Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Προσαρμογή και Ευθυγραμμία Σχεδιαστικών Υποδειγμάτων Διεπαφών σε Υπηρεσιοστρεφείς Αρχιτεκτονικές

Διδακτορική Διατριβή

Μιχαήλ Γ. Αθανασόπουλος

Αθήνα, Δεκέμβριος 2015

Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Προσαρμογή και Ευθυγραμμία Σχεδιαστικών Υποδειγμάτων Διεπαφών σε Υπηρεσιοστρεφείς Αρχιτεκτονικές

Διδακτορική Διατριβή

Μιχαήλ Γ. Αθανασόπουλος

Διπλ. Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών ΕΜΠ (2008)

Συμβουλευτική επιτροπή: Κ. Κοντογιάννης
Ι. Βασιλείου
Τ. Σελλής

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή τη 17η Δεκεμβρίου 2015.

. . . . .	. . . . .	. . . . .
Κ. Κοντογιάννης	Ι. Βασιλείου	Τ. Σελλής
Αναπλ. Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ
. . . . .	. . . . .	. . . . .
Ν. Παπασπύρου	Δ. Ασκούνης	Στ. Χριστοδουλάκης
Αναπλ. Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Καθηγητής Πολυτ. Κρήτης

. . . . . .
Y. Zou
Αναπλ. Καθηγήτρια
Queen's Univ. (Canada)

Αθήνα, Δεκέμβριος 2015

. . . . . .

Michael G. Athanasopoulos
Electrical and Computer Engineer, Ph.D., N.T.U.A.

National Technical University of Athens

School of Electrical and Computer Engineering

Division of Computer Science

Adaptation and Alignment of Interface Design Paradigms in Service-Oriented Architectures

Doctoral Dissertation

Michael G. Athanasopoulos

Athens, December 2015

NATIONAL TECHNICAL UNIVERSITY OF ATHENS

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

DIVISION OF COMPUTER SCIENCE

Adaptation and Alignment of Interface Design Paradigms in Service-Oriented Architectures

Doctoral Dissertation

by Michael G. Athanasopoulos

Diploma in Electrical and Computer Engineering (2008)

Advisory committee: K. Kontogiannis
Y. Vassiliou
T. Sellis

Approved by the seven-member examining committee on December 17th, 2015.

. . . . .	. . . . .	. . . . .
K. Kontogiannis	Y. Vassiliou	T. Sellis
Assoc. Professor NTUA	Professor NTUA	Professor NTUA
. . . . .	. . . . .	. . . . .
N. Papaspyrou	D. Askounis	S. Christodoulakis
Assoc. Professor NTUA	Professor NTUA	Professor TUC

. . . . . .
Y. Zou
Assoc. Professor
Queen's Univ. (Canada)

Athens, December 2015

. . . . . .

Michael G. Athanasopoulos
Electrical and Computer Engineer, Ph.D., N.T.U.A.

Περίληψη

Η περιοχή της Υπηρεσιοστρεφούς Υπολογιστικής (Service-Oriented Computing) αναδύθηκε ως ένα υπολογιστικό μοντέλο για τη σχεδίαση και την κατασκευή κατανεμημένων συστημάτων. Σε αυτό το μοντέλο, τα δομοστοιχεία λογισμικού παρέχουν και καταναλώνουν υπηρεσίες, οι οποίες εκτίθενται μέσω διεπαφών. Η καθιερωμένη προσέγγιση για τη σχεδίαση διεπαφών υπηρεσιών στην υπηρεσιοστρεφή υπολογιστική ακολουθεί το υπόδειγμα προσανατολισμένο σε διαδικασίες (ΠΣΔ) (procedure-oriented paradigm), στο οποίο οι δυνατότητες μιας υπηρεσίας αναλύονται και εκτίθενται ως συλλογές σχετιζόμενων λειτουργιών που μπορούν να κληθούν, ακολουθώντας το παραδοσιακό μοντέλο πρόσβασης απομακρυσμένων κλήσεων διαδικασιών. Παρόλα αυτά, κατά τη διάρκεια των τελευταίων ετών έχουμε γίνει μάρτυρες μια στροφής σε ό,τι αφορά τα υποδείγματα προγραμματιστικών μοντέλων και των αρχιτεκτονικών στυλ που χρησιμοποιούνται για τη σχεδίαση και την υλοποίηση μεγάλης κλίμακας υπηρεσιοστρεφών συστημάτων. Αυτή η στροφή αποδίδεται στην υιοθέτηση του υποδείγματος προσανατολισμένου σε πόρους (ΠΣΠ) (resource-oriented paradigm) το οποίο είναι εμπνευσμένο από αρχιτεκτονικές προσεγγίσεις και ιδιότητες συναφείς με τον Παγκόσμιο Ιστό. Το σχεδιαστικό υπόδειγμα ΠΣΠ στοχεύει στη βελτίωση της διαλειτουργικότητας, στην απλότητα μέσω της ομοιομορφίας, στην ενίσχυση των δυνατοτήτων κλιμάκωσης, στη μείωση της σύζευξης μεταξύ των δομοστοιχείων και στον περιορισμό της εξάρτησης από μεταβαλλόμενα τεχνολογικά πρότυπα. Ως αποτέλεσμα, το αρχιτεκτονικό στυλ Representational State Transfer (REST), το οποίο επιβάλει τον προσανατολισμό σε πόρους στην αρχιτεκτονική του Παγκόσμιου Ιστού, έλαβε σημαντική προσοχή από την κοινότητα των μηχανικών υπηρεσιών, καθώς στοχεύει στην αποδοτική γεφύρωση των συστημάτων επιχειρησιακού λογισμικού και του Παγκόσμιου Ιστού. Τα τελευταία χρόνια, στην περιοχή της Τεχνολογίας Λογισμικού λαμβάνει χώρα μια ανοιχτή δημόσια συζήτηση η οποία εστιάζει στους τρόπους ορθής εφαρμογής του REST τόσο σε ό,τι αφορά στην ανάπτυξη νέων επιχειρησιακών συστημάτων, όσο και στην προσαρμογή υφιστάμενων συστημάτων υπηρεσιών, έτσι ώστε οι πάροχοι υπηρεσιών να προσφέρουν τις δυνατότητες των υπηρεσιών τους μέσω επιπρόσθετων διεπαφών και συγκεκριμένα διεπαφών προσανατολισμένων σε πόρους. Επιπλέον, η συνύπαρξη υπηρεσιών ΠΣΔ και υπηρεσιών ΠΣΠ σε επιχειρησιακά περιβάλλοντα λογισμικού, εισάγει έναν αριθμό νέων προκλήσεων συναφών με την ανάπτυξη και την εξέλιξη υπηρεσιοστρεφών εφαρμογών που διαθέτουν διττές διεπαφές. Στην παρούσα διατριβή, μελετάμε δύο κύρια προβλήματα. Το πρώτο πρόβλημα αφορά σε ζητήματα και προκλήσεις που σχετίζονται με την προσαρμογή υπηρεσιών ΠΣΔ σε RESTful αρχιτεκτονικές. Για τον σκοπό αυτό, προτείνουμε μια διαδικασία κι ένα περιβάλλον-πλαίσιο το οποίο επιτρέπει την μοντελοποίηση, την ανάλυση και τον μετασχηματισμό διεπαφών ΠΣΔ έτσι ώστε να προσδιοριστούν αντίστοιχες διεπαφές ΠΣΠ. Το δεύτερο πρόβλημα αφορά στην ευθυγραμμία μεταξύ διεπαφών ΠΣΔ και διεπαφών ΠΣΠ. Πιο συγκεκριμένα, μελετάμε πως τα δύο υποδείγματα σχετίζονται εννοιολογικά και αρχιτεκτονικά και προτείνουμε μια μέθοδο κι ένα περιβάλλον-πλαίσιο για την εξέταση της υποκαταστασιμότητας μεταξύ υπηρεσιών που παρέχουν παρόμοια λειτουργικότητα αλλά εκτίθενται μέσω διαφορετικών σχεδιαστικών υποδειγμάτων διεπαφών. Η λύση σε αυτά τα προβλήματα είναι σημαντική καθώς, πρώτον, επιτρέπει την μεθοδολογική ανάλυση της σχέσης του προσανατολισμού σε διαδικασίες και του προσανατολισμού σε πόρους, δεύτερον, επιτρέπει τη σχεδίαση και την υλοποίηση περιβαλλόντων-πλαισίων τα οποία μπορούν να χρησιμοποιηθούν για να παρέχουν δυνατότητες υπηρεσιών μέσω διττών διεπαφών και τρίτον, επιτρέπει τη συνεπή συνεξέλιξη υπηρεσιών διττών διεπαφών ως προς την οπτική του παρόχου υπηρεσιών και την επικύρωση της υποκαταστασιμότητας υπό την οπτική του καταναλωτή υπηρεσιών, όταν απαιτείται η εξέταση εναλλακτικών υπηρεσιών. Οι προσεγγίσεις και τα περιβάλλοντα-πλαίσια που προτείνονται αξιολογούνται μέσω μιας σειράς πειραμάτων και περιπτωσιολογικών μελετών. Τα αποτελέσματα αυτών δείχνουν ότι οι προτεινόμενες προσεγγίσεις είναι αποτελεσματικές, ότι μπορούν να χρησιμοποιηθούν για την ανάπτυξη προσαρμογέων υποδειγμάτων διεπαφών που λειτουργούν κατά το χρόνο εκτέλεσης κι ότι μπορούν να παρέχουν μια μέθοδο για την αποδοτική αποτίμηση της υποκαταστασιμότητας και της ευθυγραμμίας μεταξύ διιστάμενων, σχετικά με το ακολουθούμενο υπόδειγμα σχεδίασης, διεπαφών υπηρεσιών.

Λέξεις κλειδιά: Αρχιτεκτονική λογισμικού, Υπηρεσιοστρεφής αρχιτεκτονική, Προσαρμογή λογισμικού

Abstract

The area of Service-Oriented Computing (SOC) has emerged as a computing paradigm for designing and constructing distributed systems. In this paradigm, software components provide and consume services, exposed through service interfaces. The de facto approach for service interface design in service-oriented computing follows a procedure-oriented paradigm, where service capabilities are decomposed and exposed as collections of related service operations, which can be invoked using a traditional remote procedure call (RPC) type of access.
However, over the past few years we have witnessed a paradigm shift on the programming models and the architectural styles that have been used to design and implement large-scale service-oriented systems. This paradigm shift is attributed to the adoption of the resource-oriented service design paradigm, which is inspired by architectural approaches and properties pertinent to the World Wide Web. Such resource-oriented design paradigm aims for enhancing interoperability and simplicity through uniformity, improving scalability, allowing for loose coupling and limiting dependence on shifting technology standards. As a result, the Representational State Transfer (REST), which has been proposed as an architectural style that establishes the resource-oriented nature of the Web's architecture, gained considerable attention in the service engineering community, since it aims for an efficient bridging of enterprise software systems and the Web. An open debate and discussion in the software engineering community focuses on how to properly apply REST not only for the development of new enterprise systems but also, for the adaptation of existing service-oriented systems so that service providers can offer their service capabilities also through resource-oriented interfaces. Furthermore, the co-existence of procedure-oriented services and resource-oriented services in enterprise environments, introduces a number of new challenges pertaining to developing, maintaining and evolving dual interface service-oriented applications. In this dissertation, we investigate two main problems. The first problem deals with issues and challenges related to the adaptation of procedure-oriented service systems to RESTful architectures. For this purpose, we propose a process and a framework, which allow for the analysis, modeling, extraction, and transformation of procedure-oriented interface models in order to identify and yield corresponding resource-oriented interfaces. The second problem deals with the alignment of procedure-oriented and resource-oriented service interfaces. More specifically, we examine how the two paradigms are related conceptually and architecturally, and we propose a method and a framework for deciding substitutability between services that offer similar functionality but are exposed through different interface paradigms (paradigm-divergent interfaces). The solution to these problems is important as first, it allows for a methodological analysis of how procedure-orientation and resource-orientation relate to each other, second, it allows for the design and implementation of run-time frameworks that can be used to offer service capabilities in dual interfaces and third, it allows for the provider-side consistent co-evolution of dual service interfaces and for the consumer-side substitution validation when alternative services need to be considered. The proposed approaches and frameworks are evaluated through a series of experiments and case studies. The results indicate that the proposed approaches are tractable, can be used to develop run-time procedure-oriented to resource-oriented interface adapters, and provide a method to efficiently assess substitutability and alignment of paradigm-divergent service interfaces.

Keywords: Software architecture, Service-oriented architecture, Software adaptationΑρχιτεκτονική λογισμικού, Υπηρεσιοστρεφής αρχιτεκτονική, Προσαρμογή λογισμικού

This dissertation is dedicated to my parents and my wife.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Kostas Kontogiannis, for his guidance, encouragement and enthusiasm that made this work possible. I would like to express my sincere gratitude for the opportunities he gave me to investigate exciting research challenges and work with interesting people.

I would like to deeply thank Professors Yannis Vassiliou and Timos Sellis for always being helpful and supportive to me. I am also very thankful to Professors Nikos Papaspyrou, Dimitrios Askounis, Stavros Christodoulakis and Ying Zou for being members of my dissertation committee and for sharing their ideas and interest in my research.

I am especially grateful to IBM CAS Research for providing me the privilege to be an IBM CAS Ph.D. Fellow from 2010 to 2015. I had the invaluable opportunity to work closely with exceptional software engineering professionals from IBM and to spend several months at the IBM Canada Lab. I owe special thanks to Chris Brealey for his insight, collaboration and incredible support. I am very grateful to Joanna Ng for her vision and ideas and I would also like to thank Tinny Ng, Dianna Lau, Jimmy Lo, Alex Lau and Debbie Kilbride for their help and support. I am thankful to ICCS and its people for handling all issues related to my fellowship as well as to my employer, National Bank of Greece, and especially its IT Division, that allowed me to be on leave of absence for a long period. I also appreciate the help and support from my coworkers at NBG.

I would like to thank my colleagues at the Software Engineering Lab of NTUA and especially George Chatzikonstantinou and Thodoris Kalamatianos for the collaboration, the long discussions and their friendship.

I want to thank all my friends and relatives in Athens, Larisa and Canada, for their support and for taking my mind off research.

I am grateful to my sisters, Maria Athanasopoulou and Theodora Athanasopoulou, who formed an unbeatable team that was always there for me.

I owe my deepest gratitude to my parents, George Athanasopoulos and Stella Athanasopoulou. Nothing would be possible without their support and inspiration.

Last, but not least, I am truly and deeply grateful to my beloved wife, Vasiliki Xanthi. Her devotion, patience, support and commitment paved the way to completing my Ph.D. journey.

BIOGRAPHICAL SKETCH

Michael Athanasopoulos graduated from the School of Electrical and Computer Engineering at National Technical University of Athens in 2008 and joined the Ph.D program during the same year. Michael joined National Bank of Greece in 2005, where he is working as an in-house software engineer in electronic banking applications. Also, from 2010 to 2015 he has been a Ph.D. Fellowship Student of the Center of Advanced Studies, IBM Canada.

Refereed publications:

- Michael Athanasopoulos, Kostas Kontogiannis. *Extracting REST Resource Models from Procedure-oriented Service Interfaces*. Journal of Systems and Software, vol. 100, pp. 149-166. 2015, Elsevier.
- George Chatzikonstantinou, Michael Athanasopoulos, Kostas Kontogiannis. *Task Specification and Reasoning in Dynamically Altered Contexts*. CAISE 2014. In Advanced Information Systems Engineering, vol. 8484, pp. 625-639. 2014, Springer International Publishing.
- George Chatzikonstantinou, Michael Athanasopoulos, Kostas Kontogiannis. *Towards a Goal Driven Task Personalization Specification Framework*. 1st International Workshop on Personalized Web Tasking (PWT 2013 at IEEE Ninth World Congress on Services), pp. 180-184. 2013, CA, USA.
- Michael Athanasopoulos, Kostas Kontogiannis, Chris Brealey. *Considerations of Adapting Service-offering Components to RESTful Architectures*. In book *Migrating to SOA and Cloud Environments: Challenges in Service Oriented Architecture and Cloud Computing Environments*, eds. A. D. Ionita, G. Lewis and M. Litoiu, pp. 303-331. 2012, IGI Global.
- Michael Athanasopoulos, Kostas Kontogiannis, Chris Brealey. *Towards an interpretation framework for assessing interface uniformity in REST*. In Proceedings of the Second International Workshop on RESTful Design (WS-REST '11 at WWW), eds. Cesare Pautasso and Erik Wilde, pp. 47-50. 2011, ACM, New York, USA.
- Michael Athanasopoulos, Kostas Kontogiannis. *Identification of REST-like Resources from Legacy Service Descriptions*. In Proceedings of the 17th

Working Conference on Reverse Engineering (WCRE) (Boston, USA), pp. 215-219. 2010, IEEE.

Workshop organization:

- Michael Athanasopoulos, Kostas Kontogiannis, Chris Brealey: *4th Workshop on Leveraging REST and Web Technologies in Enterprise Service Systems*. In Proceedings of the 2013 Conference of the Center for Advanced Studies (CASCON). Proceedings pp. 360-362.
- Michael Athanasopoulos, Kostas Kontogiannis, Chris Brealey: *3rd Workshop on Leveraging REST and Web Technologies in Enterprise Service Systems*. In Proceedings of the 2012 Conference of the Center for Advanced Studies (CASCON). Proceedings pp 243-245.
- Michael Athanasopoulos, Kostas Kontogiannis, and Chris Brealey. *2nd Workshop on Leveraging REST in Enterprise Service Systems*. At the 2011 Conference of the Center for Advanced Studies (CASCON). Proceedings pp. 342-344.
- Kostas Kontogiannis, Ying Zou, Chris Brealey, Michael Athanasopoulos: *Issues and challenges leveraging REST architectural style in enterprise service systems*. At the 2010 Conference of the Center for Advanced Studies (CASCON). Proceedings pp 368-370.

Demos/Posters:

- 2nd Workshop on Personal Web Tasking 2013: *Goal-based Web Task Templates and Reasoning: a resource-oriented application case*. Michael Athanasopoulos, George Chatzikonstantinou, Kostas Kontogiannis.
- CASCON 2013, Technology Showcase Exhibit: *Hypermedia-enabled Service Adaptation*. Michael Athanasopoulos, Kostas Kontogiannis, Chris Brealey, Joanna Ng, Diana Lau, Tinny Ng.
- CASCON 2012, Technology Showcase Exhibit: *RESTful Adaptation of Procedure-Oriented Services*. Michael Athanasopoulos, Kostas Kontogiannis, Chris Brealey.

- CASCON 2011, Technology Showcase Exhibit: *REST Resource Extraction*. Michael Athanasopoulos, Kostas Kontogiannis, Chris Brealey, Alex Lau.
- CASCON 2010, Technology Showcase Exhibit: *REST in the Enterprise*. Michael Athanasopoulos, Kostas Kontogiannis, Chris Brealey, Alex Lau.

Fellowship and Awards:

- *CAS Research Ph.D. Fellowship*, IBM Canada Ltd., 2010 - 2015.
- *CAS Research Student of the Year*, CAS Research, IBM Canada Lab, November 2015.
- *CAS Research Project of the Year*, CAS Research, IBM Canada Lab, November 2013.

Teaching Experience:

- Teaching assistant, Software Engineering (2009-2014), School of ECE at NTUA.
- Teaching assistant, Programming Techniques (2010-2012), School of ECE at NTUA.

ΠΕΡΙΕΧΟΜΕΝΑ

ΛΙΣΤΑ ΠΙΝΑΚΩΝ

ΛΙΣΤΑ ΣΧΗΜΑΤΩΝ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Στον ολοένα και πιο διασυνδεδεμένο σύγχρονο κόσμο, είναι αναμενόμενο πληροφορίες που παράγονται από ένα σύστημα να χρησιμοποιούνται από πολλές διαφορετικές διεργασίες, οι οποίες ενδέχεται να εκτελούνται παράλληλα, σε διαφορετικές συσκευές και σε διαφορετικές περιοχές. Εφαρμογές δικτύων (π.χ. εφαρμογές Παγκοσμίου Ιστού, παιχνίδια πολλών παικτών), τηλεπικοινωνιακά συστήματα (π.χ. δίκτυα ασύρματων αισθητήρων), εφαρμογές επιστημονικής υπολογιστικής (π.χ. grid computing) αποτελούν περιοχές στις οποίες οι υπολογισμοί είναι κατανεμημένοι στο φυσικό επίπεδο, ή εκτελούνται μέσω αυτόνομων διεργασιών οι οποίες επικοινωνούν μεταξύ τους μέσω μετάδοσης και ανταλλαγής μηνυμάτων. Λόγω των παραπάνω, η κατανεμημένη υπολογιστική (distributed computing) έχει καταστεί ένα πρωτεύον πεδίο μελέτης στην επιστήμη υπολογιστών καθώς τα κατανεμημένα συστήματα είναι σε θέση να παρέχουν λύσεις σε ένα ευρύ φάσμα προβλημάτων και να διευκολύνουν τη μεγάλης κλίμακας επικοινωνία και συνεργασία ανθρώπων και συστημάτων. Στο πλαίσιο αυτό, ο ρόλος του λογισμικού στην υλοποίηση κατανεμημένων συστημάτων είναι κεντρικός, καθώς επιτρέπει την αλληλεπίδραση συστημάτων, συσκευών και ανθρώπων στα πλαίσια εκτέλεσης υπολογιστικών εργασιών.

Επιπλέον, κατά τη διάρκεια της τελευταίας δεκαετίας, σχεδόν όλοι οι τομείς οικονομικής δραστηριότητας έχουν επηρεαστεί σημαντικά από τα ραγδαίως μεταβαλλόμενα επιχειρηματικά περιβάλλοντα και τις επαναστατικές προσεγγίσεις που εισήχθησαν στις επιχειρηματικές λειτουργίες και δράσης, λόγω της τεχνολογικής προόδου στον τομέα του επιχειρησιακού λογισμικού (enterprise software). Σήμερα, είναι ευρέως αποδεκτό ότι το επιχειρησιακό λογισμικό παίζει κρίσιμο ρόλο ως προς τη δυνατότητα ενός οργανισμού να ανταποκρίνεται σε συνεχείς και συχνά απρόβλεπτες μεταβολές των αναγκών του επιχειρηματικού περιβάλλοντος στο οποίο δραστηριοποιείται και ταυτόχρονα αποτελεί ρυθμιστικό παράγοντα της ικανότητάς του να υλοποιεί και να επιτυγχάνει τη στρατηγική του. Παρ' όλα αυτά, καθώς το λογισμικό εισήχθη στους οργανισμούς με στόχο την αυτοματοποίηση εργασιών που προηγουμένως υλοποιούνταν χειροκίνητα, το αντικείμενο κάθε μονάδας λογισμικού ήταν σχετικά περιορισμένο και ανεξάρτητο από το αντικείμενο άλλων μονάδων. Κατά συνέπεια, στα συστήματα αυτά δόθηκε περιορισμένη προσοχή στις δυνατότητες ολοκλήρωσης και διαλειτουργικότητας, γεγονός που οδήγησε στα λεγόμενα

μονολιθικά συστήματα ή *silos*. Παρόλο που τα επιχειρηματικά συστήματα λογισμικού συνέχισαν να γίνονται ολοένα και πιο αποκεντρωμένα, ανεξάρτητα και ετερογενή, οι απαιτήσεις για αλληλεπίδραση και διαλειτουργικότητα αποδεικνύονταν συστηματικά όλο και πιο κρίσιμες για τη δυνατότητα ενός οργανισμού να λειτουργεί αποδοτικά και να είναι ανταγωνιστικός. Αποτέλεσμα του γεγονότος αυτού ήταν η ανάδειξη ενός συνόλου υποδειγμάτων και τεχνοτροπιών κατανεμημένης υπολογιστικής (π.χ. CORBA) οι οποίες αφορούν στο πως μπορεί να κατασκευαστεί ή να προσαρμοστεί το επιχειρησιακό λογισμικό των συστημάτων έτσι ώστε να αντιμετωπιστούν μια σειρά ζητημάτων και προκλήσεων ολοκλήρωσης συστημάτων. Την ίδια ώρα, η αυξημένη ανάγκη για αντιμετώπιση των ραγδαίως εξελισσόμενων επιχειρηματικών αναγκών, κατέστησε αρκετές από αυτές τις προσπάθειες ολοκλήρωσης συστημάτων ως ανεπαρκείς, εξ' αιτίας διαφόρων εγγενών περιορισμών (π.χ. υψηλός βαθμός σύζευξης μεταξύ των αλληλεπιδρώντων δομοστοιχείων, υψηλά τεχνολογικά εμπόδια και προϋποθέσεις, σημαντικές επιβαρύνσεις ως προς την επικοινωνία και την επίδοση των συστημάτων λόγω μη αποδοτικών πρωτοκόλλων, κ.ά.). Σύντομα έγινε σαφές ότι αποτελεί σημαντική πρόκληση και ταυτόχρονα ζητούμενο η αρμονία μεταξύ επιχειρηματικών στόχων και δυνατοτήτων που παρέχουν οι τεχνολογίες πληροφορικής και πιο συγκεκριμένα τα συστήματα λογισμικού, εισάγοντας την έννοια της ευθυγράμμισης επιχείρησης - τεχνολογιών πληροφορικής (*business-IT alignment*). Η ευθυγράμμιση επιχείρησης - τεχνολογιών πληροφορικής αναγνωρίστηκε ως ζήτημα ζωτικής σημασίας για την επίτευξη των επιχειρηματικών στόχων και σύντομα διαπιστώθηκε ότι είναι απαραίτητη η μετάβαση σε ένα νέο, περισσότερο ευέλικτο υπολογιστικό υπόδειγμα το οποίο θα εξυπηρετεί καλύτερα το στόχο της παραπάνω ευθυγράμμισης. Ένα τέτοιο υπολογιστικό υπόδειγμα θα πρέπει να αναβαθμίζει την ευελιξία και ευκινησία ενός οργανισμού, μέσω βελτίωσης των μέσων που έχει στη διάθεσή του για να αναγνωρίζει, να χρησιμοποιεί και να μεταφέρει πληροφορίες εντός και εκτός των συνόρων του, δίνοντας έτσι τη δυνατότητα να χρησιμοποιηθούν αποδοτικά οι πληροφοριακές και υπολογιστικές δυνατότητές του. Κατά συνέπεια, προηγούμενες προσεγγίσεις που ήταν προσανατολισμένες σε εξειδικευμένες υπολογιστικές πλατφόρμες ή σε τεχνολογικά εξαρτημένες προσεγγίσεις στη διασύνδεση υπολογιστικών στοιχείων, έπρεπε να αντικατασταθούν από μια προσέγγιση ικανή να οδηγήσει σε ανοιχτά, ευέλικτα και διαλειτουργικά συστήματα και ταυτόχρονα να εξυπηρετήσει το ζητούμενο της ευθυγράμμισης μεταξύ επιχειρηματικών στόχων και τεχνολογιών πληροφορικής.

Στο πλαίσιο αυτό, ανέκυψε η *Υπηρεσιοστρεφής Υπολογιστικής (ΥΥ)* (*Service-*

Oriented Computing (SOC)) ως ένα υποσχόμενο υπόδειγμα κατανεμημένης υπολογιστικής για επιχειρηματικά συστήματα. Η ΥΥ ορίζει και χρησιμοποιεί τις *υπηρεσίες* (*services*) ως θεμελιώδη υπολογιστικά στοιχεία, τα οποία είναι κατανεμημένα εντός και πέρα των ορίων του οργανισμού. Στην ΥΥ, μια υπηρεσία δηλώνει μια αυτόνομη, μη συσχετισμένη και τεχνολογικά ανεξάρτητη μονάδα λογισμικού, η οποία δίνει τη δυνατότητα πρόσβασης σε ένα σύνολο επαναχρησιμοποιήσιμων και εννοιολογικά διακριτών δυνατοτήτων λογισμικού και προσδιορίζει ένα σύνολο κανόνων και πολιτικών που υπαγορεύουν την αποδεκτή χρήση της. Η εννοιολογική θεμελίωση της υπηρεσιοστρεφούς υπολογιστικής είναι ο *προσανατολισμός σε υπηρεσίες* (*service-orientation*) [45], μια αρχιτεκτονική και σχεδιαστική προσέγγιση που επιχειρεί να αντιμετωπίσει ένα σύνολο ανεπαρκειών προηγούμενων προσεγγίσεων, προτείνοντας ορισμένες αρχές ανάπτυξης και διασύνδεσης δομοστοιχείων λογισμικού. Πιο συγκεκριμένα, οι αρχές του προσανατολισμού σε υπηρεσίες περιλαμβάνει τη χρησιμοποίηση προτυποποιημένων συμβολαίων υπηρεσιών, μειωμένης σύζευξης υπηρεσιών, αφαίρεσης υπηρεσιών (σχετικά με τη λογική που υλοποιούν), επαναχρησιμοποίησης υπηρεσιών, αυτονομίας υπηρεσιών, απουσία κατάστασης (statelessness) υπηρεσιών, δυνατότητα ανακάλυψης υπηρεσιών και δυνατότητα σύνθεσης υπηρεσιών. Οι αρχές αυτές προάγουν και συνεισφέρουν στη δυνατότητα διαλειτουργικότητας των υπηρεσιών, η οποία αποτελεί ένα από τους κεντρικούς στόχους της υπηρεσιοστρεφούς υπολογιστικής. Επιπρόσθετα, οι υπηρεσίες αποδείχθηκαν ως μια βολική αντιληπτική αφαίρεση τόσο για τους μηχανικούς λογισμικού όσο και για τους αναλυτές επιχειρηματικών λειτουργιών, επιτρέποντας έτσι για την καλύτερη κατανόηση των παροχών λειτουργικότητας και των προσδοκιών κατά την κατασκευή κατανεμημένων εφαρμογών. Ως εκ τούτου, η προσανατολισμένη σε υπηρεσίες συλλογιστική θεωρείται ότι έχει υπερβεί τα όρια των τεχνολογιών πληροφορικής.

Εστιάζοντας στη διαδικασία ανάπτυξης λογισμικού, η ΥΥ επηρεάζει σημαντικά τη σχεδίαση του λογισμικού και ειδικότερα το υψηλό αφαιρετικό επίπεδο της αρχιτεκτονικής. Η *αρχιτεκτονική λογισμικού* είναι ένας όρος που χρησιμοποιείται για να υποδηλώσει έναν επιστημονικό κλάδο της τεχνολογίας λογισμικού, καθώς και για να περιγράψει συγκεκριμένα συστήματα. Με άλλα λόγια, ο όρος αυτός χρησιμοποιείται για να υποδηλώσει τόσο τη διαδικασία όσο και το αποτέλεσμα της εφαρμογής της διαδικασίας αυτής. Πιο συγκεκριμένα, ως επιστημονικός κλάδος, η αρχιτεκτονική λογισμικού αναφέρεται στη δημιουργία, διαχείριση και επεξεργασία δομών υψηλού επιπέδου που παρέχουν απεικονίσεις ενός συστήματος λογισμικού

και διευκολύνουν την κατανόηση της συμπεριφοράς του συστήματος. Τέτοιες δομές μπορεί να υποδηλώνουν μοτίβα αλληλεπιδράσεων δομοστοιχείων, ιδιότητες των στοιχείων του συστήματος ή σχέσεις μεταξύ τους. Η αρχιτεκτονική λογισμικού ενός συγκεκριμένου συστήματος λογισμικού είναι μια αφαίρεση των στοιχείων του, η οποία καταγράφεται μέσω δομών υψηλού επιπέδου που παρέχουν διαφορετικές οπτικές στα στοιχεία του συστήματος, τις σχέσεις τους, τις ιδιότητες τους και τα χαρακτηριστικά τους. Στο Κεφάλαιο 2 συζητάμε περαιτέρω την έννοια της αρχιτεκτονικής λογισμικού, καθώς παρέχει ένα πλαίσιο εντός του οποίου αναπτύσσεται η παρούσα διατριβή. Ομοίως με αρχιτεκτονικές φυσικών δομών (π.χ. κτίρια), μία αρχιτεκτονική λογισμικού μπορεί να είναι σύμφωνη με μία ή περισσότερες αρχιτεκτονικές τεχνοτροπίες ή *αρχιτεκτονικά στυλ* (*architectural style*). Ένα αρχιτεκτονικό στυλ προσδιορίζει ένα σύνολο αρχιτεκτονικών αποφάσεων σχεδίασης που ορίζονται σε ένα συγκεκριμένο πλαίσιο, υπό τη μορφή των περιορισμών που εφαρμόζονται σε τύπους αρχιτεκτονικών στοιχείων, τα χαρακτηριστικά τους και τις σχέσεις τους και οι οποίοι είναι σε θέση να αποδώσουν-επάγουν συγκεκριμένες ιδιότητες στο σύστημα που προκύπτει.

Με βάση τα παραπάνω, προτάθηκε η *Υπηρεσιοστρεφής Αρχιτεκτονική (ΥΑ)* (*Service-Oriented Architecture (SOA)*) ως το βασικό αρχιτεκτονικό στυλ στην περιοχή της ΥΥ, προκειμένου να καθορίσει ένα σύνολο αρχιτεκτονικών στοιχείων, δομοστοιχείων και ενδιαφερόμενων μερών και να καθορίσει χαρακτηριστικά, ρόλους, σχέσεις και περιορισμούς που πρέπει να ακολουθούνται προκειμένου μια αρχιτεκτονική να είναι προσανατολισμένο σε υπηρεσίες. Τα χαρακτηριστικά και οι αρχές της ΥΑ έχουν ερευνηθεί εκτενώς [45], [84], [72], [58]. Σταδιακά ο όρος SOA άρχισε να χρησιμοποιείται για την αναφορά σε ευρύτερα ζητήματα του κύκλου ζωής υπηρεσιοστρεφών συστημάτων και για το λόγο αυτό σημειώνουμε ότι στην παρούσα διατριβή, εξετάζουμε την ΥΑ από τη σκοπιά της αρχιτεκτονικής λογισμικού. Ένα βασικό χαρακτηριστικό της ΥΑ είναι ο τρόπος με τον οποίο οι δυνατότητες των υπηρεσιών γίνονται προσιτές σε δυνητικούς καταναλωτές, δηλαδή, ο τρόπος με τον οποίο είναι δημοσιεύονται και εκτίθενται. Πιο συγκεκριμένα, μια υπηρεσία οφείλει να τηρεί ένα *συμβόλαιο* (*contract*) το οποίο καθορίζει τον σκοπό της υπηρεσίας και τις δυνατότητές της [45]. Ένα συμβόλαιο υπηρεσίας επιτρέπει την έκθεση των δυνατοτήτων της υπηρεσίας μέσω τελικών σημείων αλληλεπίδρασης. Αυτά τα τελικά σημεία προσδιορίζονται συνολικά ως η *διεπαφή* (*interface*) της υπηρεσίας. Για την ακρίβεια, η σχεδίαση διεπαφών υπηρεσιών αντιμετωπίζεται ως ένα θεμελιώδες θέμα στην ΥΑ [15] και θεωρείται ως ένα σημαντικό μέλημα κατά τη διάρκεια της α-

νάπτυξης αλλά και εξέλιξης του συστήματος, προκειμένου να διατηρηθεί η συνοχή των διεπαφών υπηρεσίας, η αξιοπιστία και η δυνατότητα διακυβέρνησης.

Στην ΥΥ, οι διεπαφές υπηρεσίας αναφέρονται επίσης ως *διεπαφές προγραμματισμού εφαρμογών* (*application programming interfaces (API)*), ή API υπηρεσιών, καθώς παρέχουν τις προγραμματιστικές αφαιρέσης για τη δημιουργία και σύνθεση κατανεμημένων εφαρμογών. Ωστόσο, πρέπει να σημειωθεί ότι ο όρος API δεν είναι ειδικός ως προς τη χρήση του για τις διεπαφές υπηρεσιών, καθώς είναι σύνηθες να χρησιμοποιείται για να υποδηλώσει τις λειτουργίες, τα δεδομένα εισόδου, εξόδου και τους τύπους που ένα δομοστοιχείο λογισμικού παρέχει, όταν χρησιμοποιείται ως δομικό στοιχείο μιας εφαρμογής —δηλαδή, είτε με τη μορφή βιβλιοθήκης, είτε ως τοπική ή απομακρυσμένης υπηρεσίας. Επιπλέον, έχουν οριστεί *γλώσσες περιγραφής διεπαφών* (*interface description languages (IDL)*) προκειμένου να διευκολυνθεί η τυπική προδιαγραφή των API υπηρεσιών. Οι εν λόγω προδιαγραφές διεπαφής περιγράφουν κατά κανόνα πως η λειτουργικότητα και οι πληροφορίες που μια υπηρεσία είναι σε θέση να παρέχει, μπορούν να γίνουν προσιτές και προσβάσιμες από εφαρμογές-πελάτες. Οι προδιαγραφές διεπαφών μπορούν να επεκταθούν ή να εμπλουτιστούν με περαιτέρω περιγραφές και κανόνες, προκειμένου να καλυφθούν πρόσθετες απαιτήσεις, όπως η ασφάλεια, η ποιότητα των παρεχόμενων υπηρεσιών, η προδιαγραφή της ενδο-υπηρεσιακής συμπεριφοράς και η σημασιολογική περιγραφή των δεδομένων και των δυνατοτήτων των υπηρεσιών.

Η πρώτη τεχνολογική πλατφόρμα που χρησιμοποιήθηκε ευρέως για την πραγμάτωση υπηρεσιοστρεφών αρχιτεκτονικών είναι οι SOAP υπηρεσίες Ιστού (SOAP Web services), βασισμένες στη γλώσσα WSDL [155] για την προδιαγραφή των διεπαφών τους. Τα Web services ορίζουν μια προτυποποιημένη μέθοδο διαλειτουργικότητας μεταξύ διαφορετικών εφαρμογών λογισμικού οι οποίες εκτελούνται σε ποικίλες πλατφόρμες, ακολουθώντας τον προσανατολισμό σε υπηρεσίες. Συγκεκριμένα, ο ορισμός των Web services που χρησιμοποιείται από την αντίστοιχη ομάδα εργασίας του World Wide Web Consortium (W3C) (Web Service Architecture (WSA) Working Group [157]) δηλώνει ότι 'ένα Web service είναι ένα σύστημα λογισμικού σχεδιασμένο να υποστηρίζει τη διαλειτουργική αλληλεπίδραση μηχανής με μηχανή μέσω κάποιου δικτύου. Έχει μια διεπαφή που περιγράφεται σε μια μορφή επεξεργάσιμη από μηχανές (machine-processable) (συγκεκριμένα μορφή WSDL). Άλλα συστήματα αλληλεπιδρούν με το Web service με τρόπο που προδιαγράφεται από την περιγραφή του, χρησιμοποιώντας SOAP μηνύματα, τα οποία τυπικά

μεταφέρονται χρησιμοποιώντας το πρωτόκολλο HTTP με μια XML σειριοποίηση σε συνδυασμό με άλλα πρότυπα που σχετίζονται με τον Παγκόσμιο Ιστό' [156]. Όπως είναι φυσικό, ο ορισμός αυτός συνάδει με το εννοιολογικό μοντέλο μιας υπηρεσίας στην υπηρεσιοστρεφή υπολογιστική. Η παραπάνω προσέγγιση πραγμάτωσης υπηρεσιοστρεφών αρχιτεκτονικών, η οποία συνεχίζει να χρησιμοποιείται ευρέως για την κατασκευή υπηρεσιοστρεφών συστημάτων, αναφέρεται και ως *παραδοσιακή ΥΑ*, για να τη διαφοροποίησή της από μεταγενέστερες προσεγγίσεις.

Στο πλαίσιο των Web services που ορίστηκαν παραπάνω, το ακρώνυμο WSDL αναλύεται σε Web Service Description Language [44], [129] και αναφέρεται σε μια γλώσσα περιγραφής διεπαφών των Web services βασισμένη στη γλώσσα XML. Ένα WSDL έγγραφο προσδιορίζει τα διαφορετικά σημεία αλληλεπίδρασης με μια υπηρεσία, τα οποία αποτελούν ένα σύνολο τελικών σημείων ικανών να ενεργήσουν έναντι μηνυμάτων SOAP που στέλνονται από καταναλωτές της υπηρεσίας. Το SOAP (ένα ακρώνυμο για το Simple Object Application Protocol) είναι ένα πρωτόκολλο ορισμένο στο πλαίσιο του WSA που επιτρέπει την ανταλλαγή δομημένης πληροφορίας σε XML μορφή μεταξύ παρόχων υπηρεσιών και καταναλωτών και το οποίο λειτουργεί πάνω από άλλα πρωτόκολλα επιπέδου εφαρμογής όπως το HTTP. Αναφερόμαστε στην παραδοσιακή ΥΑ ως *προσανατολισμένη σε διαδικασίες (ΠΣΔ)*, καθώς η εκτιθέμενη λειτουργικότητα από τις υπηρεσίες που προσδιορίζονται από WSDL έγγραφα αναλύεται ρητά με τον ορισμό λειτουργιών (operations) (οι οποίες περιλαμβάνονται είτε σε στοιχεία WSDL 1.1 `PortType` ή σε στοιχεία WSDL 2.0 `Interface`).

Η έκθεση της λειτουργικότητας μιας υπηρεσίας με τη χρήση λειτουργιών, όπως συμβαίνει στα Web services, μπορεί να αποδοθεί τόσο σε τεχνολογικούς, όσο και σε ιστορικούς λόγους. Η έννοια της λειτουργίας έχει χρησιμοποιηθεί ως μια αφαίρεση για τη σήμανση και την ενθυλάκωση λειτουργικότητας ήδη από τις αρχές του προστακτικού προγραμματισμού. Ως αποτέλεσμα, η παραδοσιακή λειτουργιοκεντρική προσέγγιση στην ΥΑ βασίζεται σε ένα ΠΣΔ υπόδειγμα έκθεσης δυνατοτήτων υπηρεσιών.

Η χρήση Web services για την πραγμάτωση υπηρεσιοστρεφών αρχιτεκτονικών επιχειρεί να πετύχει την ολοκλήρωση και τη διαλειτουργικότητα συστημάτων που βασίζονται στην Αρχιτεκτονική του Παγκόσμιου Ιστού (Web Architecture), χρησιμοποιώντας ως πλατφόρμα τα πρωτόκολλα και τα πρότυπα του ίδιου του Παγκόσμιου Ιστού. Παρ' όλα αυτά, σε πολλές περιπτώσεις η προσέγγιση των Web services

έχει διαφοροποιηθεί και αποκλίνει από την καθαρολογική άποψη της συμμόρφωσης στις αρχές της Αρχιτεκτονικής του Παγκόσμιου Ιστού, όπως οι αυτές διατυπώνονται από το πρωτεύον αρχιτεκτονικό στυλ του Παγκόσμιου Ιστού, το Representational State Transfer ή REST [53]. Υπό αυτή την έννοια, έγινε γρήγορα αντιληπτό ότι εφόσον το REST παρέχει ένα αποτελεσματικό περιβάλλον-πλαίσιο για τον ορισμό και τη χρησιμοποίηση υπηρεσιών δια μέσου του Παγκοσμίου Ιστού, θα πρέπει να χρησιμοποιηθεί ως βάση για το σκοπό αυτό. Η παρατήρηση αυτή οδήγησε στην εισαγωγή της σχεδίασης υπηρεσιών προσανατολισμένων σε πόρους (ΠΣΠ) και στην αρχιτεκτονική τεχνοτροπία προσανατολισμένης σε πόρους (resource-oriented architecture - (ROA)) [128], ως ένα διακριτό αρχιτεκτονικό στυλ στην περιοχή της ΥΥ. Οι ΠΣΠ αρχιτεκτονικές βασίζονται στον ορισμό πόρων Ιστού (Web resources), αναγνωριστικών πόρων και αναπαραστάσεών τους, ως πρωτεύοντα στοιχεία των διεπαφών των ΠΣΠ υπηρεσιών και χρησιμοποιούν τυπικά το HTTP πρωτόκολλο ως το πρωτόκολλο επιπέδου εφαρμογής. Η προσέγγιση αυτή αναφέρεται συνήθως ως *RESTful υπηρεσίες Ιστού* (*RESTful Web services*) ή υπηρεσίες με REST API.

Η κοινότητα της τεχνολογίας λογισμικού έχει επί μακρόν συμμετάσχει σε μια διευρυμένη αντιπαράθεση απόψεων σχετικά με το ποια προσέγγιση πρέπει να προτιμηθεί έναντι της άλλης, κάτι που αναφέρεται συνήθως ως αντιπαράθεση *SOAP vs. REST* [122], [17], [102], [66]. Αν και τα επιχειρήματα που ανταλλάσσονται αφορούν διάφορες διαστάσεις, ένα κεντρικό σημείο της αντιπαράθεσης αυτής σχετίζεται με τη σχεδίαση των διεπαφών των υπηρεσιών και συγκεκριμένα, στον τρόπο με τον οποίο οι δυνατότητες μιας υπηρεσίας θα πρέπει να αναλύονται, να οργανώνονται και να εκτίθενται. Δίνεται ιδιαίτερη προσοχή στα θέματα αυτά, καθώς αποτελεί κοινό τόπο ότι η σχεδίαση διεπαφών παίζει σημαντικό ρόλο σε ό,τι αφορά στην υιοθέτηση και επιτυχία μιας υπηρεσίας, στην υπολογιστική επιβάρυνση που απαιτείται για την παροχή της υπηρεσίας κατά τον χρόνο εκτέλεσης, όπως και σε ένα σύνολο ιδιοτήτων της αρχιτεκτονικής του υλοποιημένου συστήματος.

Ως αποτέλεσμα των παραπάνω συζητήσεων αναγνωρίστηκαν και αναδύθηκαν δύο διακριτά υποδείγματα σχεδίασης διεπαφών υπηρεσιών, το υπόδειγμα ΠΣΔ και το υπόδειγμα ΠΣΠ το οποίο πραγματώνεται κατά κανόνα μέσω RESTful HTTP υπηρεσιών. Αν και το υπόδειγμα ΠΣΔ συνεχίζει να επικρατεί στον κόσμο του επιχειρησιακού λογισμικού, το υπόδειγμα ΠΣΠ αποκτά συνεχώς ώθηση και εμπνέει νέες αρχιτεκτονικές προσεγγίσεις στην περιοχή της υπηρεσιοστρεφούς υπολογιστικής όπως η αρχιτεκτονική μικρο-υπηρεσιών (*microservices architecture*) [77], [108].

