupanja, ki jih podajo entitete po vsaki končani interakciji,
- $\mathcal{T}$ je množica stopenj zaupanja, ki so rezultat računskega algoritma,
- $\mathcal{F}$ je množica preslikav med množicama ocen in stopenj zaupanja,
- Γ je algoritem za izračun stopnje zaupanja in
- Π je množica interakcijskih politik zaupanja.

Model je splošen in omogoča uporabo različnih računskih metrik in modelov zaupanja. Predstavili smo kvantitativni model Jøsangove algebre [47] ter nadgradili kvalitativni model zaupanja [13] z ustrezno algebro, ki modelira obnašanje agentov. S simulacijami med množico agentov (glej dodatek A), smo dodatno utemeljili ustreznost kvalitativne metodologije. Model

ATM predstavlja razširljiv mehki varnostni mehanizem, ki se lahko uporabi v porazdeljenih okoljih e-poslovanja. Poleg tega ni omejen samo na kognitivne agente (t. i. ljudi), ampak tudi na tehnološke komponente ali storitve.

Storitveno usmerjene arhitekture (SOA) predstavljajo koncept za gradnjo porazdeljenih rešitev preko osnovnih gradnikov – spletnih storitev. Te se lahko povezujejo v kompleksne poslovne procese – v nove spletne storitve preko mehanizmov orkestracije (jezik BPEL) in koreografije (jezik WS-CDL). To pomeni, da je poslovni proces, do katerega dostopa agent, sestavljen iz množice pod-storitev, ki se prožijo v določenem zaporedju. Osnovni algoritem Γ kvalitativnega in kvantitativnega modela ATM smo nadgradili, da omogoča tudi izračun stopnje zaupanja kompozitnih storitev – poslovnih procesov, opisanih v jeziku BPEL. Znotraj SOA obstaja poleg osnovnih specifikacij XML tudi množica standardov WS-* (OASIS, W3C), ki pokrivajo predvsem področje povezljivosti (interoperabilnosti) in druga področja (varnost, zanesljivost, transakcije) spletnih storitev. Področje zaupanja kot mehkega varnostnega mehanizma ne obravnava noben standard SOA. Z integracijo modela ATM kot mehkega varnostnega mehanizma v standard WS-Trust smo to pomanjkljivost odpravili. V infrastrukturo SOA smo preko razširitve standardov integrirali kvantitativni in kvalitativni model zaupanja kot standardna gradnika. Ta gradnika lahko uporabljata tudi ostale standarde WS-* (npr. WS-Policy za opis interakcijskih politik zaupanja, WS-MetadataExchange za branje zahtev oz. zmožnosti entitet, WS-Security za varnostne mehanizme). Preko standardnih gradnikov smo razvili tudi prototip sistema za obvladovanje zaupanja in ugleda *trustGuard* kot rešitev SOA s centraliziranim arhitekturnim pristopom.

6.1 Nadaljnje delo

Predstavljeno področje v disertaciji ima še številne možnosti za nadaljnje raziskave in izboljšave. Abstraktni model zaupanja lahko nadgradimo z ustreznimi dodatnimi socialnimi dejavniki zaupanja, katere smo omenili v razdelku 3.1.1, vendar bi morali vse hipoteze preveriti z ustreznimi eksperimenti (bodisi v realnem okolju ali z računalniškimi simulacijami):

- Tveganje: poleg stopnje zaupanja predstavlja dodatni element pri odločanju o prihodnjih interakcijah. Tu bi lahko uporabili model, predstavljen v razdelku 3.1.1. Ta na podlagi ustreznih parametrov G_s (faktor dobička interakcije) in subjektivnega odnosa entitete do tveganja μ za vsako interakcijo določi odločitveno ploskev $F_c(p, G_s) = p^{\frac{\mu}{G_s}}$, ki je specifična za vsako entiteto. Vrednost $p \in [0, 1]$ predstavlja stopnjo zaupanja do ciljne entitete, ki se izraža kot verjetnost. Za uporabo tega modela bi morali transformirati stopnje zaupanja $\tau \in \mathcal{T}$ znotraj ATM v zvezni interval $[0, 1]$ (npr. z uporabo multinomskih Bayesovih modelov). Razširjeni model ATM bi lahko predstavljal sistem za podporo odločanju (angl. Decision Support System – DSS).

- Korist: z združitvijo dimenzije koristi posameznih interakcij in stopnje zaupanja med entitetami, bi lahko modelirali tudi interakcije v ekonomskem smislu (racionalnost entitet glede na dobljeno materialno korist) za posamezne kontekste zaupanja. Ena od možnosti je razširitev konstrukta konteksta $\varphi = (Name, ES)$ z matrikami izidov oz. izidov (angl. pay-off) za posamezne konfiguracije dogodkov $\mathcal{X} \in \mathcal{C}_{ES}$.

- Vzajemnost: ta koncept se nanaša na obnašanje entitet, kjer se na pozitivne akcije odzivajo pozitivno, na negativne pa negativno. Uporabili bi lahko podoben princip kot Mui in Halberstadt [58] preko faktorja vzajemnosti $\gamma_{a,b}$ med dvema entitetama a in b. Ta določa delež pozitivnih (kooperativnih) akcij glede na vse interakcije. Večji delež pomeni povečanje stopnje ugleda/zaupanja in obratno – večja stopnja zaupanja pomeni večjo verjetnost za vzajemnost (povratna zanka).

- Operatorji zaupanja: množico operatorjev $\mathcal{O}$ kvalitativnega modela lahko z ustreznimi eksperimenti in/ali simulacijami razširimo z dodatnimi operatorji, ki modelirajo specifično obnašanje agentov. To pomeni tudi razširitev kvalitativne algebre z novimi pravili sklepanja.

Eden izmed problemov, ki ga disertacija ne rešuje, je problem, kako zagotoviti, da agenti (končni uporabniki) podajajo ocene po končanih interakcijah. Model ATM namreč predpostavlja podajanje ocen oz. izkušenj po interakcijah; če le-teh ni, model ne more delovati kot mehki varnostni mehanizem. Potrebno bi bilo podati ustrezne vzvode za spodbujanje in motiviranje agentov k podajanju izkušenj preteklih interakcij oz. transakcij ali njihovo indirektno zajemanje. V komercialnih sistemih [9] so ti vzvodi predvsem finančne narave (npr. gotovina pri *Epinions*(1), popusti pri *BizRate*(2)), pri ostalih sistemih pa samo nematerialni vzvodi (npr. večji/boljši status, položaj).

Veliko težje rešljiv je problem namernega podajanja napačnih ocen, bodisi neupravičeno dobrih ocen (angl. "ballot stuffing") ali neupravičeno slabih ocen (angl. "bad-mouthing"). V tem primeru pride do izračuna nezanesljivih (nizkih, previsokih) stopenj zaupanja in ugleda. Mnogi avtorji skušajo rešiti ta problem na različne načine; Dellarocas [179] podaja različne pristope za nevtralizacijo tega problema. Opisuje različne scenarije interakcij znotraj skupnosti entitet, ki jih loči v kupce in prodajalce. Glede na scenarije predlaga različne "imunizacijske" metode (za preprečevanje, odkrivanje in odpravljanje posledic napačnega ocenjevanja oz. zlonamernega obnašanja), ki temeljijo na izbranih statističnih metodah (npr. frekvenčno filtriranje) in parametrih skupnosti entitet (npr. obvladovanje anonimnosti in identitete agentov, interakcijske politike za nove agente). Nekatere od teh metod bi lahko uporabili tudi v našem sistemu za obvladovanje zaupanja in ugleda, vsekakor pa bi bile potrebne nadaljnje raziskave na tem področju, ker (še) ne obstajajo učinkoviti mehanizmi za rešitev tega problema.

(1) http://www.epinions.com/.
(2) http://www.bizrate.com/.

Velja omeniti še možnost izboljšave trenutne implementacije sistema *trustGuard*. Trenutno uporablja centraliziran arhitekturni pristop, ampak zaradi večje učinkovitosti (npr. veliko število entitet, skalabilnost), bi lahko uporabili porazdeljeni arhitekturni pristop (glej poglavje 2, razdelek 2.4.2).

6.2 Prispevki k znanosti

Izvirni prispevki disertacije so sledeči:

1. Analiza metod in mehanizmov zaupanja ter definicija pojma zaupanja ne kot zgolj varnostnega mehanizma, ampak kot manifestacije sklepanja in presojanja, ki upošteva rezultate raziskav s področij sociologije, psihologije in informacijskih znanosti. Določili smo ustrezne dejavnike oz. dimenzije zaupanja ter formalno definirali abstraktni model zaupanja $ATM = (\mathbf{G}, \Omega, \mathcal{T}, \mathcal{F}, \mathcal{O}, \Gamma, \Pi)$.

2. V okviru abstraktnega modela smo podali nove algoritme za izračun stopnje zaupanja na podlagi principa tranzitivnosti z ustreznimi predpostavkami.

3. Podrobno smo razvili kvalitativni model zaupanja z ustrezno kvalitativno algebro, ki temelji na psihološki in sociološki perspektivi uporabnika.

4. V abstraktni model zaupanja smo vpeljali mehanizem za interakcijske politike zaupanja na podlagi različice linearne temporalne logike. Definirali in razširili smo tudi ustrezne algoritme za preverjanje skladnosti politik zaupanja.

5. Analizirali smo koncept storitvenih arhitektur (SOA) v smislu kompozicije spletnih storitev preko mehanizma orkestracije (jezik BPEL) in z uporabo Petrijevih mrež razvili algoritme za izračun stopnje zaupanja kompozita spletnih storitev (t. i. poslovnih procesov).

6. V infrastrukturo SOA smo preko razširitve odprtih standardov WS-* (WS-Trust, WS-Security) integrirali kvalitativni in kvantitativni model zaupanja. Definirali smo arhitekturo rešitve sistema za obvladovanje zaupanja in ugleda z ustreznimi gradniki odprtih standardov infrastrukture SOA.

Dodatek A

Simulacije

Dodatek opisuje simulacije na osnovi kvalitativnega modela zaupanja (glej poglavje 3, razdelek 3.1.4). Programska oprema za simulacijo *TrustSimulator*[1] je napisana v programskem jeziku *C#* [180]. Zaradi poenostavitve trenutna implementacija ne vsebuje računskega modela (algoritmov) na principu tranzitivnosti, ki uporablja zaupanje v priporočilo (glej poglavje 3, razdelek 3.1.3), ampak vsebuje samo računski model na podlagi funkcionalnega zaupanja.

Cilj simulacije je najti ustrezno obnašanje kvalitativnega modela na podlagi različnih množic parametrov in njihovih vrednosti. Dobljeni rezultati tako služijo za razširitve in nadaljnje raziskave kvalitativnega modela zaupanja.

A.1 Parametri

Tip simulacij in njihovi rezultati so odvisni od uporabljenih parametrov. Slika A.1 prikazuje posamezne sklope parametrov, ki vplivajo na potek simulacije.

A.1.1 Generator naključnih števil

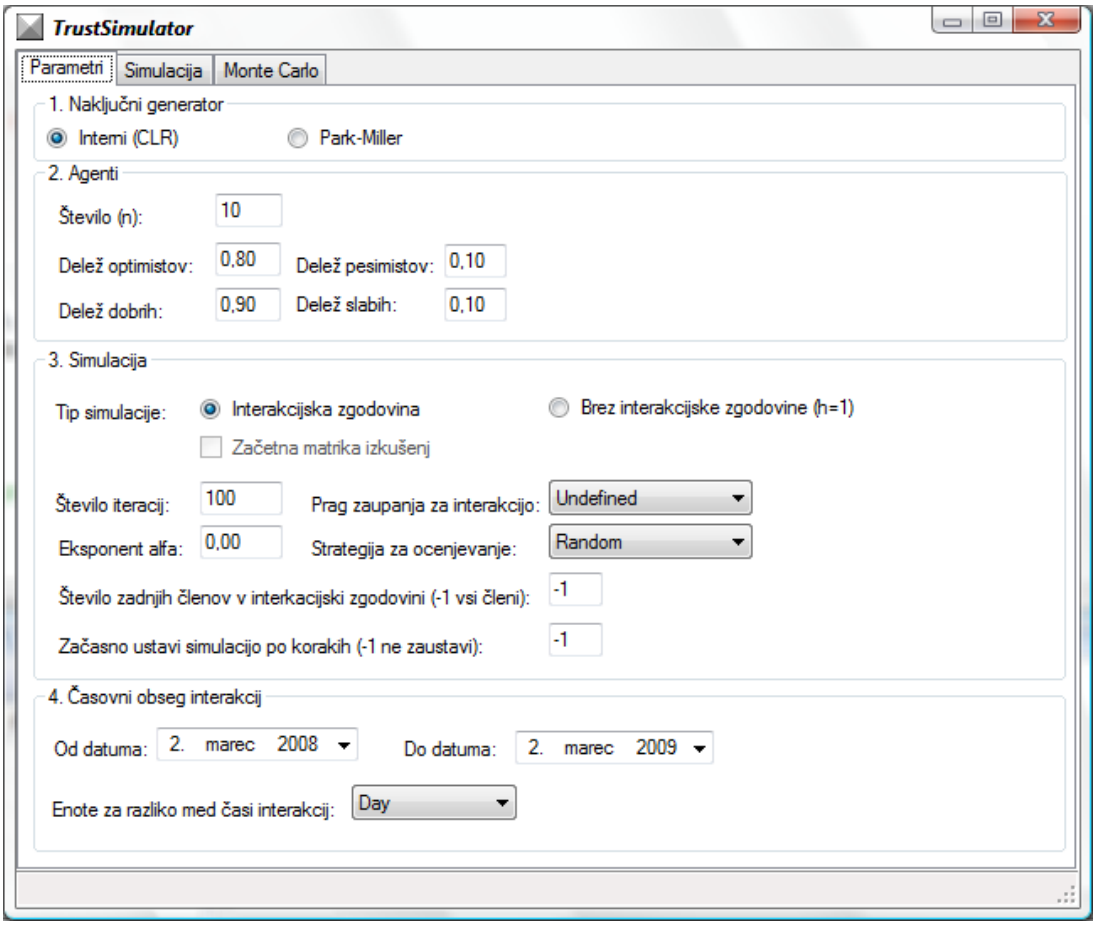
Simulator lahko uporablja vgrajeni generator naključnih števil programskega jezika *C#*[2] ali izboljšan naključni generator avtorjev Park & Miller [181].

A.1.2 Agenti

Število n določa celotno število agentov, kar določa tudi velikost matrike interakcij $\mathbb{M}(t, \varphi)$. Glede na množico operatorjev zaupanja $\mathcal{O} = \{\curlywedge, \curlyvee, \sim\}$[3] obstajajo tri vrste agentov: *optimisti*, *pesimisti* in *realisti*. Črka s označuje delež (angl. share) agentov, velja enakost: $s(\text{opt}) + s(\text{pes}) + s(\text{bal}) = 1$:

(a) *optimisti*: uporabljajo operator $\curlywedge$; $0 \leq s(\text{opt}) \leq 1$,

[1] Na voljno na priloženem mediju CD-ROM.
[2] Common Language Runtime (CLR).
[3] $\curlywedge$ je optimistični (opt), $\curlyvee$ je pesimistični (pes) in $\sim$ uravnotežen (bal) operator.



Slika A.1: Začetni ekran – parametri simulatorja.

(b) *pesimisti*: uporabljajo operator Υ; $0 \leq s(\text{pes}) \leq 1$,
(c) *realisti*: uporabljajo operator $\sim$; $s(\text{bal}) = 1 - s(\text{opt}) - s(\text{pes})$.

Dodani so tudi dodatni atributi agentov (t. i. *dobri* (angl. good), *slabi* (angl. bad), *neznani* (angl. unknown)), ki določajo obnašanje oz. ocenjevanje agentov po izvedenih interakcijah (npr. uporaba modela ocenjevanja, ki sta ga podala Theodorakopoulos in Baras [83]). Tudi v tem primeru velja enakost: $s(\text{good}) + s(\text{bad}) + s(\text{unknown}) = 1$.

A.1.3 Simulacija

Obstajata dve vrsti simulacije za izračun stopnje zaupanja: (1) *preko interakcij oz. interakcijske zgodovine* (glej poglavje 3, definicija 3.1.2.9 in 3.1.2.10) med agenti ter (2) *preko računanja vektorjev zaupanja* za vse pare agentov znotraj podane matrike izkušenj. V nadaljevanju sledi opis obeh vrst simulacij. Pred tem velja še poudariti, da je zaradi poenostavitve (ne uporabljamo interakcijskih politik) kontekst $\varphi_0 = (Name_0, ES_0) \in \Phi$ simulacij fiksen in brez dogodkov (t. i. dogodkovna struktura je prazna množica $ES = \emptyset$): $\varphi_0 = (\text{"}\texttt{DEFAULT_CONTEXT}\text{"}, \emptyset)$.

A.1.3.1 Interakcijska zgodovina

Simulacija generira interakcije med naključno izbranimi agenti; parametri, ki se upoštevajo, so sledeči:

(a) *Število interakcij*: določa število interakcij, ki se generirajo med naključno izbranimi izvornimi in ciljnimi agenti.

(b) *Prag zaupanja za interakcijo*: interakcija med izvornim a in ciljnim agentom b je možna samo v primeru, če je stopnja zaupanja $\tau_{a,b}$ (izračunana iz preteklih interakcij) večja ali enaka od praga $\tau_0 \in \mathcal{T} \cup \{-\infty\}$. V tem primeru lahko govorimo o interakcijski politiki (glej poglavje 3, razdelek 3.1.6), ki je zapisana kot $\pi \equiv \texttt{Trust}(\tau_0)$.

(c) *Začetna matrika izkušenj*: v primeru obstoja praga zaupanja $\tau_0 \neq -\infty$, lahko interaktivno vnesemo (generiramo) začetno matriko izkušenj, ki služi pri izračunu stopnje zaupanja.

(d) *Eksponent alfa*: utežni faktor $\alpha \in [0, \infty)$ služi za izračun utežene vsote $\sigma_{a,b}(t', \varphi_0) = \frac{\sum_{t \leq t'} e^{-\alpha(t'-t)} f_\Omega(x_{i,j}(t,\varphi_0).\omega))}{m}$ (glej poglavje 3, definicija 3.1.2.12) ocen v interakcijski zgodovini $h_{a,b}(\varphi_0)$. Eksponent α določa težo posameznega člena (ocene) v interakcijski zgodovini glede na oddaljenost od časa (trenutka) opazovanja t'.

(e) *Število zadnjih členov v interakcijski zgodovini*: določa koliko zadnjih členov (ocen) znotraj interakcijske zgodovine se upošteva pri izračunu utežene vsote.

(f) *Strategija za ocenjevanje*: določa pravila ocenjevanje preteklih interakcij $x_{a,b}$, ki jih poda izvorni agent a; obstajajo sledeče možnosti:

- `Random`: ocena se izbere naključno iz kvalitativne množice ocen $\Omega = \{\text{D}, \text{PD}, \text{U}, \text{PT}, \text{T}\}$ z enako verjetnostjo $p = \frac{1}{|\Omega|} = \frac{1}{5}$.
- `AgentType`[4]: ocena se določi glede na tip obnašanja agentov (t. i. `good`, `bad`, `unknown`) po pravilih (delno povzeto po viru [83]), ki jih prikazuje tabela A.1.