Η συνύπαρξη ποικίλων υπηρεσιών τόσο σε ό,τι αφορά στην αρχιτεκτονική τους όσο και την υλοποίησή τους, εισάγει ένα σύνολο ενδιαφέρων προκλήσεων [107], [66]. Στην παρούσα διατριβή, εστιάζουμε σε προκλήσεις που αφορούν στη σχεδίασης διεπαφών υπηρεσιών και προτείνουμε μεθοδολογίες και περιβάλλονται-πλαίσια τα οποία επιτρέπουν *α*) την *προσαρμογή* (*adaptation*) ΠΣΔ διεπαφών υπηρεσιών σε ΠΣΠ διεπαφές, και *β*) την εξέταση της λειτουργικής ισοδυναμίας, ή *ευθυγραμμίας* (*alignment*), μεταξύ υπηρεσιών που εκτίθενται μέσω διεπαφών διαφορετικών υποδειγμάτων σχεδίασης. Παρακάτω, αναπτύσσουμε περαιτέρω την περιγραφή του προβλήματος που μελετάμε και την αιτιολόγησή του, καθώς και τους στόχους και τις συνεισφορές της παρούσας διατριβής.

1.1 Περιγραφή προβλήματος και αιτιολόγηση

Στις μέρες μας, στην πρακτική ακολουθούνται τόσο το ΠΣΔ υπόδειγμα σχεδίασης διεπαφών υπηρεσιών όσο και το αντίστοιχο ΠΣΠ υπόδειγμα, καθώς και τα δύο έχουν τα δικά τους πλεονεκτήματα και οφέλη. Υπάρχουν περιπτώσεις και σενάρια όπου κάποιος θα πρέπει να επιλέξει κάποιο από τα δύο ως περισσότερο κατάλληλο, αλλά εν γένει, και τα δύο μπορούν να χρησιμοποιηθούν για να σχεδιαστούν και να υλοποιηθούν υπηρεσιοστρεφείς αρχιτεκτονικές. Η ΠΣΔ προσέγγιση παραμένει η de facto προσέγγιση, ειδικά σε ό,τι αφορά στα συστήματα επιχειρησιακού λογισμικού αλλά όπως έχει συζητηθεί παραπάνω, το REST και η ΠΣΠ προσέγγιση συγκεντρώνει την προσοχή της κοινότητας καθώς επιτρέπει την κατασκευή συστημάτων τα οποία βρίσκονται σε αντιστοιχία με την Αρχιτεκτονική του Παγκόσμιου Ιστού, ένα χαρακτηριστικό το οποίο είναι ιδιαιτέρως σημαντικό στις μοντέρνες εφαρμογές που βασίζονται στο Παγκόσμιο Ιστό, σε εφαρμογές mash-up και αρχιτεκτονικές microservices. Η διάσταση-απόκλιση των δύο αυτών υποδειγμάτων σχεδίασης διεπαφών υπηρεσιών, εισάγει ένα σύνολο προκλήσεων για τους αρχιτέκτονες λογισμικού και τους μηχανικούς, δύο από τις οποίες εξετάζονται στην παρούσα διατριβή. Στις παραγράφους που ακολουθούν, εισάγουμε και συζητάμε αυτές τις προκλήσεις και προχωρούμε στην αιτιολόγηση και στα κίνητρα πίσω από τη διερεύνησή τους.

Όπως συζητήθηκε προηγουμένως, η βασισμένη σε υπηρεσίες υπολογιστική, στηρίζεται σε διακριτές μονάδες λογισμικού οι οποίες παρέχουν λειτουργικές δυνατότητες στους πελάτες τους. Κάποιες υπηρεσίες ενδέχεται να βασίζονται αποκλει-

στικά στην ΠΣΔ προσέγγιση σε ό,τι αφορά στην σχεδίαση των διεπαφών τους, ενώ άλλες να βασίζονται στην ΠΣΠ προσέγγιση ή και στις δύο. Κάθε προσέγγιση έχει τα δικά της πλεονεκτήματα. Αν και η ΠΣΔ προσέγγιση παραμένει η καθιερωμένη προσέγγιση για τα επιχειρηματικά συστήματα, το REST και η ΠΣΠ προσέγγιση συγκεντρώνει σημαντική προσοχή καθώς επιτρέπει την κατασκευή συστημάτων τα οποία είναι αρχιτεκτονικά ευθυγραμμισμένα με την Αρχιτεκτονική του Παγκοσμίου Ιστού. Μια βασική πρόκληση είναι ο τρόπος με τον οποίο μπορεί να γεφυρωθεί το χάσμα μεταξύ των υπηρεσιών που προσφέρονται ακολουθώντας τα δύο αυτά υποδείγματα. Πιο συγκεκριμένα, καθώς οι επιχειρηματικές εφαρμογές βασίζονται σε συνδυασμούς και συνθέσεις υπηρεσιών, οι οποίες είναι πιθανόν να έχουν αναπτυχθεί ακολουθώντας τα δύο αυτά διαφορετικά υποδείγματα σχεδίασης διεπαφών, ένα κύριο πρόβλημα είναι η επινόηση τεχνικών και περιβαλλόντων-πλαισίων που στοχεύουν στην προσαρμογή και τη διερεύνηση της ευθυγραμμίας μεταξύ τέτοιων, διιστάμενων ως προς το ακολουθούμενο σχεδιαστικό υπόδειγμα, υπηρεσιών.

1.1.1 Προσαρμογή διεπαφών υπηρεσιών

Εξαιτίας της αυξημένης υιοθέτησης του REST, οι πάροχοι υπηρεσιών καλούνται συχνά να προσφέρουν REST API ως εναλλακτικές επιλογές στις υπάρχουσες ΠΣΔ διεπαφές υπηρεσιών, ή να εκθέσουν τα διαδικαστικά back-end δομοστοιχεία μέσω ΠΣΠ διεπαφών. Παρ' όλα αυτά, η εκ νέου υλοποίηση των δυνατοτήτων των υπηρεσιών και η συντήρηση «διπλότυπης» λειτουργικότητας αποτελούν μια σημαντική πρόκληση καθώς ενέχουν αυξημένα ρίσκα και κόστη, ενώ ταυτόχρονα μειώνουν την απόδοση των επενδύσεων σε υπάρχοντα, σταθερά και καλά δοκιμασμένα δομοστοιχεία λογισμικού. Στις περιπτώσεις αυτές, η προσαρμογή των διεπαφών των υπηρεσιών, η οποία επιτρέπει στις υπηρεσίες να εκτίθενται μέσω διαφορετικών υποδειγμάτων σχεδίασης διεπαφών, είναι συχνά η λύση που προκρίνεται.

Στην παρούσα διατριβή, αναφερόμαστε στο πρόβλημα της προσαρμογής ΠΣΔ διεπαφών σε ΠΣΠ διεπαφές ως η πρόκληση της *P2R (procedure-to-resource-orientation) προσαρμογής*. Στο πλαίσιο αυτό, διερευνούμε πρώτον, τις αρχές του ΠΣΠ υποδείγματος και παρέχουμε ένα εννοιολογικό μοντέλο για την αποτίμηση της συμμόρφωσης στον περιορισμό *Ομοιόμορφης διεπαγής* (*Uniform Interface*) του REST. Δεύτερον, προτείνουμε ένα μοντέλο διαδικασίας το οποίο αναλύει και αποσυνθέτει τη διαδικασίας P2R προσαρμογής σε διαφορετικά βήματα τα οποία

αντιμετωπίζουν με ατομικό τρόπο διαφορετικά θέματα και τρίτον, αναπτύσσουμε και παρουσιάζουμε ένα συμπαγές P2R περιβάλλον-πλαίσιο και ένα πρωτότυπο το οποίο αναλύει πληροφορίες στο επίπεδο της διεπαφής, προσανατολισμένων σε διαδικασίες υπηρεσιών, έτσι ώστε να αναγνωριστούν αντίστοιχες, προσανατολισμένες σε πόρους διεπαφές. Στη συνέχεια, η προσέγγιση που προτείνεται, αντιστοιχίζει και συσχετίζει στοιχεία της ΠΣΠ διεπαφής με στοιχεία της ΠΣΠ διεπαφής. Τελικά, παρουσιάζεται το πως τα μοντέλα που δημιουργούνται κατά τη φάση σχεδίασης της προσαρμογής, χρησιμοποιούνται από ένα δομοστοιχείο P2R προσαρμογής κατά τον χρόνο εκτέλεσης.

Κατά τη διάρκεια των τελευταίων χρόνων, η προσαρμογή διεπαφών υπηρεσιών και συγκεκριμένα η P2R προσαρμογή, έχει αναγνωριστεί στη βιβλιογραφία ως μια σημαντική πρόκληση με αποτέλεσμα να έχει προταθεί ένα σύνολο τέτοιων προσεγγίσεων. Οι περισσότερες από τις προσεγγίσεις αυτές απαιτούν σημαντική ανθρώπινη προσπάθεια και χρόνο για την αντιμετώπιση των ποικίλων μελημάτων της προσαρμογής και εν τέλει για την αναγνώριση της λογικής της προσαρμογής. Επίσης, αν και ορισμένες προσεγγίσεις επιχειρούν να αυτοματοποιήσουν τμήματα της διαδικασίας προσαρμογής, συνήθως απαιτούν εκτεταμένες πληροφορίες και στοιχεία, τα οποία ενδέχεται να είναι δύσκολο να αποκτηθούν, ή περιλαμβάνουν σημαντική προσπάθεια για να συντεθούν. Αντιθέτως, η μεθοδολογία και το περιβάλλον-πλαίσιο που προτείνεται στην παρούσα διατριβή προάγει *α*΄) την αυτοματοποίηση σε ό,τι αφορά στην αναγνώριση της P2R λογικής της προσαρμογής, που σημαίνει τον περιορισμό της ανθρώπινης συμμετοχής στην επισκόπηση και ρύθμιση της έκβασης της προσαρμογής αν αυτό απαιτείται, *β*΄) την ανεξαρτησία σχετικά με την υλοποίηση, που σημαίνει ότι απαιτούνται μόνο πληροφορίες επιπέδου διεπαφής (π.χ. περιγραφή διεπαφής) οι οποίες είναι τυπικά διαθέσιμες και *γ*΄) την αποδοτικότητα, που σημαίνει ότι η προσέγγιση δεν απαιτεί εκτεταμένους πόρους και χρόνο ώστε να ολοκληρωθεί.

1.1.2 Δια-υποδειγματική Αποτίμηση Ευθυγραμμίας

Η υποκαταστασιμότητα μεταξύ υπηρεσιών αναφέρεται στη δυνατότητα μιας υπηρεσίας να εξυπηρετεί τουλάχιστον όλους τους πιθανούς καταναλωτές μιας άλλης υπηρεσίας [150]. Η εξέταση της υποκαταστασιμότητας υπηρεσιών είναι ένα πρόβλημα που εξετάζεται σε βάθος στη βιβλιογραφία της μηχανικής υπηρεσιών και

έχει προταθεί ένα πλήθος προσεγγίσεων. Παρ' όλα αυτά, η συνύπαρξη ΠΣΔ υπηρεσιών και ΠΣΠ υπηρεσιών οι οποίες εκθέτουν την ίδια λειτουργικότητα, είτε μέσω προσαρμογής, είτε μέσω ανεξάρτητων υλοποιήσεων, εισάγει μια νέα πρόκληση σχετική με τη σύγκριση των δυνατοτήτων που παρέχονται από υπηρεσίες που ακολουθούν τα δύο διαφορετικά υποδείγματα διεπαφών. Υπό αυτή την άποψη, ορίζουμε την ευθυγραμμία (δηλαδή την ύπαρξη ευθυγράμμισης) ως μια σχέση ισοδυναμίας μεταξύ υπηρεσιών διαφορετικών σχεδιαστικών υποδειγμάτων διεπαφών, η οποία επιτρέπει την αμφίδρομη υποκαταστασιμότητα, δεδομένου ενός συνόλου δια-υποδειγματικών συσχετίσεων μεταξύ των διεπαφών των υπηρεσιών.

Η δυνατότητα αποτίμησης της υποκαταστασιμότητα, και κατ' επέκταση της ευθυγραμμίας, μεταξύ διιστάμενων ως προς το υπόδειγμα διεπαφής υπηρεσιών είναι σημαντική για αρκετούς λόγους. Για παράδειγμα, ένας πάροχος υπηρεσιών ο οποίος προσφέρει τις ίδιες δυνατότητες τόσο μέσω ΠΣΔ διεπαφών όσο και μέσω ΠΣΠ διεπαφών, θα πρέπει συνεχώς να αποτιμά κατά πόσο οι διεπαφές αυτές προσφέρουν όντως την ίδια λειτουργικότητα και πληροφορίες, έτσι ώστε όλες οι εκδόσεις των εκτιθέμενων διεπαφών να βρίσκονται σε ευθυγραμμία. Η ευθυγραμμία είναι σημαντική τόσο για λόγους πολιτικής όσο και για λόγους συνοχής, αλλά και επειδή επιτρέπει στον πάροχο υπηρεσιών να εγγυηθεί ανά πάσα στιγμή στους καταναλωτές των υπηρεσιών του ότι μπορούν με ασφάλεια να χρησιμοποιήσουν τις εναλλακτικές εκδόσεις των υπηρεσιών του αντί εκείνων που ήδη χρησιμοποιούν. Επιπλέον, η ανάγκη αποτίμησης δια-υποδειγματικής υποκαταστασιμότητας και ευθυγραμμίας υπηρεσιών είναι επίσης σημαντική υπό την οπτική του καταναλωτή υπηρεσιών, ο οποίος ενδέχεται να χρειαστεί να χρησιμοποιήσει εναλλακτικές εκθέσεις της ίδιας λειτουργικότητας (π.χ. διεκπεραίωση παραγγελίας) οι οποίες παρέχονται μέσω ΠΣΠ (ΠΣΔ) διεπαφών, εξαιτίας τεχνολογικών, οικονομικών ή άλλων λόγων. Το να είναι σε θέση να εξασφαλίσουν ότι μια μετάβαση-μετάπτωση σε εναλλακτική υπηρεσία διαφορετικού υποδείγματος δεν πρόκειται να οδηγήσει σε ασυμβατότητες δεδομένων, πληροφοριακές αναντιστοιχίες, ή διαδικαστικά αδιέξοδα είναι κρίσιμο κατά τη λήψη της απόφασης και του σχεδιασμού της μετάπτωσης. Τέλος, μια τέτοια δυνατότητα αποτίμησης της ευθυγραμμίας είναι ιδιαίτερα χρήσιμη σε σχέση με την προσέγγιση της προσαρμογής όπως παρουσιάστηκε παραπάνω, καθώς μπορεί να χρησιμοποιηθεί για να επικυρώσει και να αξιολογήσει κατά πόσο το παραχθέν REST API είναι σε θέση να εκθέσει τη λειτουργικότητα και τις πληροφορίες που παρέχονται από την αντίστοιχη ΠΣΔ διεπαφή.

Σε αυτό το πλαίσιο, μελετάμε την πρόκληση που σχετίζεται με την απόφαση του αν μια ΠΣΠ υπηρεσία μπορεί να υποκαταστήσει μια ΠΣΔ υπηρεσία (και αντίστροφα) και προτείνουμε μια μέθοδο και ένα περιβάλλον-πλαίσιο για την αντιμετώπισή της. Το προτεινόμενο περιβάλλον-πλαίσιο παρέχει τις απαιτούμενες δυνατότητες εννοιολογικής μοντελοποίησης έτσι ώστε να αναπαριστώνται εκθέσεις υπηρεσιών ορισμένες σε διαφορετικά υποδείγματα διεπαφών, όπως και να επιλύονται αντιστοιχίες των ΠΣΔ και ΠΣΠ διεπαφών υπηρεσιών μέσω ορισμού κατάλληλων ανταποκρίσεων και αντιστοιχιών. Πιο συγκεκριμένα, η προτεινόμενη προσέγγιση εισάγει εννοιολογικά μεταμοντέλα για τον προσδιορισμό ειδικών ως προς το υπόδειγμα μοντέλων υπηρεσιών και δια-υποδειγματικών συσχετίσεων των διεπαφών, με συνεπή τρόπο. Στη συνέχεια, τα μοντέλα που έχουν προσδιοριστεί αναλύονται και μετασχηματίζονται μέσω μιας διαδικασίας μετασχηματισμού μοντέλων πολλαπλών σταδίων, η οποία τελικά παρέχει δύο συγκρίσιμα τυπικά μοντέλα (open net, μια εξειδίκευση των δικτύων Petri), ένα για την ΠΣΔ υπηρεσία και ένα για την ΠΣΠ υπηρεσία. Τελικά, χρησιμοποιώντας τα μοντέλα που έχουν παραχθεί, χρησιμοποιούμε τεχνικές εξέτασης υποκαταστασιμότητας και συγκεκριμένα, χρησιμοποιείται η αποτίμηση της *συμφωνίας* ή *accordance*, μεταξύ *open nets* [148], [149], ούτως ώστε να αποφασισθεί η υποκαταστασιμότητα και ευθυγραμμία των υπηρεσιών. Αν και η υποκαταστασιμότητα υπηρεσιών έχει εξεταστεί σε βάθος στη βιβλιογραφία, οι προσεγγίσεις αυτές εξετάζουν την υποκαταστασιμότητα αποκλειστικά μεταξύ ΠΣΔ υπηρεσιών και δεν καλύπτουν υπηρεσίες που διίστανται ως προς τα ακολουθούμενα υποδείγματα διεπαφών. Αναφερόμαστε στην τελευταία περίπτωση εξέτασης υποκαταστασιμότητας ως την πρόκληση αποτίμησης *υποκαταστασιμότητα υπηρεσιών διαφορετικών υποδειγμάτων*, ή για συντομία *δια-υποδειγματική υποκαταστασιμότητα (ΔΥΥ)* (*inter-paradigm service substitutability (IPSS)*). Επίσης, αναφερόμαστε στην αμφίδρομη ΔΥΥ ως *dia-upodeigmatik'h eujugramm'ia (DUE)* υπηρεσιών (*inter-paradigm service alignment (IPSA)*). Με βάση τα στοιχεία που έχουμε υπόψη μας, το περιβάλλον-πλαίσιο για την απόφαση της ΔΥΥ και κατ' επέκταση της ΔΥΕ, το οποίο προτείνεται στην παρούσα διατριβή είναι πρωτότυπο και στοχεύει να ρίξει φως στην εξέταση ύπαρξης ευθυγράμμισης μεταξύ διιστάμενων ως προς το ακολουθούμενο υπόδειγμα διεπαφών και συγκεκριμένα στην ευθυγραμμία ΠΣΔ και ΠΣΠ εκθέσεων υπηρεσιών τόσο σε επίπεδο εννοιλογικής μοντελοποίηση όσο και μεθοδολογίας αλγοριθμικής και αυτοματοποιημένης αποτίμησης.

1.2 Συνεισφορές διατριβής

Συνεισφορές σχετικές με την προσαρμογή υποδείγματος διεπαφής

Εδώ στόχος μας είναι να παρέχουμε ένα περιβάλλον-πλαίσιο προσαρμογής το οποίο βασίζεται αποκλειστικά σε διαθέσιμες ή σε εύκολα αποκτώμενες προδιαγραφές διεπαφών και μειώνει την ανθρώπινη προσπάθεια που απαιτείται για τη διαδικασία προσαρμογής. Παράλληλα, στοχεύουμε σε μια προσέγγιση που δεν υποθέτει τη διαθεσιμότητα στοιχείων υλοποίησης (π.χ. πηγαίο κώδικα, σχήματα βάσεων δεδομένων, κλπ). Στο πλαίσιο της έρευνάς μας, προσπαθούμε να απαντήσουμε σε μια σειρά ερευνητικών ερωτημάτων τα οποία περιλαμβάνουν τα εξής:

- Ε1: Πώς μπορεί να υλοποιηθεί ένα περιβάλλον-πλαίσιο προσαρμογής, το οποίο θα είναι σε θέση να παράγει διεπαφές που συμμορφώνονται στους περιορισμούς του REST; Ποιες είναι οι προκλήσεις και τα θέματα που πρέπει να αντιμετωπιστούν;
- Ε2: Πώς μπορεί να αποτιμηθεί η συμμόρφωση στο REST;
- Ε3: Πώς μπορεί να παραχθεί μια ΠΣΠ διεπαφή υπηρεσίας με βάση την προδιαγραφή μιας αντίστοιχης ΠΣΔ διεπαφής, με αυτοματοποιημένο τρόπο· Τι τεχνικές μπορούν να χρησιμοποιηθούν για να *α*) δημιουργηθούν μοντέλα των τύπων των πόρων, *β*) εντοπισθούν πιθανές πράξεις που σχετίζονται με αυτούς τους τύπους πόρων, και να *γ*) προσδιορισθούν μοντέλα αναπαραστάσεων των πόρων;
- Ε4: Πώς μπορούν τα στοιχεία μιας ΠΣΠ διεπαφής να συσχετιστούν με τα στοιχεία μιας της αντίστοιχης ΠΣΔ διεπαφής έτσι ώστε να μπορούν να προσαρμοστούν RESTful αιτήματα και αποκρίσεις σε κλήσεις και αποτελέσματα λειτουργιών υπηρεσιών;

Συνεισφορές σχετικές με την Υποκαταστασιμότητα και την Ευθυγραμμία

Εδώ στόχος μας είναι να παρέχουμε μια μέθοδο και ένα αντίστοιχο περιβάλλον-πλαίσιο το οποίο θα επιτρέψει τη σύγκριση μεταξύ ΠΣΔ υπηρεσιών και ΠΣΠ υπηρεσιών και την αποτίμηση του αν μπορούν δύο υπηρεσίες που ακολουθούν τα δύο αυτά διαφορετικά υποδείγματα να υποκαταστήσουν η μία την άλλη. Σε αυτό το

πλαίσιο, πρέπει να προσδιοριστεί το απαιτούμενο εννοιολογικό μοντέλο για τα δύο υποδείγματα, έτσι ώστε να αναγνωριστούν τύποι συσχετίσεων μεταξύ των υποδειγμάτων. Έπειτα, οι υπηρεσίες και οι συσχετίσεις που έχουν μοντελοποιηθεί πρέπει να μετασχηματιστούν σε κατάλληλα τυπικά μοντέλα έτσι ώστε το πρόβλημα να αναχθεί σε υπάρχουσες τεχνικές αποτίμησης υποκαταστασιμότητας οι οποίες είχαν αρχικά προταθεί στα πλαίσια ελέγχου υποκαταστασιμότητας μεταξύ ΠΣΔ υπηρεσιών. Τα κύρια ερευνητικά ερωτήματα που ερευνώνται στο πλαίσιο της πρόκλησης ΔΥΥ είναι τα ακόλουθα.

- Ε5: Ποιες είναι οι κοινές αφαιρέσεις μεταξύ των ΠΣΠ και των ΠΣΔ αρχιτεκτονικών που μπορούν να χρησιμοποιηθούν για να αναγνωριστούν εννοιολογικές αντιστοιχίες ή ανταποκρίσεις μεταξύ στοιχείων των ΠΣΔ διεπαφών υπηρεσιών και ΠΣΔ διεπαφών υπηρεσιών;
- Ε6: Πως μπορούν δύο υπηρεσίες που ακολουθούν διαφορετικά υποδείγματα να εξεταστούν ως προς το αν μπορούν να υποκαταστήσουν η μία την άλλη, δεδομένου ενός συνόλου συσχετίσεων μεταξύ των στοιχείων των διεπαφών τους;

Βήματα για την επίτευξη των στόχων της διατριβής

Για την επίτευξη των παραπάνω στόχων και για την παροχή απαντήσεων στα σχετικά ερευνητικά ερωτήματα ακολουθούμε ένα σύνολο βημάτων.

1. Εξετάζουμε στυλ σχεδίασης διεπαφών υπηρεσιών έτσι ώστε να αποκτήσουμε καλύτερη κατανόηση του νεότερου ΠΣΠ υποδείγματος και μελετάμε το REST τόσο ως προς τις θεωρητικές του διαστάσεις, όσο και ως προς πρακτικές διαστάσεις, έτσι ώστε να αναγνωρίσουμε πως τα στοιχεία του και οι περιορισμοί που θέτει χρησιμοποιούνται στις RESTful υπηρεσίες Παγκοσμίου Ιστού.
2. Αναγνωρίζουμε βασικά κριτήρια για την αποτίμηση της συμμόρφωσης στους περιορισμούς του REST, εστιάζοντας ιδιαιτέρως στον περιορισμό Ομοιόμορφης διεπαφής και προτείνουμε ένα αντίστοιχο εννοιολογικό μοντέλο.
3. Μελετάμε ορισμένες απαιτήσεις σχετικές με την προσαρμογή ΠΣΔ διεπαφών σε ΠΣΠ διεπαφές, και ορίζουμε ένα μοντέλο διαδικασίας το οποίο αντιμετωπίζει ποικίλα ζητήματα που σχετίζονται με μια τέτοια διαδικασία προσαρμογής διεπαφών, με αναλυτικό και συστηματικό τρόπο.

4. Ορίζουμε μια συμπαγή διαδικασία και ένα περιβάλλον-πλαίσιο για την προσαρμογή ΠΣΔ υπηρεσιών σε υπηρεσίες ΠΣΠ, αποτελούμενα από τεχνικές που αναλύουν περιγραφές ΠΣΔ υπηρεσιών και παράγουν ΠΣΠ διεπαφές. Αξιολογούμε την ακρίβεια της προσέγγισης εξαγωγής πόρων μέσω εκτέλεσης πειραμάτων με ένα σημαντικό αριθμό υπηρεσιών.
5. Παρέχουμε μια πρωτότυπη υλοποίηση ενός προσαρμογέα χρόνου εκτέλεσης ο οποίος εφαρμόζει τη λογική προσαρμογής που έχει αναγνωριστεί στο πλαίσιο του Apache Tuscany[1] το οποίο είναι ένα Service Component Architecture[2] (SCA) περιβάλλον ανοιχτού κώδικα.
6. Ορίζουμε εννοιολογικά μεταμοντέλα τόσο ΠΣΔ όσο και ΠΣΠ υπηρεσιών και αναγνωρίζουμε ανταποκρίσεις, ή αντιστοιχίες, μεταξύ στοιχείων των δύο μοντέλων μέσω ενός τρίτου μεταμοντέλου.
7. Προτείνουμε μια διαδικασία και ένα αντίστοιχο περιβάλλον-πλαίσιο για τον μετασχηματισμό μοντέλων υπηρεσιών και συσχετίσεων μεταξύ τους σε open nets, έτσι ώστε να είναι μπορεί να αποτιμηθεί η ΔΥΥ μέσω εξέτασης της ύπαρξης accordance.

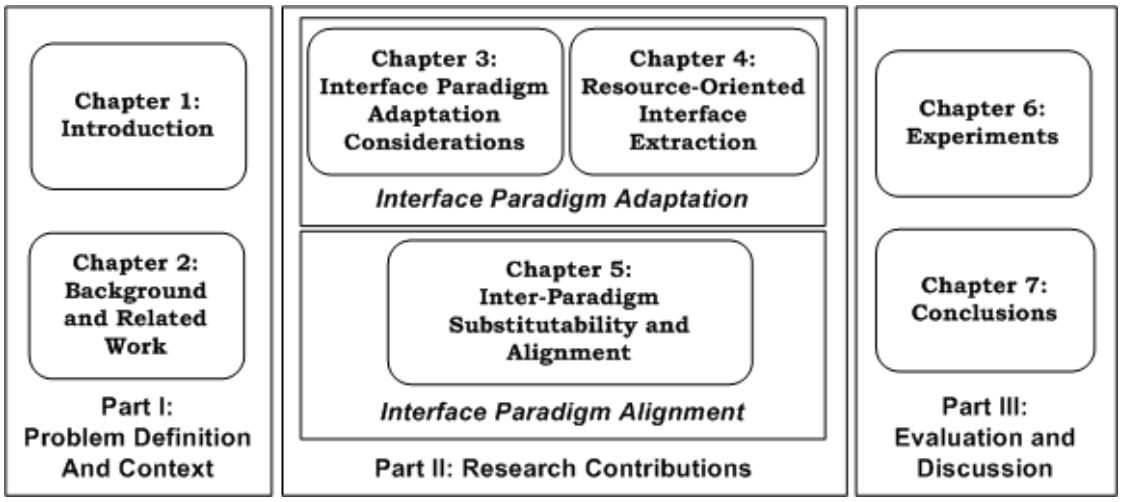
1.3 Γενική περιγραφή διατριβής

Το Σχήμα 1.1 παρέχει μια απεικόνιση της δομής της διατριβής η οποία έχει τρία κύρια μέρη, την *Περιγραφή Προβλήματος και Πλαίσιο* (*Problem Description and Context*), τις *Ερευνητικές Συνεισφορές* (*Research Contributions*) και την *Αξιολόγηση και Συζήτηση Evaluation and Discussion*. Κάθε τμήμα συντίθεται από αντίστοιχα κεφάλαια.

Τμήμα 1: Περιγραφή Προβλήματος και Πλαίσιο

Πέρα από την εισαγωγή του προβλήματος που γίνεται στο τρέχον κεφάλαιο, στο Κεφάλαιο 2 παρέχεται μια παρουσίαση και συζήτηση προηγούμενων εργασιών στην περιοχή. Συγκεκριμένα, συζητούνται πληροφορίες υποβάθρου-πλαισίου και

[1] Apache Tuscany, http://tuscany.apache.org/
[2] SCA, http://www.oasis-opencsa.org/sca



Σχήμα 1.1: Γενική περιγραφή διατριβής

προηγούμενες συνεισφορές σχετικά με την αρχιτεκτονική λογισμικού, την υπηρεσιοστρεφή υπολογιστική, το REST και την υποκαταστασιμότητα υπηρεσιών.

Τμήμα 2: Ερευνητικές Συνεισφορές

Στο Κεφάλαιο 3, εξετάζονται θέματα σχετικά με την πρόκληση P2R προσαρμογής τα οποία είναι οργανωμένα σε τρεις κατευθύνσεις: *α*) *απαιτήσεις*, όπου συζητούνται οι λειτουργικές και μη λειτουργικές απαιτήσεις για μια προσέγγιση προσαρμογής ΠΣΔ σε ΠΣΠ υπηρεσιών, *β*) *σχεδίαση*, όπου εξετάζεται ένα μοντέλο διαδικασίας υψηλού αφαιρετικού επιπέδου για μια προσέγγιση P2R προσαρμογής έτσι ώστε να αναλυθεί και να επιμεριστεί η διαδικασία προσαρμογής σε βήματα τα οποία αντιμετωπίζουν διαφορετικά μελήματα, *γ*) *αξιολόγηση και αποτίμηση*, όπου συζητούνται μέθοδοι για την αποτίμηση της ποιότητας του αποτελέσματος μιας προσέγγισης P2R προσαρμογής, δηλαδή ενός REST API.

Στη συνέχεια, στο Κεφάλαιο 4 παρουσιάζεται ένα περιβάλλον-πλαίσιο P2R προσαρμογής, το οποίο επιτρέπει την εξαγωγή μιας ΠΣΠ διεπαφής χρησιμοποιώντας ως δεδομένα εισόδου τον ορισμό μιας αντίστοιχης ΠΣΔ διεπαφής. Το περιβάλλον-πλαίσιο που παρουσιάζεται παρέχει τεχνικές για την αναγνώριση στοιχείων μιας ΠΣΠ διεπαφής, όπως ιεραρχίες πόρων και αναπαραστάσεις, μαζί με αντιστοιχίες μεταξύ των στοιχείων που έχουν αναγνωριστεί και στοιχείων της ΠΣΔ διεπαφής.

Το Κεφάλαιο 5 αναλύει την πρόκληση ΔΥΥ και παρουσιάζει ένα περιβάλλον-πλαίσιο το οποίο επιτρέπει την αποτίμηση της υποκαταστασιμότητας και της ευθυγραμμίας. Το περιβάλλον-πλαίσιο που προτείνεται παρέχει εννοιολογικά μοντέλα τα οποία προσδιορίζουν μοντέλα υπηρεσιών που φέρουν διεπαφές των δύο υποδειγμάτων, μαζί με ένα μεταμοντέλο κανόνων αντιστοίχισης ή ευθυγραμμίας, το οποίο αναγνωρίζει σημασιολογικές ανταποκρίσεις μεταξύ στοιχείων των δύο υποδειγμάτων. Τα μοντέλα αυτά χρησιμοποιούνται εν συνεχεία για να παραχθούν συγκρίσιμα τυπικά μοντέλα μέσω μιας διαδικασίας μετασχηματισμού. Τελικά, η απόφαση περί υποκαταστασιμότητας και ευθυγραμμίας μεταξύ υπηρεσιών λαμβάνεται χρησιμοποιώντας το θεωρητικό πλαίσιο ελέγχου ύπαρξης accordance μεταξύ open net.

Τμήμα 3: Αξιολόγηση και Συζήτηση

Το Κεφάλαιο 6 παρουσιάζει πειράματα και περιπτωσιολογικές μελέτες που εκτελέστηκαν για την αξιολόγηση των περιβαλλόντων-πλαισίων που προτείνονται στα Κεφάλαια 4 και 5. Η πειραματική αξιολόγηση του περιβάλλοντος-πλαισίου P2R προσαρμογής αποτιμά την ακρίβεια των τεχνικών εξαγωγής, την επίδοση της διαδικασίας εξαγωγής και τον αντίκτυπο της διαδικασίας αυτής στο πλαίσιο ενός ρεαλιστικού περιβάλλοντος ανάπτυξης. Η αποτελεσματικότητα του περιβάλλοντος-πλαισίου ΔΥΥ εξετάζεται μέσω μιας ρεαλιστικής περιπτωσιολογικής μελέτης στην οποία δύο διεπαφές υπηρεσιών, μια ΠΣΔ και μια ΠΣΠ, αναλύονται και αποτιμώνται ως προς τη μεταξύ τους υποκαταστασιμότητα.

ΚΕΦΑΛΑΙΟ 2

ΣΧΕΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

Στο παρόν κεφάλαιο παραθέτουμε το γνωσιακό υπόβαθρο και τις εργασίες σε επιστημονικές περιοχές που σχετίζονται με τις συνεισφορές της παρούσας διατριβής. Πρώτον, συζητούνται ορισμένες έννοιες από την περιοχή της αρχιτεκτονικής λογισμικού οι οποίες παρέχουν ένα πλαίσιο για την εισαγωγή του μοντέλου υπηρεσιοστρεφούς υπολογιστικής και των κύριων αρχιτεκτονικών προσεγγίσεων του μοντέλου αυτού. Πιο συγκεκριμένα, παρέχουμε μια σύντομη συζήτηση και σύγκριση μεταξύ της προσέγγισης ΠΣΔ και της προσέγγισης ΠΣΠ, βασισμένη σε μελέτες και παρατηρήσεις που εντοπίστηκαν στη σχετική βιβλιογραφία. Δεύτερον, παρουσιάζουμε το αρχιτεκτονικό στυλ REST ως προς τις αρχές και τους περιορισμούς του και συζητάμε το πως το αρχιτεκτονικό αυτό στυλ επηρεάζει τη σχεδίαση υπηρεσιών. Τρίτον, παρουσιάζονται εργασίες στην περιοχή της προσαρμογής διεπαφών υπηρεσιών, εντοπίζοντας την προσοχή μας στις προσεγγίσεις που αφορούν στην προσαρμογή ΠΣΔ διεπαφών σε ΠΣΠ διεπαφές, καθώς στην παρούσα διατριβή παρουσιάζεται ένα ανάλογο περιβάλλον-πλαίσιο προσαρμογής. Τέταρτον, συζητούνται εργασίες στην περιοχή ελέγχου υποκαταστασιμότητας μεταξύ υπηρεσιών, με στόχο την παρουσίαση του πλαίσιο στο οποίο η παρούσα διατριβή εξετάζει την πρόκληση αποτίμησης υποκαταστασιμότητας και ευθυγραμμίας υπηρεσιών που ακολουθούν διαφορετικά σχεδιαστικά υποδείγματα διεπαφών.

2.1 Αρχιτεκτονική προσέγγιση υπηρεσιοστρεφών συστημάτων

Καθώς τα συστήματα λογισμικού καλούνταν να αντιμετωπίσουν ολοένα και πιο σύνθετες απαιτήσεις, γεννήθηκε η ανάγκη καταγραφής των στοιχείων που συνθέτουν τα συστήματα αυτά όπως και των χαρακτηριστικών των στοιχείων αυτών και των μεταξύ τους σχέσεων, σε περιγραφές υψηλού αφαιρετικού επιπέδου, έτσι ώστε να είναι εφικτή η παροχή ολιστικών απεικονίσεων των συστημάτων αυτών. Η ανάγκη αυτή οδήγησε στην εισαγωγή της *αρχιτεκτονικής* προσέγγισης στη σχεδίαση, στην υλοποίηση και στην εξέλιξη συστημάτων λογισμικού. Οι αρχιτεκτονικές περιγραφές ενός συστήματος λογισμικού δύνανται να περιλαμβάνουν δομές, ιδιότητες, ρόλους και σχέσεις των δομοστοιχείων και των μονάδων λογισμικού που απαρτίζουν το σύστημα. Τα στοιχεία αυτά παρέχουν *αρχιτεκτονικές όψεις* (*architectural views*)

ενός συστήματος οι οποίες κατά κανόνα δεν είναι άμεσα διαθέσιμες στο επίπεδο υλοποίησης.

2.1.1 Αρχιτεκτονική λογισμικού και αρχιτεκτονικά στυλ

Η *αρχιτεκτονική λογισμικού* (*software architecture*) αναπτύχθηκε ως διακριτή επιστημονική περιοχή στη δεκαετία του 1990, αν και ο όρος εισήχθη στα τέλη της δεκαετίας του 1960 [73]. Τόσο η ακαδημαϊκή κοινότητα όσο και η βιομηχανία συνεισφέρουν στην περιοχή αυτή προτείνοντας μεθόδους και γλώσσες για τον ορισμό, την περιγραφή και την ανάλυση αρχιτεκτονικών προδιαγραφών συστημάτων λογισμικού. Σταδιακά, τόσο οι θεωρητικές όσο και οι πρακτικές διαστάσεις της αρχιτεκτονικής συστημάτων που στηρίζονται στο λογισμικό αποτέλεσαν αντικείμενα επιστημονικής διερεύνησης, γεγονός που οδήγησε σε ένα πλήθος μεθοδολογιών μοντελοποίησης, προτύπων και πλαισίων τυπικής ανάλυσης. Μια εκτενής εξέταση της ιστορίας και των σημαντικότερων συνεισφορών στην περιοχή της αρχιτεκτονικής λογισμικού παρέχεται στην εργασία [73].

Στη βιβλιογραφία έχει προταθεί ένα πλήθος ορισμών για την έννοια της αρχιτεκτονικής λογισμικού. Μια συλλογή τέτοιων ορισμών έχει συντεθεί από το Ινστιτούτο Τεχνολογίας Λογισμικού (Software Engineering Institute (SEI)) του πανεπιστημίου Carnegie Mellon [1]. Παραθέτουμε παρακάτω μια λίστα επιλεγμένων ορισμών της έννοιας, που αντλήθηκαν από τη λίστα του SEI και τη βιβλιογραφία και η οποία επιχειρεί να συνοψίσει τις διαφορετικές οπτικές για την έννοια όπως παρατηρούνται στην κοινότητα της τεχνολογίας λογισμικού.

- *Ένα σύνολο αρχιτεκτονικών (ή, αν θέλετε, σχεδιαστικών) στοιχείων τα οποία έχουν μια συγκεκριμένη δομή. Οι Perry και Wolf διαχωρίζουν τα στοιχεία επεξεργασίας, τα στοιχεία δεδομένων και τα στοιχεία σύνδεσης και αυτή η ταξινόμηση επικρατεί μεταξύ των περισσοτέρων από τους ορισμούς και τις προσεγγίσεις.* [120]
- *Μια αρχιτεκτονική λογισμικού είναι μια αφαίρεση των στοιχείων χρόνου εκτέλεσης ενός συστήματος λογισμικού κατά τη διάρκεια μιας φάσης λειτουργίας του. Ένα σύστημα μπορεί να έχει συντεθεί από πολλά επίπεδα αφαίρεσης και πολλές φάσεις λειτουργίας, κάθε μια εκ των οποίων με τη δική της αρχιτεκτονική λογισμικού.* [53]

- *Η αρχιτεκτονική λογισμικού ενός προγράμματος ή ενός υπολογιστικού συστήματος είναι η δομή ή οι δομές του συστήματος, οι οποίες αποτελούνται από στοιχεία λογισμικού, από τις εξωτερικά ορατές ιδιότητες των στοιχείων αυτών και τις μεταξύ τους σχέσεις.* [12]
- *Η αρχιτεκτονική λογισμικού περιλαμβάνει: (α) τη δομή και οργάνωση με βάση τις οποίες τα σύγχρονα δομοστοιχεία συστημάτων και υποσυστήματα αλληλεπιδρούν ώστε να σχηματίζουν συστήματα και (β) τις ιδιότητες συστημάτων οι οποίες μπορούν να σχεδιαστούν και να αναλυθούν με βέλτιστο τρόπο σε επίπεδο συστήματος.* [73]
- *Αρχιτεκτονική: θεμελιώδεις έννοιες ή ιδιότητες ενός συστήματος στο περιβάλλον του, όπως ενσωματώνονται στα στοιχεία του, στις σχέσεις του, και στις αρχές σχεδίαση και εξέλιξής του.* [65]

Με βάση τα παραπάνω, αν και τα ακριβή σημεία στα οποία εστιάζουν οι διάφοροι ορισμοί της έννοιας διαφέρουν, φαίνεται να υπάρχει ένας κοινός τόπος σχετικά με το αντικείμενο της αρχιτεκτονική λογισμικού ενός συστήματος και συγκεκριμένα είναι σαφής η αναφορά στα στοιχεία ενός συστήματος, στη δομή τους, στις ιδιότητές τους και στις μεταξύ τους σχέσεις και αλληλεπιδράσεις. Κατά συνέπεια, οι αρχιτεκτονικές όψεις νούνται ως προβολές ή περιγραφές των παραπάνω διαστάσεων ενός συστήματος.

Μια ακόμη βασική έννοια, η οποία εισήχθη νωρίς στην ερευνητική περιοχή της αρχιτεκτονικής λογισμικού και η οποία παίζει κεντρικό ρόλο ως προς το περιεχόμενο της παρούσας διατριβής, είναι η έννοια του *αρχιτεκτονικού στυλ* (*architectural style*). Τόσο η υπηρεσιοστρεφής αρχιτεκτονική, όσο και το REST έχουν οριστεί ως αρχιτεκτονικά στυλ λογισμικού, τα οποία βρίσκουν εφαρμογή κατά κανόνα σε κατανεμημένες, δικτυακές εφαρμογές. Στη βιβλιογραφία έχει προταθεί ένα πλήθος ορισμών για την έννοια του αρχιτεκτονικού στυλ. Ομοίως με την έννοια της αρχιτεκτονικής λογισμικού, παραθέτουμε μια λίστα με επιλεγμένους τέτοιους ορισμούς.

- *Ένα αρχιτεκτονικό στυλ ορίζει: μια οικογένεια συστημάτων σε όρους ενός προτύπου δομικής οργάνωσης, ενός λεξιλογίου δομοστοιχείων και συνδέσμων, παράλληλα με περιορισμούς ως προς το πως αυτά μπορούν να συνδυαστούν.* [136]
- *Ένα αρχιτεκτονικό στυλ είναι ένα συγκροτημένο σύνολο αρχιτεκτονικών περιορισμών οι οποίοι περιορίζουν τους ρόλους/ιδιότητες των αρχιτεκτονικών*

στοιχείων και των επιτρεπτών σχέσεων μεταξύ αυτών των στοιχείων στα πλαίσια μιας αρχιτεκτονικής που συμμορφώνεται στο στυλ αυτό. [53]

- *Ένα αρχιτεκτονικό στυλ είναι μια αναγνωρισμένη συλλογή αρχιτεκτονικών αποφάσεων σχεδίασης οι οποίες (1) είναι εφαρμόσιμες σε ένα δεδομένο πλαίσιο ανάπτυξης, (2) περιορίζουν τις αρχιτεκτονικές αποφάσεις που είναι ειδικές σε κάποιο συγκεκριμένο σύστημα στο πλαίσιο αυτό και (3) επάγουν ωφέλιμες ιδιότητες σε κάθε προκύπτου σύστημα.* [145]
- *Μια εξειδίκευση τύπων στοιχείων και σχέσεων, μαζί με ένα σύνολο περιορισμών στο πως μπορούν αυτά να χρησιμοποιηθούν.* [2]

Λαμβάνοντας υπόψη τους παραπάνω ορισμούς, μπορεί να σημειωθεί ότι η έννοια του αρχιτεκτονικού στυλ χρησιμοποιείται για να δηλώσει διακριτές και αναγνωρίσιμες συλλογές σχεδιαστικών αποφάσεων και δομικών προτύπων στοιχείων, παράλληλα με περιορισμούς που επιβάλλονται ως προς το τρόπο με τον οποίο τα στοιχεία αυτά ορίζονται ή συσχετίζονται.