Izvorni agent a	Ciljni agent b	Ocena $\omega_{a,b}$
`good`	`bad`	D
`good`	`good`	T
`good`	`unknown`	`Random`
`bad`	`bad`	T
`bad`	`good`	D
`bad`	`unknown`	`Random`
`unknown`	`*`	`Random`

Tabela A.1: Pravila za ocenjevanje glede na tip obnašanja.

[4] Velja poudariti, da je to področje še predmet nadaljnjih raziskav in je samo eksperimentalno vključeno v simulator; v vseh simulacijah se trenutno uporablja samo strategija `Random`.

(g) *Ustavitev simulacije po določenem številu korakov*: simulacijo lahko ustavimo za pregled vmesnih rezultatov (matrika izkušenj do določenega trenutka) in morebitno spreminjanje vrednosti za deleže operatorjev in izkušenj znotraj matrike $\mathbb{M}'(t, \varphi)$.

Potek simulacije preko interakcijske zgodovine je sledeč:

(1) Generiramo množico agentov $\mathcal{A}$ z ustreznimi lastnostmi glede na vrednosti parametrov (deleži operatorjev, deleži tipa obnašanja).

(2) V vsaki iteraciji naključno izberemo izvornega a in ciljnega agenta b, nato generiramo interakcijo $x_{a,b}(t, \varphi) = (\emptyset, \omega)$ s časovno oznako t (vrednost je določena naključno iz časovnega obsega interakcij). Ocena $\omega_{a,b}$ se določi glede na izbrano strategijo ocenjevanja; v primeru, da je podan prag zaupanja τ_0, je interakcija $x_{a,b}$ možna samo v primeru, ko je trenutna stopnja zaupanja $\tau_{a,b} \geq \tau_0$.

(3) Po koncu interakcij izračunamo agregirano matrika izkušenj $\mathbb{M}'(t_{Now}, \varphi_0)$ z elementi $\lambda_{a,b}(\varphi_0)$, kjer je t_{Now} trenutni čas.

(4) Iz agregirane matrike izkušenj $\mathbb{M}'(t_{Now}, \varphi_0)$ sledi izračun t. i. *matrike izračunanih stopenj* $\mathbb{M}'^{trust}(\varphi_0)$ z elementi $trust^{\mathrm{op}}_{a,b}(\varphi_0)$ (op $\in \mathcal{O}$ je operator, ki ga uporablja izvorni agent a), ki se izračuna preko algoritma Γ (glej poglavje 3, definicija 3.1.4.4):

$$trust^{\mathrm{op}}_{a,b}(\varphi_0) = \begin{cases} tvec^{\mathrm{op}}_{a,b}(\varphi_0), & \lambda_{a,b}(\varphi_0) \neq -\infty \\ rep^{\mathrm{op}}_{b}(\varphi_0), & \lambda_{a,b}(\varphi_0) = -\infty \end{cases}$$

Izračunana matrika stopenj služi za opazovanje deležev posameznih kvalitativnih stopenj množice $\mathcal{T}$ glede na izbrane parametre.

A.1.3.2 Vektorji zaupanja

Ta vrsta simulacije ne temelji na interakcijah (ne uporablja parametrov simulacije z interakcijsko zgodovino), ampak se uporabi začetna matrika izkušenj $\mathbb{M}'(t_0, \varphi_0)$ (se lahko generira naključno ali interaktivno). V vsakem koraku se izračuna nova matrika, ki vsebuje elemente – vektorje zaupanja $tvec_{a,b}(\varphi_0)$, ki se izračunajo iz elementov predhodne matrike. Postopek je sledeč (podrobnosti v [13]):

(1) Definiramo/generiramo začetno matriko izkušenj $\mathbb{M}'(t_0, \varphi_0)$.

(2) V naslednji iteraciji $k + 1$ izračunamo nove elemente $\lambda^{k+1}_{i,j}(\varphi_0)$ – vektorje zaupanja $tvec^{\mathrm{op}}_{i,j}(\varphi_0)$, kjer $i = 1, \ldots, |\mathcal{A}|, j =$ konstanten (glej poglavje 3, razdelek 3.1.4), kjer je op $\in \mathcal{O}$ operator izvornega i-tega agenta:

$$\lambda^{k+1}_{i,j}(\varphi_0) = tvec^{\mathrm{op}}_{i,j} = (\ldots((\lambda^k_{i,j} \ \mathrm{op} \ \lambda^k_{1,j}) \ \mathrm{op} \ \lambda^k_{2,j})\ldots \ \mathrm{op} \ \lambda^k_{n-1,j}) \ \mathrm{op} \ \lambda^k_{n,j}, \ \lambda^k_{i,j} \neq -\infty$$

(3) V primeru, da je nova matrika izkušenj enaka prejšnji (t. i. $\mathbb{M}'^{k+1}(t_0, \varphi_0) = \mathbb{M}'^{k}(t_0, \varphi_0)$) končamo postopek (pride do konvergence vrednosti) in gremo na naslednji korak (4). V nasprotnem primeru izvedemo prejšnji korak (2).

(4) Iz matrike $\mathbb{M}'(t_0, \varphi_0)$ s konvergiranimi vrednostmi izkušenj $\lambda_{a,b}(\varphi_0)$ se izračuna *matrika izračunanih stopenj zaupanja* $\mathbb{M}'^{trust}(\varphi_0)$ (enako kot v koraku (4) simulacije z interakcijsko zgodovino – glej razdelek A.1.3.1). Matrika služi za opazovanje deležev posameznih kvalitativnih stopenj množice $\mathcal{T}$ glede na izbrane parametre.

Tudi to vrsto simulacije lahko začasno ustavimo med izvajanjem (po določenem koraku), kjer lahko spremenimo delež posameznih operatorjev med agenti in/ali spremenimo vrednosti znotraj matrike izkušenj. To nam omogoča spremljanje konvergence vrednosti.

A.1.4 Časovni obseg interakcij

Ta sklop parametrov pride v poštev samo v primeru, ko je izbrana simulacija z interakcijsko zgodovino. Parametra sta sledeča:

(a) *Spodnja in zgornja datumska meja*: množica T možnih časovnih oznak t interakcij $x_{a,b}(t, \varphi_0)$, torej $t \in T = [\texttt{from_date}, \texttt{to_date}]$.

(b) *Enote za razliko med časovnimi oznakami interakcij*: enote za izračun razlik med časovnimi oznakami (značkami)[5] t interakcij; razlike $t' - t$ (t' je čas opazovanja, t je časovna oznaka interakcije) se rabijo pri izračunu preteklih izkušenj $\lambda_{a,b}$ (razlike nastopajo pri utežnem faktorju ocen $e^{-\alpha(t'-t)}$). Tipične enote so *sekunda, minuta, ura, dan, teden, mesec.*

A.2 Primeri simulacij

V tem razdelku so podani nekateri scenariji in rezultati različnih tipov simulacij (z interakcijsko zgodovino, z vektorji zaupanja). Rezultati so predstavljeni kot matrike oz. končni deleži posameznih stopenj zaupanja.

A.2.1 Simulacija: interakcijska zgodovina

V tem razdelku predstavimo nekaj simulacij z nekaterimi tipičnimi parametri (delež operatorjev, utežni faktor α, število interakcij (iteracij)). Simulator sicer omogoča tudi različne kombinacije drugih parametrov, vendar se omejimo samo na nekatere[6]. Omeniti je potrebno oznake agentov, ki jih označujemo z nizom `A<indeks><operator><obnašanje>` (npr. `A3o+`), kjer `indeks` $\in [1 \ldots |\mathcal{A}|]$, `operator` $\in \{$`'o','p','b'`$\}$ (`'o'` = optimist, `'p'` = pesimist, `'b'` = realist) ter `obnašanje` $\in \{$`'+','-','?'`$\}$ (`'+'` = dober (good), `'-'` = zlonameren (bad), `'?'` = neznan (unknown)).

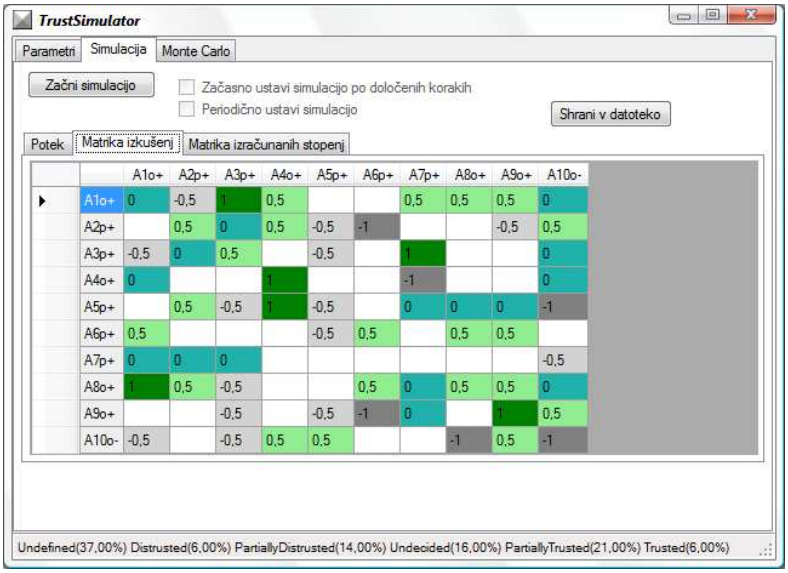
[5] Časovni trenutek (datumska oznaka), ko je prišlo do interakcije.
[6] Določeni parametri so še predmet nadaljnjih raziskav.

A.2.1.1 Primer 1: število interakcij

V tej simulaciji spreminjamo število interakcij pri fiksnem številu agentov. Začetne vrednosti parametrov so sledeče (izpis iz dnevnika simulatorja):

```
======== SIMULATION PARAMETERS (initial) ========
** Random generator: Internal *
** Agents **
  num of agents: 10
  optimistic: 0,5; pessimistic: 0,5; balanced: 0,0
  good: 0,90; bad: 0,10; unknown: 0,00
** Simulation **
  type: interactions
  num of random interactions: 100
  alpha: 0,00
  interactions threshold: NONE
  interaction assessment strategy: Random
  num of last interactions in history: -1
  steps to interrupt simulation: -1
** Interaction time properties **
  from date: 6.3.2008 11:31:46
  to date: 6.3.2009 11:31:46
  date difference interval: Day
======== END ========
```

Slika A.2 oz. A.3 prikazuje izračunano matriko izkušenj oz. stopenj (prikazane so preslikane vrednosti s preslikavo $f_\Omega \in \mathcal{F}$ – glej poglavje 3, razdelek 3.1.4) z ustreznimi oznakami agentov.



	A1o+	A2p+	A3p+	A4o+	A5p+	A6p+	A7p+	A8o+	A9o+	A10o-
A1o+	0	-0,5	1	0,5			0,5	0,5	0,5	0
A2p+		0,5	0	0,5	-0,5	-1			-0,5	0,5
A3p+	-0,5	0	0,5		-0,5		1			0
A4o+	0			1			-1			0
A5p+		0,5	-0,5	1	-0,5		0	0	0	-1
A6p+	0,5				-0,5	0,5		0,5	0,5	
A7p+	0	0	0							-0,5
A8o+	1	0,5	-0,5			0,5	0	0,5	0,5	0
A9o+			-0,5		-0,5	-1	0		1	0,5
A10o-	-0,5		-0,5	0,5	0,5			-1	0,5	-1

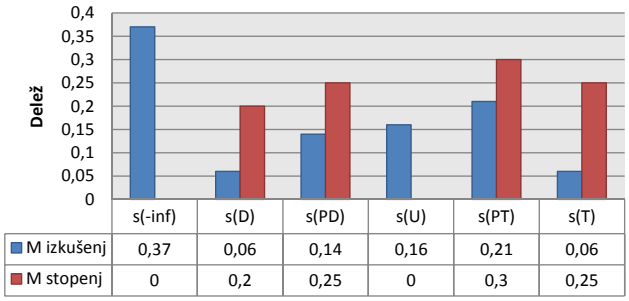
Slika A.2: Izračunana matrika izkušenj.

Simulator izračuna tudi deleže (t. i. površino) posameznih kvalitativnih vrednosti znotraj matrik (deleže označimo s črko s). Te deleže lahko predstavimo z grafikoni, iz katerih lahko

	A1o+	A2p+	A3p+	A4o+	A5p+	A6p+	A7p+	A8o+	A9o+	A10o-
A1o+	1	0,5	1	1	0,5	0,5	1	0,5	1	0,5
A2p+	-0,5	-0,5	-0,5	0,5	-0,5	-1	-1	-1	-0,5	-1
A3p+	-0,5	-0,5	-0,5	0,5	-0,5	-1	-1	-1	-0,5	-1
A4o+	1	0,5	1	1	0,5	0,5	1	0,5	1	0,5
A5p+	-0,5	-0,5	-0,5	0,5	-0,5	-1	-1	-1	-0,5	-1
A6p+	-0,5	-0,5	-0,5	0,5	-0,5	-1	-1	-1	-0,5	-1
A7p+	-0,5	-0,5	-0,5	0,5	-0,5	-1	-1	-1	-0,5	-1
A8o+	1	0,5	1	1	0,5	0,5	1	0,5	1	0,5
A9o+	1	0,5	1	1	0,5	0,5	1	0,5	1	0,5
A10o-	1	0,5	1	1	0,5	0,5	1	0,5	1	0,5

Slika A.3: Izračunana matrika stopenj.

vidimo morebitno povezavo (odvisnost) med vrednostmi določenih parametrov in deleži. Slika A.4 prikazuje deleže kvalitativnih vrednosti obeh matrik za podani nabor parametrov. Ker

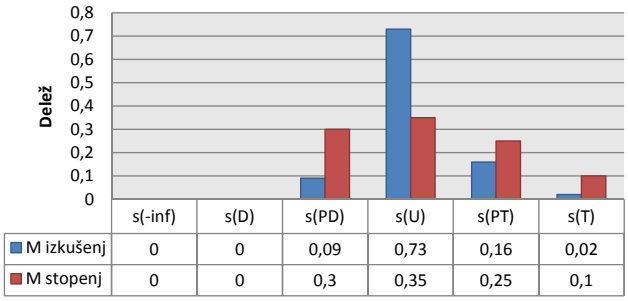


	s(-inf)	s(D)	s(PD)	s(U)	s(PT)	s(T)
M izkušenj	0,37	0,06	0,14	0,16	0,21	0,06
M stopenj	0	0,2	0,25	0	0,3	0,25

Slika A.4: Deleži kvalitativnih vrednosti ($n = 10$, $iter = 100$, $s(\text{opt}) = s(\text{pes}) = 0.5$, $s(\text{bal}) = 0$, $\alpha = 0$).

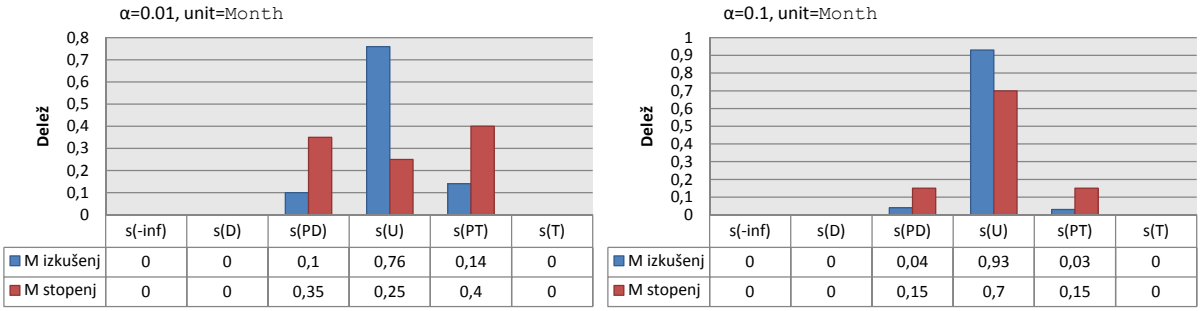
je število interakcij (100) razmeroma majhno glede na število agentov (10), je delež $s(-\infty)$ razmeroma velik (37 %); v matriki stopenj je $s(-\infty) = 0$, ker se za stopnjo zaupanja neobstoječih relacij med agentoma a in b uporabi ugled ciljnega agenta b: $rep_b^{\text{op}}(\varphi_0)$. Ker je $\alpha = 0$, imajo ocene interakcij v interakcijski zgodovini enako težo (utežni faktor je enak 1). Ob povečanju števila interakcij pričakujemo v matrikah izkušenj/stopenj največji delež $s(\text{U})$, ki predstavlja povprečno vrednost. Slika A.5 prikazuje deleže ob povečanju števila interakcij (1000).

V primeru, da je $\alpha > 0$, dobijo večjo težo zadnje interakcije, teža ostalih pa eksponentno s časovno oddaljenostjo pada proti vrednosti 0; lahko rečemo, da se ocene starih interakcij enostavno ignorira. Če vzamemo premajhno enoto za razliko med časi interakcij (npr. *dan*), je pričakovana utežena vsota ocen $\sigma_{i,j}(t', \varphi_0)$ (glej poglavje 3, definicija 3.1.2.12) enaka 0. Slika A.6 prikazuje deleže kvalitativnih ocen, kjer je $\alpha = 0.01$ (levo) in $\alpha = 0.1$ (desno) ter enota časovne razlike *mesec* (`Month`). Opazimo lahko, da so pri večji vrednosti $\alpha = 0.1$ deleži vrednosti v matriki izkušenj $\mathbb{M}'$ in stopenj $\mathbb{M}'^{trust}$ medsebojno primerljivi (sorazmerni). Pri



Slika A.5: Deleži kvalitativnih vrednosti ($n = 10$, $iter = 1000$, $s(\text{opt}) = s(\text{pes}) = 0.5$, $s(\text{bal}) = 0$, $\alpha = 0$).

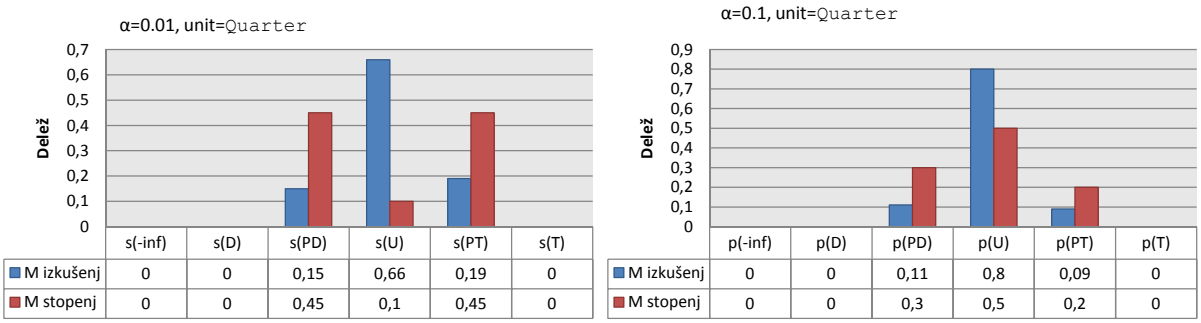
manjši vrednosti $\alpha = 0.01$ opazimo obratno sliko: v $\mathbb{M}'$ se delež $s(\text{U})$ povečuje, deleža $s(\text{PD})$ in $s(\text{PT})$ pa zmanjšujeta; v matriki $\mathbb{M}'^{trust}$ je opazno obratno – delež $s(\text{U})$ se zmanjšuje, deleža $s(\text{PD})$ in $s(\text{PT})$ pa povečujeta. Opazimo, da sta deleža $s(\text{PD})$ in $s(\text{PT})$ v obeh matrikah približno enaka, kar velja pripisati dejstvu, da je delež optimistov in pesimistov enak. Lahko rečemo, da



Slika A.6: Deleži vrednosti ($n = 10$, $iter = 1000$, $unit = \texttt{Month}$, $s(\text{opt}) = s(\text{pes}) = 0.5$, $s(\text{bal}) = 0$).

povečanje vrednosti parametra α veča delež $s(\text{U})$, ker hitreje izniči vse stare ocene.

Slika A.7 prikazuje deleže vrednosti, če spremenimo enoto časovne razlike na četrtletje (`Quarter`). Opazimo lahko podoben vzorec kot pri sliki A.6. Delež $s(\text{U})$ se v obeh matrikah zmanjša, medtem ko se deleža $s(\text{PD})$ in $s(\text{PT})$ povečata.

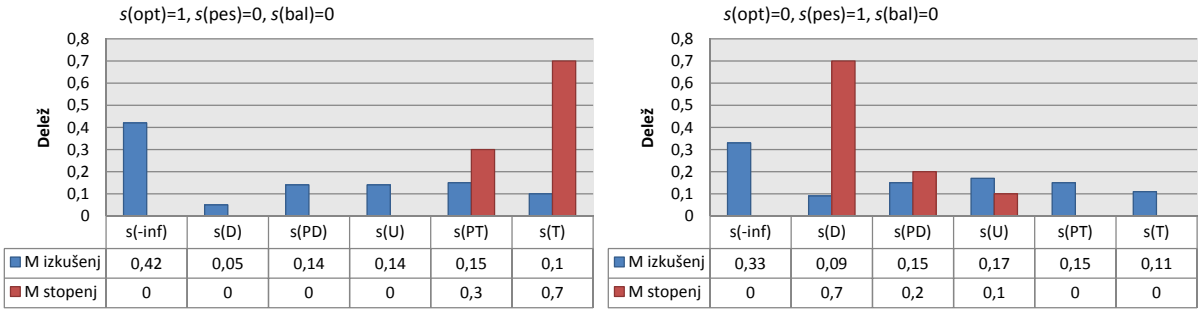


Slika A.7: Deleži vrednosti ($n = 10$, $iter = 1000$, $unit = \texttt{Quarter}$, $s(\text{opt}) = s(\text{pes}) = 0.5$, $s(\text{bal}) = 0$).

A.2.1.2 Primer 2: deleži operatorjev

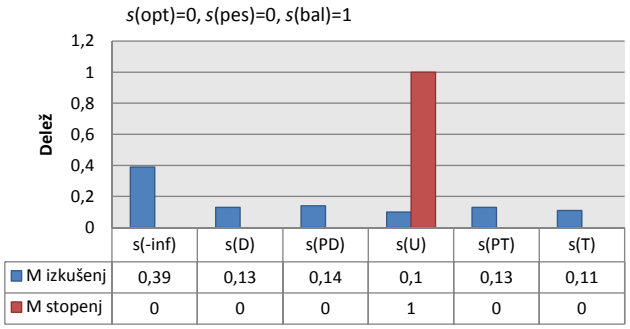
V tej simulaciji spreminjamo deleže operatorjev, ki jih uporabljajo agenti. Predpostavljamo, da večji delež optimistov poveča delež vrednosti (znotraj matrike $\mathbb{M}'^{trust}$) $\tau \in \mathcal{T} : 0 \leq f_{\mathcal{T}}(\tau) \leq 1$, večji delež pesimistov poveča delež $\tau \in \mathcal{T} : -1 \leq f_{\mathcal{T}}(\tau) \leq 0$, večji delež realistov pa najbolj poveča delež $s(\text{U})$.

Na začetku preverimo tri skrajne primere: (a) $s(\text{opt}) = 1$, (b) $s(\text{pes}) = 1$ in (c) $s(\text{bal}) = 1$. Slika A.8 prikazuje primer (a) – levi del in (b) – desni del. Naša predpostavka o povečanju deležev ustreznih vrednosti v matriki stopenj drži. Slika A.9 prikazuje primer (c), kjer nastopajo samo



Slika A.8: Deleži vrednosti za deleže operatorjev ($n = 10$, $iter = 100$, $\alpha = 0$).

realisti. Glede na definicijo operatorja $\sim$ (glej poglavje 3, definicija 3.1.4.3), prevlada delež $s(\text{U})$, kar zopet potrdi našo predpostavko.

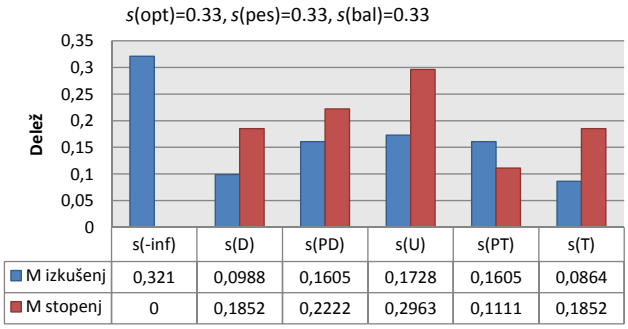


Slika A.9: Deleži vrednosti za realiste ($n = 10$, $iter = 100$, $\alpha = 0$).

Za enakomerno porazdelitev deležev kvalitativnih vrednosti predpostavljamo enak delež vseh treh operatorjev, ki je enak $\frac{1}{|\mathcal{O}|} = \frac{1}{3}$. V tem primeru znižamo število agentov na $n = 9$ (po trije agenti imajo svoj tip operatorja), število interakcij na 90. Slika A.10 prikazuje dobljene deleže kvalitativnih vrednosti; v matriki stopenj $\mathbb{M}'^{trust}$ opazimo bolj enakomerno porazdelitev deležev.

A.2.2 Simulacija: vektorji zaupanja

V tem razdelku predstavimo primere simulacij preko iterativnega računanja vektorjev zaupanja iz začetne matrike izkušenj $\mathbb{M}'(t_0, \varphi_0)$.



Slika A.10: Deleži vrednosti za realiste ($n = 9$, $iter = 90$, $\alpha = 0$).

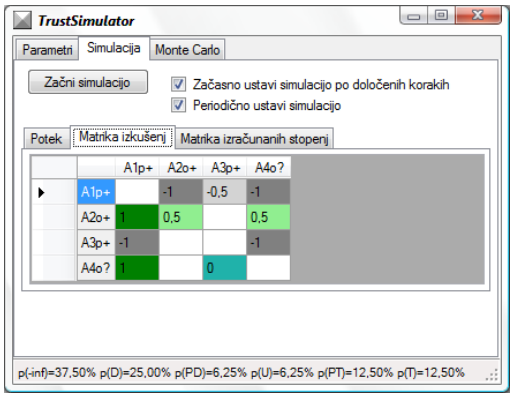
Za začetni primer vzamemo množico 4 agentov, kjer je polovica optimistov, polovica pa pesimistov. Spremljamo potek izračuna novih matrik prek vektorjev zaupanja. Nastavitve simulacije so sledeče (izpis iz dnevnika simulatorja):

```
======== SIMULATION PARAMETERS (initial) ========
** Random generator: Internal *
** Agents **
  num of agents: 4
  optimistic: 0,5; pessimistic: 0,5; balanced: 0,0
** Simulation **
  type: trust vectors computation
  num of iterations: 3
  steps to interrupt simulation: 1
======== END ========
Iter=1; [s(-inf)=37,50% s(D)=18,75% s(PD)=6,25% s(U)=12,50% s(PT)=18,75% s(T)=6,25%]
Iter=2; [s(-inf)=37,50% s(D)=25,00% s(PD)=6,25% s(U)=6,25% s(PT)=12,50% s(T)=12,50%]
Iter=3; [s(-inf)=37,50% s(D)=25,00% s(PD)=6,25% s(U)=6,25% s(PT)=12,50% s(T)=12,50%]

    | A1p | A2o | A3p | A4o         | A1p | A2o | A3p | A4o         | A1p | A2o | A3p | A4o
 ---------------------------     ---------------------------     ---------------------------
 A1p|     | 0,50|-0,50| 0,50     A1p|     |-1,00|-0,50|-1,00     A1p|     |-1,00|-0,50|-1,00
 A2o| 1,00|-1,00|     | 0,00 --> A2o| 1,00| 0,50|     | 0,50 --> A2o| 1,00| 0,50|     | 0,50
 A3p| 0,50|     |     |-1,00     A3p|-1,00|     |     |-1,00     A3p|-1,00|     |     |-1,00
 A4o|-1,00|     | 0,00|          A4o| 1,00|     | 0,00|          A4o| 1,00|     | 0,00|
```

Grafični potek simulacije prikazuje slika A.11, kjer vidimo izračunane matrike na vsakem koraku iteracije. Hitro lahko opazimo, da je matrika že v drugi iteraciji enaka prejšnji: $\mathbb{M}'^2(t_0, \varphi_0) = \mathbb{M}'^1(t_0, \varphi_0)$; vrednosti torej konvergirajo že v drugi iteraciji. To je razvidno tudi iz dnevnika simulatorja, kjer so deleži posameznih kvalitativnih stopenj znotraj matrike od 2. iteracije enaki. Slika A.12 prikazuje končno matriko po končanih iteracijah, slika A.13 pa vrednosti spremembe deležev kvalitativnih vrednosti iz iteracije 1 na iteracijo 2. Deleži skrajnih vrednosti $s(\text{D})$ in $s(\text{T})$ se povečajo, ostale zmanjšajo; z drugačnimi deležem operatorjev bi dobili drugačne deleže.

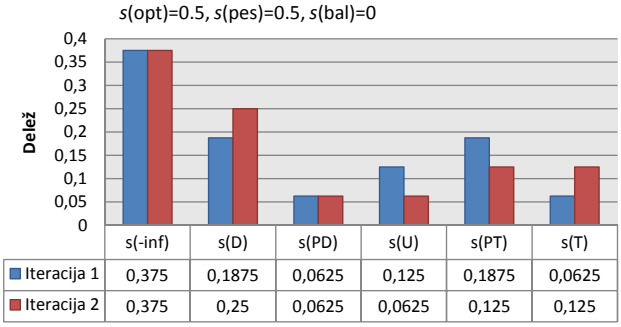
Slika A.11: Začetna in vmesne matrike ($n = 4$, $iter = 3$, $s(\text{opt}) = s(\text{pes}) = 0.5$).



Slika A.12: Končna matrika iz vektorjev zaupanja ($n = 4$, $iter = 3$, $s(\text{opt}) = s(\text{pes}) = 0.5$).

Simulator omogoča tudi spremembo deleža operatorjev in/ali vrednosti izkušenj (ocen) med izvajanjem simulacije. To spremeni vrednosti vektorjev zaupanja in posledično izračunanih matrik. Vendar sta za konvergenco vrednosti vedno potrebni dve iteraciji, četudi spremenimo vrednosti deležev operatorjev in/ali vrednosti znotraj matrike. Za primer vzemimo 9 agentov, kjer je enako število optimistov, pesimistov in realistov (t. i. $s(\text{opt}) = s(\text{pes}) = s(\text{bal}) = \frac{1}{3}$). Naj bo število iteracij 9, v vsaki tretji pa spremenimo delež operatorjev – povečujemo delež optimistov. Iz poteka simulacije vidimo, da po ažuriranju vrednosti deležev operatorjev vrednosti v matriki (in njihovi deleži) po dveh iteracijah vedno konvergirajo. To je razvidno iz sledečega poteka (izpis dnevnika simulatorja):

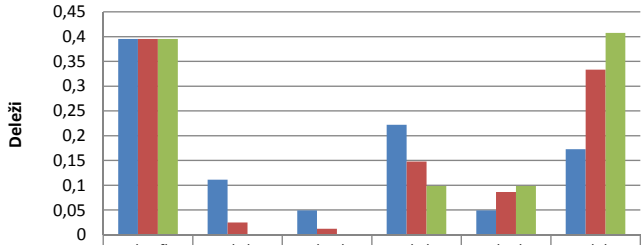
```
======== SIMULATION PARAMETERS (initial) ========
** Random generator: Internal *
** Agents **
  num of agents: 9
  optimistic: 0,33; pessimistic: 0,33; balanced: 0,34
```



Slika A.13: Sprememba deležev vrednosti med iteracijama ($n = 4$, $p(\text{opt}) = p(\text{pes}) = 0.5$).

```
** Simulation **
  type: trust vectors computation
  num of iterations: 9
  steps to interrupt simulation: 3
======== END ========
Iter=1;  [s(-inf)=39,51% s(D)=12,35% s(PD)=16,05% s(U)=6,17% s(PT)=16,05% s(T)=9,88%]
Iter=2;  [s(-inf)=39,51% s(D)=11,11% s(PD)=4,94% s(U)=22,22% s(PT)=4,94% s(T)=17,28%]
Iter=3;  [s(-inf)=39,51% s(D)=11,11% s(PD)=4,94% s(U)=22,22% s(PT)=4,94% s(T)=17,28%]
======== SIMULATION PARAMETERS (modified) ========
 interrupt at iteration: 3
** Agents **
  optimistic: 0,66; pessimistic: 0,1; balanced: 0,24
======== END ========
Iter=4;  [s(-inf)=39,51% s(D)=11,11% s(PD)=4,94% s(U)=22,22% s(PT)=4,94% s(T)=17,28%]
Iter=5;  [s(-inf)=39,51% s(D)=2,47% s(PD)=1,23% s(U)=14,81% s(PT)=8,64% s(T)=33,33%]
Iter=6;  [s(-inf)=39,51% s(D)=2,47% s(PD)=1,23% s(U)=14,81% s(PT)=8,64% s(T)=33,33%]
======== SIMULATION PARAMETERS (modified) ========
 interrupt at iteration: 6
** Agents **
  optimistic: 0,9; pessimistic: 0; balanced: 0,1
======== END ========
Iter=7;  [s(-inf)=39,51% s(D)=2,47% s(PD)=1,23% s(U)=14,81% s(PT)=8,64% s(T)=33,33%]
Iter=8;  [s(-inf)=39,51% s(D)=0,00% s(PD)=0,00% s(U)=9,88% s(PT)=9,88% s(T)=40,74%]
Iter=9;  [s(-inf)=39,51% s(D)=0,00% s(PD)=0,00% s(U)=9,88% s(PT)=9,88% s(T)=40,74%]
```

Slika A.14 prikazuje spremembo deležev kvalitativnih vrednosti ob konvergiranih vrednostih v matriki ob različnih deležih operatorjev; ker se delež optimistov povečuje $s(\text{opt}) = 0.33 \rightarrow 0.66 \rightarrow 0.9$, se posledično poveča tudi delež $s(\text{T})$. Ker vektorji zaupanja upoštevajo samo obstoječe relacije zaupanja, ostaja delež $s(-\infty)$ skozi izračunane matrike vedno enak. Pri $s(\text{opt}) = 0.9$ vidimo, da se deleža $s(\text{D})$ in $s(\text{PD})$ izničita.



	s(-inf)	s(D)	s(PD)	s(U)	s(PT)	s(T)
s(opt)=0.33 s(pes)=0.33 s(bal)=0.34	0,3951	0,1111	0,0494	0,2222	0,0494	0,1728
s(opt)=0.66 s(pes)=0.1 s(bal)=0.24	0,3951	0,0247	0,0123	0,1481	0,0864	0,3333
s(opt)=0.9 s(pes)=0 s(bal)=0.1	0,3951	0	0	0,0988	0,0988	0,4074

Slika A.14: Sprememba deležev vrednosti pri spremembah deležev operatorjev med simulacijo ($n = 9$).

A.3 Simulacije *Monte Carlo*

Znotraj tega razdelka so opisane simulacije z metodo *Monte Carlo* (MC) [182, 183]. Metoda predstavlja razred računskih algoritmov, ki temeljijo na ponavljanju računanja naključnih vzorcev za izračun končnih rezultatov. Metoda je odvisna od naključnih generatorjev števil in se uporablja predvsem na področjih, kjer je težko oz. nemogoče izračunati točen rezultat z determinističnim algoritmom (kjer nastopa npr. veliko število nedeterminističnih spremenljivk, kar velja tudi v našem primeru).

Zanima nas, pri katerih deležih operatorjev $s(\mathrm{op})$ na izbrani množici agentov $\mathcal{A}$, je delež izbrane kvalitativne vrednosti $\tau \in \mathcal{T}$ največji (maksimalen); to lahko zapišemo kot izraz (A.1), kjer $\overline{s}(\tau)$ predstavlja vrednost povprečnega deleža kvalitativne vrednosti $\tau \in \mathcal{T}$:

$$\boldsymbol{f}(s(\mathrm{op}_1), s(\mathrm{op}_2)) = \overline{s}(\tau) \text{ kjer } \underset{s(\mathrm{op}_1)+s(\mathrm{op}_2)\leq 1}{\arg\max} \overline{s}(\tau), \tag{A.1}$$
$$\mathrm{op}_1, \mathrm{op}_2 \in \mathcal{O} = \{\curlywedge, \curlyvee, \sim\}, \sum_{\mathrm{op}\in\mathcal{O}} s(\mathrm{op}) = 1.$$

Preslikava $\boldsymbol{f} : [0,1] \times [0,1] \mapsto [0,1]$ določa odvisnost povprečnega deleža $\overline{s}(\tau)$ od deležev operatorjev $s(\mathrm{op}_1)$ in $s(\mathrm{op}_2)$. Najti želimo delež operatorjev $s(\mathrm{op}_i), s(\mathrm{op}_j) \in [0,1]$, kjer je povprečni delež $\overline{s}(\tau)$ izbrane kvalitativne vrednosti največji (optimizacijski problem). Problem lahko rešimo z metodo MC, preko katere izračunamo povprečne vrednosti $\overline{s}(\tau)$ za vse možne pare deležev (od 0 do 1 z določenim korakom `step`) operatorjev $s(\mathrm{op}_1)$ in $s(\mathrm{op}_2)$, kjer $s(\mathrm{op}_1)+s(\mathrm{op}_2) \leq 1$ – torej določimo preslikavo $\boldsymbol{f}$. V osnovi je postopek izračuna sledeč:

(1) izberi število iteracij MC `iterations`, kvalitativno vrednost $\tau \in \mathcal{T}$, operatorja $\mathrm{op}_1, \mathrm{op}_2 \in \mathcal{O}$ ter korak `step`;

(2) za vse možne deleže $s(\mathrm{op}_1)$ in $s(\mathrm{op}_2)$ izvedi MC iteracijo, kjer v vsaki iteraciji $i \in [1, \texttt{iterations}]$ izvedeš:

(2.1) izvedi simulacijo z interakcijsko zgodovino z izbranimi vrednostmi parametrov: število agentov, število interakcij, trenutni delež $s(\mathrm{op}_1)$ in $s(\mathrm{op}_2)$, faktor α, izhodiščni datum t', interval za datumsko razliko;

(2.2) izračunaj matriko izkušenj $\mathbb{M}'(t', \varphi_0)$;

(2.3) iz matrike izkušenj izračunaj končno matriko stopenj zaupanja $\mathbb{M}'^{trust}$;

(2.4) izračunaj delež kvalitativne vrednosti $s_i(\tau)$ v matriki stopenj $\mathbb{M}'(t', \varphi_0)$;

(3) izračunaj povprečno vrednost deleža vrednosti: $\overline{s}(\tau) = \frac{\sum_{i=1}^{\texttt{iterations}} s_i(\tau)}{\texttt{iterations}}$;

(4) shrani povprečno vrednost pri izbranih deležih operatorjev v dvodimenzionalno polje: $\texttt{surface}[s(\mathrm{op}_1), s(\mathrm{op}_2)] = \overline{s}(\tau)$.