2.1.2 Προσανατολισμός σε υπηρεσίες και σχεδίαση διεπαφών

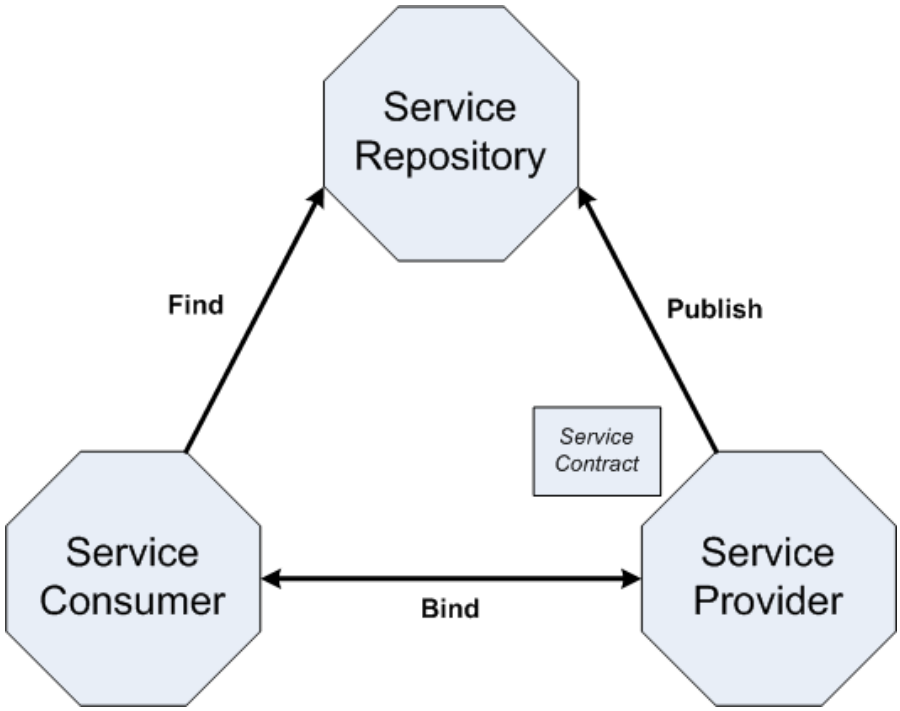
Η υπηρεσιοστρεφής αρχιτεκτονική αποτελεί ένα αρχιτεκτονικό στυλ κατανεμημένων συστημάτων λογισμικού το οποίο προωθεί την οργάνωση της λειτουργικότητας μέσω αυτοτελών, καλά καθορισμένων και επαναλήψιμων δυνατοτήτων, οι οποίες ονομάζονται *υπηρεσίες* [72]. Τα υπηρεσιοστρεφή συστήματα αποτελούνται από διακριτούς συντελεστές λογισμικού (software agents), οι οποίοι δύνανται να λειτουργούν σε ανεξάρτητα περιβάλλοντα και οι οποίοι επικοινωνούν και αλληλεπιδρούν με στόχο την εκτέλεση υπολογιστικών εργασιών και την εκπλήρωση συγκεκριμένων απαιτήσεων και στόχων. Το αρχιτεκτονικό στυλ SOA προάγει τις αρχές του προσανατολισμού σε υπηρεσίες, έτσι ώστε τα συστήματα που προκύπτουν να χαρακτηρίζονται από ένα σύνολο αντίστοιχων ιδιοτήτων όπως η λανθάνουσα/ασθενής σύζευξη (loose coupling), η δυνατότητα επαναχρησιμοποίησης (reusability), η χρήση προτυποποιημένων συμβολαίων υπηρεσιών (service contracts) [45], κ.ά.

Όπως συζητήθηκε και στο Κεφάλαιο 1, σε ό,τι αφορά στην ολοκλήρωση συστημάτων λογισμικού, η ιδέα του προσανατολισμού σε υπηρεσίες αναδύθηκε ως μια ευοίωνη εναλλακτική, η οποία αντιμετώπιζε αρκετές από τις ανεπάρκειες των

προηγούμενων προσεγγίσεων και οδήγησε στην ανάπτυξη μια ξεχωριστής περιοχής έρευνας και εφαρμογής στην τεχνολογία λογισμικού, την περιοχή της υπηρεσιοστρεφούς υπολογιστικής (ΥΥ) (service-oriented computing (SOC)) [140], [55]. Στα πλαίσια της ΥΥ, έχουν αναγνωριστεί και εξετάζονται ενεργά μια σειρά θεμάτων και προκλήσεων σχετικών με τις υπηρεσιοστρεφείς αρχιτεκτονικές. Για παράδειγμα, τα θέματα και οι προκλήσεις αυτές περιλαμβάνουν θέματα σχεδίασης διεπαφών και αρχιτεκτονικών προσεγγίσεων, ζητήματα σχετικά με τον έλεγχο και τη διασφάλιση της ποιότητας των υπηρεσιών, οργανωτικές προκλήσεις και προβλήματα διακυβέρνησης (governance) υπηρεσιοστρεφών συστημάτων, κ.ά. [70], [111], [110]. Ορισμένα από τα θέματα αυτά ξεφεύγουν από τα όρια της τεχνολογίας λογισμικού ή και των τεχνολογιών πληροφορικής γενικότερα καθώς αγγίζουν ζητήματα συνολικού μετασχηματισμού και λειτουργίας των οργανισμών στη βάση του προσανατολισμού σε υπηρεσίες. Μια εκτεταμένη παρουσίαση και συζήτηση του προσανατολισμού σε υπηρεσίες θα ξέφευγε από το εύρος και τους στόχους της παρούσας διατριβής. Ως εκ τούτου, στο κεφάλαιο αυτό, εισάγουμε με συνοπτικό τρόπο τα κύρια χαρακτηριστικά του αρχιτεκτονικού στυλ SOA και συζητάμε ένα σύνολο αρχιτεκτονικών διαστάσεων των υπηρεσιοστρεφών συστημάτων, εστιάζοντας ιδιαιτέρως στη σχεδίαση διεπαφών. Εκτιμούμε ότι μια τέτοια συζήτηση δύναται να παρέχει το απαραίτητο γνωσιακό υπόβαθρο ώστε να διευκολυνθεί η κατανόηση του περιεχομένου, του πλαισίου εφαρμογής και της χρησιμότητας των προσεγγίσεων που παρουσιάζονται στη συνέχεια της διατριβής. Ωστόσο, ενθαρρύνουμε τους αναγνώστες που ενδιαφέρονται για τη μελέτη του υπηρεσιοστρεφούς υπολογιστικού μοντέλου να εξετάσουν ορισμένες εξαιρετικές πηγές γνώσης για την ΥΥ και το αρχιτεκτονικό στυλ SOA όπως οι παρακάτω [45], [84], [140], [55], [46].

Στο αρχιτεκτονικο στυλ SOA, οι υπηρεσίες ορίζονται ως ανεξάρτητες, αυτοτελείς μονάδες λογισμικού οι οποίες παρέχουν συγκεκριμένες λειτουργικές και πληροφοριακές δυνατότητες. Σε μια υπηρεσιοστρεφή αρχιτεκτονική, υπάρχουν τρεις τύποι ενδιαφερομένων μερών, οι *πάροχοι υπηρεσιών* (*service providers*), οι *καταναλωτές υπηρεσιών* (*service consumers*) και οι *μεσολαβητές υπηρεσιών* (*service brokers*). Μια υπηρεσία παρέχεται από έναν πάροχο και μπορεί να καταναλωθεί από έναν ή περισσότερους καταναλωτές. Οι πάροχοι υπηρεσιών δημοσιεύουν τις υπηρεσίες που προσφέρουν σε *αποθετήρια υπηρεσιών* (*service repositories*) τα οποία διαχειρίζονται οι μεσολαβητές υπηρεσιών. Τα αποθετήρια υπηρεσιών είναι προσβάσιμα και μπορούν να δέχονται ερωτήματα από καταναλωτές υπηρεσιών οι οποίοι αναζητούν και ενδιαφέρονται να χρησιμοποιήσουν δυνατότητες υπηρεσιών. Οι παραπάνω αλ-

ληλεπιδράσεις μεταξύ των ενδιαφερόμενων μερών του SOA, συνθέτουν το *μοντέλο τριγωνικής αλληλεπίδρασης της υπηρεσιοστρεφούς αρχιτεκτονικής*, αναφερόμενο ως SOA interaction triangle, το οποίο απεικονίζεται στο Σχήμα 2.1.



Σχήμα 2.1: Το μοντέλο τριγωνικής αλληλεπίδρασης της υπηρεσιοστρεφούς αρχιτεκτονικής

Λόγω της αρχής χρήσης προτυποποιημένων *συμβολαίων υπηρεσιών* (*service contracts*), ένας πάροχος υπηρεσιών πρέπει να προσδιορίζει και να είναι συνεπής ως προς τον τρόπο με τον οποίο οι πιθανοί καταναλωτές των υπηρεσιών του μπορούν να προσπελάσουν την παρεχόμενη από τις υπηρεσίες λειτουργικότητα και πληροφορίες. Η σημασία των συμβολαίων υπηρεσίας στο αρχιτεκτονικό στυλ της υπηρεσιοστρεφούς αρχιτεκτονικής έγκειται στο γεγονός ότι επιτρέπουν τον διαχωρισμό και τον ανεξάρτητο προσδιορισμό *διεπαφών* και *υλοποιήσεων*. Υπό αυτή την άποψη, στα πλαίσια του αρχιτεκτονικού στυλ SOA, μια υπηρεσία εκθέτει τη λειτουργικότητά της μέσω μιας διεπαφής και η διεπαφή δύναται να προσδιοριστεί από μια προτυποποιημένη προδιαγραφή σε όρους ανεξάρτητους της υλοποίησης. Αυτός ο διαχωρισμός παρέχει ένα επίπεδο αφαίρεσης μεταξύ του προσδιορισμού διεπαφών υπηρεσιών και ζητημάτων υλοποίησης υπηρεσιών και επιτρέπει στους καταναλωτές υπηρεσιών να μην είναι στενά συζευγμένοι με τεχνολογίες που χρησιμοποιούνται από τους παρόχους των υπηρεσιών, βελτιώνοντας έτσι τις δυνατότητες

διαλειτουργικότητας. Επιπλέον, οι πάροχοι υπηρεσιών είναι ελεύθεροι να διαχειρίζονται, να εξελίσσουν ή να αντικαθιστούν τις υλοποιήσεις των υπηρεσιών τους, χωρίς αυτό να έχει αντίκτυπο στους καταναλωτές, υπό την προϋπόθεση ότι οι εξελιγμένες ή νέες υλοποιήσεις συμμορφώνονται με τις συμφωνηθείσες προδιαγραφές διεπαφών, ή με άλλα λόγια, τηρούν τα συμβόλαια των υπηρεσιών.

Αν και το αρχιτεκτονικό στυλ της υπηρεσιοστρεφούς αρχιτεκτονικής δεν καθορίζει κάποιο συγκεκριμένο μοτίβο για τη σχεδίαση διεπαφών και για την έκθεση λειτουργικότητας υπηρεσιών, το υπόδειγμα ΠΣΔ έχει χρησιμοποιηθεί κατά κόρον για τον σκοπό αυτό, τυπικά μέσω κάποιου είδους απομακρυσμένης κλήσης (remote invocation ή remote procedure call (RPC)), μέσω ενός πλήθους πρωτοκόλλων επικοινωνίας (π.χ. SOAP, HTTP, κ.ά.). Για παράδειγμα, όπως συζητήθηκε στο Κεφάλαιο 1, οι διεπαφές που βασίζονται στη γλώσσα WSDL, συμβατές είτε με την έκδοση 1.1 είτε με την έκδοση 2.0, οφείλουν να προσδιορίζουν ρητά *λειτουργίες* (*operations*) ώστε να ορίσουν στοιχεία `PortType` (WSDL 1.1) ή `Interface` (WSDL 2.0) που περιγράφουν τις διεπαφές. Η κυριαρχία του RPC ως μοντέλο αλληλεπίδρασης από τα πρώτα ήδη συστήματα υπηρεσιών έχει αποδοθεί σε ένα σύνολο λόγων το οποίο περιλαμβάνει τόσο τεχνολογικούς όσο και ιστορικούς ή «πολιτισμικούς» (cultural). Για παράδειγμα, οι διαδικασίες και οι μέθοδοι αντικειμένων αποτελούν το de facto προγραμματιστικό μοντέλο για να μπορεί ένα δομοστοιχείο λογισμικού να χρησιμοποιήσει λειτουργικότητα ενός δεύτερου δομοστοιχείου μέσω απευθείας τοπικής κλήση. Ομοίως, τα συστήματα κατανεμημένων αντικειμένων στηρίζονταν επίσης σε κλήσεις μεθόδων, διατηρώντας τον προσανατολισμό σε διαδικασίες σε ό,τι αφορά στην έκθεση και κατανάλωση λειτουργικότητας. Υπό αυτή την άποψη, η κατασκευή υπηρεσιών που εκθέτουν λειτουργίες, όπως και η υλοποίηση εφαρμογών πελάτη οι οποίες καταναλώνουν λειτουργικότητα μέσω κλήσεων λειτουργιών, ήταν σε συμφωνία με το κυρίαρχο προγραμματιστικό μοντέλο και ως εκ τούτου ο προσανατολισμός σε διαδικασίες, σε ό,τι αφορά τη σχεδίαση διεπαφών, μπορεί να θεωρηθεί ως μια φυσική επέκταση του καθιερωμένου νοητικού μοντέλου για την ολοκλήρωση δομοστοιχείων λογισμικού.

Το αρχιτεκτονικό στυλ *Μεταφοράς Αναπαραστατικής Κατάστασης* (Representational State Transfer (REST)) είναι ένα αρχιτεκτονικό στυλ που έχει χρησιμοποιηθεί επίσης για τη σχεδίαση και υλοποίηση υπηρεσιοστρεφών αρχιτεκτονικών. Το REST ορίστηκε από τον Roy Fielding [53] ως ένα αρχιτεκτονικό στυλ για κατανεμημένα συστήματα υπερμέσων, όντας ουσιαστικά μια αφαίρεση των στοιχείων

και των περιορισμών που χαρακτηρίζουν τη σύγχρονη (μετά-1994) αρχιτεκτονική του Παγκόσμιου Ιστού. Τυπικά, το REST ορίζεται ως ένα συγκροτημένο σύνολο περιορισμών το οποίο εφαρμόζεται σε μια συλλογή αρχιτεκτονικών στοιχείων, οριοθετώντας τα χαρακτηριστικά τους και τους ρόλους τους. Ένα από τα πιο σημαντικά στοιχεία που ορίζονται στο REST είναι ο *πόρος* (resource), ο οποίος ορίζεται ως ένα στοιχείο αφαίρεσης της πληροφορίας. Κάθε πληροφοριακή οντότητα η οποία φέρει διακριτή σημασιολογία και *μπορεί να ονομαστεί* [53], μπορεί να αναγνωριστεί ως πόρος. Αν και το REST ορίστηκε έχοντας υπόψη εφαρμογές κατανεμημένων υπερμέσων (όπως ο Παγκόσμιος Ιστός), έχει από τότε χρησιμοποιηθεί εκτεταμένα για την κατασκευή συστημάτων υπηρεσιών [128], [158], [46]. Σε αντίθεση με το μοντέλο αλληλεπίδρασης απομακρυσμένης κλήσης, στο οποίο χρησιμοποιούνται λειτουργίες ή δραστηριότητες ελεύθερης σημασιολογίας ως μέσα έκθεσης των δυνατοτήτων μιας υπηρεσίας, στα RESTful συστήματα υπηρεσιών η λειτουργικότητα αναλύεται και εκτίθεται μέσω ομοιόμορφων διεπαφών. Μια ομοιόμορφη διεπαφή επιτρέπει την αλληλεπίδραση με τους πόρους ενός συστήματος υπηρεσιών, χρησιμοποιώντας ένα (αμετάβλητο) σύνολο καθολικών πράξεων. Κατά αυτόν τον τρόπο, οι RESTful αλληλεπιδράσεις επιτρέπουν την ανάκληση και μεταχείριση πόρων μέσω αναπαραστάσεων των καταστάσεων των πόρων, οι οποίες αναπαραστάσεις ανταλλάσσονται μεταξύ των καταναλωτών υπηρεσιών και των παρόχων υπηρεσιών. Αντί η οργάνωση και η έκθεση των δυνατοτήτων των υπηρεσιών να γίνεται μέσω δραστηριοτήτων ή διαδικασιών, όπως συμβαίνει στις ΠΣΔ υπηρεσιοστρεφείς αρχιτεκτονικές, οι ΠΣΠ ή RESTful υπηρεσιοστρεφείς αρχιτεκτονικές ακολουθούν ένα βασισμένο σε οντότητες υπόδειγμα για τη σχεδίαση διεπαφών.

Όπως συζητήθηκε προηγουμένως, η ΠΣΔ σχεδίαση διεπαφών υπηρεσιών αποτελεί την καθιερωμένη προσέγγιση στην πραγμάτωση υπηρεσιοστρεφών αρχιτεκτονικών. Εκτός από την εδραιωμένη νοοτροπία του προσανατολισμού σε διαδικασίες που συζητήθηκε παραπάνω, η ευρεία υιοθέτηση ΠΣΔ υπηρεσιών υποστηρίζεται από την ωριμότητα της στοίβας πρωτοκόλλων, προτύπων και τεχνολογιών των παραδοσιακών SOA υλοποιήσεων. Πιο συγκεκριμένα, μια πλειάδα ζητημάτων που αφορούν τόσο σε λειτουργικές όσο και σε μη λειτουργικές απαιτήσεις (π.χ. ασύγχρονη επικοινωνία, ποιότητα υπηρεσιών (Quality-of-Service), κ.ά.) έχουν αποτελέσει αντικείμενα αντίστοιχων προτύπων και εργαλείων, επιτρέποντας στους αρχιτέκτονες και τους μηχανικούς υπηρεσιών να ανταποκριθούν σε απαιτητικά και σύνθετα σενάρια ολοκλήρωσης συστημάτων.

Παρ' όλα αυτά, είναι κοινώς αποδεκτό ότι υπάρχει ένα πλήθος εφαρμογών που μπορούν να ωφεληθούν από την ΠΣΠ έκθεση των δυνατοτήτων υπηρεσιών και συγκεκριμένα, μέσω RESTful υπηρεσιών Παγκοσμίου Ιστού.

Στο επίπεδο αρχιτεκτονικής, μελέτες [62], [117], [114] δείχνουν ότι ο προσανατολισμός σε πόρους και το REST χαρακτηρίζονται από ένα σύνολο ισχυρών σημείων σχετικά με την πραγμάτωση υπηρεσιοστρεφών συστημάτων, όπως για παράδειγμα η απλότητα των διεπαφών, η προτυποποίηση, τα χαμηλά εμπόδια ως προς την υιοθέτηση, η δυνατότητα ανακάλυψης, η βελτιωμένη ορατότητα των αλληλεπιδράσεων, κ.ά. Επίσης, έχει υποστηριχθεί ότι η ΠΣΠ σχεδίαση υπηρεσιών επιτρέπει την αντιμετώπιση ζητημάτων πολυπλοκότητας διεπαφών με καλύτερα κλιμακούμενο τρόπο από εκείνο του RPC μοντέλου αλληλεπίδρασης, το οποίο εγκαθίσταται με το ΠΣΔ υπόδειγμα [151], [49]. Αυτό επιτυγχάνεται με την αντικατάσταση των κατά παραγγελία και ειδικών ανά εφαρμογή πρωτοκόλλων τα οποία προωθούνται από το RPC και συνήθως τείνουν να αυξάνουν την πολυπλοκότητα, με τις ομοιόμορφες διεπαφές του REST. Επιπρόσθετα, συγκρινόμενες με τις ΠΣΔ διεπαφές, οι RESTful διεπαφές επιδεικνύουν ένα βελτιωμένο επίπεδο λανθάνουσας σύζευξης όπως έχει αναγνωριστεί εξετάζοντας ποικίλες διαστάσεις σχεδίασης υπηρεσιών [114]. Επιπλέον, οι υπηρεσίες που εκτίθενται ακολουθώντας το ΠΣΠ υπόδειγμα, μπορούν να επωφεληθούν από τις δυνατότητες διαστρωμάτωσης των RESTful αρχιτεκτονικών, όπως η χρήση ενδιάμεσων δομοστοιχείων κρυφής μνήμης και εξισορρόπησης φορτίου. Τελικά, η ΠΣΠ προσέγγιση θεωρείται ότι προωθεί την αυθόρμητη και δημιουργική επαναχρησιμοποίηση και σύνθεση της λειτουργικότητας και της πληροφορίας που εκτίθεται μέσω των πόρων, καθώς η αναγνώριση και η διαχείριση των πόρων μπορεί να στηριχθεί σε ευρέως χρησιμοποιούμενα ανοιχτά πρότυπα (π.χ. URI [92]) και εκτεταμένης αποδοχής πρωτόκολλα (π.χ. HTTP [50]) [153].

Πέρα από τις αρχιτεκτονικές και εννοιολογικές διαστάσεις του διλήμματος ΠΣΔ έναντι ΠΣΠ σχετικά με τη σχεδίασης διεπαφών, υπάρχουν επίσης ζητήματα που αφορούν στο επίπεδο υλοποίησης και σχετίζονται με την υιοθέτηση ενός από τα δύο υποδείγματα. Παράλληλα με την πολυπλοκότητα ορισμένων απαιτήσεων σχετικά με τον κύκλο ζωής των δομοστοιχείων και τις δυνατότητες ολοκλήρωσής τους, η πολυπλοκότητα των διεπαφών και το επίπεδο σύζευξης που επιβάλλεται από το RPC μοντέλο, συνηγορεί στη χρησιμοποίηση τυπικά απαιτητικών και ακριβών πλατφορμών (π.χ. WS-* πλατφόρμες) για την υποστήριξη υλοποιήσεων υπηρεσιών [152]. Την ίδια στιγμή, υπηρεσίες βασισμένες στο REST και το HTTP στηρίζονται τυπικά

σε ανοιχτά πρότυπα και μπορούν να υλοποιηθούν χρησιμοποιώντας τις τεχνολογίες του Παγκόσμιου Ιστού. Επιπλέον, πρόσφατες μελέτες δείχνουν ότι η επίδοση των υπηρεσιών αυτών, ως προς τον χρόνο και το μέγεθος των μηνυμάτων είναι καλύτερη σε σχέση με τις αντίστοιχες επιδόσεις των υπηρεσιών βασισμένων σε SOAP και XML [89], [33]. Επιπρόσθετα, μελέτες στις οποίες αναλύθηκε η απαιτούμενη προσπάθεια για ορισμένες δραστηριότητες της τεχνολογίας λογισμικού, δείχνουν ότι οι RESTful υπηρεσίες είναι καλύτερα συντηρήσιμες έναντι των αντίστοιχων βασισμένων σε SOAP υπηρεσιών [39] και είναι επίσης ευκολότερο να αποτελέσουν τμήματα συνθέσεων υπηρεσιών [78]. Τέλος, εφαρμογές νέας γενιάς όπως αυτές του Σημασιολογικού Ιστού (Semantic Web) και των Συνδεδεμένων Δεδομένων (Linked Data) φαίνεται να μπορούν να αντλήσουν οφέλη από την παροχή RESTful διεπαφών [14].

Σύμφωνα με τα παραπάνω, δεν αποτελεί έκπληξη το γεγονός ότι ο αριθμός των διεπαφών που χρησιμοποιούν στοιχεία του REST αυξάνεται [67], αν και αυτά τα API εμφανίζουν διάφορα επίπεδα συμμόρφωσης στους περιορισμούς του REST [127], [86], [35], ξεκινώντας από χρήση των URIs πάνω από το HTTP εν είδει σηράγγων (URI tunneling), ως RESTful διεπαφές με ενεργή χρήση υπερμέσων (hypermedia-enabled). Αν και οι περιορισμοί του REST δεν εφαρμόζονται πλήρως στην πλειονότητα των διεπαφών, είτε λόγω αρχιτεκτονικών συμβιβασμών, είτε εξαιτίας παρανοήσεων σχετικά με το ίδιο το REST, η αύξηση αυτή δείχνει μια σημαντική τάση που στοχεύει στην ολοένα και περισσότερο ΠΣΠ έκθεση των δυνατοτήτων των υπηρεσιών.

2.2 Υποδείγματα διεπαφών υπηρεσιών

Βάσει της παραπάνω συζήτησης για τα αρχιτεκτονικά στυλ των υπηρεσιών και για τη σχεδίασης διεπαφών υπηρεσιών, εισάγουμε έναν ορισμό για την έννοια του υποδείγματος διεπαφών υπηρεσιών και αντίστοιχους ορισμούς για τα δύο κύρια υποδείγματα διεπαφών στην περιοχή της υπηρεσιοστρεφούς υπολογιστικής.

Αρχικά, ορίζουμε την έννοια του *υποδείγματος διεπαφών υπηρεσιών* (*service interface paradigm*) ως εξής:

Ορισμός 1 (Υπόδειγμα διεπαφών υπηρεσιών)**.** *Ένα υπόδειγμα διεπαφών υπηρεσιών είναι μια αναγνωρισμένη συλλογή σχεδιαστικών αποφάσεων διεπαφής από τις οποίες απορρέει μια διακριτή τεχνοτροπία ανάλυσης και έκθεσης λειτουργικότητας*

υπηρεσιών.

Συνεπώς, τα δύο κυρίαρχα υποδείγματα διεπαφών υπηρεσιών που παρουσιάστηκαν παραπάνω ορίζονται ως εξής:

Ορισμός 2 (Υπόδειγμα διεπαφών υπηρεσιών προσανατολισμένο σε διαδικασίες)**.** *Το υπόδειγμα διεπαφών υπηρεσιών προσανατολισμένο σε διαδικασίες (ΠΣΔ) (procedure-oriented service interface paradigm) προβλέπει ότι μια υπηρεσία αναλύεται και εκθέτει τη λειτουργικότητά της μέσω αλληλεπιδράσεων με λειτουργίες (operations), οι οποίες υποδηλώνουν καλά ορισμένες πράξεις ή δραστηριότητες και χαρακτηρίζονται από διακριτά ονόματα και δομικές υπογραφές.*

Ορισμός 3 (Υπόδειγμα διεπαφών υπηρεσιών προσανατολισμένο σε πόρους)**.** *Το υπόδειγμα διεπαφών υπηρεσιών προσανατολισμένο σε πόρους (ΠΣΠ) (procedure-oriented service interface paradigm) προβλέπει ότι μια υπηρεσία αναλύεται και εκθέτει τη λειτουργικότητά της μέσω αλληλεπιδράσεων με πόρους (resource) όπως ορίζονται στο αρχιτεκτονικό στυλ REST.*

Στη συνέχεια της διατριβής, για λόγους οικονομίας χώρου, χρησιμοποιείται ο όρος *υπόδειγμα* αναφέρεται σε *υπόδειγμα διεπαφών υπηρεσιών*, εκτός αν σημειώνεται διαφορετικά.

2.3 Αρχές και εφαρμογή του REST

Το ΠΣΠ υπόδειγμα αναδύθηκε ως εναλλακτική επιλογή στο επικρατές ΠΣΔ υπόδειγμα και δεν έχει προς το παρόν υιοθετηθεί στο βαθμό που χρησιμοποιείται το ΠΣΔ υπόδειγμα, ενώ τα χαρακτηριστικά του και οι αρχές του είναι λιγότερο κατανοητές. Δεδομένου του ότι το REST αποτελεί προσδιοριστικό παράγοντα για το ΠΣΠ υπόδειγμα, η κατανόηση των χαρακτηριστικών του παίζει σημαντικό ρόλο στην αποτελεσματική παρουσίαση των τεχνικών και των περιβαλλόντων-πλαισίων που παρουσιάζονται στην παρούσα διατριβή. Ως εκ τούτου, στην ενότητα αυτή μελετάμε τα χαρακτηριστικά αυτού του αρχιτεκτονικού στυλ, παρουσιάζοντας τόσο τις θεωρητικές βάσεις του στυλ, όσο και πρακτικές διαστάσεις που προκύπτουν από τη χρησιμοποίησή του στην περιοχή της υπηρεσιοστρεφούς υπολογιστικής.

2.3.1 Θεωρία του REST

Περιορισμοί και επαγόμενες ιδιότητες

Το στυλ μεταφοράς αναπαραστατικής κατάστασης REpresentational State Transfer (REST) είναι ένα αρχιτεκτονικό στυλ ορισμένο για κατανεμημένα συστήματα υπερμέσων το οποίο προτάθηκε από το Roy Fielding στη διδακτορική του διατριβή [53]. Το REST αναπτύχθηκε στο πλαίσιο της σύγχρονης (μετά το 1994) αρχιτεκτονικής του Παγκόσμιου Ιστού, κατά τη διάρκεια που ο Fielding ασχολούνταν με πρότυπα του Παγκόσμιου Ιστού όπως αυτά του HTTP πρωτοκόλλου, του URI και αντίστοιχων εργαλείων (Apache Web Server). Τυπικά το REST ορίζεται ως ένα συγκροτημένο σύνολο των παρακάτω αρχιτεκτονικών περιορισμών: *Πελάτη-Εξυπηρετητή* (*Client-Server*), *Επικοινωνία χωρίς μνήμη κατάστασης* (*Stateless (communication)*), *Κρυφή μνήμη* (*Cache*), *Ομοιόμορφη διεπαφή* (*Uniform Interface*), *Διαστρωματωμένο σύστημα* (*Layered System*), και του προαιρετικού *Κατά παραγγελία κώδικα* (Code-on-Demand). Ο Πίνακας 2.1 παρέχει μια συνοπτική περιγραφή του κάθε περιορισμού. Οι τρεις πρώτοι περιορισμοί του REST (*Client-Server*, *Stateless*, και *Cache*) εφαρμόστηκαν στον Παγκόσμιο Ιστό ήδη από την αρχική του αρχιτεκτονική, ενώ οι επόμενοι τρεις (*Uniform Interface*, *Layered System*, και *Code-on-Demand*) ορίστηκαν και εφαρμόστηκαν καθώς ο Ιστός εξελισσόταν.

Πίνακας 2.1: Αρχιτεκτονικοί περιορισμοί του REST

Περιορισμός	Περιγραφή
Πελάτη-Εξυπηρετητή	Ο περιορισμός Πελάτη-Εξυπηρετητή καθιερώνει ένα μοντέλο αλληλεπίδρασης το οποίο διαχωρίζει το ρόλο της υποβολής αιτήματος για μια υπηρεσία και το ρόλο παροχής μιας υπηρεσίας.
Επικοινωνία χωρίς μνήμη κατάστασης	Ο περιορισμός επικοινωνίας χωρίς μνήμη κατάστασης αναφέρεται στην επικοινωνία μεταξύ πελάτη και εξυπηρετητή και απαιτεί κάθε αίτημα από τον πελάτη στον εξυπηρετή να είναι ανεξάρτητο από προηγούμενα αιτήματα. Συνεπώς, δεν υπάρχει στην πλευρά του εξυπηρετητή κατάσταση συνεδρίας (session state) και επιπλέον, κάθε αίτημα θα πρέπει να είναι αρκετά περιγραφικό ώστε να μπορεί να γίνεται πλήρως κατανοητό χωρίς περαιτέρω πληροφορία.
Κρυφή μνήμη	Ο περιορισμός Κρυφής μνήμης απαιτεί ότι οι απαντήσεις του εξυπηρετητή πρέπει να υποδεικνύουν ή να υπονοούν κατά πόσο μπορούν να αποθηκευτούν προσωρινά στα πλαίσια κρυφών μνημών ή όχι.
Ομοιόμορφη διεπαφή	Ο περιορισμός Ομοιόμορφης διεπαφής επιβάλλει τη γενικότητα των διεπαφών των δομοστοιχείων και απαιτεί ότι οι διεπαφές αυτές πρέπει να έχουν καθολική σημασιολογία κατά μήκος του συστήματος. Επίσης, ο περιορισμός αυτός εισάγει και περιγράφει την προσανατολισμένη σε πόρους μοντελοποίηση των περιεχομένων ενός συστήματος, όπως υπαγορεύεται από την εξάρτησή του και την αποσύνθεσή του σε περαιτέρω αρχιτεκτονικούς περιορισμούς, οι οποίοι συνήθως αναφέρονται ως οι υπο-περιορισμοί του περιορισμού Ομοιόμορφης διεπαφής. Συγκεκριμένα, το REST ορίζει ότι η ομοιομορφία των διεπαφών σε μια RESTful αρχιτεκτονική αντλείται από τη συμμόρφωσή της στους εξής τέσσερις περιορισμούς: *αναγνώριση πόρων* (*identification of resources*), *μεταχείριση πόρων μέσω αναπαραστάσεων* (*manipulation of resources through representations*), *αυτο-περιγραφικά μηνύματα* (*self-descriptive messages*) και *υπερμέσα ως ο μηχανισμός της κατάστασης της εφαρμογής* (*hypermedia as the engine of application state*) ο οποίος αναφέρεται συνήθως ως HATEOAS ή *περιορισμός υπερμέσων*. Εξαιτίας του σημαντικού του ρόλου ως προς τη σχεδίαση διεπαφών υπηρεσιών, ο περιορισμός Ομοιόμορφης διεπαφής εξετάζεται λεπτομερώς στην παρούσα διατριβή στο Κεφάλαιο 3, Ενότητα 3.3.2.
Διαστρωματωμένο σύστημα	Ο περιορισμός του Διαστρωματωμένου συστήματος απαιτεί ότι η οργάνωση του συστήματος ακολουθεί ένα ιεραρχικό, διαστρωματωμένο τρόπο, στον οποίο κάθε επίπεδο παρέχει υπηρεσίες στο επίπεδο πάνω από αυτό και καταναλώνει υπηρεσίες από το επίπεδο κάτω από αυτό.
Κατά παραγγελία κώδικας	Ο κατά παραγγελία κώδικας επιτρέπει να επεκταθεί η λογική των καταναλωτών με κώδικα ο οποίος μπορεί να μεταφορτωθεί και να εκτελεστεί. Ο τελευταίος αυτός περιορισμός αποτελεί προαιρετικό περιορισμό του REST και ο Fielding σημειώνει ότι θα πρέπει να υποστηρίζεται από μια αρχιτεκτονική που συμμορφώνεται στο REST στη γενική περίπτωση. Ωστόσο, ενδέχεται να υπάρχουν περιβάλλοντα στα οποία η συμπεριφορά αυτή είναι απενεργοποιημένη και η δυνατότητα αυτή θα πρέπει να θεωρείται αποδεκτή.

Ο περιορισμός Ομοιόμορφης διεπαφής (Uniform Interface), θεωρείται κεντρικό χαρακτηριστικό του REST και συντίθεται από τέσσερις υπο-περιορισμούς και συγκεκριμένα από τους εξής: *α*) *αναγνώριση πόρων* (*identification of resources*), *β*) *μεταχείριση πόρων μέσω αναπαραστάσεων* (*manipulation of resources through representations*), *γ*) *αυτο-περιγραφικά μηνύματα* (*self-descriptive messages*) και *δ*) *υπερμέσα ως ο μηχανισμός της κατάστασης της εφαρμογής* (*hypermedia as the engine of application state (HATEOAS)*). Μέσω περαιτέρω ανάλυσης των υποπεριορισμών, μπορεί να αναγνωριστεί ένα σύνολο δώδεκα σχεδιαστικών κριτηρίων τα οποία διευκολύνουν την πραγματοποίηση RESTful σχεδιάσεων, όπως έχει προταθεί από την ομάδα μας στην εργασία [8]. Το περιβάλλον-πλαίσιο που προτάθηκε στην εργασία αυτή συζητείται λεπτομερώς στο Κεφάλαιο 3.

Ξεκινώντας από το *κενό* (*null*) στυλ, ο Fielding δείχνει πως αντλείται το REST συνδυάζοντας περιορισμούς που εντοπίζονται σε περισσότερο λεπτομερή αρχιτεκτονικά στυλ (χαμηλότερου επιπέδου αφαίρεσης). Επίσης, στη διατριβή του ο Fielding υπογραμμίζει το γεγονός ότι το REST ορίζει ένα *συγκροτημένο* σύνολο περιορισμών, έτσι ώστε η συνδυαστική εφαρμογή τους μπορεί να επάγει ή να επηρεάσει ορισμένες ιδιότητες σε όλη την έκταση της αρχιτεκτονικής. Επιπλέον, συζητά συνοπτικά ποιες ιδιότητες επηρεάζει ο κάθε περιορισμός του REST και με ποιον τρόπο -το θέμα αυτό αποτέλεσε στη συνέχεια αντικείμενο περαιτέρω έρευνας [106], [5]. Ο Πίνακας 2.2 συνοψίζει αυτές τις σχέσεις μεταξύ περιορισμών και ιδιοτήτων, παραθέτοντας μια λίστα με επαγόμενες ή επηρεαζόμενες ιδιότητες στο συνολικό εύρος της αρχιτεκτονικής, για κάθε περιορισμό, σημειώνοντας αν υπάρχει θετική ή αρνητική επίδραση.

Πίνακας 2.2: Περιορισμοί του REST και αρχιτεκτονικές ιδιότητες

Περιορισμός	Ιδιότητες
Client-Server	portability (+), (system) simplification (+), reliability (+), flexibility (+)
Stateless (communication)	(system) simplification (+), scalability (+), reliability (+), efficiency (-)
Cache	scalability (+), efficiency (+), performance (+), reliability (-)
Uniform Interface	(system) simplification (+), reusability (+), evolvability (+), visibility (+), efficiency (-)
Layered System	scalability (+), (system) simplification (+), abstraction (+), latency (-)
Code-On-Demand	performance (+), extensibility (+), configurability (+), visibility (-)

Εξαιτίας της συγκροτημένης φύσης του συνόλου των περιορισμών, η ανάλυση που παρουσιάζεται στην εργασία [106] δείχνει ότι όταν ένα σύστημα που επιχειρεί να εφαρμόσει το REST αποκλίνει αρχιτεκτονικά από κάποιο υποσύνολο των περιορισμών του, είναι πιθανό να υπάρξουν παρενέργειες σε ό,τι αφορά τις επαγόμενες ιδιότητες. Η ανάλυση εκτελείται χρησιμοποιώντας διαγράμματα επιρροής τα οποία αντικατοπτρίζουν θετικές και αρνητικές επιπτώσεις μεταξύ των αρχιτεκτονικών περιορισμών και των ιδιοτήτων. Οι συγγραφείς σημειώνουν ότι τέτοια διαγράμματα μπορούν να κατασκευαστούν και να χρησιμοποιηθούν για τη συστηματική μελέτη του πως διαφορετικά επίπεδα συμμόρφωσης στο REST μπορεί να επηρεάζουν τις επαγόμενες ιδιότητες και για την αναγνώριση συμβιβασμών που εμπλέκονται στις αντίστοιχες αρχιτεκτονικές αποφάσεις. Επίσης, στην ανάλυση που παρουσιάζεται στο [5], πέρα από τις αρχιτεκτονικές ιδιότητες, εξετάζεται η επίδραση των περιορισμών του REST στα ποικίλα κόστη κατά τον κύκλο ζωής μιας αρχιτεκτονικής (αρχικό, κόστος συντήρησης και κόστος εξέλιξης).

Αρχιτεκτονικά στοιχεία

Στα πλαίσια του REST ορίζεται ένα σύνολο αρχιτεκτονικών στοιχείων διαφορετικών τύπων όπως *συνδέσμων* (*connectors*), δομοστοιχείων (*components*) και *στοιχείων δεδομένων* (*data elements*). Οι περιορισμοί του REST οριοθετούν τους ρόλους, τα χαρακτηριστικά και τις σχέσεις αυτών των αρχιτεκτονικών στοιχείων.

Σύνδεσμοι (Connectors): Ο ρόλος των συνδέσμων είναι να διαχειρίζονται τη δικτυακή επικοινωνία και να επιτρέπουν στα δομοστοιχεία να ανταλλάσσουν μηνύματα. Στο REST, ορίζονται οι ακόλουθοι σύνδεσμοι: *πελάτης* (*client*), *εξυπηρετητής* (*server*), *κρυφή μνήμη* (*cache*), *αναλυτής* (*resolver*) και *σήραγγα* (*tunnel*). Ο ρόλος ενός συνδέσμου πελάτη είναι να ανοίγει συνδέσεις και να αποστέλλει αιτήματα ενώ ο ρόλος ενός συνδέσμου εξυπηρετητή είναι να αναμένει εισερχόμενες συνδέσεις και να παρέχει πρόσβαση σε δεδομένα και λειτουργικότητα. Ένας σύνδεσμος κρυφής μνήμης μεσολαβεί στην επικοινωνία πελάτη-εξυπηρετητή και παρεμβαίνει όταν είναι εφικτό για την αποφυγή άσκοπης επικοινωνίας ή τη δημιουργία απαντήσεων μέσω της απομνημόνευσης και επαναχρησιμοποίησης προηγούμενων απαντήσεων. Ο ρόλος ενός συνδέσμου αναλυτή είναι να παρέχει πληροφορίες δικτύου έτσι ώστε να είναι εφικτή η πρόσβαση σε πόρους-στόχους. Ένας σύνδεσμος τύπου αναλυτή είναι σε θέση να εντοπίζει τους ζητούμενους πόρους χρησιμοποιώντας αναγνωριστικά πόρων. Τέλος, ένας σύνδεσμος σήραγγα λειτουργεί ως ένας αναμεταδότης επικοινωνίας μεταξύ των επικοινωνούντων στοιχείων και μοντελοποιείται στο REST ώστε να συμπεριλάβει περιπτώσεις στις οποίες δομοστοιχεία τα οποία μεσολαβούν ενεργά σε επικοινωνίες πρέπει να αλλάξουν το ρόλο τους και να προωθούν τα μηνύματα με ουδέτερο τρόπο.

Δομοστοιχεία (Components): Τα αρχιτεκτονικά δομοστοιχεία χρησιμοποιούν τις αφηρημένες διεπαφές των συνδέσμων για να συνδεθούν με το περιβάλλον τους και να εκπληρώσουν τους στόχους τους. Το REST ορίζει τα εξής δομοστοιχεία: *αντιπρόσωπος χρήστη* (*user agent*), *πηγαίος εξυπηρετητής* (*origin server*), *πληρεξούσιος* (*proxy*), και *δίοδος* (*gateway*). Οι αντιπρόσωποι χρήστη χρησιμοποιούν συνδέσμους πελάτες για να στείλουν αιτήματα που περιέχουν μηνύματα ή αναπαραστάσεις πόρων. Οι πηγαίοι εξυπηρετητές χρησιμοποιούν συνδέσμους πελάτες για να ανταποκριθούν σε αιτήματα, επιστρέφοντας μηνύματα αποκρίσεων ή αναπαραστάσεις πόρων πίσω στους αντιπροσώπους χρηστών. Ο πηγαίος εξυπηρετητής είναι υπεύθυνος να κρύβει τις υλοποιήσεις των πόρων και να παρέχει μία διεπα-

φή στις υπηρεσίες του υπό τη μορφή μιας ιεραρχίας πόρων. Τα δομοστοιχεία πληρεξούσιων και διόδων είναι ενδιάμεσα δομοστοιχεία, τα οποία μεσολαβούν στις επικοινωνίες και ενδέχεται να αναλάβουν δράση ώστε να βελτιώσουν την ποιότητα διαφόρων διαστάσεων των υπηρεσιών π.χ. την επίδοση και την ασφάλεια, ή να εφαρμόσουν συγκεκριμένες μεταφράσεις ή μετασχηματισμούς στα δεδομένα που ανταλλάσσονται. Ένα δομοστοιχείο πληρεξούσιου προσδιορίζεται από τον πελάτη ενώ ένα δομοστοιχείο διόδου προσδιορίζεται από το δίκτυο ή τον πηγαίο εξυπηρετητή ώστε να λειτουργεί ως ένας αντίστροφος πληρεξούσιος.

Στοιχεία δεδομένων (Data elements): Πέρα από τους συνδέσμους και τα δομοστοιχεία, τα αρχιτεκτονικά στοιχεία του REST περιλαμβάνουν και ορισμένους τύπους δεδομένων οι οποίοι χαρακτηρίζονται από συγκεκριμένη σημασιολογία και ρόλους και χρησιμοποιούνται για την προδιαγραφή των μηνυμάτων που ανταλλάσσονται μεταξύ των δομοστοιχείων. Το REST ορίζει τα εξής στοιχεία δεδομένων: *πόρος* (*resource*), *αναγνωριστικό πόρου* (*resource identifier*), *αναπαράσταση* (*representation*), *μεταδεδομένα πόρων* (*resource metadata*), και *δεδομένα ελέγχου* (*control data*), τα οποία περιγράφονται λεπτομερώς στον Πίνακα 2.3.

Πίνακας 2.3: Στοιχεία δεδομένων στο REST

Στοιχείο δεδομένων	Περιγραφή
Πόρος (Resource	Οι πόροι του REST αποτεθούν βασικές αφαιρέσεις της πληροφορίας και παρέχουν παρέχουν εννοιολογικές αντιστοιχίες σε οντότητες με διακριτό και αναγνωρίσιμο πληροφοριακό περιεχόμενο. Συνοπτικά, *οποιαδήποτε πληροφορία μπορεί να έχει όνομα* [53] μπορεί να είναι πόρος.
Αναγνωριστικό πόρου (Resource identifier)	Ένας πόρος αναγνωρίζεται από ένα ή περισσότερα αναγνωριστικά, τα οποία χρησιμοποιούνται στα μηνύματα που ανταλλάσσονται για την πρόσβαση, τη μεταχείριση και τη διασύνδεση των αντίστοιχων πόρων.
Αναπαράσταση (Representation)	Ένας πόρος μπορεί να έχει μια ή περισσότερες αναπαραστάσεις οι οποίες μεταφέρουν πληροφορίες για την κατάστασή του. Συγκεκριμένα, μια αναπαράσταση περιέχει στοιχεία *δεδομένων* (*data*) και *μεταδεδομένων* (*metadata*). Τα δεδομένα αναπαραστάσεων αποτυπώνουν και εκθέτουν την τρέχουσα, την προηγούμενη ή τη μελλοντική (επιθυμητή) κατάσταση ενός πόρου. Τα μεταδεδομένα αναπαραστάσεων παρέχουν πληροφορίες σχετικά με το πως πρέπει να ερμηνεύονται τα δεδομένα αναπαραστάσεων.
Μεταδεδομένα πόρων (Resource metadata)	Ένας πόρος μπορεί να χαρακτηρίζεται από στοιχεία μεταδεδομένων τα οποία προσφέρουν πληροφορίες για τον πόρο. Τα μεταδεδομένα αυτά παρέχονται από τον εξυπηρετητή στον πελάτη και είναι ανεξάρτητα από τις συγκεκριμένες αναπαραστάσεις που ανταλλάσσονται.
Δεδομένα ελέγχου (Control data)	Κατά τις αλληλεπιδράσεις πελάτη-εξυπηρετητή, ανταλλάσονται δεδομένα ελέγχου τα οποία μεταφέρουν την πρόθεση της αλληλεπίδρασης και παρέχουν περαιτέρω πληροφορίες στον αποδέκτη του μηνύματος έτσι ώστε η ακόλουθη συμπεριφορά του και η ερμηνεία του μηνύματος να μπορεί να προσαρμοστεί κατάλληλα.