Opisani postopek izračuna prikazuje algoritem A.1 v programskem jeziku *C#*. Znotraj polja `surface` lahko hitro najdemo para $s(\mathrm{op}_1)$ in $s(\mathrm{op}_2)$ (potemtakem je $s(\mathrm{op}_3) = 1 - s(\mathrm{op}_1) - s(\mathrm{op}_2)$), kjer je $\overline{s}(\tau)$ maksimalen. Poleg tega lahko polje `surface` prikažemo kot 3-dimenzionalno mrežo vrednosti, kjer vidimo odvisnost vrednosti $\overline{s}(\tau)$ od $s(\mathrm{op}_1)$ in $s(\mathrm{op}_2)$. Časovno zahtevnost algoritma lahko določimo po sledečem razmisleku:

- spremenljivka `top` določa število iteracij za možne deleže $s(\mathrm{op}_i)$ oz. dimenzijo mreže (t. i. matrike) `surface`; dimenzija je pravzaprav `top` + 1, ker upoštevamo tudi deleže z vrednostjo 0;
- potrebno je izračunati samo vrednosti $\overline{s}(\tau)$, kjer $s(\mathrm{op}_1) + s(\mathrm{op}_2) \leq 1$; torej je potrebno pregledati samo zgornjo polovico mreže (matrike) `surface`, kar je enako $\frac{(\texttt{top}+1)(\texttt{top}+2)}{2}$ iteracij;
- v vsaki iteraciji je potrebno izvesti simulacijo z interakcijsko zgodovino z izračunom matrik in deležev, kar predstavlja zahtevnost $O(n^2 \cdot k)$, kjer n predstavlja št. agentov, k pa število interakcij (`INTERACTIONS`);
- potemtakem je skupna časovna zahtevnost metode MC enaka $O(\frac{(\texttt{top}+1)(\texttt{top}+2)}{2} \cdot n^2 \cdot k) = O(top^2 \cdot n^2 \cdot k)$.

A.3.1 Primeri simulacij

V tem razdelku so prikazani rezultati simulacij MC pri izbranih vrednostih parametrov simulacije z interakcijsko zgodovino (število agentov, število interakcij, faktor α). Izračunane mreže `surface` prikažemo grafično s pomočjo programa *Gnuplot*[7]. V večini primerov uporabljamo sledeče vrednosti simulacije MC: `iterations` = 50 (št. iteracij), `step` = 0.05 (korak povečanja deležev $s(\mathrm{op}_1)$ in $s(\mathrm{op}_2)$). Potek simulacije MC je možno spremljati preko dnevnika simulatorja *TrustSimulator*, ki na koncu najde tudi maksimalno povprečno vrednost $\overline{s}(\tau)$ izbrane kvalitativne vrednosti $\tau \in \mathcal{T}$:

```
======== MONTE CARLO SIMULATION PARAMETERS ========
  Random generator: Internal
  Monte Carlo iterations: 50
  Operator 1: Optimistic
```

[7] Prosto dosegljiv na `http://www.gnuplot.info/`.

```
decimal[,] MC(QualitativeValue τ, Operator op1, Operator op2, int iterations, decimal step) {
  // naključni generator
  Randomizer rnd = new Randomizer(SEED);
  // število iteracij za podani korak 'step'
  int top = (int) (1.0 / step);
  // spremenljivka za izračunan delež p(τ)
  decimal area = 0.0;
  // matrika (ploskev) za izračunane vrednosti p(τ)
  decimal[,] surface = new decimal[top+1,top+1];

  // gremo čez vse možne deleže op1 in op2, omejitev: p(op1) + p(op2) <= 1
  for (int p1 = 0; p1 <= iTop; p1++) {
    for (int p2 = 0; p2 <= iTop; p2++) {
      if (p1 + p2 <= top) {
        area = 0.0;
        // MC iteracije
        for (int iter = 0; iter < iterations; iter++) {
          // nastavi deleže operatorjev na agentih
          Agents agents = SetupAgents(p1*step, p2*step);
          // simulacija z interakcijsko zgodovino
          History history = new InteractionHistory();
          int count = 0;
          while (count < INTERACTIONS) {
            // naključni izbor izvornega in ciljnega agenta
            Agent source = Agents[rnd.Next(Agents.Count)];
            Agent target = Agents[rnd.Next(Agents.Count)];
            // interakcija
            Interaction interaction = MakeInteraction(source, target);
            history.Add(interaction);

            count ++
          } // while

          // matrika izkušenj 𝕄' za izbrane parametre
          ExperienceMatrix em = Engine.CalcMatrix(history, DATE, INTERVAL, CONTEXT, ALPHA);

          // matrika stopenj 𝕄^trust
          CalculatedMatrix cm = Engine.CalcMatrix(em);

          // izračun deleža p(t) v matriki stopenj
          area = area + CalculateRatio(cm, τ);

        } // for MC iter

        // izračun povprečne vrednosti p(τ)
        surface[p1,p2] = area / iterations;

      } // if p1 + p2
    } // for p2
  } // for p1
  return surface;
}
```

Algoritem A.1: Metoda MC za izračun povprečne vrednosti deleža kvalitativne vrednosti.

```
  Operator 2: Pessimistic
  Increment step for s(op): 0,05
** Interaction simulation params **
  num of inteactions (iterations): 100
  num of agents: 10
  alpha: 0,00
  interactions threshold: NONE
  interaction assessment strategy: Random
  num of last interactions in history: -1
  from date: 12.3.2008 16:06:49
  to date: 12.3.2009 16:06:49
  date difference interval: Day
======== END ========
s(Optimistic)=0, s(Pessimistic)=0; mean_ratio(Distrusted)=0
s(Optimistic)=0, s(Pessimistic)=0,05; mean_ratio(Distrusted)=0
....
s(Optimistic)=1, s(Pessimistic)=0; mean_ratio(Distrusted)=0
 *** max mean area ***
 s(Optimistic)=0,00, s(Pessimistic)=1,00, MaxMeanArea=0,584
```

A.3.1.1 Primer 1: $\overline{s}(\mathbf{d})$, $\overline{s}(\mathbf{pd})$

V tej simulaciji poiščemo deleže operatorjev, pri katerem je povprečna vrednost $\overline{s}(\text{D})$ oz. $\overline{s}(\text{PD})$ največja. Že v simulacijah z interakcijsko zgodovino (glej razdelek A.2.1.2) smo videli, da večji delež pesimistov poveča delež $s(\text{PD})$ in $s(\text{D})$ v matriki $\mathbb{M}'^{trust}$; enak rezultat torej predpostavljamo tudi tukaj. Za ilustracijo izvedemo simulacijo pri različnih vrednostih faktorja α (enota časovne razlike je *mesec* (`Month`)); rezultate prikazuje tabela A.2. Opazimo, da pri večji

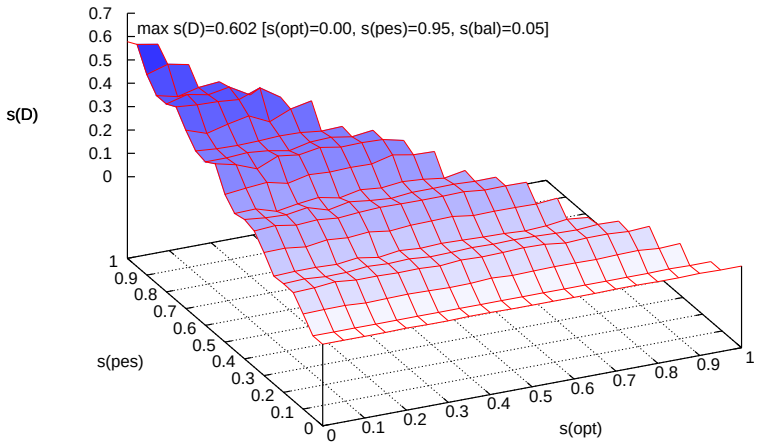
Faktor α	s(opt)	s(pes)	s(bal)	$\overline{s}$(PD)	$\overline{s}$(D)
0.00	0.00	0.95	0.05	0.352	0.602
0.01	0.00	1.00	0.00		0.576
	0.00	0.95	0.05	0.364	
0.10	0.00	0.95	0.05		0.168
	0.00	1.00	0.00	0.71	

Tabela A.2: Vpliv faktorja α na $\overline{s}(\text{PD})$ in $\overline{s}(\text{D})$ ($n = 10, \texttt{iterations} = 50, \texttt{step} = 0.05$).

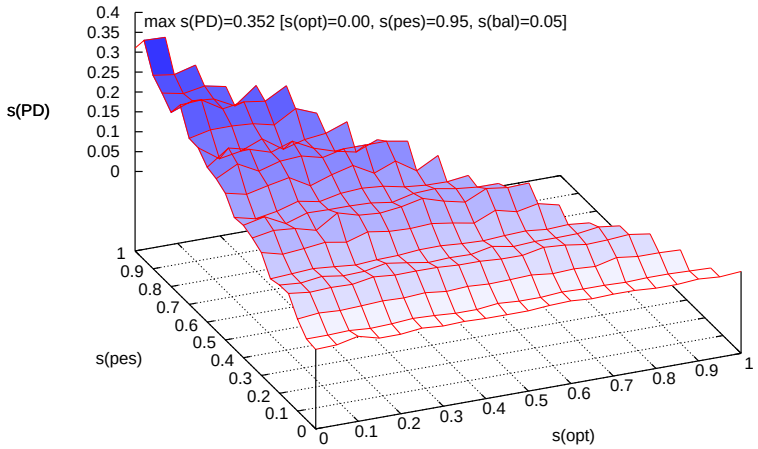
vrednosti α delež $\overline{s}(\text{PD})$ narašča, delež $\overline{s}(\text{D})$ pa pada; opazimo tudi, da je največji povprečen delež za obe vrednosti D in PD dosežen pri prevladi pesimistov. Slika A.15 oz. A.16 prikazuje mrežo povprečnih vrednosti $\overline{s}(\text{D})$ oz. $\overline{s}(\text{PD})$. Ob povečevanju deleža pesimistov sorazmerno narašča vrednost $\overline{s}(\text{D})$ in $\overline{s}(\text{PD})$; torej tako, kot smo predpostavili na začetku.

A.3.1.2 Primer 2: $\overline{s}(\mathbf{u})$

V tej simulaciji poiščemo delež optimistov, pesimistov in realistov, kjer je povprečni delež $\overline{s}(\text{U})$ največji. Predpostavljamo, da je vrednost največja, če prevladujejo realisti (ugotovitev



Slika A.15: Mreža vrednosti $\overline{s}(\text{D})$ ($n = 10$, $\alpha = 0.00$).



Slika A.16: Mreža vrednosti $\overline{s}(\text{PD})$ ($n = 10$, $\alpha = 0.00$).

v razdelku A.2.1.2). Simulacijo izvedemo za različne vrednosti faktorja α (enota časovne razlike je *mesec* (`Month`)); rezultate prikazuje tabela A.3. Vidimo, da večja vrednost α veča delež

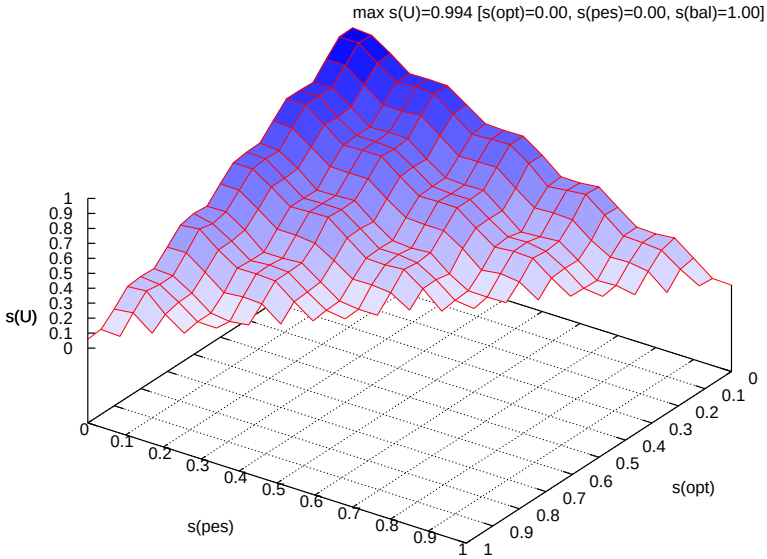
Faktor α	$s(\text{opt})$	$s(\text{pes})$	$s(\text{bal})$	$\overline{s}(\text{U})$
0.00	0.00	0.00	1.00	0.994
0.01	0.05	0.05	0.90	0.998
0.10	0.00	0.05	0.95	1.000

Tabela A.3: Vpliv faktorja α na $\overline{s}(\text{U})$ ($n = 10, \texttt{iterations} = 50, \texttt{step} = 0.05$).

$\overline{s}(\text{U})$ (stare ocene v interakcijski zgodovini se hitreje izničijo, zato konvergenca k vrednosti $f_\Omega(\text{U}) = 0$). Slika A.17 prikazuje mrežo povprečnih vrednosti $\overline{s}(\text{U})$, kjer opazimo, da z večanjem optimistov/pesimistov sorazmerno pada vrednost $\overline{s}(\text{U})$.

A.3.1.3 Primer 3: $\overline{s}(\text{pt})$, $\overline{s}(\text{t})$

Skozi to simulacijo skušamo najti deleže operatorjev, kjer je povprečna vrednost $\overline{s}(\text{PT})$ oz. $\overline{s}(\text{T})$ največja. Že v razdelku A.2.1.2 smo videli, da večji delež optimistov poveča delež kvalitativnih



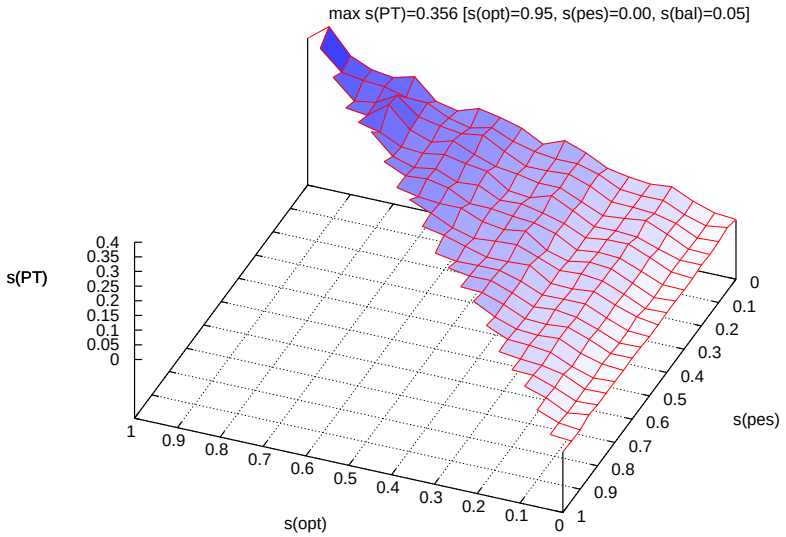
Slika A.17: Mreža vrednosti $\overline{s}(\text{U})$ ($n = 10$, $\alpha = 0.00$).

vrednosti $\text{PT}, \text{T} \in \mathcal{T}$ v matriki stopenj $\mathbb{M}^{ttrust}$; torej s simulacijo MC moramo dobiti podobne rezultate. Simulacijo izvedemo na različnih vrednostih α z enoto časovne razlike *mesec* (`Month`); rezultate prikazuje tabela A.4. Vidimo, da večja vrednost α povečuje povprečni delež $\overline{s}(\text{PT})$

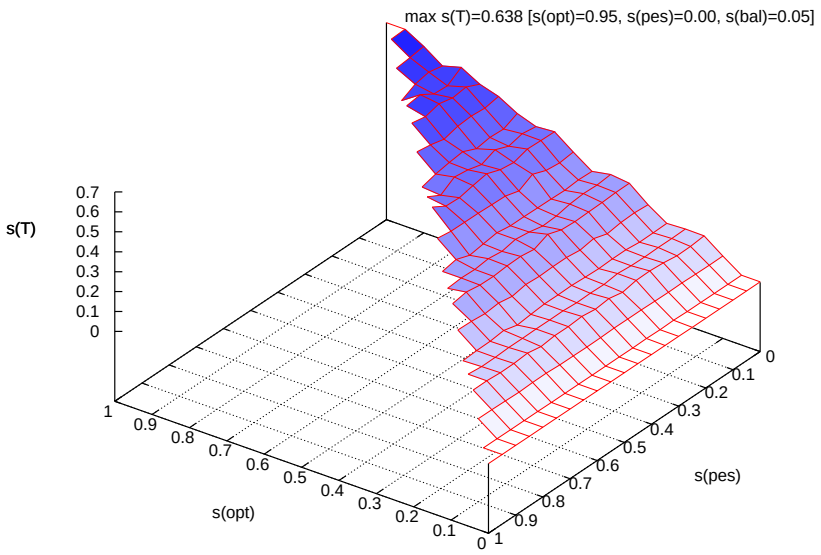
Faktor α	s(opt)	s(pes)	s(bal)	$\overline{s}(\text{PT})$	$\overline{s}(\text{T})$
0.00	0.95	0.00	0.05	0.356	0.638
0.01	0.95	0.00	0.05	0.380	0.604
0.10	0.95	0.00	0.05	0.710	0.154

Tabela A.4: Vpliv faktorja α na $\overline{s}(\text{PT})$ in $\overline{s}(\text{T})$ ($n = 10, \texttt{iterations} = 50, \texttt{step} = 0.05$).

in zmanjšuje $\overline{s}(\text{T})$ (stare ocene v interakcijski zgodovini se hitreje izničijo, zato vrednosti konvergirajo k vrednosti $f_\Omega(\text{U}) = 0$; ker je vrednost $f_\Omega(\text{PT}) = 0.5$ bližja vrednosti 0 kot $f_\Omega(\text{T}) = 1$, se delež $\overline{s}(\text{PT})$ povečuje). Slika A.18 oz. A.19 prikazuje mrežo povprečnih vrednosti $\overline{s}(\text{PT})$ oz. $\overline{s}(\text{T})$. Vidimo, da je pogoj za večjo vrednost $\overline{s}(\text{PT})$ oz. $\overline{s}(\text{T})$ čimvečji delež optimistov in čimmanjši delež pesimistov.