2.3.2 RESTful υπηρεσίες Ιστού

Το REST ορίστηκε ως αρχιτεκτονικό στυλ έχοντας υπόψη κατανεμημένα συστήματα υπερμέσων. Παρ' όλα αυτά, συγκέντρωσε σημαντική προσοχή από την κοινότητα της υπηρεσιοστρεφούς υπολογιστικής και έχει χρησιμοποιηθεί ευρέως στην κατασκευή συστημάτων που προσφέρουν υπηρεσίες. Καθώς το REST συγκέντρωνε το

ενδιαφέρον της κοινότητας του SOC, ερευνητές και μηχανικοί λογισμικού στην περιοχή πρότειναν και καθιέρωσαν μοτίβα, συμβάσεις και βέλτιστες πρακτικές για την κατασκευή υπηρεσιοστρεφών συστημάτων χρησιμοποιώντας το REST, έτσι ώστε οι αρχιτεκτονικές που παράγονται να είναι συμβατές όχι μόνο με τους περιορισμούς του αρχιτεκτονικού στυλ αλλά και να υπηρετούν και να εφαρμόζουν τις αρχές του προσανατολισμού σε υπηρεσίες [128], [158], [46]. Επιπρόσθετα, αναπτύχθηκε ένα σύνολο προτύπων και περιβαλλόντων-πλαισίων έτσι ώστε να διευκολύνεται η υλοποίηση RESTful υπηρεσιών στα πλαίσια των περισσοτέρων κύριων γλωσσών προγραμματισμού και περιβαλλόντων (π.χ. το JAX-RS στη Java [118]). Στις παραγράφους που ακολουθούν, χρησιμοποιούνται τα κύρια μοτίβα και συμβάσεις σχετικές με την κατασκευή RESTful υπηρεσιών, για να παρουσιαστεί ο τυπικός τρόπος με τον οποίο σχεδιάζονται και υλοποιούνται RESTful υπηρεσίες στην πράξη.

Όντας ένα αρχιτεκτονικό στυλ, το REST εξ ορισμού είναι τεχνολογικά ανεξάρτητο. Συνεπώς, ποικίλες τεχνολογίες μπορούν να χρησιμοποιηθούν για την κατασκευή αρχιτεκτονικών που συμμορφώνονται στο στυλ, εφόσον αυτές βρίσκονται σε αντιστοιχία με τα στοιχεία και τους περιορισμούς που θέτει το REST. Η τεχνολογική στοίβα η οποία έχει ευρέως χρησιμοποιηθεί για την πραγμάτωση RESTful αρχιτεκτονικών είναι η στοίβα των πρωτοκόλλων και των προτύπων του Παγκοσμίου Ιστού (π.χ. HTTP, URI, media types, link relations, κλπ). Υπηρεσίες που συμμορφώνονται στο REST και υλοποιούνται χρησιμοποιώντας τις τεχνολογίες του Παγκόσμιου Ιστού, αναφέρονται ως *RESTful Web services* [128]. Στο πλαίσιο αυτό, το HTTP χρησιμοποιείται ως ένα πρωτόκολλο επικοινωνίας για τον ορισμό στοιχείων που λειτουργούν ως δεδομένα ελέγχου (control data), μεταδεδομένα πόρων (resource metadata) και μεταδεδομένα αναπαραστάσεων (representation metadata), το URI πρότυπο λειτουργεί ως ένας καθολικός μηχανισμός με τον οποίο εκφράζονται αναγνωριστικά πόρων (resource identifiers), οι τύποι μέσων (media types) χρησιμοποιούνται για να προσδιορίσουν αναπαραστάσεις πόρων, οι σχέσεις σύνδεσης (link relations) εξυπηρετούν το στόχο των υπερμέσων (μεταξύ άλλων), και ούτω καθεξής. Αποτελεί συνήθη πρακτική για τις RESTful υπηρεσίες Ιστού να χρησιμοποιούν τις βασικές μεθόδους του HTTP, όπως `POST`, `GET`, `PUT`, `DELETE` ώστε να παρέχουν πρόσβαση και να δώσουν τη δυνατότητα μεταχείρισης της κατάστασης των πόρων που εκθέτουν. Επιπλέον, οι τύποι μέσων (media types), τόσο οι κατά παραγγελία, όσο και οι καταχωρημένοι στο αποθετήριο του IANA (π.χ. το ATOM Syndication

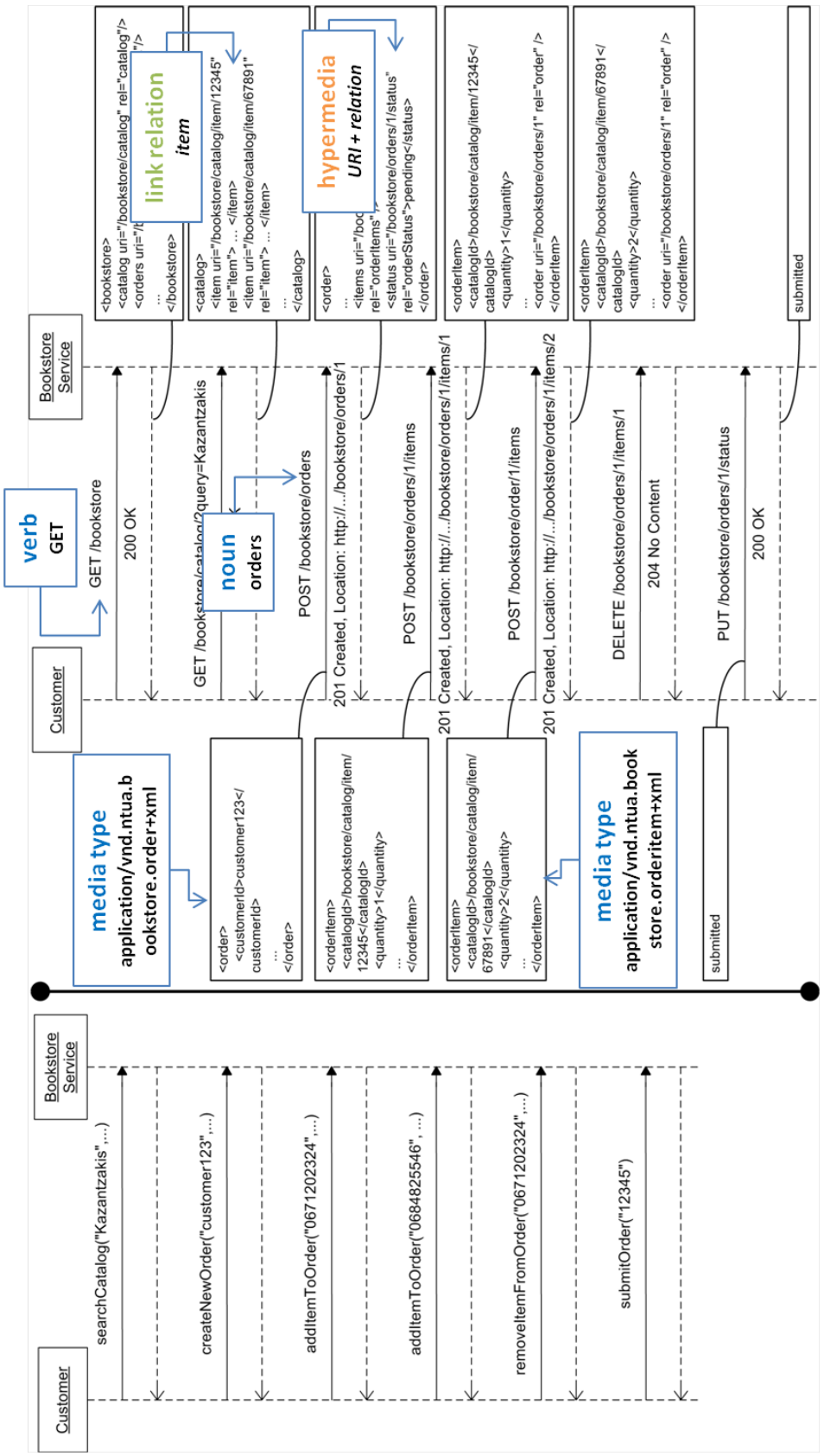
Format[1], `application/atom+xml`) χρησιμοποιούνται συνήθως για να ενσαρκώσουν αναπαραστάσεις πόρων, καθώς οι δομικοί και σημασιολογικοί προσδιορισμοί που ορίζονται στα πλαίσια ενός τύπου μέσου μπορούν να χρησιμοποιηθούν για να παρέχουν οπτικές σε καταστάσεις πόρων.

Για να δείξουμε πως το HTTP, το URI και άλλα πρότυπα και τεχνολογίες του Παγκόσμιου Ιστού μπορούν να χρησιμοποιηθούν σε ένα RESTful Web service, χρησιμοποιούμε ένα απλό παράδειγμα προμήθειας αγαθών και συγκεκριμένα μιας υπηρεσίας online καταστήματος βιβλίων, το οποίο παρουσιάζεται σε σύγκριση με μια ΠΣΔ υπηρεσίας η οποία προσφέρει την ίδια λειτουργικότητα. Το Σχήμα 2.2 απεικονίζει δύο ακολουθιακά διαγράμματα, ένα για την ΠΣΔ υπηρεσία (αριστερά) και ένα για την ΠΣΠ υπηρεσία (δεξιά).

Στην αριστερή πλευρά του σχήματος, απεικονίζεται ένα σενάριο χρήσης της ΠΣΔ διεπαφής, στο οποίο ο πελάτης διερευνά έναν κατάλογο, δημιουργεί μια παραγγελία, προσθέτει δύο αντικείμενα στην παραγγελία, αφαιρεί το πρώτο αντικείμενο από αυτή και τελικά την υποβάλλει. Οι δραστηριότητες αυτές του πελάτη αντιστοιχούν σε κλήσεις λειτουργιών της υπηρεσίας, με χρήση κατάλληλων παραμέτρων και τιμών.

Στη δεξιά πλευρά του σχήματος, απεικονίζεται το ίδιο σενάριο, αλλά αυτά τη φορά η λειτουργικότητα της υπηρεσίας καταναλώνεται μέσω μιας ΠΣΠ διεπαφής. Σε αυτό το πλαίσιο, αντί των λειτουργιών υπηρεσιών, υπάρχουν πόροι όπως οι `bookstore`, `catalog`, `orders` (συλλογή), `order`, `items`, `item`, `order status` και μέθοδοι HTTP για την πρόσβαση σε αυτούς και τη μεταχείρισή τους. Πιο συγκεκριμένα, τυπικά στις RESTful υπηρεσίες Ιστού χρησιμοποιούνται τέσσερις μέθοδοι του HTTP, η μέθοδος `GET` για την προσπέλαση ενός πόρου και την ανάκτηση της αναπαράστασής του, η μέθοδος `PUT` για την ενημέρωση της κατάστασης ενός πόρου, η μέθοδος `POST` για τη δημιουργία ενός νέου πόρου ή για την εφαρμογή ενημερώσεων της κατάστασης ενός πόρου οι οποίες δεν είναι *ταυτοδύναμες* (*idempotent*) και η μέθοδος `DELETE` για τη διαγραφή ενός πόρου, που σημαίνει ότι ο πόρος είτε καταστρέφεται είτε αφαιρείται από το ορατό πεδίο πόρων, ανάλογα με τους σκοπούς της εφαρμογής. Στο παράδειγμα που παρουσιάζεται στο Σχήμα 2.2, για την πρόσβαση στο `bookstore` αποστέλλεται ένα GET αίτημα από την εφαρμογή-πελάτη στο URI `/bookstore/`. Αυτό το HTTP αίτημα επιστρέφει ένα 200 OK κωδικό ο οποίος δείχνει ότι το αίτημα μεταφράστηκε και εξυπηρετήθη-

[1]RFC 4287, https://tools.ietf.org/html/rfc4287



Σχήμα 2.2: Παράδειγμα υπηρεσίας καταστήματος βιβλίων: WS-* και RESTful εναλλακτικές

κε επιτυχώς και επεστράφη μια αναπαράσταση του πόρου στο απαντητικό μήνυμα. Τα δεδομένα της αναπαράστασης του πόρου `bookstore` περιέχουν στοιχεία σχετικά με την κατάσταση του πόρου (δεν απεικονίζονται) και στοιχεία υπερμέσων τα οποία παρέχουν συνδέσμους σε άλλους πόρους όπως τους πόρους `catalog` και `orders` που αναγνωρίζονται από τα URIs `/bookstore/catalog/` και `/bookstore/orders/`, αντίστοιχα. Κάθε σύνδεσμος επισημειώνεται με μια σχέση σύνδεσης (link relation) μέσω της ιδιότητες rel, έτσι ώστε ο καταναλωτής της υπηρεσίας να μπορεί να ερμηνεύσει τη σημασιολογία των υπερμέσων που του παρουσιάζονται και να κατασκευάσει κατάλληλα επόμενα αιτήματα, εφόσον αυτό είναι επιθυμητό με βάση τους στόχους του. Στη συνέχεια, ένας υφιστάμενος πόρος του πόρου `catalog` προσπελάζεται, επίσης μέσω ενός αιτήματος `GET`. Συγκεκριμένα, το URI `/bookstore/catalog/?query=Kazantzakis` το οποίο περιέχει μια URI query παράμετρο, αναγνωρίζει έναν ξεχωριστό πόρο του οποίου η σημασιολογία δηλώνει μια συγκεκριμένη όψη ή προβολή των περιεχομένων του πόρου `catalog` (στο παράδειγμά μας, όλα τα βιβλία του καταλόγου των οποίων τα δεδομένα αναπαράστασης περιέχουν τη λέξη `Kazantzakis`). Ομοίως, για τη δημιουργία μιας νέας παραγγελίας αποστέλλεται ένα `POST` αίτημα στον εξυπηρετητή το οποίο αφορά στον πόρο-συλλογή `orders` και παρέχεται μια αναπαράσταση για το νέο πόρο. Ο τύπος μέσων των δεδομένων αναπαράστασης που αποστέλλονται στον εξυπηρετητή αναγνωρίζεται από ένα `Content-Type` HTTP header το οποίο έχει την τιμή `application/vnd.ntua.bookstore.order+xml`. Το αποτέλεσμα του HTTP αιτήματος αυτού είναι ένας 201 Created κωδικός παράλληλα με την αναπαράσταση του πόρου που έχει μόλις δημιουργηθεί και αναγνωρίζεται από το URI `/bookstore/orders/1/`, η οποία περιέχει επίσης συνδέσμους στον πόρο της συλλογής αντικειμένων της παραγγελίας (`items`) και στον πόρο `order status`. Ομοίως, για την προσθήκη ενός αντικειμένου στην παραγγελία που έχει δημιουργηθεί, αποστέλλεται ένα `POST` αίτημα στον πόρο `order items` ο οποίος αναγνωρίζεται από το URI `/bookstore/orders/1/items/` και για να ενημερωθεί η φάση-κατάσταση στην οποία βρίσκεται η παραγγελία που έχει δημιουργηθεί σε *submitted* αποστέλλεται ένα αίτημα PUT στον πόρο `order status` ο οποίος αναγνωρίζεται από το URI `/bookstore/orders/1/status`.

Όπως μπορεί να συναχθεί από το παράδειγμα, ο προσανατολισμός σε πόρους των REST API σχετίζεται με την ύπαρξη ενός *μοντέλου πόρων* (*resource model*), το οποίο καταγράφει, οργανώνει και εκθέτει πόρους έτσι ώστε να προσφερθούν οι δυνατότητες της υπηρεσίας, δηλαδή η λειτουργικότητα και οι πληροφορίες. Ένα

μοντέλο πόρων προσδιορίζει μια συλλογή πόρων μαζί με σχέσεις μεταξύ των πόρων, οι οποίες εκφράζουν είτε δομικά εγγενείς, υπαρξιακές σχέσεις, είτε την ίδια τη λογική της εφαρμογής. Οι υπαρξιακές σχέσεις μεταξύ των πόρων προσδιορίζουν μια ιεραρχία πόρων, ενώ οι σχέσεις που σχετίζονται με την εφαρμογή ενδέχεται να γίνουν ορατές (ως προς τον καταναλωτή της υπηρεσίας) κατά τον χρόνο εκτέλεσης, ανάλογα με τις καταστάσεις των πόρων (αναφερόμενες συνολικά ως *κατάσταση πόρων*). Χρησιμοποιώντας τις πληροφορίες που καταγράφουν τα μοντέλα πόρων μπορούν να παραχθούν αντίστοιχα αναγνωριστικά πόρων μιας υπηρεσιοστρεφούς υπηρεσίας, τα οποία παρέχουν τον νοητικό στόχο για τον ορισμό ενός σημείου αλληλεπίδρασης πελάτη-εξυπηρετητή, δηλαδή, ποιο είναι το *νόημα* της κάθε αλληλεπίδρασης. Κατά συνέπεια, στο REST, τα αναγνωριστικά πόρων θεωρούνται ως *απεριόριστα*, επιτρέποντας τους αρχιτέκτονες να προσδιορίζουν ελεύθερα οποιαδήποτε πληροφοριακή αφαίρεση απαιτηθεί και η οποία χρησιμοποιείται για να αναλύσει και αποσυνθέσει τη λειτουργικότητα με τρόπο προσανατολισμένο σε πόρους. Στην πράξη, είναι σύνηθες σε ό,τι αφορά στην αναγνώριση πόρων, αυτή να υλοποιείται μέσω ιεραρχικών HTTP URIs, κυρίως εξαιτίας της βελτιωμένης αναγνωσιμότητας τους από τους ανθρώπους. Παρ' όλα αυτά, αδιαφανή (opaque) URIs μπορούν επίσης να χρησιμοποιηθούν καθώς το REST δεν απαιτεί κάποιο συγκεκριμένο δομικό μοτίβο. Επιπλέον, συνήθως οι πόροι είναι προσανατολισμένοι σε δεδομένα και τείνουν να φέρουν τη σημασιολογία πλούσιων σε πληροφορία οντοτήτων, οι οποίες εκτίθενται και γίνονται προσβάσιμες μέσω συγκεκριμένων προβολών-οπτικών. Για παράδειγμα, οι πόροι `/orders/123/` και `/orders/?mostRecent` αποτελούν δύο σημασιολογικά διακριτούς πόρους, ο πρώτος αναφέρεται στην παραγγελία που αναγνωρίζεται από τον αριθμό `123` και η δεύτερη αναφέρεται στην πιο πρόσφατη παραγγελία που έχει καταχωρηθεί στο σύστημα. Ωστόσο, κάποια χρονική στιγμή και για κάποιο χρονικό διάστημα ενδέχεται να αντιστοιχούν στην ίδια υποκείμενη πληροφοριακή οντότητα.

Πέρα από τον ορισμό του μοντέλου πόρων, οι αρχιτέκτονες λογισμικού πρέπει να καθορίσουν (είτε αυτό σημαίνει να επιλέξουν, ή να σχεδιάσουν) κατάλληλες αναπαραστάσεις πόρων έτσι ώστε να σχεδιάσουν και να υλοποιήσουν RESTful υπηρεσίες, παρέχοντας πιθανόν περισσότερες από μια τέτοιες αναπαραστάσεις για συγκεκριμένες αλληλεπιδράσεις. Οι εναλλακτικές αναπαραστάσεις σε συνδυασμό με μηχανισμούς διαπραγματεύσεις περιεχομένου *content negotiation* μπορούν να προσφέρουν σημαντικά οφέλη στις υπηρεσίες τόσο σε ιδιότητες του κύκλου ζωής τους (π.χ. επεκτασιμότητα) όσο και κατά τον χρόνο εκτέλεσης (προσαρμογή του

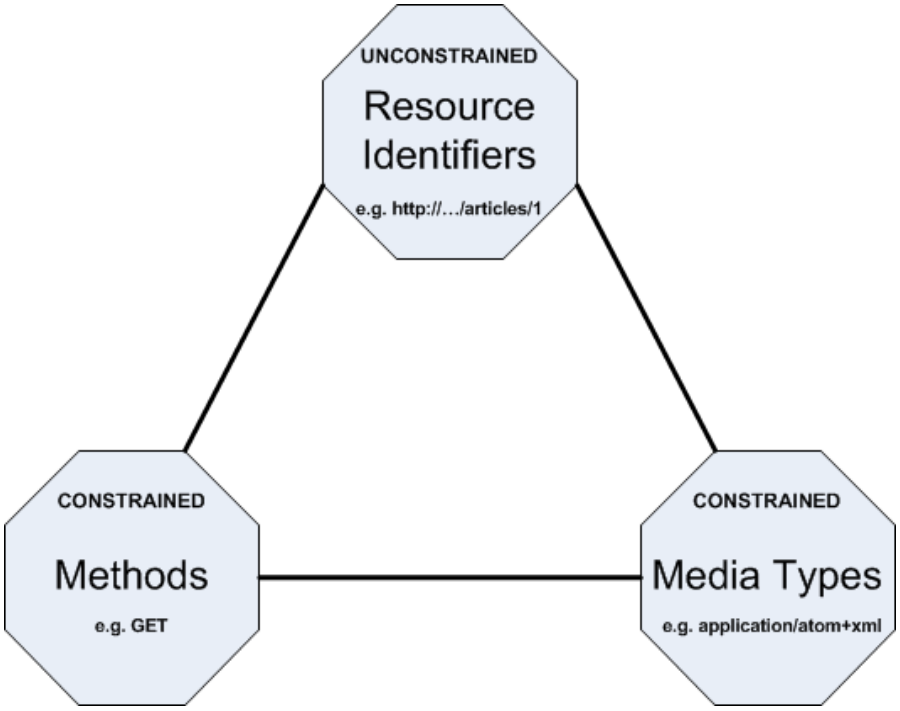
τύπου μέσων που αντλείται ανάλογα με το περιβάλλον, τις συνθήκες και τις δυνατότητες του καταναλωτή). Οι δομές και η σημασιολογία των αναπαραστάσεων προέρχεται από τους ορισμούς των τύπων μέσων, οι οποίοι μπορούν να είναι είτε προτυποποιημένοι σε παγκόσμιο επίπεδο (δηλαδή να περιλαμβάνονται στο αποθετήριο τύπων πόρων του IANA), ή κατά παραγγελία (δηλαδή να είναι ορισμένοι στο πλαίσιο της συγκεκριμένης εφαρμογής που χρησιμοποιούνται ή κάποιου ανάλογου, περιορισμένου εύρους, πεδίου εφαρμογών). Αν και οι κατά παραγγελία τύποι μέσων δεν είναι προτυποποιημένοι κάτι που ενέχει τον κίνδυνο βλάβης της ομοιομορφίας που απαιτεί ο περιορισμός Ομοιόμορφης διεπαφής, η χρήση τους θεωρείται αποδεκτή στο REST[2], με την προϋπόθεση ότι στο εύρος των οικοσυστημάτων στα οποία χρησιμοποιούνται τέτοιοι τύποι μέσων, αυτοί θα πρέπει να αντιμετωπίζονται ως πρότυπα, ή με άλλα λόγια, να θεωρούνται ως κοινή γνώση μεταξύ όλων των μερών της προκύπτουσας αρχιτεκτονικής. Υπό αυτή την έννοια, οι τύποι μέσων θεωρούνται *περιορισμένοι* σε σχέση με το εύρος μιας REST αρχιτεκτονικής. Επιπλέον, πέρα από τη μεταβίβαση πληροφοριών που σχετίζονται με την κατάσταση (των πόρων), οι τύποι μέσων αποτελούν μέσα προδιαγραφής των *προσφερόμενων δυνατοτήτων* (*affordances*) στα μηνύματα αποκρίσεων, οι οποίες αντιμετωπίζονται κατά τον χρόνο εκτέλεσης ως στοιχεία υπερμέσων.

Επιπρόσθετα, ο αρχιτέκτονας ενός RESTful Web service πρέπει να προσδιορίσει το υποσύνολο των HTTP μεθόδων το οποίο θα χρησιμοποιηθεί κατά μήκος του πεδίου πόρων που έχει οριστεί και πιο συγκεκριμένα να προσδιορίσει ποιες μέθοδοι θα επιτρέπονται και θα υλοποιηθούν για κάθε πόρο που έχει αναγνωριστεί. Οι HTTP μέθοδοι θα πρέπει να χρησιμοποιούνται ώστε να μεταβιβάζουν την πρόθεση της αλληλεπίδρασης με έναν πόρο σε συμφωνία με τη σημασιολογία που έχει προσδιοριστεί για κάθε μέθοδο στην προδιαγραφή του HTTP. Υπό αυτή την άποψη, το γεγονός ότι υπάρχει ένα σταθερό σύνολο μεθόδων οι οποίες μπορούν να χρησιμοποιηθούν, μαζί με το γεγονός ότι η σημασιολογία τους πρέπει να είναι καθολική για όλους του πόρους, καθιστά και αυτή τη διάσταση σχεδίασης στο REST επίσης *περιορισμένη*.

Το *τρίγωνο του REST*[3], το οποίο απεικονίζεται στο Σχήμα 2.3, έχει προταθεί ως ένα μοντέλο που αποτυπώνει τα κύρια μελήματα κατά τη σχεδίαση και την υλοποίηση RESTful υπηρεσιών Ιστού, τα οποία είναι τα *αναγνωριστικά πόρων* (*resource*

[2]https://groups.yahoo.com/neo/groups/rest-discuss/conversations/messages/17189
[3]REST Rewiring, http://soundadvice.id.au/blog/2008/04/18/

identifiers), οι *τύποι μέσων* (*media types*) και οι *μέθοδοι* (*methods*).



Σχήμα 2.3: Το *τρίγωνο του REST* (*REST triangle*)

Τελικά, σχετικά με τα συμβόλαια υπηρεσιών και τους επεξεργάσιμους από μηχανές ορισμούς διεπαφών, στο REST, η *κανονική* προσέγγιση, ή για ορισμένους, η καθαρολογική προσέγγιση απορρίπτει τον προσδιορισμό οποιασδήποτε άλλης πληροφορίας πέρα από την εξειδίκευση των χαρακτηριστικών της ομοιόμορφης διεπαφή που ορίζεται στο εύρος μιας αρχιτεκτονικής. Αυτά τα χαρακτηριστικά περιλαμβάνουν ένα σύνολο από καθολικές πράξεις, δεδομένα ελέγχου και μηχανισμούς διαπραγμάτευσης που υποστηρίζονται (κάτι που τυπικά υλοποιείται επιλέγοντας κάποιο πρωτόκολλο επικοινωνίας, το οποίο στα RESTful Web services είναι το HTTP), το σύνολο των τύπων μέσων οι οποίοι μπορεί να χρησιμοποιηθούν κατά τον χρόνο εκτέλεσης, το σύνολο των σχέσεων σύνδεσης ή άλλων μέσων για την παροχή ελέγχων υπερμέσων, και πιθανώς ένα ή περισσότερα αναγνωριστικά πόρων εισόδου, έτσι ώστε η αλληλεπίδραση πελάτη-εξυπηρετητή να μπορεί να ξεκινήσει. Μια τέτοια προσέγγιση επιχειρεί να μειώσει τις απαραίτητες πληροφορίες στο ελάχιστο και να αποτρέψει με αυτόν τον τρόπο ανεπιθύμητη και χωρίς ουσιαστικό λόγο ύπαρξης σύζευξη. Τέτοιου είδους σύζευξη θεωρείται ότι εισάγεται μέσω των ρητών ορισμών των υπηρεσιών οι οποίοι υπαγορεύουν εκ των προτέρων τα σημεία αλληλεπίδρα-

σης, αντί να τα αποκαλύπτουν κατά τον χρόνο εκτέλεσης. Ωστόσο, στην πράξη, κατά τη διάρκεια των τελευταίων λίγων ετών, ένας αυξανόμενος αριθμός γλωσσών και μοντέλων ορισμού REST διεπαφών έχει κάνει την εμφάνισή του, οδηγούμενα κυρίως από την ανάγκη απλοποίησης και αυτοματοποίησης ορισμένων δραστηριοτήτων της τεχνολογίας λογισμικού, όπως ο έλεγχος των API, η ταχεία παραγωγή σκελετών πελατών υπηρεσιών για τη διευκόλυνση της ανάπτυξης εφαρμογών που καταναλώνουν τις υπηρεσίες, την παραγωγή τεκμηρίωσης των υπηρεσιών κλπ. Στο Κεφάλαιο 3, Ενότητα 3.3.2 συζητείται περαιτέρω το πως ο περιορισμός της Ομοιόμορφης διεπαφής του REST μπορεί να αναλυθεί μέσω σχεδιαστικών κριτηρίων, τα οποία μπορούν τελικά να χρησιμοποιηθούν για να καθοδηγήσουν τη χρήση των αναγνωριστικών πόρων, των τύπων μέσων, των υπερμέσων και των λοιπών στοιχείων κατά το σχεδιασμό RESTful διεπαφών.

2.4 Προσαρμογή υπηρεσιών

Η προσαρμογή λογισμικού έχει αναδυθεί ως επιστημονική περιοχή εδώ και λίγα έτη [29] [32]. Παρ' όλα αυτά, το πρόβλημα προσαρμογής υφιστάμενων δομοστοιχείων λογισμικού ώστε να ανταπεξέρχονται σε εξελισσόμενες απαιτήσεις ή να αναβαθμίζουν τις δυνατότητες επαναχρησιμοποίησής τους, αποτελεί περιοχή μακράς έρευνας και διαλόγου [68]. Η προσαρμογή λογισμικού είναι η επιστημονική περιοχή η οποία στοχεύει στη σύνθεση ειδικών δομοστοιχείων λογισμικού, τα οποία καλούνται *προσαρμογείς λογισμικού* (*software adapters*), για να αντιμετωπίσουν μια ποικιλία ζητημάτων και προκλήσεων που εγείρονται όταν απαιτείται να επαναχρησιμοποιηθούν ή να συνδυαστούν εκ νέου, υπάρχουσες μονάδες λογισμικού σε νέα περιβάλλοντα και πλαίσια.

Για παράδειγμα, η αυτο-προσαρμογή ή η προσαρμοστική εξέλιξη υπαρχόντων δομοστοιχείων, επιτρέπει στα δομοστοιχεία να ανταποκρίνονται σε απαιτήσεις ή να προσαρμόζονται ως προς μεταβαλλόμενα πλαίσια λειτουργίας με το να περικλείουν και να συνεργάζονται με κατάλληλους προσαρμογείς. Ομοίως, η προσαρμοστική ολοκλήρωση θεωρεί προσαρμογείς λογισμικού ως μεσολαβούντα δομοστοιχεία τα οποία αντιμετωπίζουν δομικές, συμπεριφορικές ή μη λειτουργικές ασυμβατότητες μεταξύ υπαρχόντων δομοστοιχείων που απαιτείται να επικοινωνούν. Οι προσεγγίσεις προσαρμογών ενδέχεται να περιλαμβάνουν αναδιαρθρώσεις ή τροποποιήσεις των υπαρχόντων δομοστοιχείων, ανάλογα με αν εφαρμόζουν διεισδυτικές ή μη

διεισδυτικές τεχνικές προσαρμογής.

Οι προσαρμογείς λογισμικού αντιμετωπίζουν ελλείψεις μέσω της εφαρμογής κατάλληλης λογικής προσαρμογής. Αυτή η λογική προσαρμογής καθορίζεται συχνά ιδιοχείρως από μηχανικούς. Ωστόσο, ένας αυξανόμενος αριθμός συνεισφορών στην περιοχή της προσαρμογής λογισμικού, προωθεί την αναγνώριση ή την εξαγωγή της απαραίτητης λογικής προσαρμογής με αυτόματο ή ημι-αυτόματο τρόπο, μέσω της ανάλυσης επεξεργάσιμων από μηχανές περιγραφών δομοστοιχείων και στοιχείων υλοποίησης. Κατά κανόνα, το επίπεδο αυτοματοποίησης κατά την άντληση κατάλληλης λογικής προσαρμογής σε ένα σενάριο προσαρμογής, είναι ανάλογο με το επίπεδο στο οποίο τα δομοστοιχεία αλλά και οι στόχοι του σεναρίου προσαρμογής προσδιορίζονται τυπικά, όπως και του επιπέδου ωριμότητας και εκζήτησης των διαδικασιών ανάλυσης και εξαγωγής που χρησιμοποιούνται.

Εν ολίγοις, μια διαδικασία προσαρμογής μπορεί να χαρακτηριστεί από τις ακόλουθες παραμέτρους: *α*) κατά πόσο απαιτείται τροποποίηση υπαρχόντων δομοστοιχείων για την επίτευξη των στόχων της προσαρμογής (διεισδυτική, μη διεισδυτική), *β*) ο τρόπος με τον οποίο η λογική προσαρμογής αντλείται και εφαρμόζεται (χειροκίνητα, ημι-αυτόματα, αυτόματα), *γ*) ποιο είναι το εύρος του πεδίου της προσαρμογής (λειτουργικό, μη λειτουργικό). .

Στην παρούσα διατριβή προτείνουμε μια διαδικασία προσαρμογής λογισμικού, μαζί με ένα αντίστοιχο περιβάλλον-πλαίσιο και πρωτότυπο, έτσι ώστε να αντιμετωπίσουμε την πρόκληση P2R προσαρμογής διεπαφών υπηρεσιών. Η προτεινόμενη προσέγγιση είναι μη διεισδυτική, καθώς οι υλοποιήσεις των υφιστάμενων δομοστοιχείων και διεπαφών δεν τροποποιούνται, η λογική προσαρμογής αντλείται με ημι-αυτόματο τρόπο, περιορίζοντας τη συμμετοχή του χρήστη σε δηλωτικά δεδομένα εισόδου ή βελτιωτική ανάδραση και το εύρος του πεδίου προσαρμογής περιορίζεται σε λειτουργικές αναντιστοιχίες. Η προσέγγιση προσαρμογής που προτείνεται μπορεί να θεωρηθεί είτε ως προσέγγιση αυτο-προσαρμογής, είτε ως προσέγγιση προσαρμογής ολοκλήρωσης, ανάλογα με την οπτική που την εξετάσει κανείς. Όταν εξετάζεται από την οπτική του παρόχου υπηρεσιών, η P2R προσαρμογή αποτελεί τύπο προσαρμοστικής εξέλιξης στην οποία τα δομοστοιχεία παραμένουν ανέπαφα σε σχέση με τις προ της προσαρμογής δυνατότητές τους (δηλαδή, είναι σε θέση να εξυπηρετήσουν όλους του υπάρχοντες πελάτες τους) και αποκτούν τη δυνατότητα να παρέχουν ΠΣΠ εκθέσεις της λειτουργικότητας που υλοποιούν μέσω συνεργασίας τους με κατάλληλους προσαρμογείς. Αντίθετα, όταν η P2R προσαρμογή εξετάζεται

ως πρόκληση ολοκλήρωσης μιας ΠΣΠ εφαρμογής πελάτη με ένα ΠΣΔ εξυπηρετητή, η προτεινόμενη προσέγγιση προσαρμογής μπορεί να θεωρηθεί ως μια προσέγγιση προσαρμοστικής ολοκλήρωσης, καθώς επιτρέπει την ολοκλήρωση των μη συμβατών δομοστοιχείων.

2.4.1 Προσεγγίσεις στην προσαρμογή υπηρεσιών

Η υπηρεσιοστρεφής υπολογιστική αποτελεί πεδίο στο οποίο έχουν εφαρμοστεί εκτεταμένα προσεγγίσεις προσαρμογής λογισμικού, καθώς η ανάγκη για ολοκλήρωση ανεξάρτητα ανεπτυγμένων και συντηρημένων κατανεμημένων δομοστοιχείων είναι κάτι ιδιαίτερα σύνηθες στις υπηρεσιοστρεφείς αρχιτεκτονικές [18]. Στην υπηρεσιοστρεφή αρχιτεκτονική, τα δομοστοιχεία υπηρεσιών αλληλεπιδρούν μέσω μηνυμάτων τα οποία προσδιορίζονται από αντίστοιχες διεπαφές και μοντέλα αλληλεπίδρασης. Υπό αυτή την άποψη, καθώς ο προσανατολισμός σε υπηρεσίες αναπτύσσονταν, προτάθηκαν και χρησιμοποιήθηκαν αρκετά περιβάλλοντα πλαίσια και τεχνικές προσαρμογής λογισμικού για την ολοκλήρωση συστημάτων λογισμικού με μερικώς συμβατές διεπαφές ή μοντέλα αλληλεπίδρασης. Τέτοιου είδους προσεγγίσεις προσαρμογής εντάσσονται στην περιοχή της *προσαρμογής υπηρεσιών* (*service adaptation*) [56], [101], [121], [20]. Όταν η προσαρμογή υπηρεσιών περιορίζεται στο επίπεδο διεπαφής, δηλαδή όταν δεν υπάρχει πρόσβαση και δεν πραγματοποιούνται αναδιαρθρώσεις στις υλοποιήσεις των υπηρεσιών, αναφέρεται ως *προσαρμογή διεπαφών υπηρεσιών* (*service interface adaptation*) [42], [100]. Οι αναντιστοιχίες των διεπαφών μπορεί να σχετίζονται με διαφορές στην ονοματολογία των σημείων αλληλεπίδρασης ή των δομών των μηνυμάτων, στο πλήθος ή την έγκυρη ροή των αλληλεπιδράσεων ώστε να επιτευχθεί ένας στόχος (π.χ. συμπεριφορικές αναντιστοιχίες / αναντιστοιχίες πρωτοκόλλου), στις μη λειτουργικές ιδιότητες της επικοινωνίας ή ακόμα και στη σημασιολογία των στοιχείων των διεπαφών.

Η προσαρμογή διεπαφών υπηρεσιών έχει εξεταστεί ως προς *δομικές* διαστάσεις των στοιχείων δεδομένων των διεπαφών (π.χ. σχήματα μηνυμάτων και υπογραφές) όπως και ως προς *συμπεριφορικές* διαστάσεις των μοντέλων αλληλεπίδρασης των διεπαφών (π.χ. ενδο-υπηρεσιακά επιχειρηματικά πρωτόκολλα). Επιπλέον, πρόσφατα έχει αναδυθεί μια τρίτη οπτική σε σχέση με την προσαρμογή διεπαφών η οποία εστιάζει σε *αρχιτεκτονικές* διαστάσεις της σχεδίασης των διεπαφών, δηλαδή στην ανάλυση της λειτουργικότητας μιας υπηρεσίας και στο υπόδειγμα που ακο-

λουθεί η υπηρεσία σε σχέση με την έκθεση της λειτουργικότητας αυτής. Δεδομένων των παραπάνω, αναφερόμαστε στην ειδική περίπτωση αρχιτεκτονικής προσαρμογής μεταξύ διαφορετικών σχεδιαστικών υποδειγμάτων διεπαφών ως *προσαρμογή υποδειγμάτων* (*paradigm adaptation*).

Η δομική διάσταση της προσαρμογής (structural adaptation) υπηρεσιών σχετίζεται με ασυμβατότητες μοντελοποίησης των διεπαφών που αφορούν σε υπογραφές λειτουργιών και σχήματα και δομές μηνυμάτων. Για παράδειγμα, ο τύπος των παραμέτρων των λειτουργιών, η κωδικοποίηση των τιμών τους ή η σημασιολογία των στοιχείων των παραμέτρων μπορούν να αποτελέσουν αντικείμενα δομικής προσαρμογής. Αν και γλώσσες όπως η XSLT[4] και η XQuery[5] έχουν προταθεί έτσι ώστε να διευκολύνουν τον προσδιορισμό των μετασχηματισμών των δεδομένων και των αντιστοιχίσεων μεταξύ των στοιχείων των σχημάτων, η άντληση της λογικής δομικής προσαρμογής στηρίζεται σε τεχνικές οι οποίες αναγνωρίζουν σχέσεις μεταξύ στοιχείων διαφορετικών σχημάτων. Υπό αυτή την άποψη, οι προκλήσεις που σχετίζονται με δομική προσαρμογή υπηρεσιών σχετίζονται με περιοχές όπως αντιστοίχιση και ταίριασμα σχημάτων/οντολογιών [19], οι οποίες αποτελούν γενικά δύσκολα προβλήματα σε ό,τι αφορά στην αυτοματοποιημένη αντιμετώπισή τους, ειδικά όταν τα προβλήματα δε μπορούν να περιοριστούν ή να αναλυθούν σε μικρότερα. Παρ' όλα αυτά, στο πλαίσιο ορισμών διεπαφών υπηρεσιών, το πρόβλημα του ταιριάσματος μπορεί να αναλυθεί σε μικρότερα χρησιμοποιώντας τις πληροφορίες οργάνωσης-επιμερισμού της λειτουργικότητας τις οποίες παρέχουν οι ορισμοί αυτοί, έτσι ώστε να συγκρίνονται μικρότερα τμήματα σχημάτων [100]. Επίσης, η δομική προσαρμογή μπορεί να διευκολυνθεί μέσω σημασιολογικών σχολιασμών των στοιχείων των διεπαφών [104].

Οι προσεγγίσεις συμπεριφορικής προσαρμογής (behavioral adaptation) υπηρεσιών επιχειρούν να αντιμετωπίσουν ασυμβατότητες σε επίπεδο πρωτοκόλλου μεταξύ παρόχων και καταναλωτών υπηρεσιών, έτσι ώστε να δίνεται η δυνατότητα για ορθές αλληλεπιδράσεις μεταξύ ασύμβατων δομοστοιχείων, μέσω κατάλληλης λογικής διαμεσολάβησης. Υπό αυτή την άποψη, οι ασυμβατότητες μεταξύ ανεξάρτητα ανεπτυγμένων και συντηρημένων υπηρεσιών μπορεί να αντιμετωπιστεί με μη διεισδυτικό τρόπο. Στη βιβλιογραφία έχει προταθεί μια πλειάδα προσεγγίσεων συμπεριφορικής προσαρμογής, οι οποίες αντιμετωπίζουν προκλήσεις που σχετίζονται με την

[4]http://www.w3.org/TR/xslt
[5]http://www.w3.org/TR/xquery/

προδιαγραφή, σύνθεση και χρησιμοποίηση προσαρμογέων [56], [42], [30], [41], [101], [20], [91], [100]. Για τις περισσότερες από αυτές τις προσεγγίσεις συμπεριφορικής προσαρμογής, η δομική προσαρμογή αποτελεί προαπαιτούμενο. Κατά συνέπεια, οι προσεγγίσεις είτε ενσωματώνουν τεχνικές δοκιμής προσαρμογής, είτε χρησιμοποιούν αφαιρετική λογική υποθέτοντας ότι πιθανά ζητήματα προσαρμογής δομικής φύσης έχουν ήδη αντιμετωπιστεί.

Η προσαρμογή σχεδιαστικού υποδείγματος (paradigm adaptation) εξετάζει το πρόβλημα από ένα διαφορετικό επίπεδο αφαίρεσης, καθώς στοχεύει σε ασυμβατότητες των διεπαφών που αφορούν στην ανάλυση της λειτουργικότητας σε συνθετικά μέρη και την έκθεσή της. Υπό αυτή την άποψη, τα δομοστοιχεία υπηρεσιών που ακολουθούν διαφορετικά υποδείγματα διεπαφών πρέπει πρώτα να ευθυγραμμιστούν ως προς το τρόπο με τον οποίο οι διεπαφές τους εκθέτουν ή αναμένουν τη λειτουργικότητα. Τέτοιου είδους ασυμβατότητες σχετίζονται με την αρχιτεκτονική μοντελοποίηση των διεπαφών, δηλαδή, το εννοιολογικό μοντέλο των αλληλεπιδράσεων και τις αφαιρέσεις πληροφοριών που χρησιμοποιούνται για να εκφράσουν δυνατότητες υπηρεσιών. Όντας μια πιο ευρεία πρόκληση, τεχνικές τόσο από την περιοχή της δομικής προσαρμογής, όσο και από την περιοχή της συμπεριφορικής προσαρμογής υπηρεσιών, ενδέχεται να επαναχρησιμοποιούνται στο πλαίσιο κάποιου περιβάλλοντος-πλαισίου προσαρμογής σχεδιαστικού υποδείγματος διεπαφών υπηρεσιών. Η προσέγγιση προσαρμογής που εξετάζεται και προτείνεται στα Κεφάλαια 3 και 4, εμπίπτει σε αυτή την τελευταία κατηγορία προσαρμογής υπηρεσιών. Στις επόμενες ενότητες εξετάζουμε στενότερα την προσαρμογή υποδείγματος όπως εξετάζεται στην παρούσα διατριβή και παρέχουμε μια περίληψη ανάλογων προσεγγίσεων προσαρμογής που έχουν προταθεί στη βιβλιογραφία.

2.4.2 Προσαρμογή υποδειγμάτων διεπαφών

Οι τεχνικές προσαρμογής υπηρεσιών, τόσο δομικής όσο και συμπεριφορικής προσαρμογής, ορίζονται και εφαρμόζονται τυπικά στα πλαίσια υπηρεσιών ΠΣΔ, υποθέτοντας, έμμεσα ή άμεσα, ότι η λειτουργικότητα μοντελοποιείται και εκτίθεται μέσω λειτουργιών υπηρεσιών και στα δύο δομοστοιχεία που εξετάζονται κάθε φορά. Κατά συνέπεια, οι προσαρμογείς που παράγονται είναι σε θέση να ολοκληρώσουν υπηρεσίες ΠΣΔ αλλά ενδέχεται να υπολείπονται σε ό,τι αφορά στην αντιμετώπιση αρχιτεκτονικών και σχεδιαστικών ασυμβατοτήτων μοντελοποίησης των στοιχείων

των διεπαφών.