Slika A.18: Mreža vrednosti $\overline{s}(\text{PT})$ ($n = 10$, $\alpha = 0.00$).



Slika A.19: Mreža vrednosti $\overline{s}(\text{T})$ ($n = 10$, $\alpha = 0.00$).

Dodatek B

Pregled BPEL

Jezik BPEL temelji na sintaksi XML; poslovni proces je zapisan v obliki dokumenta XML, ki se lahko izvede na ustreznem orkestracijskem strežniku (npr. Oracle BPEL Manager, Microsoft BizTalk, ActiveBPEL). Strežnik prebere dokument in proži potrebne spletne storitve v ustreznem zaporedju (sekvenčno ali paralelno). Poslovni proces je na zunaj viden kot nova spletna storitev, ki se lahko proži kot vsaka druga spletna storitev. Jezik BPEL je do neke mere pravi visokonivojski programski jezik, vsebuje konstrukte za pogojno proženje storitev, zanke, spremenljivke, obvladovanje napak in dogodkov – vse to omogoča konstrukcijo kompleksnih poslovnih procesov.

Proces BPEL je lahko *sinhroni* (uporaba ene ali več operacij tipa zahteva-odgovor (angl. request-response)) ali *asinhroni* (uporaba ene ali več enosmernih operacij (angl. one-way)). Sinhroni proces BPEL blokira odjemalca, vse dokler operacija ni končana in vrne rezultat. Asinhroni proces ne blokira odjemalca, ampak uporablja *povratni klic* (angl. callback) ob končanju operacije.

B.1 Struktura

Proces BPEL je v osnovi sestavljen iz: (i) seznama udeleženih partnerjev v procesu in njihovih vlog, (ii) mehanizmov za delo z dogodki in izjemami, (iii) poslovnega procesa z aktivnostmi. Omenjeno strukturo prikazuje sledeči izsek sheme XML [97, 136]:

```
<element name="process" type="bpws:tProcess"/>
<complexType name="tProcess">
  <complexContent>
    <extension base="bpws:tExtensibleElements">
      <sequence>
        <element name="partnerLinks" type="bpws:tPartnerLinks" minOccurs="0"/>
        <element name="partners" type="bpws:tPartners" minOccurs="0"/>
        <element name="variables" type="bpws:tVariables" minOccurs="0"/>
        <element name="correlationSets" type="bpws:tCorrelationSets" minOccurs="0"/>
        <element name="faultHandlers" type="bpws:tFaultHandlers" minOccurs="0"/>
        <element name="compensationHandler" type="bpws:tCompensationHandler" minOccurs="0"/>
        <element name="eventHandlers" type="bpws:tEventHandlers" minOccurs="0"/>
```

```
        <group ref="bpws:activity"/>
      </sequence>
      <attribute name="name" type="NCName" use="required"/>
      <attribute name="targetNamespace" type="anyURI" use="required"/>
      <attribute name="queryLanguage" type="anyURI" />
      <attribute name="expressionLanguage" type="anyURI"/>
      <attribute name="suppressJoinFailure" type="bpws:tBoolean" default="no"/>
      <attribute name="enableInstanceCompensation" type="bpws:tBoolean" default="no"/>
      <attribute name="abstractProcess" type="bpws:tBoolean" default="no"/>
    </extension>
  </complexContent>
</complexType>
```

Proces BPEL je sestavljen iz posameznih korakov, ki se imenujejo aktivnosti. Obstajajo *osnovne (atomarne)* in *strukturirane* aktivnosti. Sledeči izsek sheme XML vsebuje seznam osnovnih aktivnosti [125]:

```
<group name="activity">
  <choice>
    <element name="empty" type="bpws:tEmpty"/>
    <element name="invoke" type="bpws:tInvoke"/>
    <element name="receive" type="bpws:tReceive"/>
    <element name="reply" type="bpws:tReply"/>
    <element name="assign" type="bpws:tAssign"/>
    <element name="wait" type="bpws:tWait"/>
    <element name="throw" type="bpws:tThrow"/>
    <element name="rethrow" type="bpws:tRethrow"/>
    <element name="terminateexit" type="bpws:tTerminate"/>
    <element name="flow" type="bpws:tFlow"/>
    <element name="switch" type="bpws:tSwitch"/>
    <element name="while" type="bpws:tWhile"/>
    <element name="sequence" type="bpws:tSequence"/>
    <element name="pick" type="bpws:tPick"/>
    <element name="scope" type="bpws:tScope"/>
    <element name="compensate" type="bpws:tCompensate"/>
  </choice>
</group>
```

Sledeči razdelki okvirno opisujejo osnovne in strukturirane aktivnosti, mehanizme za delo z dogodki, izjemami in kompenzacijske mehanizme. Določene podrobnosti so izpuščene; elementi so zaradi bolj kompaktne predstavitve podani v BNF (Backus-Naur) obliki namesto shem XML. Sledeči simboli pomenijo različno število pojavitev elementa `elem`: `elem?` (nič ali enkrat), `elem+` (enkrat ali več) in `elem*` (nič ali večkrat).

B.1.1 Osnovne aktivnosti

Znotraj tega razdelka so opisane nekatere osnovne aktivnosti, ki se uporabljajo za izmenjavo sporočil. Osnovne aktivnosti so: *invoke*, *receive*, *reply*, *throw*, *compensate*, *empty*, *exit*, *assign* in *wait*.

B.1.1.1 Invoke

Aktivnosti predstavlja proženje operacije na spletni storitvi, ki je lahko sinhrona ali asinhrona. Aktivnost je definirana kot:

```
<invoke partnerLink="ncname" portType="qname" operation="cname" inputVariable="ncname"?
        outputVariable="ncname"? standard-attributes>
</invoke>
```

Atributi `partnerLink`, `portType` in `operation` določajo ustrezno operacijo na ustreznem partnerju, končno točko prožene operacije in transportni protokol za zahteve SOAP (HTTP, SMTP, ...); `inputVariable` in `outputVariable` določajo spremenljivke kot vhod oz. izhod storitve.

B.1.1.2 Receive

Aktivnost čaka na zahtevo ustreznega partnerja in njegovo vlogo, tip kanala, operacijo in morebitne spremenljivke. Sintaksa je sledeča:

```
<receive partnerLink="ncname" portType="qname" operation="ncname" variable="ncname"?
         createInstance="yes|no"? standard-attributes>
  standard-elements
</receive>
```

B.1.1.3 Reply

Aktivnost rabi za generiranje odgovora, ki je posledica izvedbe `receive` aktivnosti. Sintaksa je sledeča:

```
<reply partnerLink="ncname" portType="qname" operation="ncname" variable="ncname"?
      faultName="qname"? standard-attributes>
  standard-elements
</reply>
```

B.1.1.4 Throw

Aktivnost eksplicitno signalizira interno napako ali izjemo. Določeno mora biti enolično ime napake/izjeme in spremenljivka, ki nosi informacijo o napaki. Signalizirane napake/izjeme ujame mehanizem za napake. Sintaksa je sledeča:

```
<throw faultName="qname" faultVariable="ncname"? standard-attributes>
  standard-elements
</throw>
```

B.1.1.5 Compensate

Aktivnost proži mehanizem za kompenzacijo (razveljavitev že izvedenih akcij), ki se nanaša na določen *blok* oz. *območju* (angl. scope). Mehanizem za kompenzacijo je lahko prožen samo s strani mehanizma za napake ali mehanizma za kompenzacijo na zunanjem območju. Sintaksa je sledeča:

```
<compensate scope="ncname"? standard-attributes>
  standard-elements
</compensate>
```

B.1.2 Strukturirane aktivnosti

Strukturirane aktivnosti BPEL omogočajo združevanje osnovnih aktivnosti na različne načine: (i) zaporedno združevanje in vejanje (`sequence` in `switch`), (ii) paralelno združevanje in sinhronizacijo (`flow`) in (iii) nedeterministična izbira (`pick`).

B.1.2.1 Sequence

Aktivnost omogoča zaporedno (sekvenčno) kompozicijo ostalih aktivnosti, ki se izvedejo v istem vrstnem redu kot so naštete. Sekvenca se konča, ko se konča zadnja aktivnost v zaporedju. Sintaksa je sledeča:

```
<sequence standard-attributes>
  standard-elementsactivity+
</sequence>
```

B.1.2.2 Flow

Aktivnost omogoča vzporedno (paralelno) izvajanje osnovnih aktivnosti; zagotavlja sočasnost in sinhronizacijo. Aktivnost se konča, ko se končajo vse vsebovane vzporedne aktivnosti. Sintaksa je sledeča:

```
<flow standard-attributes>
  standard-elements
  <links>?
    <link name="ncname">+
  </links>
  activity+
</flow>
```

B.1.2.3 Links

Ta konstrukt omogoča vzpostavitev ustreznih omejitev znotraj aktivnosti `flow`. Kontrolna povezava (t. i. `link`) določa ustrezen vrstni red izvajanja paralelnih aktivnosti. Preko uporabe kontrolnih povezav je aktivnost `flow` lahko predstavljena kot graf – vozlišča so aktivnosti, povezave pa so kontrolne omejitve, ki določajo odvisnost med aktivnostmi (vrstni red izvedbe).

Kontrolne povezave rabijo za sinhronizacijo odvisnosti med aktivnostmi. Element `link` ima svoje ime in določa povezavo med dvema aktivnostima (izvorno in ciljno). Izvorna aktivnost lahko določa tudi logični pogoj za prehod (angl. transition condition), ki je sestavljen iz procesnih spremenljivk. Kontrolna povezava med aktivnostima A in B pomeni, da se B ne more začeti preden se je A bodisi končala ali pa je bila izpuščena. B se lahko izvrši samo v primeru, če je povezan *združitveni pogoj* (angl. join condition) resničen, drugače se B izpusti. Združitveni pogoj je izražen v smislu *žetonov*, ki se prenašajo po kontrolnih povezavah. Žeton je lahko pozitiven (vrednost *true*) ali negativen (vrednost *false*). Aktivnost X odda pozitiven žeton na izhodni kontrolni povezavi L samo v primeru, ko se je X izvršila in je ovrednoten pogoj za prehod pridružen L resničen. Kontrolne povezave lahko prečkajo meje strukturiranih aktivnosti, vendar ne morejo prečkati območja, ki ga določa `scope`.

B.1.2.4 Switch

Aktivnost sestavlja zaporedje ene ali več pogojnih vej, ki so definirane preko elementa `case`, katerim lahko sledi opcijski element `otherwise`. Veje se vrednotijo v istem vrstnem redu, kot so naštete. Pogoj predstavlja logični izraz procesnih spremenljivk. Prva veja, kjer je pogoj resničen, se izvrši. Sintaksa je sledeča:

```
<switch standard-attributes>
  standard-elements
  <case condition="bool-expr">+
    activity
  </case>
  <otherwise>?
    activity
  </otherwise>
</switch>
```

Velja omeniti, da je v verziji BPEL 2.0 aktivnost `switch` preimenovana v `if`; pomen pa je enak:

```
<if standard-attributes>
  standard-elements
  <condition expressionLanguage="anyURI"?>bool-expr</condition>
    activity
  <elseif>*
    <condition expressionLanguage="anyURI"?>bool-expr</condition>
    activity
  </elseif>
  <else>?
    activity
  </else>
</if>
```

B.1.2.5 Pick

Aktivnost rabi za nedeterministično izvedbo ene od več poti v odvisnosti od zunanjega dogodka. Aktivnost `pick` vsebuje množico vej v obliki *dogodek-aktivnost*, kjer se izbere samo ena veja

glede na pridruženi izvedeni dogodek. Dogodki so lahko *sporočila*, ki jih obravnava aktivnost `onMessage` ali *alarmni dogodki*, ki jih obravnava aktivnost `onAlarm`. Če se zgodi več dogodkov, velja tisti dogodek, ki se je zgodil prvi. Aktivnost `pick` se konča, ko se končajo aktivnosti znotraj izbrane veje. Sintaksa je sledeča:

```
<pick createInstance="yes|no"? standard-attributes>
  standard-elements
  <onMessage partnerLink="ncname" portType="qname" operation="ncname" variable="ncname"?>+
    <correlations>?
      <correlation set="ncname" initiate="yes|no"?>+
    </correlations>
    activity
  </onMessage>
  <onAlarm (for="duration-expr" | until="deadline-expr")>*
    activity
  </onAlarm>
</pick>
```

B.1.2.6 Scope

Bloki oz. območja so hierarhično organizirani deli poslovnega procesa. Območje določa kontekst obnašanja za vsako vsebovano aktivnost. Vsako območje ima primarno aktivnost (osnovna ali strukturirana), ki določa obnašanje ob normalnem izvajanju. V primeru, da je aktivnost strukturirana, lahko vsebuje poljubno število vgnezdenih območij poljubne globine.

Poleg aktivnosti imajo lahko tudi območja definirane lokalne korelacijske množice, mehanizme za kompenzacijo, napake in dogodke. Definicija spremenljivk znotraj območja je veljavna vse dokler je območje aktivno. Spremenljivke so vidne samo v območju, kjer so definirane ter v podrejenih območjih. Območje postane aktivno, ko se podrejene aktivnosti lahko izvršijo; območje se konča, ko se vse podrejene aktivnosti končajo. Potrebno je izpostaviti mehanizem za obravnavo izjemnih situacij (napak) in mehanizem za kompenzacijo. Mehanizem za napake se uporablja ob izvajanju aktivnosti znotraj območja, mehanizem za kompenzacijo pa se uporablja, ko se območje uspešno konča. Mehanizem za obravnavo napak, ki se navezuje na območje, obravnava napake znotraj aktivnosti (tudi vgnezdenih). Napake, ki jih mehanizem ne prestreže, se propagirajo na hierarhično višje območje. Sintaksa je sledeča:

```
<scope variableAccessSerializable="yes|no" standard-attributes>
  standard-elements
  <variables>?
    ...
  </variables>
  <correlationSets>?
    ...
  </correlationSets>
  <faultHandlers>?
    ...
  </faultHandlers>
  <compensationHandler>?
```

```
    ...
  </compensationHandler>
  <eventHandlers>?
    ...
  </eventHandlers>
  activity
</scope>
```

B.1.2.7 CompensationHandler

Mehanizem rabi za razveljavitev uspešno končanega območja (t. i. `scope`). Mehanizem lahko dostopa do spremenljivk, ki imajo stanje enako, kot je bilo ob končanju območja; lahko tudi proži druge storitve. Mehanizem za kompenzacijo se lahko proži preko aktivnosti `compensate`, ki ima za argument ime območja, na katerega se mehanizem nanaša. Sintaksa je sledeča:

```
<compensationHandler>?
  activity
</compensationHandler>
```

B.1.2.8 FaultHandler

Ta mehanizem je odgovoren za notranje ali zunanje napake, ki se zgodijo med izvajanjem območja. Element `catch` obravnava napako z izbranim imenom. Element `catchAll` lahko obravnava katerokoli napako. Aktivnost `throw` signalizira napako in prekine normalno izvajanje območja z aktiviranjem ustreznega mehanizma za napake. Če le-ta ni definiran, se aktivira privzeti mehanizem. Ta kompenzira vsa podrejena območja, napaka pa se propagira na hierarhično višje območje. Območje, ki signalizira napako, se ne konča uspešno, njegov mehanizem za kompenzacijo se ne sme izvršiti. Sintaksa je sledeča:

```
<faultHandlers>?
  <!-- there must be at least one fault handler or default -->
  <catch faultName="qname"? faultVariable="ncname"?>*
    activity
  </catch>
  <catchAll>?
    activity
  </catchAll>
</faultHandlers>
```

B.1.2.9 EventHandler

Mehanizem za dogodke določa pravila v obliki *dogodek-akcija* v povezavi z izbranim območjem. Mehanizem se lahko izvaja vzporedno z glavno aktivnostjo znotraj območja. Ko se zgodi dogodek, ki je naveden znotraj mehanizma (naveden v elementu `onAlarm` ali `onMessage`), se izvrši pripadajoča aktivnost mehanizma, medtem ko se glavna aktivnost znotraj območja izvaja naprej. Sintaksa mehanizma za dogodke je sledeča:

```
<eventHandlers>?
  <!-- there must be at least one onMessage or onAlarm handler -->
  <onMessage partnerLink="ncname" portType="qname" operation="ncname" variable="ncname"?>*
    <correlations>?
      <correlation set="ncname" initiate="yes|no">+
    </correlations>
    activity
  </onMessage>
  <onAlarm for="duration-expr"? until="deadline-expr"?>*
    activity
  </onAlarm>
</eventHandlers>
```

Dodatek C

Pregled WS-CDL

WS-CDL je jezik za opis koreografije spletnih storitev. Koreografija je način sodelovanja, ki temelji na interakcijah (izmenjavi sporočil) vidnih iz globalne točke, kjer so vse udeležene storitve obravnavane na enak način. Vse udeležene strani znotraj koreografije se morajo zavedati poslovnega procesa, operacij za izvršitev, sporočil za izmenjavo in časovnega poteka izmenjave.

WS-CDL temelji na jeziku XML in omogoča definicijo vmesnikov (t. i. pogodb) med udeleženimi storitvami, ki opisujejo vidno obnašanje preko izmenjave sporočil. WS-CDL se lahko uporablja v navezi z BPEL pri modeliranju in implementaciji poslovnih procesov. V sledečih razdelkih je podana osnovna struktura jezika WS-CDL. Bolj specifične podrobnosti so podane v [108, 109].

C.1 Struktura

Opis koreografije preko WS-CDL je vsebovan v *paketu* (angl. choreography package), ki je glavni vsebnik za zbirko aktivnosti. Te aktivnosti izvajajo udeležene spletne storitve. Obstajajo trije tipi aktivnosti znotraj WS-CDL: (i) *aktivnosti za kontrolo toka* (angl. control flow), (ii) *aktivnosti delovne enote* (angl. workunit) in (iii) *osnovne aktivnosti*. Sintaksa korenskega paketa koreografije je sledeča (namesto shem XSD se uporablja bolj kompaktna predstavitev v obliki BNF – enako kot v dodatku B):

```
<package name="ncname" author="xsd:string"? version="xsd:string"? targetNamespace="uri"
   xmlns="http://www.w3.org/2004/12/ws-chor/cdl">
   informationType*
   token*
   tokenLocator*
   roleType*
   relationshipType*
   participantType*
   channelType*
   Choreography-Notation*
</package>
```

C.1.1 Aktivnosti za kontrolo toka (control-flow)

Te aktivnosti so blokovno strukturirane (vsebujejo pod-aktivnosti) in določajo vrstni red aktivnosti znotraj koreografije. V ta sklop spadajo sledeče aktivnosti: `sequence`, `parallel` in `choice`.