Κατά τη διάρκεια των τελευταίων χρόνων, η προσαρμογή υποδειγμάτων διεπαφών αναδύθηκε ως ένα νέο είδος προσαρμογής υπηρεσιών, καθώς εστιάζει σε μεθόδους και μηχανισμού προσαρμογής του τρόπου με τον οποίο αναλύεται και εκτίθεται η λειτουργικότητα από το σχεδιαστικό υπόδειγμα διεπαφής που ακολουθεί μια υπηρεσία, σε ένα δεύτερο υπόδειγμα διεπαφής, έτσι ώστε οι δυνατότητες της υπηρεσίας να γίνουν προσπελάσιμες μέσω του δεύτερου υποδείγματος. Συγκεκριμένα, καθώς ο προσανατολισμός σε διαδικασίες και ο προσανατολισμός σε πόρους αποτελούν τα δύο κύρια υποδείγματα διεπαφών υπηρεσιών στην υπηρεσιοστρεφή υπολογιστική, οι προσεγγίσεις προσαρμογής υποδείγματος εστιάζουν στην προσαρμογή ΠΣΔ διεπαφών σε ΠΣΠ διεπαφές και αντίστροφα. Υπό αυτή την έννοια, οι δύο κατευθύνσεις προσαρμογής που μπορούν να αναγνωριστούν είναι οι εξής: η ΠΣΔ-προς-ΠΣΠ (P2R) κατεύθυνση και η ΠΣΠ-προς-ΠΣΔ (R2P) κατεύθυνση. Για παράδειγμα, στο P2R σενάριο, ένα SOAP Web Service καταναλώνεται από μια mash-up, Web 2.0 εφαρμογή μέσω μιας HTTP διεπαφής προσανατολισμένης σε πόρους, κατά το οποίο σενάριο, ένας προσαρμογέας υποδείγματος μεταφράζει τις RESTful αλληλεπιδράσεις σε κλήσεις λειτουργικών και παροχή αποτελεσμάτων. Ομοίως, ένα REST API μπορεί να παρέχει τη λειτουργικότητά του σε μια back-end επιχειρηματική εφαρμογή μέσω ενός R2P προσαρμογέα υποδείγματος ο οποίος αντιστοιχίζει και μετατρέπει κλήσεις λειτουργιών και απαντητικά μηνύματα σε αλληλεπιδράσεις πρόσβασης και μεταχείρισης πόρων.

Στην παρούσα διατριβή, διερευνούμε την κατεύθυνση της P2R προσαρμογής υποδείγματος και συγκεκριμένα το πρόβλημα *εξαγωγής διεπαφής* (*interface extraction*), στο οποίο ο στόχος της προσαρμογής είναι η αναγνώριση κατάλληλης λογικής προσαρμογής, έτσι ώστε να παρασχεθεί μια διεπαφή προσανατολισμένη σε πόρους μέσω ενός ανάλογου προσαρμογέα χρόνου εκτέλεσης ο οποίος προσαρτάται σε ένα υφιστάμενο δομοστοιχείο ΠΣΔ υπηρεσίας. Συγκεκριμένα, στο Κεφάλαιο 3 συζητάμε ένα σύνολο σημαντικών ζητημάτων σχετικά με την εξαγωγή διεπαφής και την προσαρμογή υποδείγματος κατά τον χρόνο εκτέλεσης. Επίσης, παρουσιάζουμε ένα αφαιρετικό μοντέλο διαδικασίας προσαρμογής το οποίο αναλύει τη διαδικασία προσαρμογής σε βήματα απλούστερων εργασιών και επίσης ορίζει και εξετάζει ένα σύνολο τύπων στοιχείων/τεχνουργημάτων (artifacts) τα οποία μοντελοποιούν συνιστώσες μιας διαδικασίας προσαρμογής. Στο Κεφάλαιο 4 παρουσιάζεται ένα περιβάλλον-πλαίσιο προσαρμογής και ένα αντίστοιχο πρωτότυπο P2R εξαγωγής

διεπαφής. Πιο συγκεκριμένα, το περιβάλλον-πλαίσιο προσαρμογής που προτείνεται αναλύει σε βάθος ορισμούς διεπαφών ΠΣΔ υπηρεσιών, έτσι ώστε να εξάγει στοιχείων ΠΣΠ διεπαφών και να αναγνωρίσει αντιστοιχίες μεταξύ των στοιχείων αυτών και στοιχείων της υποκείμενης ΠΣΔ διεπαφής. Η λογική προσαρμογής που εξάγεται καταγράφεται ως μια αναλυτική προδιαγραφή την οποία καλούμε *οδηγία προσαρμογής* (*adaptation prescription*), η οποία εκτελείται τελικά από έναν κατάλληλο προσαρμογέα χρόνου εκτέλεσης.

2.4.3 Εργασίες προσαρμογής ΠΣΔ υπηρεσιών και REST API

Κατά τη διάρκεια των τελευταίων ετών, το ερευνητικό πρόβλημα προσαρμογής υποδείγματος έχει συγκεντρώσει σημαντική προσοχή τόσο από την ακαδημαϊκή κοινότητα όσο και από τη βιομηχανία λογισμικού και στο πλαίσιο αυτό, έχει προταθεί ένα σύνολο προσεγγίσεων προσαρμογής υποδείγματος (κυρίως P2R) [6], [69], [75], [76] [80], [146], [142]. Στις επόμενες παραγράφους εξετάζουμε ορισμένες από τις σημαντικότερες προσεγγίσεις της περιοχής.

Στην εργασία [75] εισάγεται μια προσέγγιση βασισμένη σε UML μοντέλα, για την αφαίρεση και μοντελοποίηση υπαρχόντων API σε ένα κανονικοποιημένο μοντέλο διεπαφής το οποίο μπορεί να χρησιμοποιηθεί για να την έκθεση REST-like πόρων. Η μέθοδος στην [75] απαιτεί ένα σύνολο UML μοντέλων το οποίο περιγράφει λεπτομερώς τόσο τις δομικές όσο και τις συμπεριφορικές διαστάσεις της διεπαφής. Τα UML μοντέλα που απαιτούνται τοποθετούνται πέρα από τις τυπικές περιγραφές διεπαφών που βασίζονται σε γλώσσες προδιαγραφής διεπαφών και δεν είναι συνήθως διαθέσιμα στην πράξη, ενώ απαιτείται συνήθως σημαντική προσπάθεια για την κατασκευή τους. Η μεταγενέστερη εργασία [76], περιγράφει μια διαδικασία οδηγούμενη από μοντέλα (model-driven) για τον σταδιακό μετασχηματισμό ΠΣΔ μοντέλων προδιαγραφής (π.χ. ένα ακολουθιακό διάγραμμα δομοστοιχείων στο υψηλότερο επίπεδο) σε ΠΣΠ διεπαφές. Παρ' όλα αυτά, η προσέγγιση υποθέτει σημαντική εμπλοκή του χρήστη στη διαδικασία, καθώς πρέπει να μεταφράσει χειροκίνητα τις ΠΣΔ προδιαγραφές σε ένα πληροφοριακό μοντέλο το οποίο χρησιμοποιείται στη συνέχεια για την παραγωγή του μοντέλου πόρων. Η προσέγγιση στην παρούσα διατριβή επιχειρεί να αυτοματοποιήσει τη διαδικασία εξαγωγής πληροφορίας (information extraction) που περιλαμβάνεται στην πρόκληση εξαγωγής πόρων, περιορίζοντας σημαντικά την εμπλοκή και συμμετοχή του χρήστη στη δια-

δικασία.

Μια άλλη προσέγγιση παρουσιάζεται στην εργασία [80], όπου οι συγγραφείς προτείνουν μια διαδικασία re-engineering υφιστάμενων συστημάτων στο REST. Η προσέγγιση ξεκινά με την ανάλυση του πηγαίου κώδικα του συστήματος. Επίσης, απαιτεί την ανάλυση άλλων στοιχείων όπως μοντέλων ER, διαγραμμάτων κλάσεων αλλά και μοντέλων απαιτήσεων και τεκμηρίωσης. Αντίθετα, η προσέγγιση στην παρούσα διατριβή είναι ουδέτερη και διαφανής ως προς την υλοποίηση των υπηρεσιών καθώς δεν υποθέτει τη διαθεσιμότητα ή τη δυνατότητα πρόσβασης στα στοιχεία της υλοποίησης.

Μια ακόμη προσέγγιση για τη μετάπτωση υπηρεσιών που στηρίζονται στο SOAP σε υπηρεσίες που βασίζονται στο REST συζητείται στην εργασία [146]. Σχετικά με την αναγνώριση πόρων, η προσέγγιση στη [146] διαμερίζει το σύνολο των υπογραφών λειτουργιών σε συστάδες, όπου κάθε συστάδα υποδεικνύει έναν πόρο. Οι λέξεις που περιέχονται στις υπογραφές κάθε συστάδας επεξεργάζονται για να αναγνωριστούν επιπλέον πόροι και να προσδιοριστεί μια αλληλουχία των λέξεων αυτών, η οποία χρησιμοποιείται στη δόμηση αναγνωριστικών πόρων. Η προσέγγιση έχει ομοιότητες με την προσέγγιση που παρουσιάζεται στην παρούσα διατριβή σχετικά με τη χρήση μεθόδων επεξεργασίας φυσικής γλώσσας. Παρ' όλα αυτά, εστιάζει στον σχηματισμό συστάδων μέσω επίβλεψης (από άνθρωπο) και στην αναγνώριση πόρων και σχέσεων μεταξύ τους μέσω των δομών των υπογραφών και των σημασιολογικών σχέσεων μεταξύ των λέξεων, όπως αυτές ανακτώνται από μια οντολογία φυσικής γλώσσας (WordNet [99]). Αντίθετα, η προσέγγιση που προτείνεται στο Κεφάλαιο 4 εστιάζει στην αναγνώριση μοντέλων πόρων μέσω της ανάλυσης όρων και των σχέσεων μεταξύ αυτών των όρων που εμφανίζονται στις λειτουργίες των ΠΣΔ διεπαφών. Η ανάλυση που υλοποιείται στηρίζεται αφενός στη γραμματική χρήση των όρων, όπως εμφανίζεται στην ονοματολογία των στοιχείων των διεπαφών και αφετέρου σε επακόλουθες τεχνικές επεξεργασίας γράφων.

Επίσης, έχει προταθεί ένας αριθμός προσεγγίσεων οι οποίες στηρίζονται στη συμμετοχή του χρήστη σε wizard-like διαδικασίες εξαγωγής πληροφορίας [69], [154]. Η κύρια διαφορά των εργασιών αυτών από αυτή που προτείνεται στην παρούσα διατριβή έγκειται στο ότι οι πρώτες οδηγούνται από το χρήστη και απαιτούν δεδομένα εισόδου και αλληλεπίδραση με αυτόν σε κάθε βήμα.

Σε μια προηγούμενη προσέγγιση του δικής μας ομάδας [6], παρουσιάσαμε μια

τεχνική για την αναγνώριση πόρων χρησιμοποιώντας περιγραφές υφιστάμενων ΠΣΔ υπηρεσιών, στις οποίες εφαρμόζονταν τεχνικές δομικής ανάλυσης των υπογραφών των λειτουργιών των υπηρεσιών. Πιο συγκεκριμένα, στην εργασία [6] οι δομές μηνυμάτων εισόδου και εξόδου αναλύονταν έτσι ώστε να εξαχθούν πρότυπα αναγνωριστικών πόρων. Η ανάλυση αυτή απαιτούσε τη χρήση μεταδεδομένων τα οποία είχαμε υποθέσει ότι παρέχονται από τον χρήστη και αφορούσαν στην σημασιολογική κατάταξη των στοιχείων WSDL εγγράφων, π.χ. μια παράμετρος μπορούσε να χαρακτηριστεί ως Container (υποδοχέας). Η προσέγγιση που παρουσιάζεται στην παρούσα διατριβή διαφέρει από την προσέγγιση στη [6] ως προς τα εξής: *α*′) δεν υποθέτει την ύπαρξη και διαθεσιμότητα μεταδεδομένων όπως αυτών που περιγράφηκαν παραπάνω, *β*′) βασίζεται αποκλειστικά σε περιγραφές WSDL για την εξαγωγή μοντέλων πόρων και τέλος, *γ*′) εξετάζει υπογραφές λειτουργιών υπηρεσιών για την αναγνώριση πόρων πέρα από τα δομικά τους χαρακτηριστικά.

Στις εργασίες [71] και [137] εξετάζεται η πρόκληση της γεφύρωσης της σημασιολογικής απόστασης μεταξύ των RESTful υπηρεσιών και της ΠΣΔ προσέγγισης, προς την αντίθετη κατεύθυνση, δηλαδή την R2P. Συγκεκριμένα, το hRESTS που προτάθηκε στην εργασία [71] είναι μια προδιαγραφή microformat η οποία μπορεί να χρησιμοποιηθεί για την αναγνώριση RESTful σημείων αλληλεπίδρασης ως λειτουργίες με παραμέτρους εισόδου και εξόδου και παράλληλα η προδιαγραφή μπορεί να λάβει τη μορφή επισημειώσεων, ενσωματωμένων σε περιγραφές διεπαφών. Το SA-REST [137] επεκτείνει το hRESTS μέσω περαιτέρω επισημειώσεων οι οποίες επιτρέπουν τον ορισμό *όψεων* των υπηρεσιών (π.χ. υποστηριζόμενες δομές δεδομένων, συνδέσεις (bindings), κ.ά). Οι παραπάνω R2P προσεγγίσεις επιχειρούν να αναγνωρίσουν λειτουργίες μέσω αναλύσεων της λειτουργικότητας που παρέχεται μέσω πόρων. Ωστόσο, οι σχετικές συνεισφορές περιορίζονται σε λεξιλόγια μεταδεδομένων χωρίς να προτείνονται μεθοδολογίες ή τεχνικές εξαγωγής της απαραίτητης λογικής προσαρμογής.

Αν και κάθε P2R προσέγγιση που συζητήθηκε παραπάνω εστιάζει περισσότερο σε κάποια σημεία απ΄ ότι σε άλλα, γίνεται σαφές ότι σε ένα P2R σενάριο διαυποδειγματικής προσαρμογής πρέπει να αντιμετωπιστούν δύο μείζονα ζητήματα: *α*′) ο τρόπος αναγνώρισης πόρων και μοντέλων πόρων, *β*′) ο τρόπος συσχέτισης των στοιχείων των ΠΣΔ διεπαφών και των ΠΣΠ διεπαφών. Οι μέθοδοι και οι τεχνικές που προτείνονται ποικίλουν ως προς το επίπεδο αυτοματοποίησης και το επίπεδο συμμόρφωσης του αποτελέσματος στο REST. Για παράδειγμα, οι μέθοδοι αναγνώρισης

πόρων ποικίλουν, ξεκινώντας από το χειροκίνητο προσδιορισμό που στηρίζεται στη συμμετοχή ειδικών, ως τεχνικές αυτοματοποιημένης εξαγωγής πληροφορίας που στηρίζονται σε περιγραφές των διεπαφών που μπορούν να αναγνωστούν και να αναλυθούν μηχανιστικά.

2.5 Υποκαταστασιμότητα υπηρεσιών

Μια πρόκληση που σχετίζεται στενά με την προσαρμογή υπηρεσιών είναι η *υποκαταστασιμότητα υπηρεσιών*. Μια υπηρεσία μπορεί να υποκαταστήσει μια άλλη υπό την προϋπόθεση ότι είναι σε θέση να εξυπηρετήσει τουλάχιστον όλους τους *δυνατούς* καταναλωτές της υπηρεσίας αυτής. Υπό αυτή την έννοια, μια τέτοια υποκατάσταση θα ήταν διαφανής από την οπτική του πελάτη, εφόσον η απόφαση σχετικά με την τοποθεσία της προς κατανάλωση υπηρεσίας έχει επίσης αντιμετωπιστεί (π.χ. αποτελεί παράμετρο που μπορεί να ρυθμιστεί). Ομοίως με την προσαρμογή υπηρεσιών, η πρόκληση της υποκαταστασιμότητας υπηρεσιών έχει επίσης διερευνηθεί σημαντικά στο πλαίσιο των ΠΣΔ υπηρεσιών, κάνοντας έτσι αντίστοιχες υποθέσεις για τον τρόπο με τον οποίο η λειτουργικότητα των υπηρεσιών εκτίθεται. Αποτέλεσμα αυτής της διερεύνησης είναι να έχει προταθεί ένα σύνολο μεθοδολογιών για τη σύγκριση υπηρεσιών και διεπαφών υπηρεσιών στη σχετική βιβλιογραφία. Οι προσεγγίσεις αυτές απαιτούν συνήθως τη μοντελοποίηση των δομικών και συμπεριφορικών διαστάσεων των υπηρεσιών χρησιμοποιώντας αντίστοιχους μηχανισμούς τυπικής μοντελοποίησης και εν συνεχεία, περιλαμβάνουν τη χρησιμοποίηση αλγοριθμικών και αλγεβρικών διαδικασιών για την εξέταση της συμβατότητας, της ισοδυναμίας, της υποκαταστασιμότητας, ή εξειδικεύσεις και παραλλαγές αυτών των εννοιών [27], [90], [24], [21], [26], [28], [141].

Πηγαίνοντας ένα βήμα πέρα από την απόφαση της υποκαταστασιμότητας στο πλαίσιο αποκλειστικά ΠΣΔ υπηρεσιών, ένα ενδιαφέρον και απαιτητικό πρόβλημα είναι η διερεύνηση της μεθόδου λήψης απόφασης για το αν υπηρεσίες που ακολουθούν διαφορετικά υποδείγματα διεπαφών μπορούν να υποκαταστήσουν η μια την άλλη, υπό την εμβέλεια κάποιων αρχιτεκτονικών ή εννοιολογικών συσχετίσεων των στοιχείων των διεπαφών τους. Μια τέτοια δυνατότητα εξέτασης υποκαταστασιμότητας είναι χρήσιμη σε ένα σύνολο σεναρίων, όπως συζητήθηκε στο Κεφάλαιο 1.

Στο Κεφάλαιο 5, παρουσιάζουμε διεξοδικά ένα περιβάλλον-πλαίσιο εξέτασης υποκαταστασιμότητας μεταξύ υπηρεσιών το οποίο βασίζεται σε μια εξειδίκευση των δικτύων Petri που ονομάζονται open nets ή Open WorkFlow Nets (OWFN) και το οποίο έχει προταθεί για την αποτίμηση της δυνατότητας υποκατάστασης μεταξύ ΠΣΔ υπηρεσιών χρησιμοποιώντας έννοιας της *συμφωνίας* ή *accordance*. Χρησιμοποιώντας το πλαίσιο αυτό και με στόχο να δοθεί η δυνατότητα ελέγχου της ΔΥΥ και της ΔΥΕ, εισάγουμε ορισμένους μηχανισμούς εννοιολογικής μοντελοποίησης οι οποίοι επιτρέπου την καταγραφή ειδικών ως προς το υπόδειγμα πληροφοριών και δια-υποδειγματικών συσχετίσεις με σαφή τρόπο. Έπειτα, μέσω μιας διαδικασίας μετασχηματισμού μοντέλων, δείχνουμε πως τα εννοιολογικά μοντέλα μπορούν να μετασχηματιστούν σε open net μοντέλα, τα οποία συγκρίνονται τελικά από το περιβάλλον-πλαίσιο εξέτασης για accordance έτσι ώστε να αποφασισθεί η υποκαταστασιμότητα.

ΚΕΦΑΛΑΙΟ 3

ΖΗΤΗΜΑΤΑ ΠΡΟΣΑΡΜΟΓΗΣ ΥΠΟΔΕΙΓΜΑΤΩΝ

Στο παρόν κεφάλαιο συζητείται η πρόκληση προσαρμογής υποδειγμάτων διεπαφών, στην P2R κατεύθυνση, δηλαδή προσανατολισμού σε διαδικασίες προς προσανατολισμού σε πόρους και διερευνάται ένα σύνολο ζητημάτων σχετικών με τον ορισμό μιας προσέγγισης P2R προσαρμογής. Συγκεκριμένα, στις ενότητες που ακολουθούν, εξετάζουμε την P2R προσαρμογή ως προς τις παρακάτω διαστάσεις:

- *Απαιτήσεις*: αναγνωρίζονται και συζητούνται ένα σύνολο λειτουργικών και μη λειτουργικών απαιτήσεων για μια προσέγγιση P2R προσαρμογής. Συγκεκριμένα, εστιάζουμε σε χαρακτηριστικά τα οποία θεωρούνται σημαντικά για μια τέτοια προσέγγιση προσαρμογής, έτσι ώστε να είναι χρήσιμη και εφαρμόσιμη σε ρεαλιστικά σενάρια τεχνολογίας λογισμικού και περιβάλλοντα εφαρμογής. Επιπλέον, συζητάμε σύντομα ορισμένα θέματα τα οποία είναι κάθετα στη μοντελοποίηση δυνατοτήτων αλλά πρέπει να εξεταστούν πριν την εφαρμογή μιας διαδικασίας P2R προσαρμογής.
- *Σχεδίαση*: η πρόκληση της P2R προσαρμογής υποδείγματος αναλύεται μέσω ενός διαδικαστικού μοντέλου υψηλού αφαιρετικού επιπέδου. Ο στόχος του διαδικαστικού μοντέλου που προτείνεται είναι να αναδείξει τα θέματα και τις προκλήσεις της συγκεκριμένης εργασίας προσαρμογής, να παρέχει μια συστηματική ανάλυση του προβλήματος σε μικρότερα, και να υπηρετήσει ως ένα σχέδιο δράσης για την πραγμάτωση ενός περιβάλλοντος-πλαισίου προσαρμογής.
- *Αξιολόγηση και αποτίμηση*: εξετάζονται μέθοδοι αξιολόγησης της ποιότητας του αποτελέσματος μιας προσέγγισης P2R προσαρμογής, δηλαδή της ΠΣΠ διεπαφής. Υπό αυτήν την άποψη, συζητάμε μοντέλα αξιολόγησης της ωριμότητας και αποτίμησης της συμμόρφωσης ενός API ως προς το REST και εισάγουμε ένα περιβάλλον πλαίσιο συμμόρφωσης στον περιορισμό Ομοιόμορφης διεπαφής του REST.

Στην Ενότητα 3.1, θέτουμε το πλαίσιο σε ό,τι αφορά στις λειτουργικές απαιτήσεις της προσαρμογής υποδειγμάτων για την P2R κατεύθυνση και αναλύουμε τα μη λειτουργικά χαρακτηριστικά τα οποία πρέπει να εμφανίζει μια τέτοια προσέγγιση προσαρμογής, προκειμένου να έχει αναβαθμισμένη πρακτική αξία σε ρεαλιστικές συνθήκες εφαρμογής.

Στη συνέχεια, στην Ενότητα 3.2, εισαγάγουμε ένα αφαιρετικό μοντέλο διαδικασίας P2R προσαρμογής που είναι συνεπές με τις απαιτήσεις που συζητήθηκαν. Η διαδικασία προσαρμογής αποσυντίθεται σε *φάσεις*, οι οποίες καθορίζονται από *βήματα* προσαρμογής που παράγουν ή καταναλώνουν *αντικείμενα* (δηλαδή μοντέλα, έγγραφα και περιγραφές). Κάθε φάση προσαρμογής του διαδικαστικού μοντέλου εξετάζει ξεχωριστά ζητήματα τα οποία συνδέονται με αρχιτεκτονικούς περιορισμούς του REST ή τις περαιτέρω λειτουργικές ή μη-λειτουργικές απαιτήσεις που συζητούνται στην Ενότητα 3.1. Επιπλέον, ένα βήμα προσαρμογής καθορίζει μια ενέργεια (π.χ. υπολογισμός ή μετασχηματισμός) έναντι συγκεκριμένων στοιχείων εισόδου (π.χ. προδιαγραφές, στοιχεία διαμόρφωσης, ενδιάμεσα μοντέλα) και παράγει συγκεκριμένα στοιχεία εξόδου.

Τέλος, στην Ενότητα 3.3 συζητάμε την έννοια της συμμόρφωσης στο REST και εξετάζουμε πλαίσια αξιολόγησης της ωριμότητας RESTful υπηρεσιών και API, όπως προτείνεται στη βιβλιογραφία. Επιπλέον, παρουσιάζουμε το *Εννοιολογικό Περιβάλλον-Πλαίσιο Ομοιόμορφης Διεπαφής* (*Uniform Interface Conceptual Framework (UICF)*), το οποίο παρέχει μια μέθοδο αποτίμησης της συμμόρφωσης ως προς τον περιορισμό της Ομοιόμορφης διεπαφής του REST. Η χρήση τέτοιων πλαισίων μπορεί να επιτρέψει την αξιολόγηση της ποιότητας του αποτελέσματος μιας προσέγγισης P2R προσαρμογής, δηλαδή μιας ΠΣΠ διεπαφής.

3.1 Απαιτήσεις προσέγγισης προσαρμογής

Σε αυτήν την ενότητα, συζητάμε τις απαιτήσεις που επιχειρεί να καλύψει η προσέγγιση προσαρμογής υποδείγματος που προτείνεται στην παρούσα διατριβή. Η εξέταση των απαιτήσεων αυτών αναλύεται σε ξεχωριστή παρουσίαση των *λειτουργικών* και των *μη λειτουργικών* απαιτήσεων προσαρμογής. Οι λειτουργικές απαιτήσεις προσδιορίζουν τη φύση και την έκταση της προσαρμοστικής λογικής που πρέπει να αναγνωριστεί από την προσέγγιση προσαρμογής, καθώς και το πώς και το πότε εφαρμόζεται η προσέγγιση. Οι μη λειτουργικές απαιτήσεις καλύπτουν ορισμένα ποιοτικά χαρακτηριστικά που θεωρούνται σημαντικά για μια προσέγγιση P2R προσαρμογής διεπαφών.

3.1.1 Λειτουργικές απαιτήσεις

Η προσαρμογή λογισμικού, ως πρακτική, αναφέρεται στις ενέργειες που εκτελούνται κατά τον χρόνο εκτέλεσης από το δομοστοιχείο-προσαρμογέα. Ωστόσο, ένα σημαντικό τμήμα μιας διαδικασίας προσαρμογής, επικεντρώνεται στην απόκτηση ή άντληση των κανόνων διαμεσολάβησης και μετασχηματισμού, που πρέπει να εφαρμοστούν κατά τον χρόνο εκτέλεσης (δηλαδή η προσαρμοστική λογική ή λογική προσαρμογής), η οποία υλοποιείται συνήθως απογραμμικά. Καθώς η προσαρμογή υποδειγμάτων διεπαφών στοχεύει στην αντιμετώπιση αρχιτεκτονικών παρεκκλίσεων ως προς τη σχεδίαση των διεπαφών, η κύρια λειτουργική απαίτηση μιας τέτοιας προσέγγισης προσαρμογής είναι να εντοπιστούν οι απαραίτητες αντιστοιχίες μεταξύ των μοντέλων των διεπαφών των δύο διαφορετικών υποδειγμάτων, έτσι ώστε μια δυνατότητα υπηρεσίας (π.χ. λειτουργικότητα ή πληροφορίες) που παρέχεται από υπηρεσία που ακολουθεί το ένα υπόδειγμα, να μπορεί να γίνει προσβάσιμη μέσω μιας διεπαφής που ακολουθεί το δεύτερο υπόδειγμα. Όσον αφορά στις προσεγγίσεις P2R προσαρμογής, όπως αυτή που προτείνεται στην παρούσα διατριβή, η κύρια λειτουργική απαίτηση πέρα από τη δυνατότητα υποκατάστασης της υφιστάμενης διεπαφής, είναι η συμμόρφωση της παραγόμενης διεπαφής στους περιορισμούς του REST που αφορούν στη σχεδίαση των διεπαφών και στις αρχές του προσανατολισμού σε πόρους. Την ίδια στιγμή, ένα πρακτικό ζήτημα και απαίτηση σε σχέση με το περιβάλλον-πλαίσιο που υλοποιεί τη διαδικασία, είναι ότι θα πρέπει να είναι αρκετά ευέλικτο ώστε να επιτρέπει την παροχή αποτελεσμάτων που ενδέχεται να εμφανίζουν μερική συμμόρφωση. Κάτι τέτοιο είναι σημαντικό καθώς συχνά οι αρχιτέκτονες επιλέγουν μερικώς RESTful διεπαφές λόγω συμβιβασμών που είναι απαραίτητοι να γίνουν στα πλαίσια συγκεκριμένων περιβαλλόντων ανάπτυξης και λειτουργίας.

Ωστόσο, ανάλογα με το αν και οι δύο προδιαγραφές διεπαφών είναι ήδη διαθέσιμες ή αν είναι μόνο μία από αυτές, μπορούν να προσδιοριστούν δύο τύποι προσεγγίσεων προσαρμογής, η *meet-in-the-middle (MITM)* προσαρμογή υποδείγματος και η *one-way (OW)* (*μιας κατεύθυνσης*) προσαρμογή υποδείγματος. Ο *MITM* τύπος προσαρμογής υποδείγματος προβλέπει ότι το δομοστοιχείο προσαρμογής μεσολαβεί μεταξύ δύο διεπαφών και μεταφράζει μηνύματα που απευθύνονται σε μια προκαθορισμένη διεπαφή υπηρεσίας η οποία ακολουθεί ένα συγκεκριμένο υπόδειγμα, σε μια δεύτερη προκαθορισμένη διεπαφή η οποία ακολουθεί διαφορετικό υπόδειγμα. Υπό την άποψη αυτή, η προσαρμοστική λογική περιλαμβάνει όλες

τις απαιτούμενες αντιστοιχίες μεταξύ των στοιχείων που ορίζονται στις δύο προδιαγραφές διεπαφών, έτσι ώστε οι δυνατότητες των υπηρεσιών να προσαρμόζονται να εκτίθενται με αποτελεσματικό τρόπο. Μια προσέγγιση MITM προσαρμογής είναι εξ ορισμού αμφίδρομη. Ως εκ τούτου, υποστηρίζει τόσο την P2R και την R2P κατεύθυνση προσαρμογής. Ο OW τύπος προσαρμογής υποδείγματος, προβλέπει ότι το δομοστοιχείο προσαρμογής δημιουργεί και εκθέτει μια διεπαφή η οποία ακολουθεί ένα συγκεκριμένο υπόδειγμα, έτσι ώστε οι δυνατότητες της υπηρεσίας να αντιστοιχούνται και να εκτίθενται με αποτελεσματικό τρόπο. Υπό την άποψη αυτή, η προσαρμοστική λογική πρέπει να καθορίζει τόσο την εκτεθειμένη διεπαφή, όσο και να παρέχει τις απαιτούμενες αντιστοιχίσεις μεταξύ των στοιχείων των δύο προδιαγραφών των διεπαφών. Μια προσέγγιση OW προσαρμογής, υποστηρίζει μία από τις κατευθύνσεις προσαρμογής P2R ή R2P.

Μια μέθοδος άντλησης της προσαρμοστικής λογικής στα πλαίσια μιας προσέγγισης MITM προσαρμογής απαιτεί μια μέθοδο εξαγωγής αντιστοιχιών, κατά τη διάρκεια της οποίας αναγνωρίζονται συσχετισμοί μεταξύ των στοιχείων των δύο προδιαγραφών, έτσι ώστε να μπορούν να εφαρμοστούν κατάλληλοι μετασχηματισμοί από το δομοστοιχείο-προσαρμογέα. Σχετικά με την OW προσαρμογή, η μέθοδος άντλησης της προσαρμοστικής λογικής μπορεί να αναλυθεί σε δύο μεθόδους, συγκεκριμένα στη μέθοδο εξαγωγής διεπαφής, κατά τη διάρκεια της οποίας αναγνωρίζονται τα πρωτεύοντα στοιχεία της διεπαφής-στόχου (εκείνης που εκτίθεται) και μια μέθοδο εξαγωγής αντιστοιχιών, όμοια με εκείνη που περιλαμβάνεται σε μια προσέγγιση MITM προσαρμογής. Υπό αυτήν την έννοια, οποιεσδήποτε τεχνικές επινοούνται και εφαρμόζονται στο πλαίσιο μιας προσέγγισης OW προσαρμογής για την υλοποίηση της μεθόδου εξαγωγής αντιστοιχιών, μπορούν επίσης να χρησιμοποιηθούν για να υποστηρίξουν σενάρια MITM προσαρμογής, εφόσον υπάρχουν καλά ορισμένα και σαφή όρια μεταξύ των δύο μεθόδων που προσδιορίζονται στα πλαίσια μιας OW προσέγγισης (δηλαδή η μέθοδος εξαγωγής αντιστοιχιών που επινοείται δεν πρέπει να στηρίζεται σε ενδιάμεσα αποτελέσματα της μεθόδου εξαγωγής διεπαφής).

Αφού αναγνωριστεί η προσαρμοστική λογική, αποτυπώνεται και καταγράφεται τυπικά ως ένα έγγραφο προδιαγραφής ή οδηγίας προσαρμογής. Η οδηγία προσαρμογής παρέχεται σε ένα δομοστοιχείο-προσαρμογέα, έτσι ώστε η προσαρμοστική λογική να μπορεί να εφαρμοστεί κατά τον χρόνο εκτέλεσης. Το δομοστοιχείο-προσαρμογέα πρέπει να είναι σε θέση να φορτώνει, να προ-επεξεργάζεται και να

εφαρμόζει την προσαρμοστική λογική που περιλαμβάνεται σε αρχεία οδηγιών προσαρμογής, με τρόπο ανεξάρτητο της συγκεκριμένη εφαρμογής ή πεδίου εφαρμογών. Η προσαρμογή κατά τον χρόνο εκτέλεσης θα πρέπει να εφαρμόζεται μέσω ερμηνείας των μηνυμάτων που περιέχουν αιτήματα ΠΣΠ (τα οποία τυπικά περιλαμβάνουν δεδομένα ελέγχου, αναγνωριστικά και αναπαραστάσεις πόρων) σε κλήσεις λειτουργικών υπηρεσιών ΠΣΔ, και μέσω της μετάφρασης των αποτελεσμάτων των κλήσεων αυτών σε ΠΣΠ αποκρίσεις (οι οποίες τυπικά περιλαμβάνουν αναπαραστάσεις πόρων, μεταδεδομένα πόρων και δεδομένα ελέγχου).

Στην παρούσα διατριβή, διερευνούμε και εξετάζουμε την πρόκληση της P2R προσαρμογής υποδείγματος, προτείνοντας μια προσέγγιση OW προσαρμογής, της οποίας η μέθοδος άντλησης προσαρμοστικής λογικής ακολουθεί την ανάλυση σε φάσεις *εξαγωγής διεπαφής* και *εξαγωγής αντιστοιχιών*. Υπό αυτό το πρίσμα, οι αλγόριθμοι και οι τεχνικές που προτείνονται στο πλαίσιο της μεθόδου εξαγωγής αντιστοιχιών της προσέγγισης που προτείνεται, μπορούν επίσης να χρησιμοποιηθούν σε προβλήματα MITM προσαρμογής. Επιπρόσθετα, το περιβάλλον-πλαίσιο προσαρμογής που προτείνεται περιλαμβάνει ένα δομοστοιχεία προσαρμογέας χρόνου εκτέλεσης, το οποίο έχει τη δυνατότητα να εκθέσει ένα REST API εφαρμόζοντας την οδηγία προσαρμογής που έχει αναγνωριστεί.

3.1.2 Μη λειτουργικές απαιτήσεις

Η αυτοματοποίηση βελτιώνει τη χρηστικότητα όπως και τις πιθανότητες υιοθέτησης μιας προσέγγισης προσαρμογής. Το να αναγνωριστεί η προσαρμοστική λογική που απαιτείται για την προσαρμογή υποδείγματος ιδιοχείρως από τον μηχανικό, μπορεί να είναι μια απαιτητική, κοπιώδης και επιρρεπής σε λάθη εργασία. Συνεπώς, αυτοματοποιώντας τμήματα ή ολόκληρη τη διαδικασία προσαρμογής μπορεί να διευκολυνθεί η άντληση ακριβών προσαρμογέων με περιορισμένη ανθρώπινη προσπάθεια. Επιπλέον, οι διαδικασίες προσαρμογής που εμφανίζουν υψηλά επίπεδα αυτοματοποίησης μπορούν να διευκολύνουν έργα προσαρμογής μεγάλης κλίμακας (π.χ. πλήρη χαρτοφυλάκια υπηρεσιών οργανισμών) και να παρέχουν σημαντική ευελιξία στους οργανισμούς ως προς την αντιμετώπιση των μονίμως εξελισσόμενων απαιτήσεων ολοκλήρωσης. Εξαιτίας της σύνθετης φύσης της προσαρμογής υποδείγματος, είναι συνήθως αναμενόμενο μια προσέγγιση προσαρμογής να απαιτεί κάποιου είδους ανθρώπινη συμμετοχή. Ωστόσο, όπως συζητείται στο

Κεφάλαιο 2, Ενότητα 2.4.3, οι περισσότερες από τις προσεγγίσεις που έχουν προταθεί στη βιβλιογραφία ως τώρα εξαρτώνται σημαντικά από ανθρώπους-έμπειρους χρήστες που εκτελούν χειροκίνητες εργασίες, περιορίζοντας έτσι το επίπεδο αυτοματοποίησης. Η προσέγγιση που προτείνεται στην παρούσα διατριβή χειρίζεται την αυτοματοποίηση ως ένα από τα πιο σημαντικά ζητήματα και επιχειρεί να περιορίσει την ανθρώπινη συμμετοχή κυρίως στην επισκόπηση και στη βελτίωση των αποτελεσμάτων της προσαρμογής.

Μια άλλη πτυχή της προσαρμογής σχετίζεται με το αν εφαρμόζονται αναδιαρθρώσεις (refactorings) ή τροποποιήσεις στα δομοστοιχεία που προσαρμόζονται, προκειμένου να επιτευχθούν οι στόχοι της προσαρμογής. Στο πλαίσιο αυτό, μια δεύτερη μη λειτουργική απαίτηση που θεωρούμε στην προσέγγιση προσαρμογής υποδείγματος η οποία προτείνεται στην παρούσα διατριβή, είναι η *μη παρεμβατικότητα*, δηλαδή τα υπάρχοντα δομοστοιχεία δεν πρέπει να μεταβάλλονται με οποιονδήποτε τρόπο. Ο προσαρμογέας κατά τον χρόνο εκτέλεσης πρέπει επίσης να λειτουργεί διαφανώς ως προς τα υπάρχοντα δομοστοιχεία υπηρεσιών, επιτρέποντάς τους να συνεχίσουν να προσφέρουν τις υπηρεσίες τους στους καταναλωτές τους, ακριβώς με τον τρόπο που το έκαναν. Με άλλα λόγια, ο P2R προσαρμογέας χρόνου εκτέλεσης θα πρέπει να γίνει αντιληπτός από το δομοστοιχείο υπηρεσίας το οποίο προσαρμόζεται ως *ένας ακόμη καταναλωτής*.

Επιπρόσθετα στις παραπάνω απαιτήσεις, προκειμένου η άντληση της προσαρμοστικής λογικής της προτεινόμενης προσέγγισης να έχει πρακτική αξία σε ρεαλιστικές συνθήκες, θεωρούμε δύο επιπλέον μη λειτουργικές απαιτήσεις, συγκεκριμένα τις: *α*) *ανεξαρτησία ως προς την υλοποίηση*: δε θα πρέπει να απαιτείται διαθεσιμότητα και πρόσβαση στον πηγαίο κώδικα, σε σχήματα βάσεων δεδομένων ή προδιαγραφές της υλοποίησης της υπηρεσίας και *β*) *αποδοτικότητα*: θα πρέπει ο χρόνος επεξεργασίας και η επιβάρυνση λόγων των υπολογισμών να είναι περιορισμένα. Αυτό επιτρέπει στη διαδικασία εξαγωγής να μπορεί να ενσωματωθεί στην πρακτική της τεχνολογίας λογισμικού, μέσω μιας αλληλεπιδραστικής και οδηγούμενης από τον άνθρωπο δραστηριότητας, η οποία μπορεί να υποστηριχθεί από ένα αντίστοιχο δομοστοιχείο ενός ολοκληρωμένου περιβάλλοντος ανάπτυξης (IDE).

Η απαίτηση της ανεξαρτησίας από την υλοποίηση, περιορίζει τις πληροφορίες και τα αντικείμενα τα οποία πρέπει να παρέχονται ως δεδομένα εισόδου στη διαδικασία προσαρμογής που θα εφαρμοστεί, το οποίο είναι συχνό προβληματικό σημείο των σχετικών συνεισφορών στη βιβλιογραφία. Ειδικότερα, θεωρούμε ως ελάχιστη

είσοδο για τις πυρηνικές δραστηριότητες προσαρμογής της προτεινόμενης διαδικασίας προσαρμογής (δηλαδή εξαγωγή διεπαφής και εξαγωγή αντιστοιχιών), ένα μόνο IDL έγγραφο που περιγράφει τη διαδικαστική διεπαφή, όπως για παράδειγμα μια περιγραφή WSDL ενός παραδοσιακού Web service. Επίσης, αν και η διαδικασία προσαρμογής μπορεί να υπόκειται σε επιλογές διαμόρφωσης, οι επιλογές αυτές θα πρέπει να περιορίζονται μόνο σε προτιμήσεις σχετικές με την προσαρμογή καθαυτή και δε θα πρέπει να ανακλούν πληροφορίες που σχετίζονται με τη συγκεκριμένη υπηρεσία που προσαρμόζεται. Ωστόσο, θα πρέπει να σημειωθεί ότι ενδεχομένως να απαιτούνται περαιτέρω πληροφορίες από το περιβάλλον προσαρμογής, όταν στο πλαίσιό του υλοποιούνται επιπλέον δυνατότητες που εξειδικεύουν και να εμπλουτίζουν το προκύπτον REST API (π.χ. φίλτρα πόρων) -σαφώς, οι δυνατότητες αυτές δε θεωρούνται πυρηνικές δραστηριότητες προσαρμογής. Με τον τρόπο αυτό, μια προσέγγιση προσαρμογής που είναι ανεξάρτητη από την υλοποίηση, έχει ένα πολύ ευρύ φάσμα εφαρμογής, δεδομένου ότι καλύπτει κάθε ΠΣΔ υπηρεσία, ή γενικά, κάθε διαδικαστικό δομοστοιχείο με προδιαγεγραμμένη διεπαφή, ανεξαρτήτως από τη διαθεσιμότητα πληροφοριών σχετικών με τον πηγαίο κώδικα, μοντέλων πεδίου, σχημάτων βάσεων δεδομένων και τεχνολογιών που χρησιμοποιούνται, ή οποιωνδήποτε άλλων τεχνικών στοιχείων και προδιαγραφών που περιγράφουν την υλοποίηση του δομοστοιχείου.

Αντιμετωπίζουμε την αποδοτικότητα ως μια σημαντική πτυχή της μεθόδου εξαγωγής της προσαρμοστικής λογικής, διότι θεωρούμε την εφαρμογή μιας τέτοιας μεθόδου ως μια τακτική δραστηριότητα ενός μηχανικού, η οποία θα πρέπει να μπορεί να πραγματοποιηθεί διαδραστικά και στα πλαίσια ενός ολοκληρωμένου περιβάλλοντος ανάπτυξης. Υπό αυτήν την άποψη, θα πρέπει να αναμένονται πολλαπλές εκτελέσεις της διαδικασίας προσαρμογής, είτε λόγω αναθεωρήσεων των σημείων διαμόρφωσης της διαδικασίας προσαρμογής, είτε μετά από τροποποιήσεις που γίνονται σε ήδη προσαρμοσμένα δομοστοιχεία ως μέρος της συντήρησης και της εξέλιξή τους. Τέλος, η έννοια της αποδοτικότητας θα πρέπει επίσης να χαρακτηρίζει και το δομοστοιχείο-προσαρμογέα ενός πλαισίου προσαρμογής υποδείγματος. Συγκεκριμένα, η ερμηνεία και η εφαρμογή της προσαρμοστικής λογικής που έχει αναγνωριστεί, θα πρέπει να είναι ιδιαίτερα αποδοτική, έτσι ώστε η υπολογιστική επιβάρυνση λόγω της διαμεσολάβησης του προσαρμογέα να μην έχει σημαντική επίπτωση στην επίδοση και την ποιότητα της υπηρεσίας, όπως αυτές γίνονται αντιληπτές από την πλευρά των καταναλωτών και να μην περιορίζει τις δυνατότητες κλιμάκωσης της προσέγγισης. Η απαίτηση αυτή έχει συνέπειες τόσο σε ότι αφορά

στην εκφραστικότητα των προδιαγραφών προσαρμογής, καθώς και σε ό,τι αφορά στη σχεδίαση και την υλοποίηση του προσαρμογέα χρόνου εκτέλεσης.

3.1.3 Κάθετα ζητήματα

Όπως συζητήθηκε στο Κεφάλαιο 2, η πρόκληση προσαρμογής υποδείγματος εστιάζει στην αντιμετώπιση αναντιστοιχιών στον τρόπο έκθεση των δυνατοτήτων των υπηρεσιών, δηλαδή στα σχεδιαστικά υποδείγματα των διεπαφών. Ωστόσο, όταν προσαρμόζονται ΠΣΔ υπηρεσίες σε ΠΣΠ υπηρεσίες, υπάρχει ένα σύνολο θεμάτων που πρέπει να θεωρηθούν ως κάθετα στη μοντελοποίηση δυνατοτήτων και στην ανάλυση-οργάνωση της λειτουργικότητας, τα οποία ενδεχομένως θα πρέπει επίσης να αντιμετωπίζονται μέσω κατάλληλων μεθόδων. Στις παραγράφους που ακολουθούν, συνοψίζουμε ορισμένα τέτοια βασικά ζητήματα.

Απουσία κατάστασης και ιδιότητες συναλλαγών. Η επικοινωνία με υπηρεσίες η οποία χαρακτηρίζεται από ύπαρξη κατάστασης αποτελεί ένα σημαντικό θέμα στην υπηρεσιοστρεφή υπολογιστική το οποίο πρέπει να αντιμετωπιστεί προτού εφαρμοστεί οποιαδήποτε τεχνική προσαρμογής ως προς τη λειτουργικότητα. Θα πρέπει να σημειωθεί ότι οι απαιτήσεις του REST σχετικά με την απουσία κατάστασης αφορά στην επικοινωνία μεταξύ των αρχιτεκτονικών στοιχείων και όχι στην προσφερόμενη υπηρεσία καθαυτή (οι πόροι είναι εγγενώς οντότητες που έχουν κατάσταση). Ειδικότερα, το REST απαιτεί ότι οποιοδήποτε αίτημα δεν πρέπει να εξαρτάται από κάποιο προηγούμενο αίτημα ώστε να γίνει κατανοητό και να μπορεί να ερμηνευτεί. Υπό αυτό το πρίσμα, σε ένα RESTful σύστημα δεν πρέπει να υπάρχει κατάσταση συνεδρίας στην πλευρά του εξυπηρετητή (ή τουλάχιστον, αυτή δεν μπορεί να είναι παρατηρήσιμη) και τα μηνύματα πρέπει να είναι αυτο-περιγραφικά, υποδεικνύοντας πλήρως πως μπορούν να γίνουν κατανοητά και να ερμηνευτούν, ανεξαρτήτως από προηγούμενες αλληλεπιδράσεις. Εφόσον η διατήρηση κατάστασης συνεδρίας που δεν αποθηκεύεται δεν είναι αποδεκτή σε RESTful αρχιτεκτονικές, εγείρεται μια πρόκληση ως προς την προσαρμογή στο REST συστημάτων προσανατολισμένων σε διαδικασίες που ταυτόχρονα διατηρούν κατάσταση συνεδρίας στην πλευρά του εξυπηρετητή, η οποία αφορά στο πως η σημασιολογία των ιδιοτήτων των συναλλαγών (κάτι που διεθνώς αναφέρεται ως transactionality) μπορεί να μοντελοποιηθεί και να υλοποιηθεί με έναν πραγματικά RESTful τρόπο. Η γενική προσέγγιση που έχει προταθεί στη βιβλιογραφία είναι η επέκταση του μοντέλου

πόρων με κατάλληλους πόρους οι οποίοι μοντελοποιούν και αποθηκεύουν χαρακτηριστικά εκτέλεσης και συντονισμού συναλλαγών. Για παράδειγμα στην εργασία [113] ορίζονται κατάλληλοι πόροι *συμμετέχοντα* και *συντονιστή*, διαχωρισμένοι από το υπόλοιπο μοντέλο, οι οποίοι προτείνονται ως ένα σχεδιαστικό υπόδειγμα για να επιτραπεί η υλοποίηση ατομικών κατανεμημένων συναλλαγών. Επίσης, στο ίδιο πλαίσιο, έχει προταθεί από το REST-* initiative[1] μια προδιαγραφή για την υποστήριξη της ατομικότητας σε σενάρια κατανεμημένων συναλλαγών με συντονισμένα αποτελέσματα, τα οποία είναι βασισμένα στο REST (π.χ. πρωτόκολλα two phase commit). Η προδιαγραφή αυτή βασίζεται επίσης σε πόρους συντονισμού συναλλαγών και πόρους συμμετοχής. Καθώς ο ορισμός της προσαρμογής υποδειγμάτων διεπαφής αφορά σε θέματα ανάλυσης-οργάνωσης της λειτουργικότητας, θεωρούμε την ενσωμάτωση τέτοιων επεκτάσεων του μοντέλου πόρων ως μια ανεξάρτητη εργασία προσαρμογής.

Για λόγους πληρότητας, θα πρέπει να σημειωθεί ότι το αν το REST ως αρχιτεκτονικό στυλ αποτελεί γενικότερα καλή επιλογή για την υποστήριξη μοντέλων κατανεμημένων συναλλαγών έχει αποτελέσει θέμα δημοσίου διαλόγου [79], [112]. Με βάση αυτό το διάλογο, έχει προταθεί στη βιβλιογραφία ένα πλήθος διαφορετικών μοντέλων και επεκτάσεων, είτε στο αρχιτεκτονικό επίπεδο, είτε σε επίπεδο τεχνολογιών που χρησιμοποιούνται από ΠΣΠ υπηρεσίες (π.χ. για το HTTP), έτσι ώστε να αντιμετωπιστούν διάφορες απαιτήσεις σχετικές με συναλλαγές [125], [112], [88], [36], [37], [40].

Ποιότητα υπηρεσιών και χαρακτηριστικά. RESTful υπηρεσίες οι οποίες είναι δεδομενο-κεντρικές (π.χ. Web 2.0, mash-ups) έχουν συνήθως χαλαρότερες απαιτήσεις όσον αφορά σε χαρακτηριστικά ποιότητας υπηρεσιών (QoS), σε σύγκριση με τα σενάρια επιχειρηματικών συστημάτων υπηρεσιών. Είναι γενικώς αποδεκτό ότι οι WS-* QoS προδιαγραφές συνήθως χαρακτηρίζονται από υψηλό επίπεδο εκφραστικότητας. Ωστόσο, είναι συχνά ιδιαιτέρως πολύπλοκες και αυτός είναι ο πρωταρχικός λόγος για την περιορισμένη υιοθέτησή τους. Οι συζητήσεις γύρω από τη δημιουργία RESTful επιχειρηματικών συστημάτων με τις εν λόγω απαιτήσεις, φαίνεται να οδηγούν στην άποψη ότι πρέπει να προηγείται κάθε φορά προσεκτική ανάλυση του σκεπτικού και στόχων των υπηρεσιών και συχνά υποστηρίζεται ότι οι υφιστάμενες τεχνολογίες του Παγκόσμιου Ιστού, ως μέσα για την εκπλήρωση των απαιτήσεων ποιότητας, είναι συνήθως επαρκή. Μέχρι σήμερα, η εφαρμογή των

[1] http://www.jboss.org/reststar/specifications/transactions.html

τυποποιημένων πλαισίων QoS δεν έχει εξεταστεί ευρέως και υπάρχουν μόνο λίγες ανάλογες πρωτοβουλίες και πρότυπα. Το γεγονός αυτό μπορεί να αποδοθεί σε μια επικρατούσα άποψη στην κοινότητα του REST, ότι η απλότητα και η γενικότητα των RESTful HTTP εφαρμογών αποτελούν σημαντικά πλεονεκτήματα τα οποία οι αρχιτέκτονες λογισμικού πρέπει να προσπαθούν να διατηρούν, ακόμα και σε πολύπλοκα επιχειρηματικά σενάρια -οι λύσεις για προβλήματα που υπάρχουν στο επίπεδο επιχειρηματικής λειτουργίας δε θα πρέπει να είναι τεχνικής φύσεως, με την έννοια της χρήσης ολοένα και πιο περίπλοκων και απαιτητικών προτύπων και περιβαλλόντων-πλαισίων.

Αν και οι QoS απαιτήσεις για το προσαρμοσμένο σύστημα μπορεί να είναι αρκετά σημαντικές και θα πρέπει να ληφθούν υπόψη πριν από την έναρξη μιας προσπάθειας προσαρμογής υποδείγματος, η αντιμετώπιση των QoS χαρακτηριστικά γενικότερα μπορεί να θεωρηθεί ως ένα ζήτημα που εκτείνεται πέραν του πεδίου προσαρμογής υποδείγματος, καθώς το σημείο εστίασης της προσαρμογής αυτής είναι να παρέχει μια διαφορετική έκθεση των δυνατοτήτων των υπηρεσιών, όσον αφορά στην παρεχόμενη λειτουργικότητα, στις πληροφορίες και στα μοντέλα αλληλεπίδρασης. Ως εκ τούτου, θεωρούμε ότι οι απαιτήσεις QoS εξετάζονται χωριστά. Τα χαρακτηριστικά ποιότητας μπορούν να θεωρηθούν ως ρυθμιζόμενα από το υποκείμενο τεχνολογικό επίπεδο, όπως γίνεται για παράδειγμα στο SCA, μέσω του ανεξάρτητου χειρισμού των πτυχών ποιότητας υπηρεσιών στο [43]. Με τον τρόπο αυτό, ορισμένα χαρακτηριστικά ποιότητας μπορούν να επιτευχθούν μέσω κατάλληλων επιλογών διαμόρφωσης του υποκείμενου περιβάλλοντος. Για παράδειγμα, ένα τυπικό σημείο ρύθμισης θα ήταν να χρησιμοποιηθεί το HTTPS αντί για του HTTP, προκειμένου να καλυφθεί κάποια απαίτηση ασφαλείας. Τούτου λεχθέντος, θα πρέπει επίσης να σημειωθεί ότι υπάρχουν ορισμένα χαρακτηριστικά ποιότητας στα υπηρεσιοστρεφή συστήματα, η υποστήριξη των οποίων είναι υπό αμφισβήτηση όταν χρησιμοποιούνται οι περισσότερες από τις υπάρχουσες τεχνολογίες και πρακτικές υλοποίησης RESTful υπηρεσιών. Τέτοια ζητήματα αποτελούν ακόμη ανοικτές ερευνητικές προκλήσεις στην κοινότητα. Όσον αφορά στην P2R προσαρμογή υποδείγματος, ένας αρχιτέκτονας θα πρέπει πρώτα να εξετάσει κατά πόσον οι QoS απαιτήσεις για το σύστημα μπορούν να επιτευχθούν στο πλαίσιο μιας αντίστοιχης RESTful αρχιτεκτονικής στη βάση των δυνατοτήτων των σχετικών τεχνολογιών, ή μέσω κατάλληλων RESTful σχεδιάσεων και να αποφασίσει το κατά πόσον η εφαρμογή της προσαρμογής υποδείγματος αποτελεί φρόνιμη επιλογή.

3.2 Μοντέλο διαδικασίας προσαρμογής

Η αντιμετώπιση της πρόκλησης P2R προσαρμογής με έναν σημαντικά αυτοματοποιημένο, ανεξάρτητο σε σχέση με την υλοποίηση και ταυτόχρονα αποδοτικό τρόπο θεωρείται αρκετά σύνθετο πρόβλημα καθώς το αντίστοιχο περιβάλλον-πλαίσιο προσαρμογής πρέπει όχι μόνο να παράγει ως αποτέλεσμα συμβατές με το REST διεπαφές, αλλά και να το επιτυγχάνει αυτό με την ελάχιστη δυνατή ανθρώπινη συμμετοχή και με περιορισμένες, αν όχι τις ελάχιστες δυνατές, πηγές πληροφορίας. Ως εκ τούτου, πριν την παρουσίαση ενός συμπαγούς περιβάλλοντος-πλαισίου P2R προσαρμογής το οποίο υλοποιεί τις μεθόδους εξαγωγής διεπαφής και εξαγωγής αντιστοιχιών και αντιμετωπίζει ζητήματα που αφορούν στον χρόνο εκτέλεσης, παρουσιάζουμε και αναλύουμε ένα αφαιρετικό μοντέλο διαδικασίας το οποίο μπορεί να λειτουργήσει ως μια εννοιολογική ανάλυση-διαίρεση του προβλήματος σε απλούστερα. Η πρόκληση προσαρμογής αναλύεται σε διαδικασίες ή βήματα τα οποία παρουσιάζονται μαζί με τις πληροφορίες που απαιτούν και που πρέπει να παράγουν, παρέχοντας με αυτό τον τρόπο ένα υψηλού αφαιρετικού επιπέδου σχέδιο για την υλοποίηση μιας κατάλληλης προσέγγισης προσαρμογής. Το αφαιρετικό μοντέλο διαδικασίας που συζητείται στην παρούσα ενότητα αποτελεί βελτίωση του μοντέλου που προτάθηκε στο [9].

Ο κύριος στόχος του μοντέλου που προτείνεται είναι να παρέχει μια συστηματική διερεύνηση του πως τα ζητήματα που εκλύονται από την πρόκληση της P2R προσαρμογής και περιγράφονται παρακάτω μπορούν να αντιμετωπιστούν μέσω μιας αναλυτικής διαδικασίας αποτελούμενη από *α*') δραστηριότητες προσαρμογής και *β*') πληροφοριακές οντότητες που παράγονται ή καταναλώνονται από τις δραστηριότητες αυτές.

Αρχικά, η διαδικασία προσαρμογής διαιρείται σε δύο φάσεις βάσει του χρόνου εφαρμογής των σχετικών εργασιών προσαρμογής, δηλαδή στη *φάση του χρόνου σχεδίασης* (*desing-time phase*) και στη *φάση του χρόνου εκτέλεσης* (*run-time phase*). Η φάση του χρόνου σχεδίασης εστιάζει στην άντληση της απαραίτητης προσαρμοστικής λογικής ώστε να συντεθεί μια RESTful διεπαφή όπως και μια αντίστοιχη οδηγία προσαρμογής, χρησιμοποιώντας ως δεδομένα εισόδου μια προδιαγραφή της ΠΣΔ διεπαφής. Στο πλαίσιο αυτό, τα ζητήματα που αντιμετωπίζει η φάση του χρόνου σχεδίασης χωρίζονται σε δύο σύνολα, το σύνολο των *στατικών/διαρθρωτικών* (*static/structural*) ζητημάτων και στο σύνολο των *δυναμικών/συμπεριφορικών*

ζητημάτων. Τα στατικά ή διαρθρωτικά ζητήματα προέρχονται κυρίως από τον περιορισμό Ομοιόμορφης διεπαφής του REST και σχετίζονται με την αναγνώριση των στοιχείων REST διεπαφών τα οποία εξυπηρετούν την επισημείωση και την έκθεση της λειτουργικότητας, όπως και με την αναγνώριση των συσχετίσεων των στοιχείων αυτών με στοιχεία της υπάρχουσας ΠΣΔ διεπαφής. Τα δυναμικά ή διαρθρωτικά ζητήματα ενός REST API σχετίζονται με τις ενδείξεις προσωρινής αποθήκευσης, οι οποίες προέρχονται από τον περιορισμό Κρυφής μνήμης του REST, με δυνατότητες υπερμέσων, οι οποίες αποτελούν επίσης τμήμα του περιορισμού Ομοιόμορφης Διεπαφής και με επιλογές απόδοσης και εξατομίκευσης των διεπαφών (π.χ. περιορισμός ή αναδρομολόγηση της πρόσβασης σε ορισμένους πόρους). Η φάση του χρόνου εκτέλεσης αντιμετωπίζει τους περιορισμούς του REST Πελάτη-Εξυπηρετητή, Διαστρωματωμένου συστήματος και Επικοινωνίας χωρίς μνήμη κατάστασης. Ο στόχος της προσαρμογής είναι ότι η εξωτερική όψη του προσαρμοσμένου συστήματος μπορεί τελικά να συμμορφώνεται στο REST, μέσω του σεβασμού των χαρακτηριστικών που κάθε περιορισμός του REST αποδίδει στην προκύπτουσα αρχιτεκτονική. Ωστόσο, η διαδικασία προσαρμογής πρέπει να προβλέπει τη διευκόλυνση αρχιτεκτονικών συμβιβασμών οι οποίοι ενδέχεται να εμποδίζουν την πλήρη συμμόρφωση στο REST. Υπό αυτό το πρίσμα, πρέπει να επιτρέπεται η δυνατότητα διαμόρφωσης και βελτίωσης (tuning) της προσαρμογής από την πλευρά του χρήστη. Το Σχήμα 3.1 δείχνει μια γραφική απεικόνιση του αφαιρετικού μοντέλου διαδικασίας που προτείνεται. Στην υποενότητα που ακολουθεί παρέχουμε μια γενική περιγραφή και ένα περίγραμμα του μοντέλου διαδικασίας. Έπειτα, συνεχίζουμε συζητώντας καθένα από τα συστατικά του, τα οποία περιλαμβάνουν τόσο υπολογιστικά δομοστοιχεία που υλοποιούν δραστηριότητες προσαρμογής, όσο και πληροφοριακά τεχνουργήματα.

3.2.1 Περίγραμμα διαδικασίας

Όπως έχει ήδη αναφερθεί, η διαδικασία προσαρμογής χωρίζεται σε δύο φάσεις: στη φάση του χρόνου σχεδίασης και στη φάση του χρόνου εκτέλεσης. Κατά τη διάρκεια της φάσης του χρόνου σχεδίασης εφαρμόζεται ένα σύνολο τεχνικών στις προδιαγραφές και στα στοιχεία περιγραφής του συστήματος όπως και στα μεταδεδομένα διαμόρφωσης της προσαρμογής. Η συμμετοχή του χρήστη μοντελοποιείται με τη μορφή του ελέγχου του αποτελέσματος και της επανεξέτασής του, η οποία

Σχήμα 3.1: Μοντέλο διαδικασίας προσαρμογής

εξαρτάται κυρίως από την ευστοχία των χρησιμοποιούμενων τεχνικών μαζί με τις στοχεύσεις του χρήστη ως προς την εφαρμογή ειδικών αρχιτεκτονικών αποφάσεων (π.χ. χαλάρωση κάποιου περιορισμού στο πλαίσιο ενός αναγνωρισμένου αρχιτεκτονικού συμβιβασμού). Η φάση του χρόνου εκτέλεσης καταναλώνει το αποτέλεσμα της φάσης του χρόνου σχεδίασης και δεν απαιτεί τη συμμετοχή του χρήστη, εκτός από τη διαμόρφωση του περιβάλλοντος εκτέλεσης του προσαρμογέα (π.χ. σημεία διαμόρφωσης ενός SCA runtime). Κατά τη διάρκεια του χρόνου εκτέλεσης λαμβάνει χώρα η αυτοματοποιημένη αντιστοίχιση και μετατροπή των μηνυμάτων όπως και η διαχείριση των αλληλεπιδράσεων, οι οποίες ενέργειες μπορούν να ρυθμιστούν σε διάφορα επίπεδα *ευφυίας*, δεδομένων των στόχων της εφαρμογής -για παράδειγμα, να δοθεί η δυνατότητα στον προσαρμογέα να διαχειρίζεται επιπλέον εικονικούς πόρους, οι οποίοι αναπαριστούν στιγμιότυπα διαδικασιών και δεν υφίστανται ως οντότητες στην υπάρχουσα υπηρεσία.

Η φάση του χρόνου σχεδίασης διασπάται περαιτέρω σε υποφάσεις, βάσει του διαχωρισμού μεταξύ στατικών/διαρθρωτικών ζητημάτων και δυναμικών/συμπεριφορικών ζητημάτων. Στην πρώτη υποφάση, η διαδικασία λαμβάνει ως είσοδο την ΠΣΔ περιγραφή του API και τα μεταδεδομένα διαμόρφωσης της προσαρμογής και στη συνέχεια εφαρμόζει εξειδικευμένες τεχνικές και αλγόριθμους εξόρυξης, προκειμένου να παράγει ένα RESTful μοντέλο πόρων ως έξοδο, μαζί με τις πληροφορίες ανατροφοδότησης που μπορούν να είναι αναθεωρηθούν για να βελτιώσουν την επίδοση των τεχνικών (π.χ. την ακρίβεια της εξαγωγής, τις απαιτήσεις σε χρόνο ή χώρο, κλπ). Κατά τη διάρκεια αυτού του σταδίου, εξάγονται πόροι από την περιγραφή της υπάρχουσας διεπαφής και οργανώνονται σε ένα μοντέλο πόρων που βασίζεται στις ιδιότητες και στις σχέσεις τους. Στη συνέχεια, λαμβάνει χώρα η δεύτερη υποφάση, όπου το RESTful μοντέλο υπηρεσίας, ένα μοντέλο εξαρτήσεων των λειτουργιών της υπό προσαρμογή υπηρεσίας και ένα σύνολο ορισμένων από τον χρήστη πολιτικών σχετικά με την προσωρινή αποθήκευση και την έκθεση των πόρων, υποβάλλονται σε μια διαδικασία η οποία δημιουργεί την τελική προδιαγραφή ή οδηγία προσαρμογής. Η προδιαγραφή που συντίθεται καλύπτει διάφορες δυναμικές πτυχές μιας RESTful προσαρμογής. Επίσης, η οδηγία προσαρμογής χρησιμοποιείται στη συνέχεια ως είσοδος στην φάση του χρόνου εκτέλεσης για τη ρύθμιση ενός προσαρμογέα ο οποίος θα χειριστεί τα αιτήματα, θα τα αντιστοιχίσει σε κλήσεις στο υπάρχον δομοστοιχείο και θα παρέχει τις απαντήσεις του αφού τις μετασχηματίσει σε αποκρίσεις συμβατές με το REST (ή κάποιο επιλεγμένο συμβιβασμό). Στην επόμενη ενότητα συζητούνται περαιτέρω οι φάσεις της προσαρμογής και παρέχο-

νται περιγραφές κάθε δομοστοιχείου προσαρμογής όπως και κάθε πληροφοριακού στοιχείου που χρησιμοποιείται ως είσοδος ή έξοδος.

3.2.2 Περιγραφή δομοστοιχείων μοντέλου

Στην παρούσα υποενότητα συζητάμε σε μεγαλύτερη λεπτομέρεια τις συνιστώσες του μοντέλου διαδικασίας προσαρμογής, τα οποία περιλαμβάνονται στο Σχήμα 3.1.

Στατικά/Διαρθρωτικά ζητήματα

Κατά τη διάρκεια της φάσης του χρόνου σχεδίασης, τα πρώτα ζητήματα που αντιμετωπίζονται είναι τα στατικά ή διαρθρωτικά ζητήματα, τα οποία σχετίζονται με τον ορισμό των σημείων αλληλεπίδρασης της διεπαφής, αναγνωρίζοντας με αυτό τον τρόπο μια τοπολογία για την ανάλυση-οργάνωση της λειτουργικότητας της υπηρεσίας. Ως εκ τούτου, στις παραγράφους που ακολουθούν, συζητούνται τα βήματα και τα πληροφοριακά αντικείμενα που συμμετέχουν στο πρώτο τμήμα ή υποφάση της φάσης του χρόνου σχεδίασης.

Προδιαγραφή ΠΣΔ API. Θεωρούμε ότι η διαδικασία προσαρμογής λαμβάνει χώρα σε επίπεδο υπηρεσίας, όπου ως υπηρεσία θεωρείται ένα σύνολο μιας ή περισσοτέρων λειτουργιών, οι οποίες παρέχονται από ένα δομοστοιχείο παροχής υπηρεσιών, έτσι ώστε να πραγματοποιούν μια ή περισσότερες, καλά ορισμένες εργασίες και να παρέχουν λειτουργικότητα ή/και πληροφοριακό περιεχόμενο σε καταναλωτές της υπηρεσίας. Στο πλαίσιο αυτό, μια προδιαγραφή ενός ΠΣΔ API μπορεί να είναι οποιαδήποτε προγραμματιστική περιγραφή της διεπαφής της εφαρμογής η οποία ακολουθεί εγγενώς το ΠΣΔ υπόδειγμα, όπως ορίστηκε στο Κεφάλαιο 2, Ενότητα 2.2 (Ορισμός 2). Οι λειτουργίες των υπηρεσιών μπορούν να κληθούν ώστε να επεξεργαστούν δεδομένα εισόδου που περιλαμβάνονται στα μηνύματα κλήσης και έπειτα να παραγάγουν και να επιστρέψουν δεδομένα εξόδου. Τόσο τα δεδομένα εισόδου όσο και τα δεδομένα εξόδου παρέχονται στην υπηρεσία και επιστρέφονται από την υπηρεσία ακολουθώντας προκαθορισμένες δομές, δηλαδή, παραμέτρους εισόδου και εξόδου των οποίων οι τύποι ορίζονται, περιγράφονται και είναι γνωστοί πριν την κλήση. Για παράδειγμα, γλώσσες προδιαγραφής διεπαφών όπως η WSDL και γενικά οποιαδήποτε προτυποποιημένη γλώσσα που μπορεί να χρησιμοποιηθεί για να εκφράσει υπογραφές λειτουργιών και τύπους παραμέτρων, θα μπορούσε να

θεωρηθεί αρκετή ώστε να παρέχει μια προδιαγραφή API όπως αυτή που συζητάμε. Πηγαίνοντας ένα βήμα παραπέρα, διαφορετικά συντακτικά θα μπορούσαν να είναι αποδεκτά υπό την προϋπόθεση ότι μπορούν να μεταφραστούν σε WSDL PortType (WSDL 1.1) ή WSDL Interface (WSDL 2.0) στοιχεία, δηλαδή συντακτικά των οποίων η εκφραστικότητα θα μπορούσε να αντιστοιχιστεί με συνεπή τρόπο στη γλώσσα του W3C για την παροχή περιγραφών Web services (π.χ. Java interfaces).

Μεταδεδομένα διαμόρφωσης της προσαρμογής. Η διαδικασία της προσαρμογής θα πρέπει να επιτρέπει ποικίλα σημεία διαμόρφωσης έτσι ώστε να είναι αρκετά ευέλικτη στο να αντιμετωπίζει ειδικές απαιτήσεις χρηστών -δηλαδή διαφορετικούς συμβιβασμούς οι οποίοι απαιτούνται για να αντληθούν επιθυμητές αρχιτεκτονικές ιδιότητες στο πλαίσιο μιας ΠΣΠ προσέγγισης. Αντίθετα, το σύνολο των υποψήφιων σημείων διαμόρφωσης πρέπει να εξετάζεται ενδελεχώς έτσι ώστε τα σημεία εκείνα που τελικά επιλέγονται να μη δημιουργούν συγκρούσεις και η ερμηνεία τους να μην αφορά αλληλοεπικαλυπτόμενα θέματα σχετικά με το αποτέλεσμα της προσαρμογής. Ένα τέτοιο σύνολο μεταδεδομένων διαμόρφωσης καλύπτει κυρίως τα παρακάτω θέματα:

- Την επιλογή τεχνικών και αλγόριθμων που θα εφαρμοστούν έτσι ώστε να εκτελεστεί η αναγνώριση πόρων, σχέσεων μεταξύ των πόρων, δομών των αναπαραστάσεων, δια-υποδειγματικών συσχετίσεων στοιχείων των διεπαφών, κλπ. Για παράδειγμα, η επιλογή των αλγορίθμων και των τεχνικών εξόρυξης πληροφορίας που πρόκειται να χρησιμοποιηθούν για την εξαγωγή πόρων αποτελεί ένα από τα πιο σημαντικά σημεία διαμόρφωσης καθώς η ποιότητα του μοντέλου πόρων που εξάγεται επηρεάζει τη συνολική ποιότητα του αποτελέσματος.
- Την επιλογή του πρωτοκόλλου επικοινωνίας το οποίο θα χρησιμοποιηθεί (π.χ. HTTP 1.1) και πιθανώς τις RESTful συμβάσεις χρήσης του επιλεγμένου πρωτοκόλλου που θα πρέπει να χρησιμοποιηθούν για τον ορισμό της ΠΣΠ διεπαφής (π.χ. κάποιο συγκεκριμένο υποσύνολο των στοιχείων δεδομένων ελέγχου που παρέχονται από το πρωτόκολλο επικοινωνίας).
- Την επιλογή του μηχανισμού αναγνωριστικών ο οποίος θα χρησιμοποιηθεί για να αποδώσει αναγνωριστικά στους πόρους που εξάγονται και την επιλογή των προκαθορισμένων κανόνων ή προτύπων οι οποίοι πρέπει να ακολουθηθούν έτσι ώστε να καλυφθούν απαιτήσεις σχετικές με τη σχεδίαση των αναγνωριστικών (π.χ. γενικά URI Templates).

Το RESTful Service Model (RSM) και η διαδικασία παραγωγής του. Η διαδικασία παραγωγής του RSM (RSM Generation) λαμβάνει ως δεδομένα εισόδου τα στοιχεία που αναφέρθηκαν παραπάνω (προδιαγραφή της ΠΣΔ διεπαφής και μεταδεδομένα διαμόρφωσης) και παράγει ένα μοντέλο υπηρεσίας το οποίο ιδανικά συμμορφώνεται στον περιορισμό Ομοιόμορφης διεπαφής και ειδικά στους υπο-περιορισμούς *αναγνώριση πόρων* (*identification of resources*), *μεταχείριση πόρων μέσω αναπαραστάσεων* (*manipulation of resources through representations*) και *αυτο-περιγραφικά μηνύματα* (*self-descriptive messages*) (βλέπε Κεφάλαιο 2). Η συμμόρφωση στους περιορισμούς αυτούς εγκαθίσταται μέσω της εξαγωγής κατάλληλων στοιχείων δεδομένων του REST τα οποία συνθέτουν τη RESTful διεπαφή. Υπό αυτήν την έννοια, χρησιμοποιώντας την ορολογία που εισήχθη στο Κεφάλαιο 2 και νωρίτερα στο τρέχον κεφάλαιο, η διαδικασία παραγωγής του RSM υλοποιεί τη μέθοδο *εξαγωγής διεπαφής*. Μολαταύτα, θα πρέπει να επισημάνουμε για ακόμη μια φορά ότι, όπως έχει ήδη συζητηθεί, η διαδικασία προσαρμογής θα πρέπει να επιτρέπει την ύπαρξη διαμορφώσεων που αντικατοπτρίζουν αρχιτεκτονικούς συμβιβασμούς, οι οποίοι με τη σειρά τους οδηγούν σε μερική συμμόρφωση στους παραπάνω περιορισμούς. Το παραγόμενο RESTful Service Model περιέχει ένα σύνολο εξαχθέντων πόρων, ένα σύνολο αναγνωριστικών τα οποία αντιστοιχούν στους πόρους που έχουν αναγνωρισθεί, ένα σύνολο σχέσεων μεταξύ των πόρων, ένα σύνολο δυνατών πράξεων μεταχείρισης οι οποίες ανήκουν σε ένα προκαθορισμένο και σταθερό σύνολο πράξεων και ένα σύνολο αναπαραστάσεων για τους πόρους αυτούς. Για κάθε πόρο, αναγνωριστικό και αναπαράσταση που εξάγεται, εξάγονται επίσης αντιστοιχίες οι οποίες οδηγούν πίσω στην αρχική διαδικαστική υπηρεσία οι οποίες αποτυπώνονται στο μοντέλο υπηρεσίας. Συνεπώς, η παραγωγή του RSM υλοποιεί επίσης την *εξαγωγή αντιστοιχιών*.

Όπως συζητήθηκε ήδη, το RSM περιέχει σχέσεις μεταξύ των εξαχθέντων πόρων. Πιο συγκεκριμένα, αν και ο περιορισμός των υπερμέσων δεν αντιμετωπίζεται πλήρως στο σημείο αυτό, ορισμένοι τύποι σχέσεων, όπως σχέσεις συμπερίληψης και δομικές σχέσεις ιδιοκτησίας, εξάγονται στο στάδιο αυτό. Η κύρια συνεισφορά των σχέσεων αυτών είναι ότι μπορούν να χρησιμοποιηθούν για να ορίσουν μια ιεραρχία πόρων, καθώς επιβάλουν υπαρξιακές εξαρτήσεις μεταξύ των πόρων. Αναλόγως με τις επιλογές που έχουν εκφραστεί μέσω της διαμόρφωσης, ορισμένοι τύποι σχέσεων υπερμέσων μπορούν να αναγνωριστούν στο βήμα αυτό και μπορούν να καταγραφούν μέσω αντίστοιχων σχέσεων πόρων (π.χ. `prev` και `next` σχέσεις).

Επιπρόσθετα, το σύνολο των REST πόρων που αναγνωρίζεται κατά το στάδιο αυτό, μπορεί να επεκταθεί με *εικονικούς πόρους*, έτσι ώστε να υποστηριχθούν συγκεκριμένα μοτίβα μοντελοποίησης και αντιστοίχισης. Για παράδειγμα, οι εικονικοί πόροι μπορούν να χρησιμοποιηθούν για να μοντελοποιήσουν διαδικαστικά στιγμιότυπα κλήσεων ως πόρους με πλήρεις κύκλους ζωής, επιτρέποντας με αυτόν τον τρόπο την οντοκεντρική έκθεση των δυνατοτήτων υπηρεσιών που είναι κατεξοχήν προσανατολισμένες σε πράξεις-εντολές.

Σχετικά με τις τεχνικές και τους αλγόριθμους εξαγωγής, το σύνολο των πόρων καθώς και των σχέσεων των πόρων μπορεί να εντοπισθεί χρησιμοποιώντας συμβάσεις δομής και ονοματοδοσίας οι οποίες περιλαμβάνονται σε μια προδιαγραφή ΠΣΔ API. Επιπλέον, για να ανατεθούν πιθανές πράξεις στους πόρους που έχουν αναγνωριστεί, οι λειτουργίες του διαδικαστικού μέρους πρέπει να χαρακτηριστούν ως προς τη σημασιολογίας τους και τις ιδιότητές τους. Κάτι τέτοιο μπορεί να επιτευχθεί επίσης μέσω της ανάλυσης των ονομάτων των λειτουργιών και των υπογραφών τους και όταν αυτό συμβεί, μπορούν να προσδιοριστούν αντίστοιχες RESTful αλληλεπιδράσεις. Μια τέτοια διαδικασία αναγνώρισης ενδέχεται να καθοδηγείται από μεταδεδομένα διαμόρφωσης τα οποία προσδιορίζουν συμβάσεις ή μοτίβα χρήσης του πρωτοκόλλου επικοινωνίας που χρησιμοποιείται (π.χ. χρησιμοποίηση των μεθόδων του HTTP ως CRUD-like ρήματα, σύμβαση χρήσης του HTTP με μόνο GET-POST, κλπ.) μέσα από σύνολα προκαθορισμένων εναλλακτικών. Το RESTful μοντέλο υπηρεσίας που παράγεται από τη P2R διαδικασία προσαρμογής πρέπει επίσης να συντηρεί εσωτερικά κατάλληλες πληροφορίες αντιστοίχισης, οι οποίες απαιτούνται για να κατασκευαστούν τελικά έγκυρα μηνύματα κλήσεων στο υπάρχον back-end δομοστοιχείο. Συχνά η εξαγωγή διεπαφής και η εξαγωγή αντιστοιχιών συγχωνεύονται σε μια ενιαία διαδικασία εξαγωγής. Ωστόσο, όπως συζητήθηκε προηγουμένως στο παρόν κεφάλαιο (Ενότητα 3.1), η αναγνώριση τέτοιας πληροφορίας αντιστοίχισης μπορεί να υλοποιηθεί ανεξάρτητα από τη μέθοδο εξαγωγής των στοιχείων της διεπαφής, μέσω εφαρμογής διακριτών τεχνικών εξαγωγής οι οποίες μπορούν επίσης να υποστηρίξουν σενάρια MITM προσαρμογής. Στο περιβάλλον-πλαίσιο που παρουσιάζεται στο Κεφάλαιο 4, υλοποιούμε τη διαδικασία παραγωγής του RSM βασισμένοι στη δεύτερη, διακριτή προσέγγιση.

Μοτίβα και κανόνες για το RSM. Πέρα από τα μεταδεδομένα διαμόρφωσης, η παραγωγή του RSM πραγματώνεται μέσω της εφαρμογής ορισμένων κανόνων μετασχηματισμού και μοτίβων αντιστοίχισης τα οποία εφαρμόζονται σε στοιχεία της

αρχικής διεπαφής. Τέτοιοι κανόνες και μοτίβα καθοδηγούν και διαμορφώνουν τη διαδικασία εξαγωγής του μοντέλου πόρων, την ανάθεση σχέσεων μεταξύ πόρων και την αντιστοίχιση μεταξύ της σημασιολογίας των λειτουργιών, τις ιδιότητες και τα στοιχεία ελέγχου του πρωτοκόλλου επικοινωνίας και τη δημιουργία εικονικών πόρων, έτσι ώστε να υποστηριχθούν διάφορα επίπεδα συμμόρφωσης στους περιορισμούς του REST. Η διαδικασία παραγωγής του RSM επιλέγει μεταξύ των διαφορετικών τύπων κανόνων και μοτίβων και αποδίδει τιμές σε σημεία μεταβλητότητας. Συνεπώς οι κανόνες και τα μοτίβα για το RSM δε θα πρέπει να απαιτούνται για την εφαρμογή του σταδίου παραγωγής του RSM και έτσι θεωρούνται προαιρετικά. Ειδικότερα, οι επιλεγμένοι κανόνες, μοτίβα και τιμές μεταβλητών παρέχονται στον τελικό χρήστη μετά την εφαρμογή της παραγωγής του RSM, ο οποίος μπορεί να τα επιθεωρήσει, και πιθανώς να τα βελτιώσει και να επανεκκινήσει τη διαδικασία. Υπό αυτό το πρίσμα, οι κανόνες και τα μοτίβα για το RSM παρέχουν μια δηλωτική μέθοδο μειωμένης προσπάθειας για την ενσωμάτωση ανθρώπινων βελτιώσεων σε διαδικασίες εξαγωγής. Ορισμένες από τις προκλήσεις οι οποίες αντιμετωπίζονται από τους κανόνες και τα μοτίβα αυτά, είναι οι εξής:

- Η επιλογή και χρήση των συγκεκριμένων μοτίβων μοντελοποίησης του μοντέλου πόρων από ένα σύνολο τέτοιων προκαθορισμένων μοτίβων, η επιλογή των μοτίβων αντιστοίχισης που έχουν υψηλότερη προτεραιότητα και η επιλογή γενικών/διαρθρωτικών σχέσεων πόρων οι οποίες ενδιαφέρουν και ευρετικών που πρέπει να χρησιμοποιηθούν για την εξαγωγή των σχέσεων αυτών.
- Η επιλογή και χρήση συγκεκριμένων κανόνων για τον χαρακτηρισμό των ιδιοτήτων των λειτουργιών (π.χ. safety, idempotency, κλπ), και των ευρετικών που πρέπει να χρησιμοποιηθούν για την *κανονικοποίηση* των προθέσεων αλληλεπίδρασης, όπως αυτές αναγνωρίζονται κατά τη διαδικασία παραγωγής του RSM.

Διαδικασία αναθεώρησης/εξειδίκευσης της παραγωγής του RSM. Η διαδικασία παραγωγής του RSM θεωρείται ως η πιο σημαντική και επίπονη διαδικασία σε ό,τι αφορά στην πλήρη αυτοματοποίησή της, εξαιτίας του γεγονότος ότι προδιαγραφές του επιπέδου διεπαφής δεν παρέχουν συνήθως όλη την απαραίτητη πληροφορία για τον χαρακτηρισμό των λειτουργιών και την εξαγωγή σημαντικών πόρων. Επίσης, οι εμφανείς αποκλίσεις της μοντελοποίησης ΠΣΔ και ΠΣΠ εφαρμογών ενδέχεται ορισμένες φορές να καθιστά μια τέτοια διαδικασία παραγωγής

ως μη αποτελεσματική, ειδικά όταν αντικείμενο της προσαρμογής είναι διεπαφές προσανατολισμένες σε εντολές (π.χ. λειτουργίες μιας λέξης με γενικευμένους τύπους δεδομένων εισόδου και εξόδου). Υπό αυτήν την έννοια, ενσωματώνουμε στο μοντέλο τη συμμετοχή του χρήστη ως έναν τρόπο αναθεώρησης της διαδικασία παραγωγής μέσω εξειδίκευσης ή αναθεώρησης των τιμών που έχουν αποδοθεί σε σημεία προσαρμογής τα οποία επηρεάζουν την επιλογή των κανόνων και μοτίβων εξαγωγής και αντιστοίχισης. Με τον τρόπο αυτό, μπορεί να παραχθεί μια δραστική έκδοση του RSM η οποία περιλαμβάνει κυρίως το σύνολο των πόρων και των αναπαραστάσεών τους, ένα υποσύνολο των σχέσεων των πόρων και αντιστοιχίσεις μεταξύ πράξεων μεταχείρισης πόρων και στοιχείων αναπαραστάσεων με στοιχεία κλήσεων των back-end λειτουργιών. Στη συνέχεια τα δεδομένα που έχουν εξαχθεί επιθεωρούνται, πριν την εστίαση των προσπαθειών εξόρυξης πληροφορίας σε άλλες διαστάσεις, όπως για παράδειγμα η προδιαγραφή των δυναμικών χαρακτηριστικών της RESTful διεπαφής. Ωστόσο, θα πρέπει να σημειωθεί ότι στο προτεινόμενο μοντέλο η συμμετοχή του χρήστη θεωρείται προαιρετική, εννοώντας ότι η διαδικασία παραγωγής του RSM θα πρέπει να είναι αρκετά εξελιγμένη ώστε να είναι σε θέση να αναγνωρίζει ποια μοτίβα και κανόνες πρέπει να εφαρμόσει και πως. Μια διαδικασία P2R προσαρμογής πρέπει πάντα να παρέχει μια δυνατότητα ρητής αναθεώρησης του RSM, η οποία να μπορεί να γίνεται από τον άνθρωπο που οδηγεί τη διαδικασία προσαρμογής. Παρ' όλα αυτά, όταν ο χρήστης συμμετέχει με δηλωτικό τρόπο στη διαδικασία, θεωρούμε ότι υπάρχουν δύο βασικά πλεονεκτήματα: *α'*) απαιτείται λιγότερη προσπάθεια καθώς ο χρήστης καλείται κατά κύριο λόγο να επιλέξει μεταξύ προκαθορισμένων συνόλων που αντιστοιχούν σε σημεία προσαρμογής, αντί της χειροκίνητης σύνθεσης δεδομένων ή λογικής εξαγωγής και *β'*) είναι σύνηθες να παρατηρούνται επαναλαμβανόμενα μοτίβα χρήσης των κανόνων και των μοτίβων για το RSM στο πλαίσιο σχετικών υπηρεσιών (π.χ. υπηρεσίες που παράγονται από το ίδιο τμήμα ή οργανισμό, ή υπηρεσίες που ανήκουν στο ίδιο πεδίο). Υπό αυτήν την έννοια, παρόλο που η διαδικασία προσαρμογής θεωρείται ότι εφαρμόζεται ανά υπηρεσία, οι κανόνες και τα μοτίβα για το RSM που προσδιορίζονται μπορούν να επαναχρησιμοποιηθούν για να βελτιώσουν την ακρίβεια και την επίδοση μελλοντικών προσαρμογών παρόμοιων υπηρεσιών, ή μετεξελίξεων και μελλοντικών εκδόσεων της υπηρεσίας που προσαρμόζεται.

Δυναμικά/Συμπεριφορικά ζητήματα

Κατά τη διάρκεια του δεύτερου τμήματος της φάσης του χρόνου σχεδίασης εξετάζεται ένα σύνολο *δυναμικών* ή *συμπεριφορικών* ζητημάτων που σχετίζονται με την παροχή RESTful διεπαφών. Πιο συγκεκριμένα, στη δεύτερη αυτή υποφάση του χρόνου σχεδίασης, το μοντέλο υπηρεσίας που παρήχθη εμπλουτίζεται με επιπλέον πληροφορίες οι οποίες μοντελοποιούν και υπαγορεύουν ένα πλήθος διαστάσεων σχετικών με τη δυναμική συμπεριφορά του συστήματος που προσαρμόζεται, όπως αυτές αποτυπώνονται στις διεπαφές που συμμορφώνονται στο REST αρχιτεκτονικό στυλ. Οι δυναμικές διαστάσεις που εξετάζονται στο στάδιο αυτό αφορούν: *α*′) στον προσδιορισμό πολιτικών προσωρινής αποθήκευσης ως προς τους πόρους που εκτίθενται, *β*′) στην ενεργοποίηση των υπερμέσων ως μηχανισμό καθοδήγησης της κατάστασης της εφαρμογής και *γ*′) στη διαμόρφωση ή στην προσαρμογή της έκθεσης του παραχθέντος συνόλου πόρων (π.χ. επιλογές φιλτραρίσματος).

Στις παραγράφους που ακολουθούν συζητάμε τα δομοστοιχεία και τα πληροφοριακά αντικείμενα που συμμετέχουν στην αντιμετώπιση των παραπάνω ζητημάτων.

Ενδο-υπηρεσιακό πρωτόκολλο. Οι γλώσσες IDL και οι συνήθεις περιγραφές ΠΣΔ διεπαφών οι οποίες επικεντρώνονται στην αποτύπωση των υπογραφών των λειτουργιών, όπως οι περιγραφές σε WSDL, δεν παρέχουν συνήθως πληροφορίες σχετικά με τη σειρά και τις συνθήκες υπό τις οποίες κάθε λειτουργία της υπηρεσίας μπορεί ή πρέπει να κληθεί. Τα ενδο-υπηρεσιακά πρωτόκολλα των ΠΣΔ υπηρεσιών είναι συνήθως εννοούμενα, ή περιγράφονται σε έγγραφα φυσικής γλώσσας τα οποία συνοδεύουν την υπηρεσία. Συνεπώς, η πρόκληση προσδιορισμού συμπεριφορικών μοντέλων για υπηρεσίες, μέσω περιγραφών των υπογραφών των λειτουργιών τους έχει διερευνηθεί στο πλαίσιο ενός συνόλου ερευνητικών περιοχών, συμπεριλαμβανομένων της περιοχής αυτοματοποιημένης σύνθεσης Web services και της περιοχής αυτοματοποιημένων δοκιμών ελέγχου των Web services. Στο πλαίσιο αυτό, έχουν προταθεί μια σειρά τεχνικών με αντικείμενο την εξαγωγή μοντέλων εξαρτήσεων μεταξύ των λειτουργιών [61], [10], [23]. Στο μοντέλο διαδικασίας προσαρμογής που προτείνεται, θεωρούμε ότι τέτοιου τύπου πληροφορίες είναι διαθέσιμες είτε μέσω τεχνικών αυτόματης εξαγωγής τους, είτε παρέχονται ως προδιαγραφές. Το ενδο-υπηρεσιακό πρωτόκολλο χρησιμοποιείται για να προσδιοριστούν διμερής σχέσεις μεταξύ των λειτουργιών της υπηρεσίας που προσαρμόζεται και να εξαχθούν σχέσεις και σύνδεσμοι μεταξύ των πόρων, οι οποίες τελικές επιτρέπουν στη σύνθεση

ενός μηχανισμού για την καθοδήγηση της κατάστασης της εφαρμογής. Όταν ένας τέτοιος μηχανισμός ενεργοποιείται από το δομοστοιχείο-προσαρμογέα του χρόνου εκτέλεσης, είναι εφικτό να ενσωματωθούν στοιχεία υπερμέσων κατά τη διάρκεια των αλληλεπιδράσεων, καλύπτοντας έτσι τον περιορισμό υπερμέσων του REST. Καθώς υπάρχει μια ποικιλία μοντέλων και φορμαλισμών οι οποίες μπορούν να χρησιμοποιηθούν για να προσδιορίσουν διαδικαστικά ενδο-υπηρεσιακά πρωτόκολλα, θεωρούμε ότι η εκφραστικότητα του φορμαλισμού που θα χρησιμοποιείται κινείται στα πλαίσια της εκφραστικότητας των ακολουθιακών διαγραμμάτων της UML 2.0, σε ό,τι αφορά σε διαδικαστική σημασιολογία και τους χειριστές ελέγχου.