Aktivnost `sequence` določa eno ali več aktivnosti, ki se izvedejo v sekvenčnem zaporedju; aktivnost `parallel` pa določa aktivnosti, ki se lahko izvedejo vzporedno. Aktivnost `choice` omogoča izbiro iz množice aktivnosti. Izbira je dveh tipov: (a) *podatkovno vodena* (angl. data-driven), ki temelji na logičnem pogoju podatkovnih spremenljivk in (b) *dogodkovno vodena* (angl. event-driven), ki je odvisna od pojavitve enega od možnih dogodkov. Pojavitev dogodka je posledica interakcije ali akcije znotraj koreografije, kar se odrazi v spremembi vrednosti nekaterih spremenljivk.

C.1.2 Aktivnost delovne enote (workunit)

Ta vrsta aktivnosti predpisuje omejitve, ki jih je potrebno izpolniti znotraj predpisane koreografije. Gre za pogojne in ponavljajoče izvajanje aktivnosti. Aktivnost delovne enote `workunit` ima več delov, ima referenco na priloženo aktivnost ter pogoje tipa `guard`, `block`, in `repeat`, ki so logični izrazi.

Pogoja `guard` in `repeat` določata, ali se aktivnost izvede enkrat ali večkrat. Najprej se preveri veljavnost `guard` pogoja; če je veljaven, se izvede priložena aktivnost; pogoj `repeat` pa določa, ali se ta aktivnost lahko izvede večkrat.

Pogoj `block` določa, ali naj pogoja `guard` in `repeat` čakata na razpoložljivost spremenljivk pred logičnim vrednotenjem. Aktivnost `workunit` se uspešno konča, ko se uspešno končajo vse priložene aktivnosti. Sintaksa je sledeča:

```
<workunit name="ncname"
      guard="xsd:boolean XPath-expression"?
      repeat="xsd:boolean XPath-expression"?
      block="true|false"? >
      Activity-Notation
</workunit>
```

C.1.3 Osnovne aktivnosti

Ta tip aktivnosti predstavlja akcije tipa *zahteva-odgovor* za lokalne aktivnosti in proženje zunanjih storitev. Obstaja pet vrst osnovnih aktivnosti: `interaction`, `perform`, `assign`, `noaction` in `silentaction`.

Aktivnosti `noaction` in `silentaction` določata točke znotraj koreografije, kjer izvedena akcija udeleženca oz. vloge ne vpliva na sam proces koreografije. Aktivnost `assign` omogoča prirejanje vrednosti spremenljivkam. Aktivnost `perform` rabi za klic druge koreografije znotraj konteksta trenutno izvajajoče koreografije. Klicana koreografija je lahko definirana znotraj istega

paketa ali pa je iz ločenega (t. i. uvoženega) paketa. Aktivnost `perform` ima tudi mehanizem za prenos vrednosti spremenljivk na klicano koreografijo.

Najpomembnejša aktivnost je `interaction`, ki opisuje izmenjavo informacij med dvema udeležencema s poudarkom na prejemniku informacij. Interakcija lahko definira informacije tipa *zahteva*, *odgovor* ali *zahteva-odgovor*. Vsebuje tri dele, ki se nanašajo na (i) udeležence izmenjave (element `participate`), (ii) informacijo za izmenjavo in (iii) kanal za izmenjavo. Informacija je predstavljena preko spremenljivk; obstajajo trije različni tipi: *aplikacijsko odvisne informacije* (npr. šifra izdelka), *informacije o stanju* (npr. naročilo sprejeto) in *informacije o kanalu*. Podrobnosti z ustrezno sintakso so opisane v [108, 109].

Dodatek D

Primer preslikave procesa BPEL v PN

Dodatek prikazuje primer preslikave enostavnega procesa BPEL s strukturirano aktivnostjo `while` v PN preko orodja BPEL2PNML [119]. Preslikava temelji na opisanih konceptih v poglavju 4 (razdelek 4.2.5). Prikazani proces prikazuje proženje sinhrone spletne storitve preko zanke `while`.

D.1 Proces BPEL

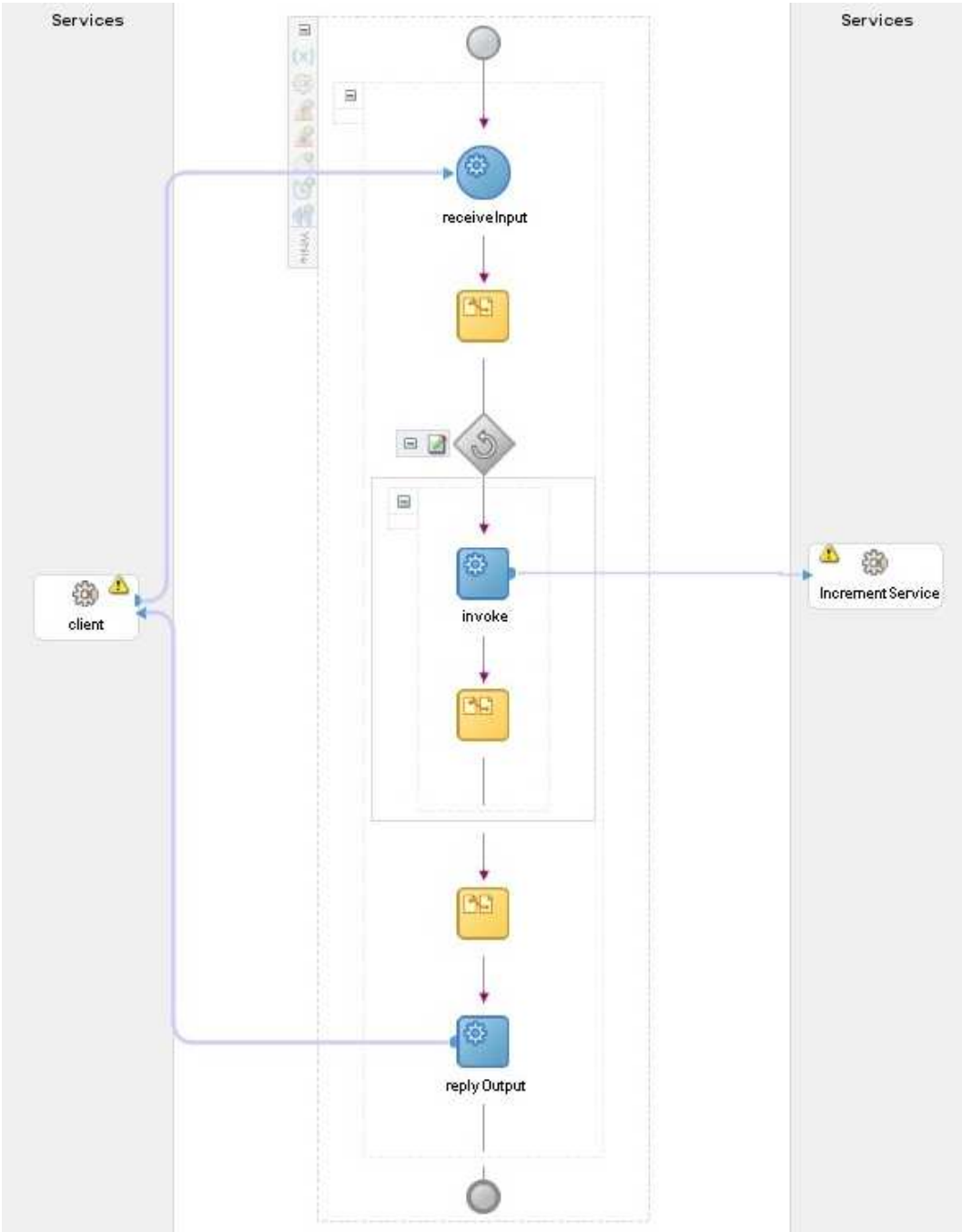
Odjemalec proži proces z začetno vrednostjo, ki jo predstavlja spremenljivka `input`. Spremenljivka `request` dobi na začetku vrednost 0. Proces kliče sinhrono spletno storitev `IncrementService` (definirana preko `partnerLink`), ki prejme kot vhod spremenljivko `request`, ter vrne povečano vrednost preko spremenljivke `response`. Storitev se kliče vse dokler je izpolnjen pogoj `input` $\geq$ `request`. Slika D.1 prikazuje grafični prikaz procesa v orodju *Oracle JDeveloper*(1), medtem ko je koda BPEL sledeča:

```
<!-- Proces sinhrono proži spletno storitev n-krat preko zanke while -->
<process name="While"
         targetNamespace="http://samples.otn.com/while"
         suppressJoinFailure="yes"
         xmlns:tns="http://samples.otn.com/while"
         xmlns:services="http://services.otn.com"
         xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
  <partnerLinks>
    <partnerLink name="client"
             partnerLinkType="tns:While"
             myRole="WhileProvider"
             partnerRole="WhileRequester"/>
    <partnerLink name="IncrementService"
             partnerLinkType="services:IncrementService"
             partnerRole="IncrementServiceProvider"/>
  </partnerLinks>
```

(1) Dostopno na `http://www.oracle.com/technology/products/jdev/index.html`.

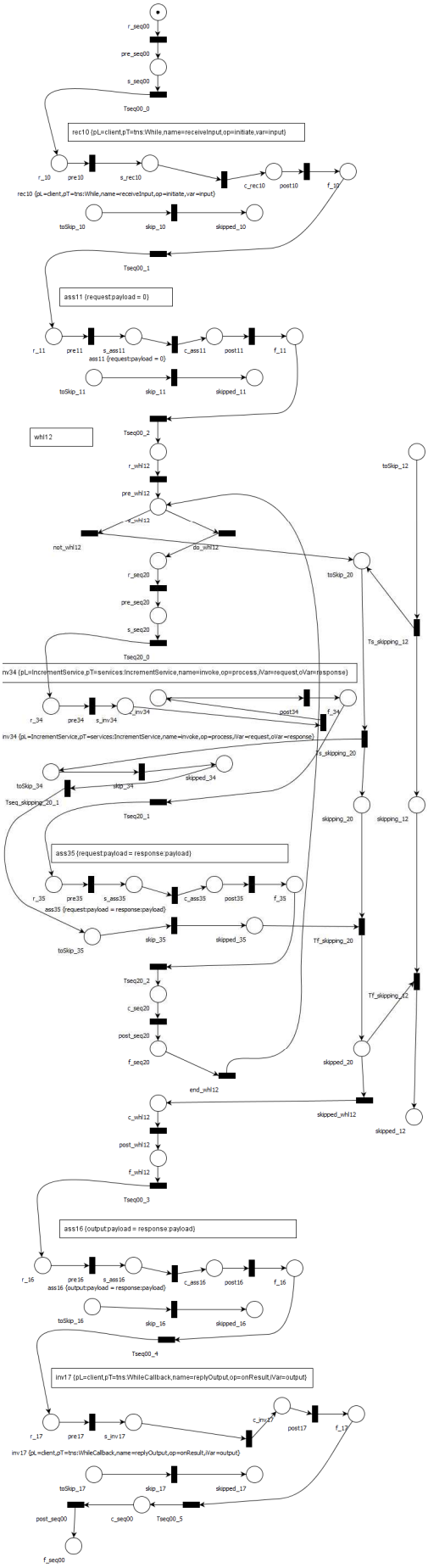
```
<variables>
  <variable name="input" messageType="tns:WhileRequestMessage"/>
  <variable name="output" messageType="tns:WhileResultMessage"/>
  <variable name="request"
            messageType="services:IncrementServiceRequestMessage"/>
  <variable name="response"
            messageType="services:IncrementServiceResponseMessage"/>
</variables>
<sequence>
  <!-- sprejmi klic od odjemalca -->
  <receive name="receiveInput" partnerLink="client"
           portType="tns:While"
           operation="initiate" variable="input"
           createInstance="yes"/>
  <assign>
    <copy>
      <from expression="0"/>
      <to variable="request" part="payload" query="/services:value"/>
    </copy>
  </assign>
  <!-- ponavljaj klic storitve dokler input < request -->
  <while condition=
    "bpws:getVariableData('input', 'payload', '/tns:value')
     >=
     bpws:getVariableData('request', 'payload', '/services:value')">
    <sequence>
      <!--  sinhrono proženje -->
      <invoke name="invoke" partnerLink="IncrementService"
              portType="services:IncrementService"
              operation="process" inputVariable="request"
              outputVariable="response"/>
      <assign>
        <copy>
          <from variable="response" part="payload" query="/services:result"/>
          <to variable="request" part="payload" query="/services:value"/>
        </copy>
      </assign>
    </sequence>
  </while>
  <assign>
    <copy>
      <from variable="response" part="payload" query="/services:result"/>
      <to variable="output" part="payload" query="/tns:result"/>
    </copy>
  </assign>
  <!-- odgovor nazaj na odjemalca (proženje storitve) -->
  <invoke name="replyOutput" partnerLink="client"
          portType="tns:WhileCallback"
```

```
            operation="onResult" inputVariable="output"/>
  </sequence>
</process>
```



Slika D.1: Grafični prikaz procesa BPEL v orodju *Oracle JDeveloper*.

Slika D.2 prikazuje rezultat preslikave z orodjem BPEL2PNML, ki je Petrijeva mreža v PNML prikazana z orodjem PIPE.



Slika D.2: Proces BPEL preslikan v Petrijevo mrežo.

Dodatek E

Prototip sistema *trustGuard*

Izvorna koda sistema[1] z namestitvijo in dokumentacijo, je dostopna na priloženem optičnem nosilcu (CD-ROM).

[1] Verzija v .NET in Javi.

Kratice, okrajšave, simboli

Kratica	Opis
ACID	Atomicity, Consistency, Isolation, Durability
ACL	Access Control List
BNF	Backus–Naur Form
BPEL	Business Process Execution Language
CCS	Calculus of Communicating Systems
CLR	Common Language Runtime
CORBA	Common Object Requesting Broker Architecture
CRL	Certificate Revocation List
DAG	Directed Acyclic Graph
DCOM	Distrubuted Component Object Model
DFS	Depth First Search
DNS	Domain Name Server
IPSec	Internet Protocol Security
JEE	Java Enterprise Edition
LTL	Linear Temporal Logic
LTS	Labeled Transition Systems
P2P	Peer-to-Peer
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PGP	Pretty Good Privacy
PICS	Platform for Internet Content Selection
PIPE	Platform Independent Petri net Editor
PKI	Public Key Infrastructure
PNML	Petri Net Markup Language
PPLTL	Pure Past Linear Temporal Logic
QoS	Quality of Service
Referee	Rule-Controlled Environment for Evaluation or Rules and Everything Else
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RSA	Rivest-Shamir-Adleman
RST	Request Security Token
RSTR	Request Security Token Response

Kratica	Opis
SAML	Security Assertion Markup Language
SCC	Strong Connected Components
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
STS	Security Token Service
TLS	Transport Layer Security
TPL	Trust Policy Language
UDDI	Universal Description Discovery and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
WCF	Windows Communication Foundation
WS-CDL	Web Services Choreography Language
WSCI	Web Service Choreography Interface
WSDL	Web Service Definition Language
XACML	Extensible Access Control Markup Language
XML	Extensible Markup Language
XSD	XML Schema Definition

Slike

2.1 Centraliziran sistem za obvladovanje zaupanja/ugleda. 8
2.2 Porazdeljen sistem za obvladovanje zaupanja/ugleda. 8

3.1 Možne odvisnosti med kontekstom, izvorom in ponorom. 30
3.2 Nagnjenost k tveganju kot odločitvena ploskev ($\mu = 10000$). 31
3.3 Nagnjenost k tveganju kot odločitvena ploskev ($\mu = 1000$). 32
3.4 Princip tranzitivnega zaupanja. 34
3.5 Modeliranje zaupanja. 36
3.6 Primer dogodkovne strukture za scenarij eBay. 38
3.7 Interakcije kot usmerjen graf. 40
3.8 Tranzitivna veriga zaupanja. 43
3.9 Paralelna veriga zaupanja. 44
3.10 Graf zaupanja na podlagi matrik $\mathbb{M}'(\varphi)$ in $\mathbb{M}'^r(\varphi)$. 52

4.1 Kompozicija spletnih storitev z orkestracijo. 72
4.2 Kompozicija spletnih storitev s koreografijo. 73
4.3 Primer proženja prehoda t: (a) pred in (b) po proženju. 78
4.4 Osnovna aktivnost kot PN. 79
4.5 Strukturirane aktivnosti kot PN. 79
4.6 Koraki analize procesa BPEL (BPEL2PNML, WofBPEL, PIPE). 81
4.7 Mehanizem procesiranja sporočil v strežniku BPEL. 82
4.8 Redukcijska pravila PN. 83
4.9 Redukcija iz PN v PN' (a) in bisimulacija (b). 84
4.10 Proces BPEL kot PN. 88
4.11 Drevo pokritosti Petrijeve mreže. 89
4.12 Primer acikličnega usmerjenega grafa. 91
4.13 Pretvorba drevesa $\mathscr{T}$ v graf komponent G^{SCC}. 93
4.14 Drevo pokritosti $\mathscr{T}$ (a), graf komponent G^{SCC} (b), končni DAG (c). 94

5.1 Trikotnik SOA z entitetami in operacijami. 101
5.2 Trikotnik SOA s spletno storitvijo zaupanja (TWS). 101

5.3 Varnostni model zaupanja WS-Trust preko STS. 112
5.4 Integracija sistema za OZ/OU v ogrodje WS-Trust. 123
5.5 Arhitekturni koncept sistema za OZ/OU. 124
5.6 Mehanizem za preverjanje skladnosti politik kot del sistema za OZ/OU. 135
5.7 Diagram zaporedja interakcij. 138

A.1 Začetni ekran – parametri simulatorja. 144
A.2 Izračunana matrika izkušenj. 148
A.3 Izračunana matrika stopenj. 149
A.4 Deleži kvalitativnih vrednosti ($n = 10$, $iter = 100$, $s(\text{opt}) = s(\text{pes}) = 0.5$, $s(\text{bal}) = 0$, $\alpha = 0$). 149
A.5 Deleži kvalitativnih vrednosti ($n = 10$, $iter = 1000$, $s(\text{opt}) = s(\text{pes}) = 0.5$, $s(\text{bal}) = 0$, $\alpha = 0$). 150
A.6 Deleži vrednosti ($n = 10$, $iter = 1000$, $unit =$ `Month`, $s(\text{opt}) = s(\text{pes}) = 0.5$, $s(\text{bal}) = 0$). 150
A.7 Deleži vrednosti ($n = 10$, $iter = 1000$, $unit =$ `Quarter`, $s(\text{opt}) = s(\text{pes}) = 0.5$, $s(\text{bal}) = 0$). 150
A.8 Deleži vrednosti za deleže operatorjev ($n = 10$, $iter = 100$, $\alpha = 0$). 151
A.9 Deleži vrednosti za realiste ($n = 10$, $iter = 100$, $\alpha = 0$). 151
A.10 Deleži vrednosti za realiste ($n = 9$, $iter = 90$, $\alpha = 0$). 152
A.11 Začetna in vmesne matrike ($n = 4$, $iter = 3$, $s(\text{opt}) = s(\text{pes}) = 0.5$). 153
A.12 Končna matrika iz vektorjev zaupanja ($n = 4$, $iter = 3$, $s(\text{opt}) = s(\text{pes}) = 0.5$). . . . 153
A.13 Sprememba deležev vrednosti med iteracijama ($n = 4$, $p(\text{opt}) = p(\text{pes}) = 0.5$). 154
A.14 Sprememba deležev vrednosti pri spremembah deležev operatorjev med simulacijo ($n = 9$). 155
A.15 Mreža vrednosti $\overline{s}(\text{D})$ ($n = 10$, $\alpha = 0.00$). 159
A.16 Mreža vrednosti $\overline{s}(\text{PD})$ ($n = 10$, $\alpha = 0.00$). 159
A.17 Mreža vrednosti $\overline{s}(\text{U})$ ($n = 10$, $\alpha = 0.00$). 160
A.18 Mreža vrednosti $\overline{s}(\text{PT})$ ($n = 10$, $\alpha = 0.00$). 161
A.19 Mreža vrednosti $\overline{s}(\text{T})$ ($n = 10$, $\alpha = 0.00$). 161