Πολιτικές προσωρινής αποθήκευσης. Η δυνατότητα προσωρινής αποθήκευσης αποτελεί κεντρική έννοια στο REST και στις ΠΣΠ αρχιτεκτονικές, τόσο στη θεωρία όσο και στην πράξη. Ο αντίστοιχος περιορισμός του REST ορίζει ότι οι απαντήσεις αναμένεται να υποδεικνύουν τη δυνατότητα προσωρινής αποθήκευσης των αναπαραστάσεων ή των μηνυμάτων τα οποία μεταφέρουν. Αποθηκεύοντας προσωρινά αναπαραστάσεις πόρων, βελτιώνεται η αποδοτικότητα της επικοινωνίας καθώς η επίδοση που γίνεται αντιληπτή από τον χρήστη αναβαθμίζεται, ενώ το φορτίο στην πλευρά του εξυπηρετητή μειώνεται καθώς ανατίθεται σε ενδιάμεσα δομοστοιχεία κρυφής μνήμης. Βελτιώσεις σαν και αυτή είναι κρίσιμες για τη βελτίωση των δυνατοτήτων κλιμάκωσης κατανεμημένων δικτυακών συστημάτων. Αντιθέτως, διαδικαστικά δομοστοιχεία υπηρεσιοστρεφών αρχιτεκτονικών συνήθως δεν υποδεικνύουν κατά πόσο οι απαντήσεις σε κλήσεις λειτουργιών, ή ποια τμήματα αυτών, μπορούν να αποθηκευτούν προσωρινά ή για πόσο. Επίσης, ακόμα και όταν το κάνουν, αυτό γίνεται μέσω ειδικών ανά εφαρμογή συμβάσεων ή τεχνικών σε ό,τι αφορά στο επίπεδο της διεπαφής, ή το καταφέρουν μέσω εξωτερικών μέσων (π.χ. μέσω τεκμηρίωσης της υπηρεσίας που παρέχεται σε φυσική γλώσσα). Θεωρούμε ότι μια διαδικασία P2R προσαρμογής πρέπει να είναι σε θέση να ενσωματώνει πληροφορίες που αφορούν στη δυνατότητα προσωρινής αποθήκευσης για τις αναπαραστάσεις που έχουν αναγνωριστεί και για τους τύπους αλληλεπιδράσεων, χρησιμοποιώντας πολιτικές προσωρινής αποθήκευσης που παρέχονται εξωτερικά. Τέτοιες πολιτικές προσωρινής αποθήκευσης μπορούν να παρέχονται είτε από τον χρήστη, είτε να εξάγονται μέσω τεχνικών που αναλύουν δυναμικώς παραγόμενα δεδομένα χρήσης των υπηρεσιών. Ωστόσο, καθώς οι λανθασμένες πολιτικές προσωρινή αποθήκευσης ενέχουν κινδύνους ως προς τη συνέπεια, θεωρούμε ότι οι πολιτικές που εξάγονται από κάποια τεχνική επιθεωρούνται και ενδεχομένως διορθώνονται προτού ενσωματωθούν στην προδιαγραφή της προσαρμογής που θα εφαρμοστεί στον

χρόνο εκτέλεσης. Σημειώνουμε ότι σε περίπτωση απουσίας πολιτικών προσωρινής αποθήκευσης, το αποτέλεσμα της προσαρμογής μπορεί να συνεχίζει να είναι συμμορφωμένο ως προς το REST, αφού είναι αποδεκτό για μια RESTful υπηρεσία να παρέχει αποκλειστικά αναπαραστάσεις που δεν αποθηκεύονται προσωρινά, αν και κάτι τέτοιο δεν ενδείκνυται.

Πολιτικές έκθεσης πόρων. Συχνά, ο στόχος της P2R προσαρμογής μπορεί να είναι η παροχή κάποιας τμηματικής έκθεσης της λειτουργικότητας και των πληροφοριών που είναι διαθέσιμες στο ΠΣΔ σύστημα. Με άλλα λόγια, μπορεί να είναι επιθυμητό να παρέχονται μόνο συγκεκριμένοι τύποι σεναρίων χρήσης των υπηρεσιών μέσω διεπαφών REST και μια απαίτηση σαν και αυτή είναι ιδιαίτερα σημαντική όταν τα υφιστάμενα ΠΣΔ δομοστοιχεία σχεδιάστηκαν για εσωτερική χρήση (δηλαδή για χρήση εντός των ορίων του οργανισμού στον οποίο ανήκει η υπηρεσία) και ενδέχεται να είναι εκτεταμένα ή ιδιαιτέρως λεπτομερή. Επιπλέον, ενδεχομένως να απαιτείται να εμφανίζονται επιπλέον σχέσεις μεταξύ των πόρων, από αυτές που προσδιορίζονται είτε από τη διαδικασία παραγωγής του RSM, είτε μέσω της διαδικασία αναγνώρισης υπερμέσων. Κάτι τέτοιο είναι επιθυμητό για τον εμπλουτισμό του API αλλά και για την υποστήριξη περαιτέρω απαιτήσεων. Για το λόγο αυτό, μοντελοποιούμε στο προτεινόμενο μοντέλο διαδικασίας προσαρμογής τη δυνατότητα να ενσωματώνονται πολιτικές που εξειδικεύουν την έκθεση των πόρων. Ωστόσο, θα πρέπει να διευκρινιστεί ότι μια τέτοια δυνατότητα θεωρείται προαιρετική και ότι δε θεωρείται μέρος των πυρηνικών δραστηριοτήτων προσαρμογής, αφού δεν αντιμετωπίζει ζητήματα που σχετίζονται με τους περιορισμούς του REST. Μοντελοποιείται ως μέσο υποβοήθησης των αρχιτεκτόνων για την τελική διαμόρφωση του REST API που εκτίθεται, το οποίο βεβαίως απαιτεί αντίστοιχες πληροφορίες που θα πρέπει να παρέχονται από εκείνους.

Το μοντέλο δυναμικής συμπεριφοράς και η διαδικασία παραγωγής του. Στο REST υπάρχουν ορισμένες διαστάσεις που αφορούν στη δυναμική συμπεριφορά των διεπαφών οι οποίες πρέπει να ρυθμίζονται κατάλληλα έτσι ώστε οι υπηρεσίες να προσφέρουν αυξημένη πρακτική αξία. Για παράδειγμα, τέτοιες διαστάσεις δυναμικής συμπεριφοράς επιτρέπουν να παρέχεται η λειτουργικότητα της υπηρεσίας με χρήση των υπερμέσων, δηλαδή οι καταναλωτές της υπηρεσίας να ενημερώνονται και να κατευθύνονται από την υπηρεσία σχετικά με δυνατές επόμενες αλληλεπιδράσεις κατά τον χρόνο εκτέλεσης. Συνεπώς, πρέπει να εφαρμόζονται ορισμένες μέθοδοι κατά τη διάρκεια της φάσης του χρόνου σχεδίασης ώστε να αντιμετωπίζο-

νται τα συμπεριφορικά ζητήματα και να υπάρχουν οι αντίστοιχες προβλέψεις στην παραγόμενη προδιαγραφή της προσαρμογής. Ειδικότερα, η ανάλυση η οποία εφαρμόζεται κατά τη διάρκεια του σταδίου παραγωγής του δυναμικού μοντέλου συμπεριφοράς (Dynamic Behavior Model (DBM) Generation) έχει τρεις πλευρές. Πρώτον, το ενδο-υπηρεσιακό πρωτόκολλο της εφαρμογής για την ΠΣΔ υπηρεσία υποβάλλεται σε επεξεργασία έτσι ώστε να μεταφραστεί σε όρους των πόρων που περιλαμβάνονται στο RSM, υποδεικνύοντας σχέσεις μεταξύ των πόρων οι οποίες ενδέχεται να χρειάζεται να μεταφέρονται στους καταναλωτές της υπηρεσίας κατά τον χρόνο εκτέλεσης. Δεύτερον, οι πολιτικές που σχετίζονται με την κατάσταση και τον κύκλο ζωής του πληροφοριακού περιεχομένου που εκτίθεται εξετάζονται ώστε να παρασχεθούν πληροφορίες προσωρινής αποθήκευσης στους πελάτες των υπηρεσιών. Αυτές οι πολιτικές θεωρούνται ρητές και εκφράζονται σε όρους του εξαχθέντος μοντέλου πόρων του RSM. Τρίτον, ένα σύνολο πολιτικών έκθεση πόρων τίθενται σε επεξεργασία έτσι ώστε να ενσωματωθούν στην οδηγία της προσαρμογής και να δίνεται η δυνατότητα παροχής πληροφοριών για τις απαιτήσεις πρόσβασης στην λειτουργικότητα και τις πληροφορίες που παρέχει το REST API. Το DBM που παράγεται στο στάδιο αυτό συνδυάζεται με το RSM που παρήχθη σε προηγούμενο στάδιο, για να παραχθεί η τελική προδιαγραφή της προσαρμογής.

Η δυνατότητα να αποτυπώνονται και να χρησιμοποιούνται οι πτυχές δυναμικής συμπεριφοράς μιας υπηρεσίας, όπως αυτό μπορεί να επιτευχθεί με τη δημιουργία και χρήση ενός DBM, αντιμετωπίζει αποτελεσματικά τον HATEOAS περιορισμό, ο οποίος θεωρείται κεντρικό ζήτημα για την παροχή ενός πραγματικά RESTful συστήματος. Επιπλέον, χρησιμοποιώντας ένα DBM μπορεί επίσης να αντιμετωπιστεί ο περιορισμός Κρυφής μνήμης του REST, κάτι που ενισχύει τις δυνατότητες κλιμάκωσης και βελτιώνει την επίδοση που γίνεται αντιληπτή από τον χρήστη στα RESTful συστήματα. Τέλος, επιτρέποντας τον καθορισμό σαφών πολιτικών έκθεσης των πόρων, ένα DBM μπορεί να παρέχει ευελιξία όσον αφορά στην εξειδικευμένη διαμόρφωση της REST διεπαφής που εκτίθεται και να επιτρέψει την καλύτερη ευθυγράμμιση του τελικού αποτελέσματος της διαδικασίας με τους γενικούς στόχους της προσαρμογής.

Κανόνες και μοτίβα για το DBM. Ο τρόπος με τον οποίο οι συγκεκριμένες δυναμικές διαστάσεις της συμπεριφοράς ενός συστήματος υπηρεσιών μοντελοποιούνται, καθοδηγείται από αντίστοιχους κανόνες και μοτίβα. Η συμμετοχή των κανόνων και των μοτίβων αυτών στη διαδικασία προσαρμογής ακολουθεί το παράδειγμα

των κανόνων και των μοτίβων για το RSM που συζητήθηκε παραπάνω. Η συμμετοχή του χρήστη μοντελοποιείται ξανά μέσω ενός βρόχου αναθεώρησης/εξειδίκευσης και είναι επίσης προαιρετικός.

Διαδικασία αναθεώρησης/εξειδίκευσης της παραγωγής του DBM. Σε αυτό το βήμα, ο χρήστης επιθεωρεί την έξοδο της διαδικασίας παραγωγής του DBM μαζί με το σύνολο των κανόνων και των μοτίβων που χρησιμοποιήθηκαν προκειμένου να αποδοθεί το δυναμικό μοντέλο συμπεριφοράς του συστήματος. Ο χρήστης είναι τότε σε θέση να τροποποιήσει το σύνολο αυτό, είτε αλλάζοντας τις τιμές των σημείων ρύθμισης των κανόνων, ή να αλλάξει τα προκαθορισμένα μοτίβα που είναι διαθέσιμα για κάθε διάσταση δυναμικής συμπεριφοράς. Οι ενέργειες αυτές αντανακλούν κυρίως στις πολιτικές προσωρινής αποθήκευσης και στις πολιτικές έκθεσης των πόρων. Προφανώς, μετά την αναθεώρηση/εξειδίκευση, η διαδικασία παραγωγής θα πρέπει αντίστοιχα να δημιουργεί ένα πιθανώς προσαρμοσμένο DBM το οποίο να ανταποκρίνεται στις προσδοκίες του χρήστη.

RESTful προδιαγραφή προσαρμογής. Όπως συζητήθηκε παραπάνω, το μοντέλο δυναμικής συμπεριφοράς αποτελεί τμήμα του τελικού αποτελέσματος της φάσης του χρόνου σχεδίασης. Ουσιαστικά, το RSM και το DBM αποτελούν τη RESTful προδιαγραφή της προσαρμογής. Το μεταμοντέλο για την προδιαγραφή αυτή πρέπει να είναι αρκετά εκφραστικό ώστε να καλύπτει και τις δύο κατηγορίες ζητημάτων (στατικά/διαρθρωτικά και δυναμικά/συμπεριφορικά). Η προδιαγραφή της προσαρμογής παρέχει όλες τις απαραίτητες πληροφορίες για να εκκινηθεί η φάση του χρόνου εκτέλεσης. Με άλλα λόγια, οτιδήποτε εξήχθη, αντιστοιχίστηκε, μοντελοποιήθηκε και πιθανώς αναθεωρήθηκε κατά τη διάρκεια της φάσης του χρόνου σχεδίασης θα πρέπει να περιλαμβάνεται ή να περιγράφεται στην τελική προδιαγραφή, η οποία καθορίζεται ως είσοδος στη διαμόρφωση της παράταξης του δομοστοιχείου-προσαρμογέα.

Διαμόρφωση περιβάλλοντος εκτέλεσης. Το μοντέλο διαδικασίας της προσαρμογής υποθέτει ότι υπάρχει ένα στρώμα δομοστοιχείων υποδομής τα οποία είναι σε θέση να αντιμετωπίσουν ένα εύρος τεχνικών θεμάτων, όπως η παροχή υλοποιήσεων των πρωτοκόλλων που επικοινωνίας χρησιμοποιούνται για την RESTful προσαρμογή ενός συστήματος, η αντιμετώπιση QoS απαιτήσεων, η παροχή μέσων ελέγχου και διαχείρισης, κλπ. Ο προσαρμογέας χρόνου εκτέλεσης αναμένεται να λειτουργεί πάνω από ένα τέτοιο στρώμα δομοστοιχείων υποδομής, έτσι ώστε να επιτυγχάνεται καλύτερος διαχωρισμός των μελημάτων. Παρ' όλα αυτά, μπορεί επίσης

να υλοποιηθεί ως ένα ξεχωριστό και αυτόνομο δομοστοιχείο για την υποστήριξη ανεξάρτητης διαμεσολάβησης μειωμένων απαιτήσεων (lightweight). Σε αμφότερες τις περιπτώσεις, υπάρχει ένα πλήθος των σημείων διαμόρφωσης σχετικών με τα δομοστοιχεία που εφαρμόζουν την προσαρμογή, που αφορούν σε ζητήματα υποδομής και διαχείρισης, τα οποία μοντελοποιούνται από το πληροφοριακό αντικείμενο της διαμόρφωσης του περιβάλλοντος εκτέλεσης (runtime configuration).

Έξυπνη προσαρμογή χρόνου εκτέλεσης

Σε αυτό το σημείο, η RESTful προδιαγραφή προσαρμογής τίθεται σε επεξεργασία και εφαρμόζεται η προσαρμογή υποδειγμάτων διεπαφών. Το δομοστοιχείο-προσαρμογέας είναι σε θέση να δεχθεί και να επεξεργαστεί RESTful αιτήματα, να διαχειριστεί πόρους, να αντιστοιχίσει τα αιτήματα σε back-end κλήσεις υπηρεσιών, να λάβει της απαντήσεις από τις κλήσεις υπηρεσιών και να επιστρέψει RESTful αποκρίσεις, οι οποίες περιλαμβάνουν πληροφορίες ελέγχου, πληροφορίες κατάστασης των πόρων, μεταδεδομένα πόρων και προσφερόμενες δυνατότητες υπερμέσων. Επιπλέον, ανάλογα με το εξαχθέν μοντέλο πόρων, ο προσαρμογέας μπορεί επίσης να λειτουργήσει ως εξυπηρετητής προέλευσης για εικονικούς πόρους, δηλαδή πόρους δεν αντιστοιχούν κατ' ανάγκη σε υπάρχουσες πληροφοριακές οντότητες στο ΠΣΔ σύστημα. Όσον αφορά στους περιορισμούς του REST, ο προσαρμογέας ακολουθεί και εφαρμόζει το Client-Server στυλ αλληλεπίδρασης το οποίο απαιτεί μια RESTful αρχιτεκτονική. Επίσης σέβεται τον περιορισμό Διαστρωματωμένου συστήματος, καθώς η διεπαφή που εκτίθεται παρέχεται ως ένα συμβατικό REST API, δηλαδή χωρίς περαιτέρω παραδοχές ή απαιτήσεις για την πραγμάτωση επιτυχών αλληλεπιδράσεων. Από την άποψη αυτή, ο πελάτης δε θα πρέπει να είναι σε θέση να διακρίνει αν αλληλεπιδρά με έναν παραδοσιακό εξυπηρετητή προέλευσης, ή με τον προσαρμογέα που παίζει το ρόλο ενός στοιχείου διαμεσολάβησης για την παροχή της λειτουργικότητας της υπηρεσίας. Αναφερόμαστε στην *ευφυΐα* ως ένα χαρακτηριστικό το οποίο πρέπει να χαρακτηρίζει την προσαρμογή του χρόνου εκτέλεσης, υποδηλώνοντας ότι υφίσταται ένα επίπεδο εμβάθυνσης ως προς τη λογική της διαμεσολάβησης, το οποίο απαιτείται για επιτυχή προσαρμογή. Για παράδειγμα, σε ένα ιδανικό σενάριο, οι αλληλεπιδράσεις πελάτη-εξυπηρετητή ξεκινούν από συγκεκριμένα αναγνωριστικά πόρων τα οποία αποτελούν σημεία εισόδου (entry-points). Η αλληλεπίδραση πελάτη-εξυπηρετητή πέρα των σημείων εισόδου θα πρέπει να παραδίδει τη λειτουργικότητα της υπηρεσίας μέσω της ενεργοποίησης του κατανα-

λωτή έναντι προσφερόμενων δυνατοτήτων και υπερμέσων, τα οποία παρέχονται από τον προσαρμογέα κατά τον χρόνο εκτέλεσης. Αυτή η ικανότητα εξαρτάται από την αποτίμηση συνθηκών που σχετίζονται με την κατάστασης των πόρων, με τις τιμές που περιλαμβάνονται σε αιτήματα ή με συνθήκες του περιβάλλοντος, παράλληλα με τη συνεπή εφαρμογή των πολιτικών έκθεσης.

Η διαδικασία προσαρμογής κατά τον χρόνο εκτέλεσης θεωρείται ως πλήρως αυτοματοποιημένη ως προς την παραγωγή και τη διαμόρφωση του προσαρμογέας, καθώς όλη η απαιτούμενη λογική προσαρμογής θα πρέπει να παρέχεται μέσω της τελικής προδιαγραφής της προσαρμογής.

RESTful διεπαφή υπηρεσίας ((entry-points) και περιγραφές). Το αποτέλεσμα της προσαρμογής είναι ένα σύνολο RESTful σημείων αλληλεπίδρασης τα οποία συνθέτουν την ΠΣΠ διεπαφή, πιθανόν μαζί με μια ή περισσότερες περιγραφές της διεπαφής που παράγεται και απευθύνονται σε μηχανές ή/και ανθρώπους, έτσι ώστε να υποστηριχθεί ο έλεγχος του προσαρμογέα αλλά και η ανάπτυξη εφαρμογών ΠΣΠ καταναλωτών. Μια περιγραφή θα πρέπει να περιλαμβάνει το μοντέλο πόρων, τους τύπους αναπαραστάσεων που χρησιμοποιούνται και πιθανώς προτυποποιημένη ή εξειδικευμένη σημασιολογία σχέσεων μεταξύ πόρων η οποία καθοδηγεί τη μετάβαση μεταξύ των καταστάσεων, αφήνοντας εκτός του φάσματός της οποιαδήποτε πληροφορία αφορά στην υποκείμενη ΠΣΔ υπηρεσία.

3.3 Αποτίμηση αποτελεσμάτων προσαρμογής

Το αποτέλεσμα μιας P2R διαδικασίας προσαρμογής είναι μια ΠΣΠ διεπαφή η οποία πρέπει είναι συμβατή με τις αρχές του REST και να συμμορφώνεται με τους περιορισμούς του. Κατά συνέπεια, οι μηχανισμοί που μπορούν να χρησιμοποιηθούν για την εξέταση του κατά πόσο υφίσταται μια τέτοια συμμόρφωση, είναι κρίσιμοι για την αξιολόγηση της αποτελεσματικότητας μιας διαδικασίας προσαρμογής και την αποτίμηση της ποιότητας του αποτελέσματός της. Επιπρόσθετα, η δυνατότητα αποτίμησης του επιπέδου συμμόρφωσης στους περιορισμούς του REST συστημάτων για τα οποία υπάρχει ο ισχυρισμός ότι το ακολουθούν, είναι σημαντική σε δύο ακόμα περιπτώσεις, πέρα από την P2R προσαρμογή υποδείγματος. Πρώτον, το REST είναι αρχιτεκτονικό στυλ και ως τέτοιο χρησιμοποιείται για να διευκολύνει την επικοινωνία και την αλληλοκατανόηση μεταξύ των αρχιτεκτόνων λογισμικού, των

προγραμματιστών και των σχεδιαστών, στα πλαίσια αρχιτεκτονικών συζητήσεων και αποφάσεων. Το να χαρακτηρίζονται συστήματα ως RESTful ενώ δεν είναι (κάτι που είναι σύνηθες για υφιστάμενα Web API [127]), ενδέχεται να οδηγήσει σε παρερμηνείες μεταξύ των συμμετεχόντων στη διαδικασία ανάπτυξης λογισμικού και τελικά να δημιουργήσει παρανοήσεις σχετικά με το τι πραγματικά σημαίνει το REST. Δεύτερον, το REST περιλαμβάνει ένα συγκροτημένο σύνολο περιορισμών, πράγμα που σημαίνει ότι όταν αυτοί οι περιορισμοί εφαρμόζονται συνολικά, αναμένεται να αντληθούν συγκεκριμένες ιδιότητες σε επίπεδο αρχιτεκτονικής. Πρακτικά όμως, υπάρχουν αρχιτεκτονικές αποφάσεις οι οποίες λαμβάνονται για να αντιμετωπίσουν εξειδικευμένες ως προς τις εφαρμογές απαιτήσεις, οι οποίες ενδέχεται να χαλαρώνουν τη συμμόρφωση ως προς ένα ή περισσότερους περιορισμούς, μεταβάλλοντας έτσι το σύνολο ή τη φύση των ιδιοτήτων που επάγονται. Υπό αυτό το πρίσμα, για να είμαστε σε θέση να αναγνωρίζουμε αρχιτεκτονικές παρεκκλίσεις από το REST και για να είμαστε σε θέση να τεκμηριώσουμε το πως θα μεταβληθούν συγκεκριμένες ιδιότητες της αρχιτεκτονικής, απαιτείται να ακολουθηθεί μια συστηματική προσέγγιση κατά την εξέταση της συμμόρφωσης στους περιορισμούς του REST. Βάσει των παραπάνω, καθώς αφενός η εφαρμογή του REST στο σύνολό του δεν είναι πάντοτε η βέλτιστη προσέγγιση, και αφετέρου εναλλακτικές ή συμβιβασμοί θα υπάρχουν πάντα στην πράξη, η αξιολόγηση ως προς το επίπεδο "RESTfulness" μέσω της εξέτασης της συμμόρφωσης μιας αρχιτεκτονικής στους περιορισμούς του REST μπορεί να βοηθήσει τους αρχιτέκτονες που σχεδιάζουν RESTful υπηρεσίες Ιστού να λάβουν καλύτερες αποφάσεις και να προβλέπουν και να προλαμβάνουν τις πιθανές επιπτώσεις των αποφάσεων αυτών.

Καθώς η ανάλυση συμμόρφωσης ενός συστήματος στις αρχές του REST και στους περιορισμούς του είναι σημαντική τόσο για την αξιολόγηση των αποτελεσμάτων της P2R προσαρμογής, όσο και για την κατασκευή νέων συστημάτων, εξετάζουμε στη συνέχεια δύο προσεγγίσεις και αντίστοιχα μοντέλα τα οποία μπορούν να χρησιμοποιηθούν για την αποτίμηση του βαθμού συμμόρφωσης μιας ΠΣΠ διεπαφής στο REST.

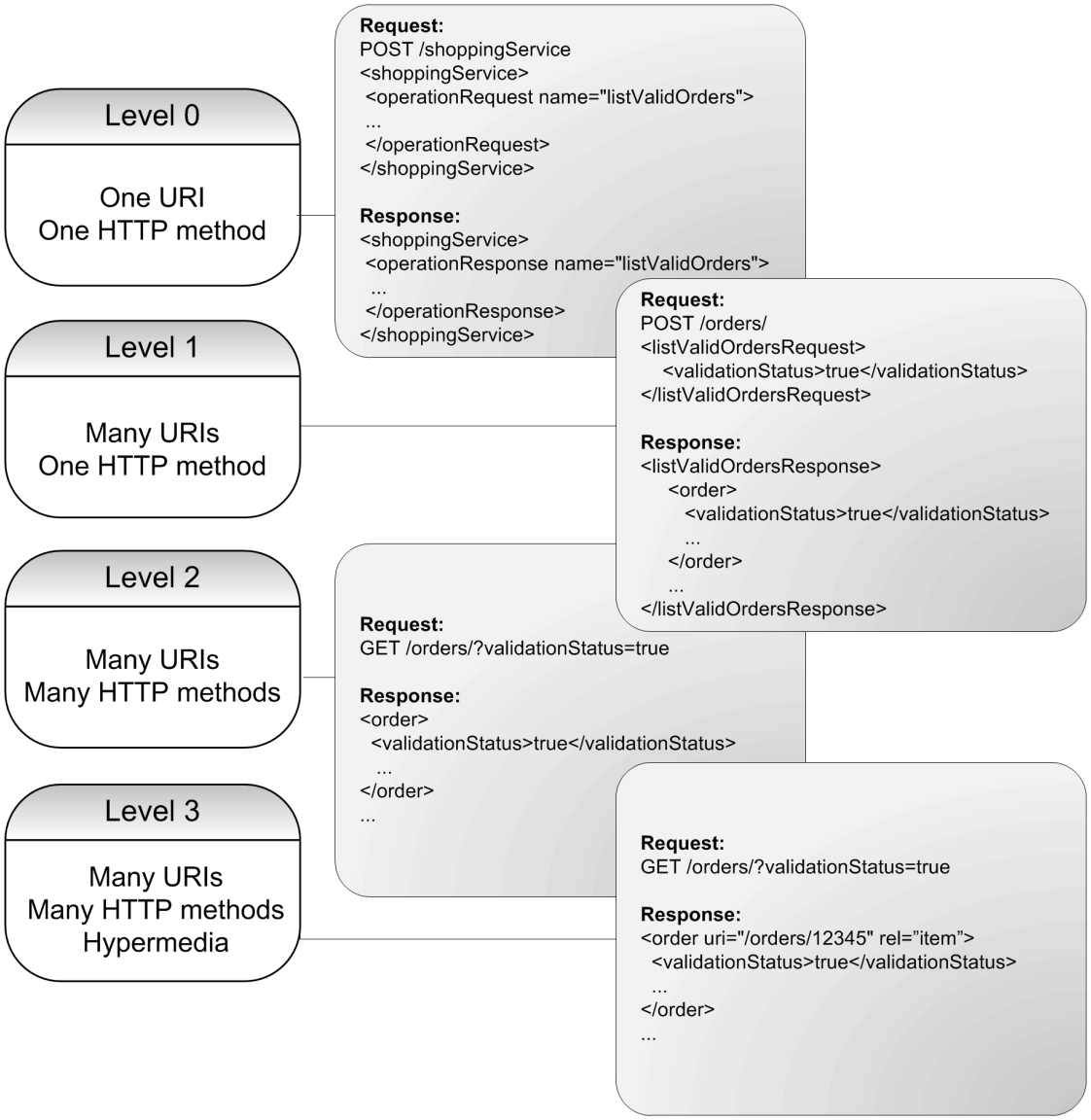
3.3.1 Ωριμότητα των REST API

Το πόσο εκτεταμένα μια αρχιτεκτονική ακολουθεί τις αρχές του REST θεωρείται ενδεικτικό της *ωριμότητάς* της ως προς το REST. Ανάλογα επίπεδα REST ωριμό-

τητας, που χρησιμοποιούνται για να χαρακτηρίσουν συστήματα υπηρεσιών που βασίζονται στο HTTP, έχουν οργανωθεί εμπειρικά σε ένα μοντέλο από τον Leonard Richardson και συγκεκριμένα στο ομώνυμο *Richardson's Maturity Model (RMM)* [54]. Το RMM ορίζει τέσσερα επίπεδα, ξεκινώντας από ΠΣΔ προσεγγίσεις σχεδίασης υπηρεσιών (Επίπεδο 0) και συνεχίζοντας με σταδιακή εφαρμογή των αρχών του REST (Επίπεδα 1 ως 3). Το Σχήμα 3.2 παρουσιάζει τα επίπεδα του RMM και απεικονίζει παραδείγματα αλληλεπιδράσεων βασισμένων στο HTTP ($L0$, $L1$, $L2$, $L3$). Ο στόχος κάθε παραδείγματος είναι να ανακαλέσει μια λίστα έγκυρων παραγγελιών οι οποίες αποτελούν ένα υποσύνολο όλων των παραγγελιών μιας υπηρεσίας αγορών.

Ξεκινώντας από το Επίπεδο 0, το HTTP πρωτόκολλο, αν και αποτελεί πρωτόκολλο επιπέδου εφαρμογής, χρησιμοποιείται ως ένας μηχανισμός μεταφοράς, κυρίως για την κλήση απομακρυσμένων διαδικασιών. Στο πρώτο αυτό επίπεδο, τα συστήματα υπηρεσιών παρέχουν συνήθως ένα μοναδικό URI ως endpoint το οποίο χρησιμοποιούν οι καταναλωτές για να στείλουν μηνύματα στον εξυπηρετητή που διαχειρίζεται το URI ώστε αυτά να τύχουν επεξεργασίας. Το παράδειγμα $L0$ στο Σχήμα 3.2 απεικονίζει ένα μοναδικό endpoint το οποίο παραλαμβάνει μια κλήση για τη λειτουργία *listValidOrders* χρησιμοποιώντας τη μέθοδο POST του HTTP και επιστρέφει μια λίστα παραγγελιών η οποία περιλαμβάνεται σε μια γενική δομή φακέλου. Τα μηνύματα συνήθως περιέχουν δομημένα δεδομένα σε μορφές όπως XML ή JSON, με ή χωρίς πρωτόκολλα όπως το SOAP για την ενθυλάκωση των δεδομένων. Τα XML-RPC, XML-JSON και WS-* Web Services πάνω από το HTTP αποτελούν τυπικά παραδείγματα τέτοιων υπηρεσιών.

Το Επίπεδο 1 αποτελεί την πρώτη μετάβαση προς μια περισσότερο ΠΣΠ προσέγγιση, μέσω της ανάλυσης των endpoint σε περισσότερα, τα οποία παρέχουν σημασιολογικά διακριτή λειτουργικότητα και δεδομένα στα πλαίσια μιας υπηρεσίας. Αυτά τα endpoint αναγνωρίζονται από και προσπελάζονται μέσω διαφορετικών URI. Με άλλα λόγια, το Επίπεδο 1 εισάγει τη χρήση διαφορετικών αναγνωριστικών πόρων ως ένα μέσο μοντελοποίησης και έκθεσης διαφορετική-διακριτής λειτουργικότητας. Ως το επίπεδο αυτό οι μέθοδοι του HTTP δεν χρησιμοποιούνται απαραίτητα σύμφωνα με την ορισμένη σημασιολογία τους και το HTTP χρησιμοποιείται κυρίως ως μια 'σήραγγα' προώθησης των αιτημάτων, αντί ως ένας τρόπος να μεταφερθεί η πρόθεση της αλληλεπίδρασης μεταξύ του πελάτη και του εξυπηρετητή. Στο Σχήμα 3.2 το παράδειγμα $L1$ δείχνει ότι στα `orders` αποδίδεται ένα ξεχωριστό URI και η κλήση απευθύνεται στο URI αυτό (συνεχίζεται η χρήση του POST ως



Σχήμα 3.2: Επίπεδα του Richardson Maturity Model με παραδείγματα

μέθοδο του HTTP για την αλληλεπίδραση). Η απάντηση περιλαμβάνει μια ένδειξη της λειτουργικότητας που κλήθηκε και μια λίστα με παραγγελίες.

Το Επίπεδο 2 εισάγει τη χρήση HTTP μεθόδων σύμφωνα με τη σημασιολογία τους ώστε να μεταφέρουν στον εξυπηρετητή (και ενδεχομένως σε ενδιάμεσα στοιχεία) τον σκοπό του αιτήματος. Γενικεύοντας αυτήν την έννοια, το Επίπεδο 2 περιλαμβάνει υπηρεσίες που χρησιμοποιούν τα δεδομένα ελέγχου του HTTP για να υποδείξουν τη σημασιολογία και τις ιδιότητες της αλληλεπίδρασης (στην έκταση που αυτό μπορεί να γίνει μέσω ενός προκαθορισμένου συνόλου δεδομένων ελέγχου). Για παράδειγμα, η μέθοδος GET χρησιμοποιείται για την ανάκληση αναπαραστάσεων και η ιδιότητα *ασφάλειας* (*safety*) (δηλαδή το γεγονός ότι δεν πρέπει να υπάρχουν ορατές παρενέργειες στον πλευρά του εξυπηρετητή εξαιτίας της αλληλεπίδρασης) γίνεται σεβαστή. Συνήθως, υπηρεσίες που βασίζονται σε CRUD πράξεις (Create, Read, Update, Delete) και δημιουργούνται για τη μεταχείριση πλούσιων σε δεδομένα πόρων, χαρακτηρίζονται από αυτό το επίπεδο ωριμότητας. Το παράδειγμα *L2* δείχνει τη χρήση ενός GET αντί ενός POST και τη χρήση ενός URI για την αναγνώριση της κατάλληλης συλλογής πόρων στην οποία επενεργεί η συγκεκριμένη αλληλεπίδραση, δηλαδή της έγκυρες παραγγελίες. Η απάντηση περιλαμβάνει μια αναπαράσταση του πόρου η οποία είναι μια λίστα από έγκυρες παραγγελίες.

Το Επίπεδο 3 αναφέρεται στον περιορισμό υπερμέσων. Οι RESTful υπηρεσίες πρέπει να παρέχουν στοιχεία υπερμέσων στα πλαίσια των αποκρίσεων ώστε να καθοδηγούν τους καταναλωτές υπηρεσιών ως προς το ποιες είναι οι πιθανές μελλοντικές αλληλεπιδράσεις, καθιστώντας ουσιαστικά τα υπερμέσα ως τη μηχανή για τις μεταβάσεις κατάστασης της εφαρμογής. Το Επίπεδο 3 του RMM απαιτεί να ανταλλάσσονται στοιχεία υπερμέσων υπό τη μορφή συνδέσμων, φορμών ή στοιχείων ελέγχου. Στο παράδειγμα *L3* που απεικονίζεται στο Σχήμα 3.2, πέρα από τη χρήση του ρήματος GET και της ορθής χρήσης ενός URI για την αναγνώριση του πόρου, η απάντηση από τον εξυπηρετητή περιλαμβάνει επίσης στοιχεία υπερμέσων τα οποία μπορούν να χρησιμοποιηθούν από τον πελάτη για να μεταφραστεί με σωστό τρόπο η απάντηση και να προγραμματιστεί η επόμενη αλληλεπίδραση.

Το RMM επιτυγχάνει να αποτυπώσει διαφορετικά επίπεδα ωριμότητας ως προς το REST συσχετίζοντας εμπειρική γνώση μοτίβων ανάπτυξης σε υπάρχοντα Web API που βασίζονται στο HTTP και αρχές του REST όπως ορίζονται στον θεωρητικό ορισμό του. Παρ' όλα αυτά, αν και οι περισσότερες από τις αρχές του REST που εξετάζονται στο RMM βασίζονται στον περιορισμό Ομοιόμορφης διεπαφής, το RMM

δεν μπορεί να χρησιμοποιηθεί για τη συστηματική μελέτη της συμμόρφωσης στον περιορισμό, καθώς ορισμένες πτυχές του δεν εξετάζονται (π.χ. αυτο-περιγραφικά μηνύματα). Συνεπώς, για υπάρχει η δυνατότητα αναφοράς απευθείας στον ορισμό του REST όταν εξετάζουμε μια RESTful διεπαφή, έχουμε προτείνει ένα εννοιολογικό περιβάλλον-πλαίσιο το οποίο επιτρέπει τη συστηματική μελέτη συμμόρφωσης στον περιορισμό Ομοιόμορφης διεπαφής [8]. Το σκεπτικό για την πρόταση ενός τέτοιου περιβάλλοντος-πλαισίου, καθώς και του ίδιου το περιβάλλοντος πλαισίου συζητούνται λεπτομερώς στις επόμενες παραγράφους.

3.3.2 Ομοιομορφία των REST API

Η δυνατότητα γενίκευσης που προσφέρει το επίπεδο διεπαφής των συστημάτων υπηρεσιών επιτρέπει την αφαίρεση και την γενίκευση των υποδειγμάτων αλληλεπίδρασης και την ανεξαρτησία από τις τεχνολογίες υλοποίησης και ακόμα και από εξειδικευμένες οπτικές του πεδίου ή της εφαρμογής. Ειδικότερα, η έννοια της ομοιομορφίας αναβαθμίζει την ορατότητα, η οποία στο συγκεκριμένο πλαίσιο ορισμού REST διεπαφών μπορεί να προσφέρει μια σημαντική μείωση στο κόστος ολοκλήρωσης των δομοστοιχείων. Επιπλέον, η αφαίρεση επιτρέψει μεγαλύτερους βαθμούς αποσύζευξης και ανεξαρτησίας μεταξύ των δομοστοιχείων, οδηγώντας σε αρχιτεκτονικές οι οποίες επιδεικνύουν σημαντικά βελτιωμένα επίπεδα ανοχής στην εξέλιξη και δυνατοτήτων επαναχρησιμοποίησης. Σύμφωνα με τον Fielding [53] ο περιορισμός της Ομοιόμορφης διεπαφής ο οποίος περιλαμβάνεται στο REST θεωρείται ως 'το κεντρικό χαρακτηριστικό που διαχωρίζει το REST από άλλα δικτυακά αρχιτεκτονικά στυλ'. Ωστόσο, ο περιορισμός αυτός αποδεικνύεται ως ένας από τους πιο δύσκολους να εφαρμοστεί και να ελεγχθεί στην πράξη και είναι πιθανότατα μια από τις έννοιες που έχει συζητηθεί περισσότερο στην κοινότητα του REST. Σύμφωνα με τον ορισμό του REST, ο περιορισμός της Ομοιόμορφης διεπαφής αναλύεται σε τέσσερις αρχιτεκτονικούς υπο-περιορισμούς οι οποίοι πρέπει να καθοδηγούν τη συμπεριφορά των δομοστοιχείων και τα χαρακτηριστικά των διεπαφών τους. Συγκεκριμένα οι υπο-περιορισμοί: *α*) *αναγνώριση πόρων* (*identification of resources*), *β*) *μεταχείριση πόρων μέσω αναπαραστάσεων* (*manipulation of resources through representations*), *γ*) *αυτο-περιγραφικά μηνύματα* (*self-descriptive messages*) και *δ*) *υπερμέσα ως ο μηχανισμός της κατάστασης της εφαρμογής* (*hypermedia as the engine of application state (HATEOAS)*). Η διατριβή του Fielding παρέχει μια συ-

ζήτηση η οποία σχετίζεται με τους παραπάνω υπο-περιορισμούς, χωρίς όμως να δίνει ρητούς ορισμούς. Συνεπώς, στη διάρκεια των χρόνων, οι περιορισμοί αυτοί αποτέλεσαν αντικείμενο πολλαπλών ερμηνειών ως προς το πως πρέπει να πραγματώνονται σε μια RESTful αρχιτεκτονική. Στο παρόν κεφάλαιο, εισάγουμε και συζητάμε ένα εννοιολογικό περιβάλλον πλαίσιο προσανατολισμένο σε στόχους το οποίο αποτελείται από τρία επίπεδα που σχετίζονται με την *αρχιτεκτονική* (*architecture*), τη *σχεδίαση* (*design*) και το *πλαίσιο πραγμάτωσης* (*instantiation context*). Στο επίπεδο αρχιτεκτονικής το περιβάλλον-πλαίσιο συμβαδίζει με τις ιδιότητες του περιορισμού Ομοιόμορφης διεπαφής όπως παρουσιάστηκαν στο [53] ενώ στο επίπεδο σχεδίασης και πραγμάτωσης παρουσιάζει μια συλλογή από στοιχεία καθοδήγησης τα οποία μπορούν να χρησιμοποιηθούν για την αξιολόγηση και την αποτίμηση του κατά πόσο ένα API βασισμένο στο REST συμμορφώνεται πραγματικά στον περιορισμό Ομοιόμορφης διεπαφής για κάποιο πεδίο εφαρμογής και παράταξης. Υπό αυτό το πρίσμα, το προτεινόμενο περιβάλλον-πλαίσιο λαμβάνει υπόψη όχι μόνο τα χαρακτηριστικά της διεπαφής τα οποία πρέπει να οριστούν για μια υπηρεσία, αλλά επίσης το πλαίσιο και το πεδίο στο οποίο χρησιμοποιείται ένα API βασισμένο στο REST. Στο περιβάλλον-πλαίσιο που προτείνεται, εστιάζουμε ειδικά και αναλύουμε το δεύτερο επίπεδο (*σχεδίαση*), το οποίο εισάγει μια συλλογή κριτηρίων αξιολόγησης συμβατότητας. Επιπρόσθετα, το περιβάλλον-πλαίσιο που προτείνεται θα μπορούσε να επιτρέψει την ανάπτυξη κοινής κατανόησης μεταξύ των ενδιαφερομένων που αλληλεπιδρούν κατά τη διάρκεια κατασκευής και συντήρησης RESTful συστημάτων, πέρα από την αποτίμηση συμμόρφωσης στον περιορισμό.

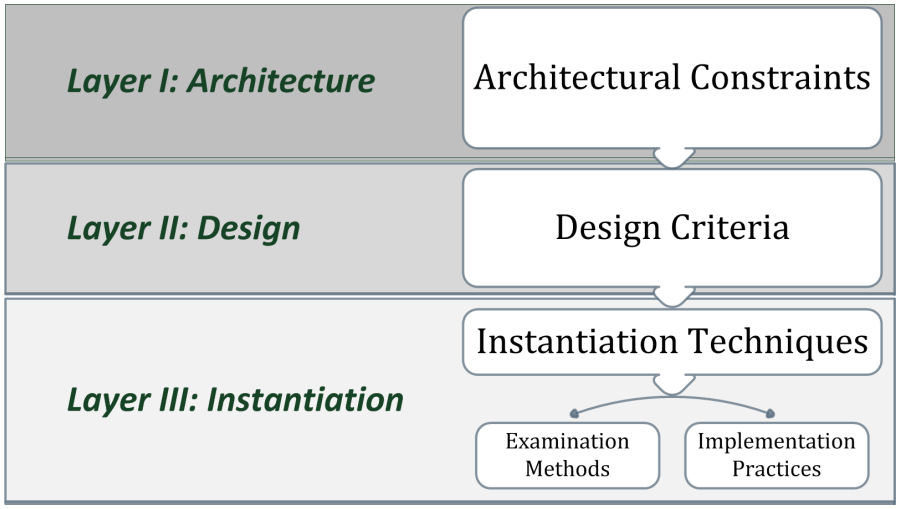
Σύμφωνα με τον ορισμό του REST [53], ο περιορισμός Ομοιόμορφης διεπαφής, μαζί με τους περιορισμούς Διαστρωματωμένου συστήματος και Κατά παραγγελία κώδικα, αποτελούν το υποσύνολο περιορισμών του REST που προστέθηκαν στην αρχική αρχιτεκτονική του Παγκόσμιου Ιστού για να δρομολογήσουν την εξέλιξή της. Ο Fielding συζητά επίσης, εν συντομία, το συμβιβασμό που πρέπει να γίνει όταν οι ορισμοί διεπαφών πρέπει να αντιστοιχιστούν σε γενικευμένες διεπαφές και αναγνωρίζει ότι αυτό θα μπορούσε να οδηγήσει σε υποβάθμιση της απόδοσης, σε σχέση με ειδικές ως προς τις εφαρμογές ανάγκες, οι οποίες θα μπορούσαν να εξυπηρετούνται καλύτερα με τον ορισμό αυθαίρετων διεπαφών. Από την άλλη πλευρά, η τυποποίηση των διεπαφών αποσυνδέει την υλοποίηση από την αλληλεπίδραση, προσθέτοντας ένα επίπεδο αφαίρεσης ή *μόνωσης* που συντηρεί τη σταθερή συμπεριφορά των αλληλεπιδρώντων δομοστοιχείων, πράγμα που αποτελεί ζήτημα ζωτικής σημασίας όταν απαιτείται μεγάλης κλίμακας ολοκλήρωση συστημάτων. Επιπλέον,

η απαίτηση για ομοιόμορφες διεπαφές των δομοστοιχείων σε μια RESTful αρχιτεκτονική είναι επίσης στενά συνδεδεμένη με την ίδια την έννοια του πόρου, όπως (ξανα)ορίζεται στο πλαίσιο του REST. Με επίκεντρο την ιδέα του πόρου ως εννοιολογικό στόχο μιας αναφοράς υπερμέσων και μακριά από τους ορισμούς μορφών του σώματος των μηνυμάτων ή των τύπων μέσων, ο περιορισμός Ομοιόμορφης διεπαφής ουσιαστικά τυποποιεί το μοντέλο αλληλεπίδρασης με την υποκείμενη σημασιολογία ενός πόρου χωρίς την ανάγκη κατανόησης αυτής καθαυτής της ειδικής ανά πόρο σημασιολογίας. Από την άποψη αυτή, οι δυνατότητες των υπηρεσιών (όπως δεδομένα και λειτουργικότητα) μπορούν να τυγχάνουν χρήσης και μεταχείρισης με τρόπο ομοιόμορφο, ενώ η πρωθύστερη γνώση που απαιτείται για τέτοιες αλληλεπιδράσεις μπορεί να ελαχιστοποιηθεί σε κοινώς αποδεκτά και σαφώς ορισμένα πληροφοριακά στοιχεία (πρωτόκολλα επικοινωνίας, μεταδεδομένα, μορφές μηνυμάτων, κλπ). Ωστόσο, όπως συζητήθηκε παραπάνω, στην πράξη και ιδίως στο πλαίσιο των υπηρεσιών βασισμένων στον Παγκόσμιο Ιστό, αποτελεί σύνηθες φαινόμενο αρχιτέκτονες (σκόπιμα ή μη) να μην εφαρμόζουν την ομοιομορφία που το REST ως αρχιτεκτονικό στυλ απαιτεί, αν και εξακολουθούν να χαρακτηρίζουν τα συστήματα υπηρεσιών που δημιουργούν ως RESTful. Το γεγονός αυτό έχει αναγνωριστεί από διάφορους ερευνητές και επαγγελματίες στο χώρο του REST και έχει επίσης οδηγήσει τον ίδιο τον Fielding να δημοσιεύσει κανόνες για ορθή σχεδίαση των REST API στο ιστολόγιό του [51], οι περισσότεροι εκ των οποίων αντανακλούν άμεσα ή έμμεσα περιορισμούς που επιβάλλονται από τον περιορισμό Ομοιόμορφης διεπαφής. Οι κανόνες αυτοί έχουν μελετηθεί και αναλύονται περαιτέρω σε σχετικές συζητήσεις από τα μέλη της κοινότητα, ενώ χρησιμοποιήθηκαν επίσης ως είσοδος για τον καθορισμό του περιβάλλοντος-πλαισίου που παρουσιάζεται στις επόμενες παραγράφους. Επιπλέον, τόσο το Richardson Maturity Model [54] που συζητήθηκε παραπάνω, όσο και ορισμένα άλλα μοντέλα εξέτασης της συμμόρφωσης στο REST [106], [5] λήφθηκαν υπόψη κατά τον προσδιορισμό του προτεινόμενου περιβάλλοντος-πλαισίου αποτίμησης του επιπέδου ομοιομορφίας.