D.1 Grafični prikaz procesa BPEL v orodju *Oracle JDeveloper*. 177
D.2 Proces BPEL preslikan v Petrijevo mrežo. 178

Tabele

2.1 Matrika izidov za igro *Zaporniška dilema*. 23

3.1 Pravila za izračun vrednosti izkušnje zaupanja preko tranzitivnosti. 44
3.2 Interakcije med entitetami. 56
3.3 Interpretacija temporalnih operatorjev. 62

5.1 Predpone in pripadajoči imenski prostori WSS. 103
5.2 Preslikava PPLTL v elemente XML. 129

A.1 Pravila za ocenjevanje glede na tip obnašanja. 145
A.2 Vpliv faktorja α na $\overline{s}(\text{PD})$ in $\overline{s}(\text{D})$ $(n = 10, \texttt{iterations} = 50, \texttt{step} = 0.05)$. 158
A.3 Vpliv faktorja α na $\overline{s}(\text{U})$ $(n = 10, \texttt{iterations} = 50, \texttt{step} = 0.05)$. 159
A.4 Vpliv faktorja α na $\overline{s}(\text{PT})$ in $\overline{s}(\text{T})$ $(n = 10, \texttt{iterations} = 50, \texttt{step} = 0.05)$. 160

Algoritmi

3.1 Izračun tranzitivne ovojnice matrike $\mathbb{M}'^r$. 49
3.2 Izračun izpeljanega funkcionalnega zaupanja. 50
3.3 Iskanje tranzitivnih poti. 51
3.4 Metode za implementacijo semantike PPLTL. 66
3.5 Preverjanje skladnosti s politko π. 67
4.1 Konstrukcija drevesa pokritosti $\mathscr{T}$. 87
4.2 Topološko sortiranje DAG. 90
4.3 Izračun stopnje zaupanja kompozita ($\mathscr{T}$ brez ciklov). 90
4.4 Izračun stopnje zaupanja za proces BPEL. 97
A.1 Metoda MC za izračun povprečne vrednosti deleža kvalitativne vrednosti. 157

Primeri

4.1 Ogrodje procesa BPEL za preslikavo v PN. 88
5.1 Sporočilo SOAP z dodano varnostno glavo. 102
5.2 Šifrirano sporočilo SOAP. 104
5.3 Digitalni podpis znotraj sporočila SOAP. 106
5.4 Deljeni varnostni kontekst. 111
5.5 Zahteva RST za certifikat X.509. 114
5.6 Odgovor RSTR s certifikatom X.509. 115
5.7 Zahteva RST za žeton zaupanja. 118
5.8 Odgovor RSTR z vsebovanim žetonom zaupanja. 119
5.9 Zahteva RST za vpis ocene zaupanja. 122
5.10 Odgovor RSTR s statusnim žetonom. 123
5.11 Varnostna politika za telo sporočila SOAP. 128
5.12 Interakcijska politika zaupanja znotraj WS-Policy. 132
5.13 Zahteva RST z interakcijskimi politikami. 134
5.14 Odgovor RSTR z žetonom politike. 135
5.15 Zahteva WS-MEX za branje politike storitve. 137

Literatura

[1] S. P. Marsh. *Formalising Trust as a Computational Concept.* PhD thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.

[2] V. Reding. Safety on the Net. International High Level Research Seminar on "Trust in the Net", Vienna, 9 February 2006.

[3] T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design.* Prentice Hall PTR, August 12 2005.

[4] M. B. Jurič. Storitvena arhitektura - zgolj kompozicija spletnih storitev? Objavljeno v *Objektna tehnologija v Sloveniji OTS'2005*, 2005. URL: `http://lisa.uni-mb.si/~juric/SOA_ss.pdf`.

[5] D. Trček. *Managing Information Systems Security and Privacy.* Springer–Verlag, Berlin, 2006.

[6] K. E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills in L. Yu. Requirements for Policy Languages for Trust Negotiation. Objavljeno v *Proceedings of the 3rd international Workshop on Policies For Distributed Systems and Networks (Policy'02)*, strani 68–79, Los Alamitos, California, 2002. IEEE CS Press.

[7] K. Ren, T. Li, Z. Wan, F. Bao, R. H. Deng in K. Kim. Highly reliable trust establishment scheme in ad hoc networks. *Computer Networks*, 45(6):687–699, 2004.

[8] B. Schneier. *Applied Cryptography.* John Wiley & Sons, 1996.

[9] A. Jøsang, R. Ismail in C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[10] A. Jøsang. Prospectives for Online Trust Management. Working Paper. Submitted to IEEE Transactions on Knowledge and Data Engineering, 2008.

[11] A. Jøsang. Trust and Reputation Systems. Objavljeno v A. Aldini in R. Gorrieri, urednika, *Foundations of Security Analysis and Design IV, FOSAD 2006/2007 Tutorial Lectures*, Bertinoro, Italy, September 2007. Springer LNCS 4677.

[12] M. Zhu, L. Liu in Z. Jin. A Social Trust Model for Services. Objavljeno v *Proceedings of 11th Australian Workshop on Requirements Engineering (AWRE'06), December 9, 2006, Adelaide, SA, Australia*, 2006.

[13] D. Trček. A Formal Apparatus for Modeling Trust in Computing Environments. *Mathematical and Computer Modelling*, 49(1–2):226–233, 2009.

[14] C. N. Ziegler in G. Lausen. Propagation Models for Trust and Distrust in Social Networks. *Information Systems Frontiers*, 7(4-5):337–358, 2005.

[15] T. Dimitrakos in J. Bicarregui. Towards A Framework for Managing Trust in e–Services. Objavljeno v B. Gavish, urednik, *Proceedings of the Fourth International Conference on Electronic Commerce Research, ATSMA, IFIP, INFORMS 2*, strani 360–381, Dallas, Texas, USA, November 2001.

[16] T. Grandison in M. Sloman. A Survey of Trust in Internet Applications. *IEEE Communications Surveys and Tutorials*, 3(4):2–13, September 2000.

[17] A. Jøsang in S. Lo Presti. Analysing the Relationship Between Risk and Trust. Objavljeno v C. Jensen, S. Poslad in T. Dimitrakos, uredniki, *Second International Conference on Trust Management (iTrust 2004)*, volume LNCS 2995, strani 135–145. Springer, 2004.

[18] D. Trček. Towards trust management standardization. *Computer Standards & Interfaces*, 26(6):543–548, 2004.

[19] C. Castelfranchi in R. Falcone. Trust Is Much More than Subjective Probability: Mental Components and Sources of Trust. Objavljeno v *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6*, stran 6008, Washington, DC, USA, 2000. IEEE Computer Society.

[20] K. Aberer in Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. Objavljeno v *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, strani 310–317, New York, NY, USA, 2001. ACM.

[21] A. Jøsang in S. Pope. Semantic Constraints for Trust Transitivity. Objavljeno v *APCCM '05: Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling*, strani 59–68, Darlinghurst, Australia, 2005. Australian Computer Society, Inc.

[22] H. Li in M. Singhal. Trust Management in Distributed Systems. *Computer*, 40(2):45–53, 2007.

[23] P. Resnick, K. Kuwabara, R. Zeckhauser in E. Friedman. Reputation Systems. *Communications of the ACM*, 43(12):45–48, 2000.

[24] Z. Despotovic in K. Aberer. P2P reputation management: probabilistic estimation vs. social networks. *Computer Networks*, 50(4):485–500, 2006.

[25] G. Suryanarayana, J. R. Erenkrantz in R. N. Taylor. An Architectural Approach for Decentralized Trust Management. *IEEE Internet Computing*, 9(6):16–23, 2005.

[26] K.-J. Lin, H. Lu, T. Yu in C. en Tai. A Reputation and Trust Management Broker Framework for Web Applications. Objavljeno v *EEE '05: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce and e-Service*, strani 262–269, Washington, DC, USA, 2005. IEEE Computer Society.

[27] A. A. Selcuk, E. Uzun in M. R. Pariente. A reputation-based trust management system for P2P networks. Objavljeno v *CCGRID '04: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid*, strani 251–258, Washington, DC, USA, 2004. IEEE Computer Society.

[28] M. Czenko, H. Tran, J. Doumen, S. Etalle, P. Hartel in J. den Hartog. Nonmonotonic Trust Management for P2P Applications. *Electronic Notes in Theoretical Computer Science*, 157:113–130, 2006.

[29] F. Martinelli in M. Petrocchi. On Relating and Integrating Two Trust Management Frameworks. *Electronic Notes in Theoretical Computer Science*, 168:191–205, 2007.

[30] F. Martinelli in M. Petrocchi. A Uniform Framework for Security and Trust Modeling and Analysis with Crypto-CCS. *Electronic Notes in Theoretical Computer Science*, 186:85–99, 2007.

[31] S. Guemara-ElFatmi, N. Boudriga in M. S. Obaidat. Relational-based calculus for trust management in networked services. *Computer Communications*, 27(12):1206–1219, 2004.

[32] TRUSTe. TRUSTe Security Guidelines 2.0, 2005. URL: `http://www.truste.org/pdf/SecurityGuidelines.pdf`.

[33] C. Adams in S. Farrell. RFC2510 – Internet X.509 Public Key Infrastructure Certificate Management Protocols, 1999. URL: `http://www.ietf.org/rfc/rfc2510.txt`.

[34] A. Jøsang. An algebra for assessing trust in certification chains. Objavljeno v *Network and Distributed Systems Security Symposium (NDSS 99)*, San Diego, USA, 1999. The Internet Society.

[35] A. Jøsang. A Subjective Metric of Authentication. Objavljeno v *ESORICS '98: Proceedings of the 5th European Symposium on Research in Computer Security*, strani 329–344, London, UK, 1998. Springer-Verlag.

[36] P. R. Zimmermann. *The Official PGP User's Guide.* MIT Press, 1995.

[37] M. Blaze, J. Feigenbaum in J. Ioannidis. RFC2704 – The KeyNote Trust-Management System Version 2. The Internet Society, 1999. URL: `http://www.crypto.com/papers/rfc2704.txt`.

[38] J. Miller, P. Resnick in D. Singer. Rating Services and Rating Systems (and Their Machine Readable Descriptions) Version 1.1. W3C Recommendation, 1996. URL: `http://www.w3.org/TR/REC-PICS-services`.

[39] A. Herzberg, Y. Mass, J. Michaeli, Y. Ravid in D. Naor. Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers. Objavljeno v *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, stran 2, Washington, DC, USA, 2000. IEEE Computer Society.

[40] O. E. Williamson. Calculativeness, Trust, and Economic Organization. *Journal of Law & Economics*, 36(1):453–86, April 1993.

[41] D. M. Kreps. *A Course in Microeconomic Theory.* Princeton University Press, 1990.

[42] D. Kovač in D. Trček. Metode in modeli zaupanja v porazdeljenih sistemih. Objavljeno v B. Zajc in A. Trost, urednika, *Zbornik šestnajste mednarodne Elektrotehniške in računalniške konference ERK 2007, 24. - 26. september 2007*, Portorož, Slovenija, 2007. IEEE Region 8, Slovenska sekcija IEEE.

[43] K. Krukow in M. Nielsen. From Simulations to Theorems: A Position Paper on Research in the Field of Computational Trust. *Lecture Notes in Computer Science*, 4691:96–111, 2007.

[44] P. Nurmi. A Bayesian framework for online reputation systems. Objavljeno v *AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*, stran 121, Washington, DC, USA, 2006. IEEE Computer Society.

[45] A. Gelman, J. B. Carlin, H. S. Stern in D. B. Rubin. *Bayesian Data Analysis.* Chapman & Hall, 2004.

[46] G. Shafer. *A Mathematical Theory of Evidence.* Priceton University Press, 1979.

[47] A. Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge–Based Systems*, 9(3):279–311, 2001.

[48] M. Ceglowski, A. Coburn in J. Cuadrado. Semantic search of unstructured data using contextual network graphs. Preliminary white paper, National Institute for Technology and Liberal Education, Middlebury College, Middlebury, Vermont, 05753 USA, 2003.

[49] L. Page, S. Brin, R. Motwani in T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Techologies Project, 1998.

[50] J. Nash. Non-Cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[51] K. Aberer in Z. Despotovic. On Reputation in Game Theory - Application to Online Settings. Working paper, Swiss Federal Institute of Technology (EPFL), 2004.

[52] E. Faingold in Y. Sannikov. Equilibrium Degeneracy and Reputation Effects in Continuous-Time Games. 2006 Meeting Papers 694, Society for Economic Dynamics, december 2006.

[53] D. M. Kreps in R. Wilson. Sequential Equilibria. *Econometrica*, 50(4):863–94, July 1982.

[54] D. M. Kreps in R. Wilson. Reputation and imperfect information. *Journal of Economic Theory*, 27(2):253–279, August 1982.

[55] R. Axelrod. *The Evolution of Cooperation.* Basic Books, New York, 1985.

[56] R. Jurca in B. Faltings. An Incentive Compatible Reputation Mechanism. Objavljeno v *Proceedings of the IEEE International Conference on E-Commerce (CEC'03)*, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[57] M. S. Al-Mutairi, K. W. Hipel in M. S. Kamel. Trust and cooperation from a fuzzy perspective. *Mathematics nad Computers in Simulation*, 76(5-6):430–446, 2008.

[58] L. Mui, M. Mohtashemi in A. Halberstadt. A Computational Model of Trust and Reputation for E-businesses. Objavljeno v *HICSS '02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 7*, stran 188, Washington, DC, USA, 2002. IEEE Computer Society.

[59] M. Nielsen in K. Krukow. Towards a Formal Notion of Trust. Objavljeno v *PPDP '03: Proceedings of the 5th ACM SIGPLAN international conference on Principles and practice of declaritive programming*, strani 4–7, New York, NY, USA, 2003. ACM.

[60] M. Carbone, M. Nielsen in V. Sassone. A Formal Model for Trust in Dynamic Networks. Objavljeno v *Proceedings of the First International Conference on Software Engineering and Formal Methods (SEFM03)*, strani 54–61, September 2003.

[61] D. Cvrček, V. Matyáš in A. Patel. Evidence processing and privacy issues in evidence-based reputation systems. *Computer Standards & Interfaces*, 27(5):533–545, 2005.

[62] D. Kovač in D. Trček. Towards Standardization of Trust in WS-Trust. Working paper, 2007.

[63] D. Kovač in D. Trček. Qualitative Trust Modeling in SOA. *Journal of Systems Architecture*, 55(4):255–263, 2009.

[64] J. Piaget. *Judgement and Reasoning in the Child.* Routledge, London, 1999.

[65] S. Staab, B. Bhargava, L. Lilien, A. Rosenthal, M. Winslett, M. Sloman, T. S. Dillon, E. Chang, F. Khadeer Hussain, W. Nejdl, D. Olmedilla in V. Kashyap. The pudding of trust. *IEEE Intelligent Systems*, 19(5):74–88, 2004.

[66] D. W. Manchala. Trust Metrics, Models and Protocols for Electronic Commerce Transactions. Objavljeno v *ICDCS '98: Proceedings of the The 18th International Conference on Distributed Computing Systems*, stran 312, Washington, DC, USA, 1998. IEEE Computer Society.

[67] R. L. Trivers. The Evolution of Reciprocal Altruism. *The Quarterly Review of Biology*, 46(1):35–57, 1971.

[68] P. Dasgupta. *Trust: Making and Breaking Cooperative Relations*, poglavje Trust as a Commodity, strani 49–72. Department of Sociology, University Oxford, 2000.

[69] A. Jøsang, E. Gray in M. Kinateder. Simplification and analysis of transitive trust networks. *Web Intelligence and Agent System*, 4(2):139–161, 2006.

[70] A. Gutscher. A Trust Model for an Open, Decentralized Reputation System. Objavljeno v S. Etalle in S. Marsh, urednika, *IFIP International Federation for Information Processing, Trust Management*, volume 238, strani 285–300. Springer Boston, 2007.

[71] A. Jøsang, E. Gray in M. Kinateder. Analysing Topologies of Transitive Trust. Objavljeno v T. Dimitrakos in F. Martinelli, urednika, *Proceedings of the First International Workshop on Formal Aspects in Security & Trust (FAST2003)*, strani 9–22, Pisa, Italy, 2003.

[72] D. Kovač in D. Trček. Modeling Trust in Service Oriented Environments. Objavljeno v B. Zajc in A. Trost, urednika, *Zbornik sedemnajste mednarodne Elektrotehniške in računalniške konference ERK 2008, 29. september – 1. oktober 2007*, Portorož, Slovenija, 2008. IEEE Region 8, Slovenska sekcija IEEE.

[73] E. Nuutila. Efficient Transitive Closure Computation in Large Digraphs. *Acta Polytechnica Scandinavia: Math. Comput. Eng.*, 74:1–124, 1995.

[74] N. Prijatelj. *Matematične strukture II.* Državna založba Slovenije, Ljubljana, 1990.

[75] B. A. Davey in H. A. Piestly. *Introduction to Lattices and Order.* Cambridge University Press, 2002.

[76] K. Krukow, M. Nielsen in V. Sassone. A Formal Framework for Concrete Reputation-Systems with Applications to History-Based Access Control. Objavljeno v *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, strani 260–269, New York, NY, USA, 2005. ACM.

[77] M. Nielsen in K. Krukow. On the Formal Modelling of Trust in Reputation-Based Systems. *Lecture Notes in Computer Science*, 3113:192–204, 2004.

[78] P. Herrmann. Temporal Logic-Based Specification and Verification of Trust Models. Objavljeno v K. Stølen, W.H. Winsborough, F. Martinelli in F. Massacci, uredniki, *Proceedings of the 4th International Conference on Trust Management, iTrust 2006, Pisa, Italy, May 16-19*, volume 3986 of *Lecture Notes in Computer Science*, strani 105–119. Springer, 2006.

[79] C. Liu, M. A. Ozols in M. Orgum. A Temporalised Belief Logic for Specifying the Dynamics of Trust for Multi-agent Systems. *Lecture Notes in Computer Science*, 3321:142–156, 2005.

[80] M. Richardson, R. Agrawal in P. Domingos. Trust Management for the Semantic Web. Objavljeno v *Proceedings of the Second International Semantic Web Conference, Sanibel Island, FL*, Lecture Notes in Computer Science, strani 351–368. Springer, 2003.

[81] Y. C. Jiang, Z. Y. Xia, Y. P. Zhong in S. Y. Zhang. Autonomous trust construction in multi-agent systems: a graph theory methodology. *Advances in Engineering Software*, 36(2):59–66, 2005.

[82] B. Christianson in W. S. Harbison. Why Isn't Trust Transitive? Objavljeno v *Proceedings of the International Workshop on Security Protocols*, strani 171–176, London, UK, 1997. Springer-Verlag.

[83] G. Theodorakopoulos in J. S. Baras. Trust Evaluation in Ad-Hoc Networks. Objavljeno v *WiSe '04: Proceedings of the 3rd ACM workshop on Wireless security*, strani 1–10, New York, NY, USA, 2004. ACM.

[84] A. Jakoby, M. Liśkiewicz in R. Reischuk. Space Efficient Algorithms for Directed Series-Parallel Graphs. *J. Algorithms*, 60(2):85–114, 2006.

[85] T. H. Cormen, C. E. Leiserson, R. L. Riverst in C. Stein. *Introduction to Algorithms*. MIT Press, Second edition, 2001.

[86] M. Finger in D. M. Gabbay. Adding a temporal dimension to a logic system. *Journal of Logic, Language and Information*, 1(3):203–233, 1992.

[87] M. A. Orgun in C. Liu. Reasoning about Dynamics of Trust and Agent Beliefs. Objavljeno v *Information Reuse and Integration, 2006 IEEE International Conference*, strani 105–110, September 2006.

[88] T. Latvala, A. Biere, K. Heljanko in T. Junttila. Simple is better: Efficient bounded model checking for past LTL. Objavljeno v R. Cousout, urednik, *VMCAI '05: Proceedings of the 6th International Conference on Verification, Model Checking, and Abstract Interpretation*, volume 3385 of *Lecture Notes in Computer Science*, strani 380–395. Springer Verlag, 2005.

[89] A. P. Sistla in E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM*, 32(3):733–749, 1985.

[90] M. Pradella, P. San Pietro, P. Spoletini in A. Morzenti. Practical Model Checking of LTL with Past. Objavljeno v *Taiwan University*, 2003.

[91] K. Havelund in G. Rosu. Synthesizing Monitors for Safety Properties. Objavljeno v *TACAS '02: Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, strani 342–356, London, UK, 2002. Springer-Verlag.

[92] J. Kozak. *Podatkovne strukture in algoritmi.* DMFA, Ljubljana, 1997.

[93] R. L. Graham, D. E. Knuth in O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science.* Addison-Wesley, 1988.

[94] E. Newcomer in G. Lomow. *Understanding SOA with Web services (Independent Technology Guides).* Addison-Wesley, Pearson Education, 2005.

[95] M. B. Jurič. Zakaj spletne storitve niso zgolj novi komponentni model. Objavljeno v *Objektna tehnologija v Sloveniji*, 2002. URL: `http://lisa.uni-mb.si/~juric/zakajSpletne.pdf`.

[96] W3C. SOAP Version 1.2. W3C, 2007. URL: `http://www.w3.org/TR/soap12`.

[97] W3C. XML Schema 1.1. W3C, 2005. URL: `http://www.w3.org/XML/Schema`.

[98] W3C. Web Services Description Language (WSDL) Version 2.0. W3C Recommendation 26 June 2007, W3C, 2007. URL: `http://www.w3.org/TR/wsdl20`.

[99] OASIS. UDDI Version 3.0.2. UDDI Spec Technical Committee Draft, Dated 20041019, 2004. URL: `http://uddi.org/pubs/uddi_v3.htm`.

[100] D. Kovač. Arhitektura za podporo heterogenih poslovnih modelov s spletnimi storitvami. Magistrska naloga, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2006.

[101] S. Weerawarana, F. Curbera, F. Leymann, T. Storey in D. F. Ferguson. *Web Services Platform Architecture.* Prentice Hall PTR, 2005.

[102] D. C. Chou in K. Yurov. Security development in Web Services environment. *Computer Standards & Interfaces*, 27(3):233–240, 2005.

[103] C. Peltz. Web Services Orchestration and Choreography. *Computer*, 36(10):46–52, October 2003.

[104] OASIS. Web Services Business Process Execution Language Version 2.0. OASIS Standard, OASIS WSBPEL TC, 2007. URL: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

[105] D. Kovač in D. Trček. A Survey of Web services Orchestration and Choreography with Formal Models. Working paper, 2007.

[106] M. B. Jurič. A Hands-on Introduction To BPEL. Oracle Technology Network. URL: http://www.oracle.com/technology/pub/articles/matjaz_bpel1.html.

[107] M. B. Jurič. A hands-on Introduction To BPEL, part 2: Advanced BPEL. Oracle Technology Network. URL: http://www.oracle.com/technology/pub/articles/matjaz_bpel2.html.

[108] W3C. Web Services Choreography Description Language Version 1.0. W3C Candidate Recommendation 9 November 2005, W3C, 2005. URL: http://www.w3.org/TR/ws-cdl-10/.

[109] A. Barros, M. Dumas in P. Oaks. A Critical Overview of the Web Services Choreography Description Language (WS-CDL). BPTrends, March 2005.

[110] W3C. Web Service Choreography Interface (WSCI) 1.0. W3C Note 8 August 2002, W3C, 2002. URL: http://www.w3.org/TR/wsci/.

[111] OASIS. Web Services Reliable Messaging (WS-ReliableMessaging). Committee Draft 04, August 11, 2006, 2006. URL: http://docs.oasis-open.org/ws-rx/wsrm/200608/wsrm-1.1-spec-cd-04.pdf.

[112] W3C. Web Services Addressing (WS-Addressing). W3C Member Submission 10 August 2004, W3C, 2004. URL: http://www.w3.org/Submission/ws-addressing/.

[113] OASIS. Web Services Coordination (WS-Coordination) Version 1.1. OASIS Standard, 16 April 2007, 2007. URL: http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.1-spec-os.pdf.

[114] S. Kendall. *UML Explained*. Addison-Wesley Professional, 2001.

[115] OASIS. Web Services Business Activity (WS-BusinessActivity) Version 1.1. OASIS Standard, 16 April 2007, 2007. URL: http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec-os.pdf.

[116] Y. Qian, Y. Xu, Z. Wang, G. Pu, H. Zhu in C. Cai. Tool Support for BPEL Verification in ActiveBPEL Engine. Objavljeno v *ASWEC '07: Proceedings of the 2007 Australian Software Engineering Conference*, strani 90–100, Washington, DC, USA, 2007. IEEE Computer Society.

[117] W.-L. Dong, H. Yu in Y.-B. Zhang. Testing BPEL-based Web Service Composition Using High-level Petri Nets. Objavljeno v *EDOC '06: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference*, strani 441–444, Washington, DC, USA, 2006. IEEE Computer Society.

[118] S. Nakajima. Model-Checking Behavioral Specification of BPEL Applications. *Electronic Notes in Theoretical Computer Science*, 151(2):89–105, May 2006.

[119] C. Ouyang, E. Verbeek, W. M. P. van der Aalst, S. Breutel, M. Dumas in A. H. M. ter Hofstede. Formal semantics and analysis of control flow in WS-BPEL. *Science of Computer Programming*, 67(2-3):162–198, 2007.

[120] M. Viroli. A core calculus for correlation in orchestration languages. *Journal of Logic and Algebraic Programming*, 70(1):74–95, 2007.

[121] M. Viroli. Towards a Formal Foundation to Orchestration Languages. *Electronic Notes in Theoretical Computer Science*, 105:51–71, December 2004.

[122] J. Shen, G. Grossmann, Y. Yang, M. Stumptner, M. Schrefl in T. Reiter. Analysis of business process integration in Web service context. *Future Generation Computer Systems*, 23(3):283–294, 2007.

[123] R. Hamadi in B. Benatallah. A Petri Net-based Model for Web Service Composition. Objavljeno v *ADC '03: Proceedings of the 14th Australasian database conference*, strani 191–200, Darlinghurst, Australia, 2003. Australian Computer Society, Inc.

[124] G. Pu, Z. Xiangpeng, W. Shuling in Q. Zongyan. Towards the Semantics and Verification of BPEL4WS. *Electronic Notes in Theoretical Computer Science*, 151(2):33–52, May 2006.

[125] R. Lucchi in M. Mazzara. A pi-calculus based semantics for WS-BPEL. *Journal of Logic and Algebraic Programming*, 70(1):96–118, January 2007.

[126] G. Salaün, L. Bordeaux in M. Schaerf. Describing and Reasoning on Web Services using Process Algebra. Objavljeno v *ICWS '04: Proceedings of the IEEE International Conference on Web Services*, stran 43, Washington, DC, USA, 2004. IEEE Computer Society.

[127] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[128] J. Esparza in M. Nielsen. Decidability issues for Petri nets. *Petri Nets Newsletter*, 52:245–262, 1994.

[129] P. Jančar in J. Esparza. Deciding Finiteness of Petri Nets Up To Bisimulation. Objavljeno v *ICALP '96: Proceedings of the 23rd International Colloquium on Automata, Languages and Programming*, strani 478–489, London, UK, 1996. Springer-Verlag.

[130] L. Aceto, A. Ingólfsdóttir, K. G. Larsen in J. Srba. *Reactive Systems*. Cambridge University Press, The Edinburgh Building, Cambridge CB2 8RU, UK, 2007.

[131] Ch. Ouyang, E. Verbeek, S. Breutel, M. Dumas in A. H. M. Ter Hofstede. WofBPEL: A tool for automated analysis of BPEL processes. Objavljeno v *ICSOC 2005 proceedings, December, Lecture Notes in Computer Science*, strani 484–489. Springer, 2005.

[132] P. Bonet, C. Llado, R. Puijaner in W. Knottenbelt. Pipe v2.5.: A Petri net Tool For Performance Modelling. Objavljeno v *23rd Latin American Conference on Informatics*, October 2007.

[133] J. Dehnert. *A Methodology for Workflow Modelling: From business process modelling towards sound workflow specification*. PhD thesis, Technical University of Berlin, School of Electrical Engineering and Computer Sciences, Berling, Germany, August 2003.

[134] H. M. W. Verbeek, W. M. P. Van, Der Aalst in A. H. M. Ter Hofstede. Verifying workflows with cancellation regions and OR-joins: an approach based on invariants. Objavljeno v *Proceedings of BPM 2006, volume 4102 of LNCS*, 2006.

[135] P. Jančar in F. Moller. Checking Regular Properties of Petri Nets. Objavljeno v *CONCUR '95: Proceedings of the 6th International Conference on Concurrency Theory*, strani 348–362, London, UK, 1995. Springer-Verlag.

[136] OASIS. Schema for Executable Process for WS-BPEL 2.0. OASIS Standard, 11th April 2007, 2006. URL: `http://docs.oasis-open.org/wsbpel/2.0/CS01/process/executable/ws-bpel_executable.xsd`.

[137] T. H. Cormen, C. E. Leiserson in R. L. Rivest. *Introduction to Algorithms*. The MIT Press, McGraw-Hill, 1999.

[138] A. B. Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562, 1962.

[139] E. Nuutila in E. Soisalon-soininen. On Finding the Strong Components in a Directed Graph, 1993. Helsinki University of Technology, Finland.

[140] R. Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[141] L. Zhengping, L. Xiaoli, W. Guoqing, Y. Min in Z. Fan. A formal Framework for Trust management of Service-oriented Systems. Objavljeno v *SOCA '07: Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*, strani 241–248, Washington, DC, USA, 2007. IEEE Computer Society.

[142] E. Chang, T. S. Dillon in F. K. Hussain. Trust and reputation relationships in service-oriented environments. Objavljeno v *Third International Conference on Information Technology and Applications, ICITA 2005*, volume 1, strani 4–14, 2005.

[143] F. K. Hussain, E. Chang in T. S. Dillon. Defining Reputation in Service Oriented Environment. Objavljeno v *AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*, stran 177, Washington, DC, USA, 2006. IEEE Computer Society.

[144] E. Chang, F. Hussain in T. Dillon. *Trust and Reputation for Service-Oriented Environments: Technologies For Building Business Intelligence And Consumer Confidence*. John Wiley & Sons, 2005.

[145] A. Michlmayr, F. Rosenberg, C. Platzer, M. Treiber in S. Dustdar. Towards Recovering the Broken SOA Triangle: A Software Engineering Perspective. Objavljeno v *IW-SOSWE '07: 2nd international workshop on Service oriented software engineering*, strani 22–28, New York, NY, USA, 2007. ACM.

[146] R. Hollar in R. Murhpy. *Enterprise Web Services Security*. Charles River Media, Inc., 2006.

[147] OASIS. Web Services Security: SOAP Message Security 1.1, OASIS Standard Specification, 1 February 2006. OASIS WSS TC, 2006. URL: `http://docs.oasis-open.org/wss/v1.1/`.

[148] W3C. Namespaces in XML 1.0 (Second Edition). W3C Recommendation 16 August 2006, W3C, 2006. URL: `http://www.w3.org/TR/REC-xml-names/`.

[149] W3C. XML Encryption Syntax and Processing. W3C Recommendation 10 December 2002, W3C, 2002. URL: `http://www.w3.org/TR/xmlenc-core/`.

[150] IETF in W3C. XML-Signature Syntax and Processing. W3C Recommendation 12 February 2002, The Internet Society & W3C, 2002. URL: `http://www.w3.org/TR/xmldsig-core/`.

[151] OASIS. Web Services Security UsernameToken Profile 1.1. OASIS Standard Specification, 1 February 2006, OASIS WSS TC, 2006. URL: `http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-UsernameTokenProfile.pdf`.

[152] OASIS. Web Services Security X.509 Certificate Token Profile 1.1. OASIS Standard Specification, 1 February 2006, OASIS WSS TC, 2006. URL: http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-x509TokenProfile.pdf.

[153] ITU-T. Public-key and attribute certificate frameworks Technical Corrigendum 1. ITU-T Recommendation X.509. Corrigendum 1, 2000. URL: http://www.itu.int/dms_pub/itu-t/rec/x/T-REC-X.509-200110-I!Cor1!PDF-E.pdf.

[154] B. Kaliski. RFC2315 – PKCS #7: Cryptographic Message Syntax Version 1.5, 1998. URL: http://www.ietf.org/rfc/rfc2315.txt.

[155] OASIS. Web Services Security Kerberos Token Profile 1.1. OASIS Standard Specification, 1 February 2006, OASIS WSS TC, 2006. URL: http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-KerberosTokenProfile.pdf.

[156] J. Kohl in C. Neuman. RFC1510 – The Kerberos Network Authentication Service (V5), 1993. URL: http://www.ietf.org/rfc/rfc1510.txt.

[157] J. Linn. RFC1964 – The Kerberos Version 5 GSS-API Mechanism, 1996. URL: http://www.ietf.org/rfc/rfc1964.txt.

[158] OASIS. Web Services Security: SAML Token Profile 1.1. OASIS Standard, 1 February 2006, OASIS WSS TC, 2006. URL: http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SAMLTokenProfile.pdf.

[159] OASIS. SAML Specifications, 2005. URL: http://saml.xml.org/saml-specifications.

[160] OASIS. Web Services Security Rights Expression Language (REL) Token Profile 1.1. OASIS Standard: 1 February 2006, OASIS WSS TC, 2006. URL: http://docs.oasis-open.org/wss/v1.1/oasis-wss-rel-token-profile-1.1.pdf.

[161] ISO/IEC 21000-5:2004. Multimedia framework (MPEG-21) – Part 5: Rights Expression Language, 2004. URL: http://www.iso.org.

[162] OASIS. WS-SecureConversation 1.3. OASIS Standard 1 March 2007, OASIS WSS Exchange TC, 2007. URL: http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.pdf.

[163] OASIS. WS-Trust 1.3, OASIS Standard, 19 March 2007. OASIS Web Service Secure Exchange TC, 2007. URL: http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.pdf.

[164] E. Kleiner in A. W. Roscoe. On the Relationship Between Web Services Security and Traditional Protocols. *Electronic Notes in Theoretical Computer Science*, 155:583–603, May 2006.

[165] T. Dierks in E. Rescorla. RFC5246 – The Transport Layer Security (TLS) Protocol Version 1.2, 2008. URL: http://www.ietf.org/rfc/rfc5246.txt.

[166] W3C. Web Services Policy 1.5 - Framework. W3C Recommendation 04 September 2007, 2007. URL: http://www.w3.org/TR/ws-policy/.

[167] W3C. Web Services Policy 1.2 – Attachment (WS-PolicyAttachment). W3C Member Submission 25 April 2006, W3C, 2006. URL: http://www.w3.org/Submission/WS-PolicyAttachment/.

[168] OASIS. Web Services Federation Language (WS-Federation) Version 1.2. Committee Draft 01, OASIS WSFED TC, 2008. URL: http://docs.oasis-open.org/wsfed/federation/v1.2/ws-federation.html.

[169] OASIS. WS-Trust 1.3 XML Schema Definition. OASIS, 2005. URL: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust.xsd.

[170] OASIS. WS-Trust 1.3 WSDL. OASIS, 2005. URL: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3.wsdl.

[171] J. Lövy. *Programming WCF Services*. O'Reilly Media, 2007.

[172] K. Haase. *The Java EE 5 Tutorial*. Addison Wesley, 3 edition, 2007.

[173] W3C. Web Services Metadata Exchange 1.1 (WS-MetadataExchange). W3C Member Submission 13 August 2008, 2008. URL: http://www.w3.org/Submission/WS-MetadataExchange/.

[174] A. H. Anderson. An Introduction to the Web Services Policy Language (WSPL). Objavljeno v *POLICY '04: Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks*, Washington, DC, USA, 2004. IEEE Computer Society.

[175] OASIS. eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard, 1 Feb 2005, 2005. URL: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

[176] W3C. W3C XML Schema defined in the Web Services Policy 1.5 Framework specification, 2006. URL: http://www.w3.org/2006/07/ws-policy.xsd.

[177] OASIS. WS-SecuritPolicy 1.2. OASIS Standard, 2007. URL: http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.pdf.

[178] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, B. C. de Laat, M. Holdrege in D. Spence. RFC 2904 – AAA Authorization Framework, 2000. URL: http://www.ietf.org/rfc/rfc2904.txt.

[179] C. Dellarocas. The Design of Reliable Trust Management Systems for Electronic Communities. Working paper, Sloan School of Management, MIT, 2001.

[180] D. Marshall. *Programming Microsoft Visual C# 2008: The Language*. Microsoft Press, 2008.

[181] S. K. Park in K. W. Miller. Random number generators: good ones are hard to find. *Communications of the ACM*, 31(10):1192–1201, 1988.

[182] C. P. Robert in G. Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, 2004.

[183] R. Y. Rubistein in D. P. Kroese. *Simulation and the Monte Carlo Method*. John Wiley & Sons, 2007.