Εννοιολογικό περιβάλλον-πλαίσιο ομοιόμορφης διεπαφής

Το *Εννοιολογικό Περιβάλλον-Πλαίσιο Ομοιόμορφης Διεπαφής* (*Uniform Interface Conceptual Framework (UICF)*) το οποίο συζητείται στην παρούσα ενότητα ακολουθεί τη δομή ενός απλοποιημένου δένδρου στόχων (goal tree) [57] το οποίο μπορεί να διαιρεθεί σε τρία εννοιολογικώς διακριτά επίπεδα: *αρχιτεκτονική*, *σχεδίαση* και

πραγμάτωση. Το UICF επεκτείνει τη σημασιολογία των κόμβων ώστε να αποτυπώσει ειδικούς, ανά επίπεδο, τύπους κόμβων. Το Σχήμα 3.3 παρουσιάζει ένα μοντέλο υψηλού αφαιρετικού επιπέδου του UICF.



Σχήμα 3.3: Αφαιρετική οπτική του Εννοιολογικού Περιβάλλοντος-Πλαισίου Ομοιόμορφης Διεπαφής

Το Επίπεδο Ι περιέχει κόμβους *αρχιτεκτονικών περιορισμών* (*architectural constraints*), το Επίπεδο ΙΙ περιέχει κόμβους *κριτηρίων σχεδίασης* (*design criteria*) και το Επίπεδο ΙΙΙ περιέχει κόμβους *τεχνικών πραγμάτωσης* (*ινσταντιατιου τεχνιχυες*), δεδομένου ενός συγκεκριμένου οικοσυστήματος-πλαισίου και ενός σαφούς εύρους της πραγματοποιούμενης ανάλυσης.

Οι αρχιτεκτονικοί περιορισμοί στο πρώτο επίπεδο έχουν μια ευθεία αναφορά στον ορισμό του REST σύμφωνα με την [53] ενώ τα κριτήρια σχεδίασης του δευτέρου επιπέδου αποτελούν πρακτικές ερμηνείες αυτών των αρχιτεκτονικών περιορισμών, αλλά με τρόπο τεχνολογικά ουδέτερο. Τα κριτήρια αυτά συχνά αναπαριστούν συμβιβασμούς ή συμβάσεις που προκύπτουν μετά από σχολαστική διερεύνηση του πως μπορούν να υλοποιηθούν αφαιρετικές αρχιτεκτονικές έννοιες ώστε να αντληθεί μια ομοιόμορφη διεπαφή χωρίς αυτό να γίνει με εξειδικευμένο τρόπο ως προς συγκεκριμένα περιβάλλοντα ή τεχνολογίες. Ειδικότερα, συνθέσαμε το επίπεδο σχεδίασης χρησιμοποιώντας ερμηνείες τις οποίες αντλήσαμε μέσω εξέτασης της βιβλιογραφίας και δημοσιεύσεων σχετικών με το REST και καταλήξαμε σε ένα σύνολο κριτηρίων τα οποία καλύπτουν ένα σημαντικό εύρος ζητημάτων που αντιμετωπίζει ένας σχεδιαστής REST συστημάτων. Το κύριο χαρακτηριστικό διαφοροποίησης των σχεδιαστι-

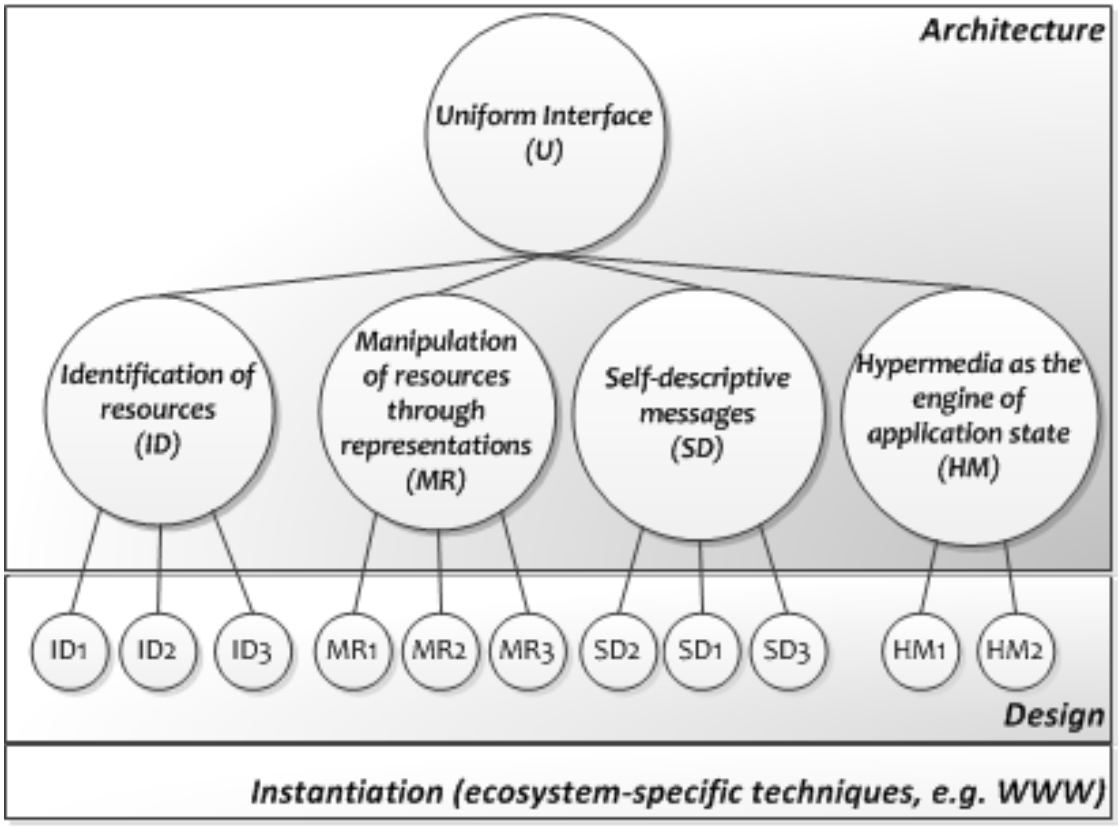
κών κριτηρίων στο δεύτερο επίπεδο του περιβάλλοντος-πλαισίου που προτείνεται, σε σύγκριση με τους αρχιτεκτονικούς περιορισμούς του πρώτου επιπέδου (βλέπε Σχήμα 3.3) είναι ότι μειώνει το επίπεδο αφαίρεσης εισάγοντας ένα σύνολο διακριτών, συμπαγών και προσανατολισμένων στην πρακτική εννοιολογικών στοιχείων, τα οποία μπορούν να χρησιμοποιηθούν για την καθοδήγηση ή την αποτίμηση μιας σχεδίασης με τεχνολογικά ουδέτερο τρόπο, αλλά ταυτόχρονα τεχνολογικά ενήμερο. Τέλος, τα κριτήρια σχεδίασης του δευτέρου επιπέδου εκδηλώνονται ως τεχνικές πραγμάτωσης στο τρίτο επίπεδο. Υπό αυτήν την έννοια, μπορούμε να θεωρήσουμε ότι το επίπεδο πραγμάτωσης συντίθεται με παραμετρικές τεχνικές πραγματοποίησης, οι οποίες μπορούν να χρησιμοποιηθούν είτε για να ελέγξουν τη συμμόρφωση μιας διεπαφής στο αρχιτεκτονικό στυλ, είτε για να καθοδηγήσουν την υλοποίηση συστημάτων ώστε να συμμορφώνονται σε αυτό.

Το περιβάλλον-πλαίσιο που προτείνεται (UICF) απεικονίζεται στο Σχήμα 3.4. Στις παραγράφους που ακολουθούν, εστιάζουμε στα κριτήρια σχεδίασης τα οποία ορίζονται στο UICF και μπορούν να χρησιμοποιηθούν για την αποτίμηση της ομοιομορφίας στο REST.

Αναγνώριση πόρων. Στο πλαίσιο του UICF, ο υπο-περιορισμός αναγνώρισης πόρων εξετάζεται ως προς δύο πτυχές: *α*) σχετικά με τη σχεδίαση του μοντέλου πόρων στο επίπεδο της διεπαφής και *β*) σχετικά με την ανάθεση αναγνωριστικών στους πόρους.

ID1: Πόροι με ευδιάκριτο πληροφοριακό περιεχόμενο. Η σχεδίαση διεπαφών στο REST και η μοντελοποίηση πόρων πρέπει να βασίζεται σε εννοιολογικές συλλήψεις των σημαντικών πόρων προσανατολισμένες στο περιεχόμενο, αντί φτωχών σε πληροφορίες, έμμεσων αναφορές πράξεων. Οι προσπελάσιμοι πόροι θα πρέπει γενικά να αποτυπώνουν ρητό και ευδιάκριτο πληροφοριακό περιεχόμενο και να παρέχουν ενδεχομένως αναφορές σε άλλους πόρους. Σύμφωνα με αυτή τη λογική, θα πρέπει να υπάρχει ένας συνεπής υποκείμενος μηχανισμός για την αντιστοίχιση του πληροφοριακού περιεχομένου που ανταλλάσσεται μεταξύ των δομοστοιχείων σε ειδικές ως προς τις εφαρμογές οντότητες, διαδικασίες, μη λειτουργικά χαρακτηριστικά ή συνδυασμούς αυτών.

ID2: Πειθαρχημένη έκθεση λειτουργικότητας. Οι πόροι πρέπει να φέρουν κατάσταση η οποία μεταφέρεται μέσω αναπαραστάσεων, μεταδίδοντας σημασιολογικά ενδιαφέρουσες και χρήσιμες πληροφορίες. Οι επιλογές μεταχείρισης της κατάστα-



Σχήμα 3.4: Εννοιολογικό Περιβάλλον-Πλαίσιο Ομοιόμορφης Διεπαφής

σης των πόρων σε συνδυασμό με τη σημασιολογία των στοιχείων αναπαράστασης θα πρέπει να είναι ο προτιμώμενος τρόπος έκθεσης της λειτουργικότητας. Μια τέτοια αντιστοίχιση απαιτεί συχνά σημαντικό επίπεδο ισομορφισμού μεταξύ της μοντελοποίησης μιας διεπαφής και της υλοποίησης. Ωστόσο, υπάρχουν περιπτώσεις στις οποίες η έκθεση της λειτουργικότητας μπορεί να πραγματοποιηθεί μέσω της αναγνώρισης πόρων οι οποίοι ανακλούν σημασιολογία στιγμιοτύπων εκτέλεσης διαδικασιών.

ID3: Οικουμενικός μηχανισμός ονοματοδοσίας. Ο μηχανισμός αναγνώρισης που χρησιμοποιείται για την απόδοση αναγνωριστικών σε πόρους πρέπει να είναι συνεπής και οικουμενικός στο εύρος του οικοσυστήματος στο οποίο το σύστημα που σχεδιάζεται υπάρχει ή θα παραταχθεί. Ένας τέτοιος μηχανισμός στοχεύει στην παροχή ενός γενικού μετα-μοντέλου για συγκεκριμένες δομές αναγνωριστικών οι οποίες τελικά θα χρησιμοποιούνται στην πράξη από το σύστημα. Επιπλέον, ο μη-

χανισμός ονοματοδοσίας θα πρέπει να επιτρέπει τη διατήρηση της αναγνώρισης και της αλληλεπίδρασης ως δύο διαφορετικά μελήματα. Τέλος, η οικουμενικότητα υποδεικνύει επίσης ότι κάθε ενδιαφερόμενο μέλος του οικοσυστήματος θεωρείται ότι έχει, ή ότι είναι σε θέση να αποκτήσει πλήρη κατανόηση του μηχανισμού.

ID4: Διαφάνεια αναγνώρισης. Τα αναγνωριστικά πρέπει να μπορούν να υποκατασταθούν οποιαδήποτε στιγμή χωρίς αυτό να έχει παρενέργειες ή επιπτώσεις στη λειτουργία των πελατών, υπό την προϋπόθεση ότι το σύνολο των entry-point αναγνωριστικών και το σύνολο των πιθανών μηχανισμών πρόσβασης παραμένουν σταθερά. Τα αναγνωριστικά δε θα πρέπει να υπερφορτώνονται με οποιαδήποτε μορφής πληροφορία (π.χ. σημασιολογία πράξεων ως πόροι, σημασιολογία τύπων πόρων, υποδήλωση σχέσεων πόρων, κλπ.), πέρα από το ελάχιστο σύνολο δεδομένων που απαιτούνται για την ανάλυση του αναγνωριστικού (δηλαδή την ένδειξη πιθανών μηχανισμών για την έμμεση αναφορά (dereference) του αναγνωριστικού). Επίσης, καμία άλλη πληροφορία πέρα από τον οικουμενικό μηχανισμό ονοματοδοσίας δε θα πρέπει να θεωρείται διαθέσιμη ως προς το πως πρέπει να γίνεται η κατασκευή και η σύνθεση μοτίβων ή προτύπων αναγνωριστικών πόρων, προ του χρόνου εκτέλεσης.

Μεταχείριση πόρων μέσω αναπαραστάσεων. Οι αναπαραστάσεις των πόρων, η μεταχείριση μέσω αυτών των αναπαραστάσεων και η σχέση μεταξύ της μεταχείρισης και της σημασιολογίας των αναπαραστάσεων εξετάζεται στο προτεινόμενο περιβάλλον-πλαίσιο μέσω των ακόλουθων κριτηρίων.

MR1: Αναγνωρίσιμοι, οικουμενικοί και εκφραστικοί τύποι μέσων. Ο τύπος των δεδομένων που ανταλλάσσονται μέσω της μεταφοράς αναπαραστάσεων πόρων πρέπει να συμμορφώνεται σε οικουμενικά, σαφώς ορισμένα σχήματα, των οποίων οι ορισμοί είναι προσπελάσιμοι σε οποιονδήποτε εντός του οικοσυστήματος. Οι περιγραφές των τύπων μέσων θα πρέπει να καθοδηγούν τα μέρη που αλληλεπιδρούν τόσο ως προς το συντακτικό όσο και ως προς το σημασιολογικό επίπεδο ανάλυσης, κατανόησης και επεξεργασίας των αναπαραστάσεων, στη βάση των στόχων της αλληλεπίδρασης. Οι περιγραφές αυτές ενδέχεται να εξελίσσονται στην πορεία του χρόνου και κάθε οικογένεια εκδόσεων θα πρέπει να αναγνωρίζεται μέσω ενός μοναδικού αναγνωριστικού.

MR2: Οικουμενικές πράξεις σχετικές με πόρους. Οι δυνατές προθέσεις που μεταδίδονται μέσω αιτημάτων ανάκτησης και μεταχείρισης θα πρέπει να κανονι-

κοποιούνται σε ένα σύνολο πράξεων οι οποίες έχουν οικουμενική σημασιολογία σε όλο το εύρος των αναγνωρίσιμων πόρων. Η σημασιολογία αυτή θα πρέπει να αντιστοιχίζεται με συνεπή τρόπο στα χρησιμοποιούμενα πρωτόκολλα επικοινωνίας, διασφαλίζοντας τον σεβασμό των αναμενόμενων ιδιοτήτων των πράξεων που ορίζονται στα πρωτόκολλα. Επιπλέον, όλοι οι παραλήπτες των αιτημάτων (δηλαδή, τα ενδιάμεσα δομοστοιχεία και οι εξυπηρετητές προέλευσης) θα πρέπει να είναι σε θέση να κατανοούν αυτές τις οικουμενικές πράξεις και να λειτουργούν βάσει των ρόλων τους.

MR3: Συνέπεια μεταχείρισης. Τα πρωτόκολλα κατά κανόνα φέρουν ήδη σημασιολογία και ιδιότητες, οι οποίες όμως είναι συνήθως αρκετά ευρείες για να καλύπτουν ένα μεγάλο φάσμα χρήσεων. Ωστόσο, οι περιγραφές διεπαφών και οι τύποι μέσων συσχετίζουν συχνά τμήματα των ορισμών τους με συγκεκριμένα πρωτόκολλα και ενδέχεται να προσδιορίζουν ρητά συγκεκριμένα μοτίβα χρήσης των πρωτοκόλλων, αποδίδοντας με τον τρόπο αυτό εξειδικευμένη σημασιολογία στις αντίστοιχες αλληλεπιδράσεις. Αυτού του είδους η εξειδικευμένη σημασιολογία ενδέχεται να παραβιάζει τη σημασιολογία που ορίζεται στο πρωτόκολλο ή τις αναμενόμενες ιδιότητες (π.χ. idempotency). Για να αποφεύγεται τέτοιου είδους σύγχυση και για να προστατεύεται η συνοχή, οι περιγραφές διεπαφών και τύπων μέσων που χρησιμοποιούνται πρέπει να είναι πλήρως συμβατές με τον ορισμό του πρωτοκόλλου επικοινωνίας και να μπορούν μόνο να συγκεκριμενοποιούν λεπτομέρειές του ή να αποσαφηνίζουν κομμάτια της σημασιολογίας του πρωτοκόλλου τα οποία ενδέχεται να είναι ασαφή όταν χρησιμοποιούνται υπό ειδικές συνθήκες, μέσω ορισμού συνεπών συμβάσεων κατά μήκος των ενδιαφερόμενων μερών του συστήματος.

Αυτο-περιγραφικά μηνύματα. Η αυτο-περιγραφικότητα των μηνυμάτων αξιολογείται χρησιμοποιώντας τα σημασιολογικώς διακριτά στοιχεία περιγραφικής επισημείωσης, τα οποία μπορούν να περιλαμβάνονται στα ανταλλασσόμενα μηνύματα και του επιπέδου συνέπειας της ερμηνεία αυτών.

SD1: Έλεγχος αλληλεπίδρασης βασισμένος στο πρωτόκολλο. Η σημασιολογία της πρόθεσης των αλληλεπιδράσεων (αιτημάτων και απαντήσεων) μπορεί να μεταδοθεί μέσω δεδομένων ελέγχου που ορίζονται στο πρωτόκολλο (π.χ. πράξεις, τύποι απαντήσεων, ενδείξεις προσωρινής αποθήκευσης). Με τον τρόπο αυτό, η συμπεριφορά των αποδεκτών μπορεί να προσανατολιστεί όπως περιγράφεται στο πρωτόκολλο επικοινωνίας. Όταν η αντιστοίχιση της λογικής της εφαρμογής στη σημασιολογία ελέγχου του πρωτοκόλλου δεν είναι δυνατή ή είναι ιδιαιτέρως δύ-

σκολο να επιτευχθεί, θα πρέπει να επανεξετάζεται η επιλογή πρωτοκόλλου. Υπό αυτήν την έννοια, δε θα πρέπει να χρησιμοποιούνται κατά παραγγελία και ειδικά ως προς την εφαρμογή δεδομένα ελέγχου έτσι ώστε να καθοδηγείται ο έλεγχος της αλληλεπίδρασης.

SD2: Οικουμενικά μεταδεδομένα πόρων και αναπαραστάσεων. Τα μεταδεδομένα παρέχουν ένα γενικό μέσο μετάδοσης στοιχείων περιγραφής για τους πόρους και τις αναπαραστάσεις. Όλοι οι αποδέκτες των μηνυμάτων θα πρέπει να είναι σε θέση να αναγνωρίζουν, να ξεχωρίζουν και να κατανοούν αυτά τα μεταδεδομένα, δεδομένου του πλαισίου της αρχιτεκτονικής. Συνήθως τέτοιες περιγραφές μεταδεδομένων περιλαμβάνονται στον ορισμό του πρωτοκόλλου επικοινωνίας. Οι περιγραφές ενδέχεται επίσης να αναφέρονται σε εξωτερικούς συνδέσμους (π.χ. το αποθετήριο τύπων μέσων του IANA) ή να παρέχουν υποδοχές επέκτασης, έτσι ώστε να μπορούν να χρησιμοποιηθούν κάθετοι και σαφώς ορισμένοι φορμαλισμοί.

SD3: Συνεπής ερμηνεία της περιγραφικής πληροφορίας. Η ερμηνεία οποιουδήποτε στοιχείου περιγραφικής επισημείωσης (π.χ. δεδομένα ελέγχου, μεταδεδομένα) τα οποία περιλαμβάνονται στα μηνύματα που ανταλλάσσονται θα πρέπει να είναι συνεπή με τη σημασιολογία που προβλέπεται από τον ορισμό τους (π.χ. η ερμηνεία των δεδομένων αναπαράστασης θα πρέπει να είναι σύμφωνη με τις αντίστοιχες δημοσιευμένες προδιαγραφές των τύπων μέσων που χρησιμοποιούνται).

Υπερμέσα ως η μηχανή της κατάστασης της εφαρμογής. Τα υπερμέσα πρέπει να αποτελούν τον αποκλειστικό τρόπο προώθησης της κατάστασης της εφαρμογής όταν απαιτείται αλληλεπίδραση. Προσδιορίζονται δύο θεμελιώδη κριτήρια για να υποδείξουν τη συμμόρφωση στον περιορισμό.

HM1: Μηνύματα εφοδιασμένα με υπερμέσα. Τα δεδομένα και τα μεταδεδομένα που ανταλλάσσονται μέσω μηνυμάτων πρέπει εμμέσως ή ρητώς να δηλώνουν την ύπαρξη αναφορών υπερμέσων. Οι αναφορές αυτές μπορούν στη συνέχεια να χρησιμοποιηθούν για την μετάβαση σε επόμενες καταστάσεις της εφαρμογής. Υπό αυτήν την έννοια, δε θα πρέπει να υπάρχει άλλος τρόπος για την κατασκευή αναγνωριστικών πόρων πέρα από τη χρησιμοποίηση της ανταλλασσόμενης πληροφορίας -με την εξαίρεση των entry-point πόρων. Ωστόσο, δεν απαιτείται όλα τα μηνύματα να φέρουν υπερμέσα εάν δεν επηρεάζουν τις αποφάσεις περί επόμενων καταστάσεων της εφαρμογής.

HM2: Εκφραστικότητα στοιχείων ελέγχου υπερμέσων. Τα στοιχεία ελέγχου υ-

περμέσων τα οποία υπάρχουν στα μηνύματα που ανταλλάσσονται πρέπει να είναι αρκούντως εκφραστικά έτσι ώστε να καθοδηγούν τις μεταβάσεις της κατάστασης της εφαρμογής και να αντιμετωπίζουν πλήρως μελήματα σχετικά με το πεδίο εφαρμογής. Τέτοια στοιχεία ελέγχου υπερμέσων μπορούν να λάβουν τις παρακάτω μορφές: *α*) στοιχεία ενσωματωμένα στα μηνύματα με σημασιολογία ορισμένη στην προδιαγραφή των τύπων μέσων, *β*) αναφορές σε επιπρόσθετα στοιχεία υπερμέσων που ορίζονται από κάθετες προδιαγραφές, ή, *γ*) προϊόντα-αποτελέσματα υπολογισμού έναντι δυναμικά παρεχόμενων δεδομένων συμπεριλαμβάνοντας εξειδικευμένες εντολές επεξεργασίας.

Στην εργασία [8] εφαρμόζουμε το UICF για την αποτίμηση της ομοιομορφίας τριών δημοφιλών API της εποχής, τα οποία αναφέρονταν από τους δημιουργούς τους ως RESTful, συγκεκριμένα το Flickr, το Twitter και το Sun Cloud API. Συνοπτικά, το Flickr API αξιολογήθηκε θετικά σε 4 από τα 12 κριτήρια σχεδίασης, το Twitter API αξιολογήθηκε θετικά σε 5 από αυτά και το Sun Cloud API αξιολογήθηκε θετικά σε 11 από τα 12 κριτήρια σχεδίασης που ορίζονται στο UICF.

Ως τελική παρατήρηση, θα θέλαμε να επισημάνουμε ότι εκτός από τη χρήση τους στην αξιολόγηση των αποτελεσμάτων της P2R προσαρμογής, ή γενικότερα των REST API, μοντέλα όπως το RMM και το UICF ενισχύουν την κατανόηση και τροφοδοτήσουν τη συζήτηση και την έρευνα γύρω από ζητήματα σχεδίασης REST διεπαφών. Στην περίπτωσή μας, έστω και αν σε αυτό το κεφάλαιο η περιγραφή του αφαιρετικού μοντέλου διαδικασία P2R προσαρμογής παρέχεται πριν από τη συζήτηση των UICF καθώς συνδέεται καλύτερα με την παρουσίαση των ζητημάτων προσαρμογής, η έρευνά μας σχετικά με τα δύο θέματα διεξήχθη με την αντίστροφη χρονολογική σειρά, καθώς αρχικά μελετήθηκε ο περιορισμός Ομοιόμορφης διεπαφής του REST, ενώ το μοντέλο διαδικασίας προσαρμογής σχεδιάστηκε αργότερα. Η διερεύνηση του REST, και ειδικότερα της ομοιομορφίας των διεπαφών, παρείχε καλύτερη κατανόηση των διαφόρων ζητημάτων που πρέπει να αντιμετωπιστούν στο πλαίσιο μιας διαδικασίας P2R προσαρμογής. Από αυτήν την άποψη, γνώσεις και ιδέες για το REST και τη RESTful σχεδίαση διεπαφών που αποκτήθηκαν ορίζοντας το UICF στην εργασία [8], χρησιμοποιήθηκαν κατά τον ορισμό του μοντέλου διαδικασίας προσαρμογής στην εργασία [9].

ΚΕΦΑΛΑΙΟ 4

ΕΞΑΓΩΓΗ ΔΙΕΠΑΦΗΣ ΠΡΟΣΑΝΑΤΟΛΙΣΜΕΝΗΣ ΣΕ ΠΟΡΟΥΣ

Στο παρόν κεφάλαιο παρουσιάζουμε ένα περιβάλλον-πλαίσιο P2R προσαρμογής υποδειγμάτων διεπαφών το οποίο χρησιμοποιεί προδιαγραφές ΠΣΔ διεπαφών και παράγει μια οδηγία/προδιαγραφή προσαρμογής η οποία ορίζει ένα μοντέλο πόρων και ένα σύνολο αντιστοιχιών των στοιχείων του μοντέλου αυτού με στοιχεία της ΠΣΔ διεπαφής. Η αναγνώριση της προσαρμοστικής λογικής εκτελείται με αυτοματοποιημένο τρόπο χωρίς τη συνδρομή του χρήστη μέσω εκτεταμένης ανάλυσης της προδιαγραφής της ΠΣΔ υπηρεσίας. Επιπλέον, υλοποιείται ένα δομοστοιχείο P2R προσαρμογής το οποίο εφαρμόζει την προσαρμογή κατά τον χρόνο εκτέλεσης και εκθέτει τις δυνατότητες της ΠΣΔ υπηρεσίας μέσω της ΠΣΠ διεπαφής που εξήχθη.

Το περιβάλλον-πλαίσιο που προτείνεται, τμήμα του οποίου παρουσιάστηκε πρώτη φορά στην εργασία [7] υλοποιεί πυρηνικές δραστηριότητες εξαγωγής προσαρμοστικής λογικής, οι οποίες αφορούν σε στατικά/διαρθρωτικά ζητήματα σχεδίασης RESTful διεπαφών, όπως περιγράφεται στο μοντέλο διαδικασίας προσαρμογής που παρουσιάστηκε στο Κεφάλαιο 3. Συγκεκριμένα, το περιβάλλον-πλαίσιο που υλοποιείται εφαρμόζει τη διαδικασία *παραγωγής του RSM* της φάσης προσαρμογής του χρόνου σχεδίασης, όπως απεικονίζεται στο Σχήμα 3.1 και περιγράφεται στο Κεφάλαιο 3, Ενότητα 3.2 ενώ παρέχει και έναν P2R προσαρμογέας, ο οποίος αντιστοιχεί στη φάση του χρόνου εκτέλεσης που περιγράφεται στο μοντέλο διαδικασίας.

Οι τεχνικές αναγνώρισης της λογικής της προσαρμογής που παρουσιάζονται στο παρόν κεφάλαιο μπορούν να θεωρηθούν ως τεχνικές εξόρυξης πληροφορίας καθώς επιχειρούν να αντλήσουν μια ΠΣΠ διεπαφή και τους κανόνες αντιστοίχισής της σε μια υφιστάμενη ΠΣΔ διεπαφή, χρησιμοποιώντας αλγορίθμους εξόρυξης πληροφοριών σε δεδομένα και ιδιότητες προδιαγραφών αναγνώσιμων από μηχανές. Οι τεχνικές αυτές αναπτύχθηκαν για να αντικαταστήσουν τη χειροκίνητη αναγνώριση της προσαρμοστικής λογικής και για να μειώσουν την εμπλοκή του χρήστη, περιορίζοντάς την, στην επισκόπηση και στην αναθεώρηση του αποτελέσματος της προσέγγισης. Επιπλέον, το δομοστοιχείο προσαρμογής που έχει υλοποιηθεί, δηλαδή ο P2R προσαρμογέας χρόνου εκτέλεσης, λειτουργεί διαφανώς ως προς την υλοποίηση της υφιστάμενης υπηρεσίας και είναι τοποθετημένο στο πλαίσιο ενός SCA runtime, ορισμένο ως ένας νέος τύπος διασύνδεσης διεπαφής (binding) για

υφιστάμενες ΠΣΔ υπηρεσίες.

Το υπόλοιπο του κεφαλαίου είναι οργανωμένο ως εξής: στην Ενότητα 4.1 ορίζουμε την εμβέλεια και τα σημεία εστίασης του περιβάλλοντος-πλαισίου προσαρμογής που προτείνεται και συνοψίζουμε τα χαρακτηριστικά των σχετικών προσεγγίσεων (οι οποίες συζητήθηκαν λεπτομερώς και συγκριτικά ως προς τη δική μας προσέγγιση στο Κεφάλαιο 2, Ενότητα 2.4.3). Επίσης, συζητάμε πως το προτεινόμενο περιβάλλον-πλαίσιο επιχειρεί να καλύψει το σύνολο απαιτήσεων για την P2R προσαρμογή που παρουσιάστηκαν στο Κεφάλαιο 3. Επιπλέον, στην ίδια ενότητα περιγράφουμε μια απλή ΠΣΔ υπηρεσία η οποία χρησιμοποιείται ως συνοδευτικό παράδειγμα σε όλες τις ενότητες του κεφαλαίου έτσι ώστε να διευκολυνθεί η παρουσίαση των μοντέλων, των αλγορίθμων και των τεχνικών που εισάγονται από την προτεινόμενη προσέγγιση. Η Ενότητα 4.2 παρουσιάζει μια τεχνική εξόρυξης πληροφορίας οδηγούμενη από μοντέλα, η οποία αναφέρεται ως *εξαγωγή πόρων* και η οποία απαιτεί ως δεδομένα εισόδου μια αναγνώσιμη από μηχανές προδιαγραφή ΠΣΔ διεπαφής βασισμένη σε μια IDL γλώσσα και επιστρέφει ως αποτέλεσμα ένα σύνολο πόρων, ιεραρχικά οργανωμένων. Στην Ενότητα 4.3, συζητάμε την έννοια των σχέσεων *βασικής ανταπόκρισης* μεταξύ λειτουργιών υπηρεσιών και τύπων πόρων και παρουσιάζουμε μια αντίστοιχη τεχνική η οποία χρησιμοποιείται για να εξάγει τέτοιες ανταποκρίσεις, καθώς οι βασικές ανταποκρίσεις χρησιμοποιούνται τόσο στην εξαγωγή αναπαραστάσεων πόρων όσο και αντιστοιχιών μεταξύ των διεπαφών. Έπειτα, στην Ενότητα 4.6 παρουσιάζεται ο τρόπος με τον οποίο πραγματώνεται η φάση προσαρμογής του χρόνου εκτέλεσης, παραθέτοντας τα χαρακτηριστικά του P2R προσαρμογέα που έχει υλοποιηθεί. Τελικά, συζητούνται στην Ενότητα 4.7 ένα σύνολο τεχνικών και τεχνολογικών πτυχών που σχετίζονται με την υλοποίηση του προτεινόμενου περιβάλλοντος-πλαισίου.

4.1 Από ΠΣΔ σε ΠΣΠ διεπαφές

Όπως συζητήθηκε στο Κεφάλαιο 1, η προσαρμογή υποδειγμάτων διεπαφών υπηρεσιών αποτελεί χρήσιμο εργαλείο σε ένα πλήθος πρακτικών σεναρίων και περιβαλλόντων. Συνοπτικά, οι πάροχοι υπηρεσιών οδηγούνται συχνά στην εκ νέου υλοποίηση λειτουργικότητας λόγω αρχιτεκτονικών ή τεχνολογικών αναντιστοιχιών, έτσι ώστε να επιτρέψουν τους καταναλωτές των υπηρεσιών τους να έχουν πρόσβαση στις δυνατότητες που προσφέρουν μέσω εναλλακτικών υποδειγμάτων διεπαφών. Παρ' όλα

αυτά, η επανα-υλοποίηση υπάρχουσας λειτουργικότητας και η συντήρηση δυϊκών υλοποιήσεων ενέχουν σημαντικά ρίσκα και κόστη για τους παρόχους υπηρεσιών. Την ίδια στιγμή, εκθέτοντας μια υπηρεσία μέσω ενός μόνο υποδείγματος και η δέσμευση σε αυτό το υπόδειγμα, ενδέχεται να οδηγήσει σε περιορισμένη υιοθέτηση και επιτυχία για την υπηρεσία, ανεξαρτήτως της ποιότητας και του εύρους των δυνατοτήτων που παρέχει.

Υπό αυτό το πρίσμα, η προσαρμογή υποδειγμάτων διεπαφών ανέκυψε ως μια αποδοτική και οικονομική λύση στην πρόκληση διαθεσιμότητας πολλαπλών υποδειγμάτων διεπαφών καθώς επιτρέπει την αποσύζευξη της ανάπτυξης και συντήρησης λειτουργικότητας από την έκθεση δυνατοτήτων υπηρεσιών και τη μοντελοποίηση διεπαφών. Συνεπώς, η προσαρμογή υποδειγμάτων μπορεί να συνεισφέρει σημαντική αξία σε υπηρεσιοστρεφείς αρχιτεκτονικές, ειδικά μέσω περιβαλλόντων όπου η υλοποίηση δομοστοιχείων και η έκθεση υπηρεσιών θεωρούνται διακριτά ζητήματα. Ένα τέτοιο περιβάλλον είναι το μοντέλο σύνθεσης υπηρεσιών Service Component Architecture (SCA) [43], το οποίο χρησιμοποιείται για τον ορισμό και τη σύνθεση υπηρεσιοστρεφών δομοστοιχείων.

Τέλος, όπως παρουσιάστηκε στο Κεφάλαιο 2, οι περισσότερες από τις προσεγγίσεις προσαρμογής υποδειγμάτων που έχουν προταθεί στη βιβλιογραφία εξετάζουν την P2R κατεύθυνση της πρόκλησης προσαρμογής υποδείγματος. Η τάση αυτή αποδίδεται στο γεγονός ότι ο προσανατολισμός σε διαδικασίες προϋπήρχε του προσανατολισμού σε πόρους ως υπόδειγμα σχεδίασης διεπαφών.

4.1.1 Αναγνώριση προσαρμοστικής λογικής

Αν και η εφαρμογή της προσαρμοστικής λογικής αποτελεί εργασία η οποία μπορεί εύκολα να αυτοματοποιηθεί από ένα προσαρμογέα χρόνου εκτέλεσης, η αναγνώριση της προσαρμοστικής λογικής η οποία περιλαμβάνει την αναγνώριση στοιχείων διεπαφών ΠΣΠ και στοιχείων δια-υποδειγματικών αντιστοιχιών, θεωρείται αρκετά πιο απαιτητική εργασία και αναμένεται να απαιτεί σημαντική προσπάθεια από την πλευρά του ανθρώπου που οδηγεί την προσαρμογή.

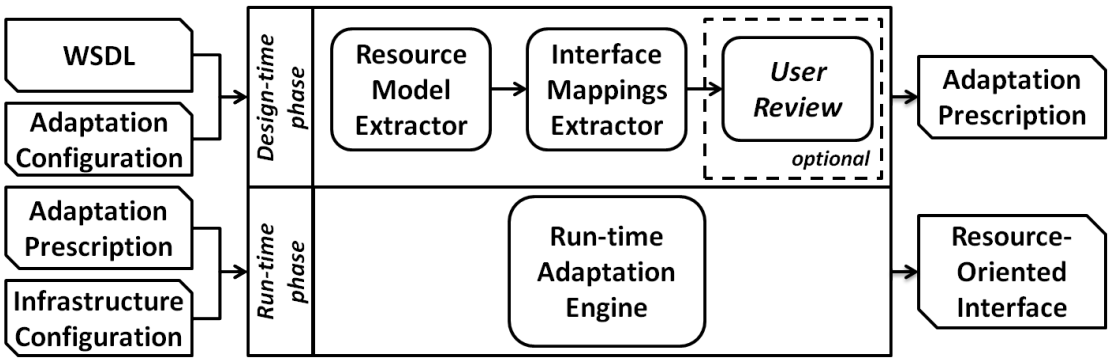
Υπό αυτή την έννοια, ένα πλήθος εργαλείων P2R προσαρμογής έχουν σχεδιαστεί για να βοηθήσουν έμπειρους χρήστες στην αναγνώριση της προσαρμοστικής λογικής. Πιο συγκεκριμένα, τέτοια εργαλεία ορίζονται συνήθως έτσι ώστε *α*) να

εφαρμόζουν μια συγκεκριμένη διαδικασία βήμα προς βήμα (π.χ. μέσω ενός γραφικού περιβάλλοντος αλληλεπίδρασης οδηγού (wizard)) και *β*) να βοηθά στη μοντελοποίηση, στην επεξεργασία και στην αποθήκευση των πληροφοριακών στοιχείων και των προδιαγραφών της προσαρμοστικής λογικής. Ωστόσο, η ουσιαστική αναγνώριση των κατάλληλων μοντέλων πόρων και πιθανώς των αντιστοιχίσεων στην ΠΣΔ διεπαφή-στόχο, αποτελούν εργασίες που εκτελούνται ιδιοχείρως από τους έμπειρους χρήστες. Ως εκ τούτου, ένας έμπειρος χρήστης θα πρέπει και πάλι να αναλύσει πηγές πληροφοριών ως προς τις υφιστάμενες διαδικαστικές υπηρεσίες, να τις συνδυάσει με τις δικές του γνώσεις, εμπειρία και κρίση, έτσι ώστε να προσδιορίσει τα διάφορα τμήματα που συνθέτουν μια κατάλληλη προδιαγραφή προσαρμογής. Ορισμένα P2R περιβάλλοντα-πλαίσια προσαρμογής πηγαίνουν ένα βήμα παραπέρα και εκτός από το να εφαρμόζουν μια συγκεκριμένη διαδικασία, παρέχουν επίσης προτάσεις-συστάσεις ως προς πιθανά στοιχεία προσαρμογής για κάποιο υποσύνολο των βημάτων της διαδικασία, έτσι ώστε να βοηθήσουν τον χρήστη στην εκτέλεσή της προσαρμογής. Όπως συζητείται στο Κεφάλαιο 2, Ενότητα 2.4.3, οι P2R προσεγγίσεις προσαρμογής που προτείνονται στη βιβλιογραφία επιδεικνύουν ποικίλα επίπεδα αυτοματοποίησης, ξεκινώντας από απλή υποστήριξη του μη-μηχανικού προσδιορισμού της προσαρμοστικής λογικής όπως συζητείται παραπάνω, μέχρι την εξόρυξη προδιαγραφών για την αναγνώριση στοιχείων REST διεπαφών και πιθανώς, αντιστοιχίσεων σε στοιχεία της διαδικαστικής διεπαφής. Μολαταύτα, αποτελεί κοινό χαρακτηριστικό των προσεγγίσεων αυτών να απαιτούν εκτεταμένες πηγές ή σύνθετα μοντέλα και προδιαγραφές της υφιστάμενης διεπαφής. Απαιτήσεις όπως οι προηγούμενες ενδέχεται να επιφορτίσουν μια προσέγγιση προσαρμογής με σημαντική προσπάθεια ή και να έχουν μη αμελητέα υπολογιστική επιβάρυνση, περιορίζοντας έτσι τη χρηστικότητα της προσέγγισης,

Το περιβάλλον-πλαίσιο προσαρμογής υποδείγματος που προτείνεται στο παρόν κεφάλαιο, επιχειρεί να αυτοματοποιήσει την αναγνώριση της P2R προσαρμοστικής λογικής, μέσω χαμηλών απαιτήσεων και ανεξάρτητων ως προς την υλοποίηση τεχνικών, καλύπτοντας τις σχετικές απαιτήσεις που συζητούνται στο Κεφάλαιο 3. Επίσης, οι τεχνικές που επινοήθηκαν, αναπτύσσονται στο πλαίσιο μιας συγκεκριμένης παρατήρησης-υπόθεσης σχετικά με τη σύνταξη των ονομάτων των λειτουργιών, η οποία διερευνήθηκε και επικυρώθηκε στατιστικά πριν την ανάπτυξη της προσέγγισης, χρησιμοποιώντας ένα σημαντικό αριθμό λειτουργιών ΠΣΔ υπηρεσιών.

4.1.2 Περίγραμμα περιβάλλοντος-πλαισίου

Η διαδικασία που υλοποιείται από το P2R περιβάλλον-πλαίσιο προσαρμογής που προτείνεται στο παρόν κεφάλαιο είναι συνεπής ως προς το αφαιρετικό μοντέλο διαδικασίας που παρουσιάστηκε στο Κεφάλαιο 3. Το Σχήμα 4.1 παρέχει μια υψηλού αφαιρετικού επιπέδου οπτική στο περιβάλλον-πλαίσιο που έχει υλοποιηθεί.



Σχήμα 4.1: Περιβάλλον-πλαίσιο εξαγωγής ΠΣΠ διεπαφής

Η υλοποίηση της φάσης του χρόνου σχεδίασης ξεκινά με τη διαδικασία εξαγωγής του μοντέλου πόρων, όπου η περιγραφή της υπηρεσίας που δίνεται ως είσοδος αναλύεται ώστε να παραχθεί ένα μοντέλο τύπων πόρων, σχέσεων μεταξύ αυτών των τύπων πόρων και αναπαραστάσεων πόρων. Το βήμα αυτό υλοποιείται από το δομοστοιχείο *Resource Model Extractor* του περιβάλλοντος-πλαισίου το οποίο καταναλώνει μια WSDL προδιαγραφή της ΠΣΔ υπηρεσίας ως την de facto περιγραφή ΠΣΔ υπηρεσιών που βασίζεται σε IDL, όπως και ένα αρχείο διαμόρφωσης της προσαρμογής. Η εξαγωγή του μοντέλου πόρων αναλύεται σε δυο βήματα, το βήμα εξαγωγής πόρων και το βήμα εξαγωγής αναπαραστάσεων και παράγει ένα ενδιάμεσο αρχείο προδιαγραφής της προσαρμογής το οποίο περιλαμβάνει το εξαχθέν μοντέλο πόρων. Στη συνέχεια, εφαρμόζεται μια μέθοδος εξαγωγής αντιστοιχιών που υλοποιείται από το δομοστοιχείο *Interface Mappings Extractor*, έτσι ώστε να αναγνωριστούν κανόνες αντιστοίχισης μεταξύ στοιχείων της ΠΣΔ διεπαφής και του εξαχθέντος μοντέλου πόρων. Το ενδιάμεσο αρχείο προδιαγραφής της προσαρμογής εμπλουτίζεται με τις εξαχθείσες αντιστοιχίες έτσι ώστε να παραχθεί η *οδηγία της προσαρμογής* (*adaptation prescription*), η οποία αποτελεί προϊόν της φάσης του χρόνου σχεδίασης της διαδικασία προσαρμογής. Προαιρετικά, ο χρήστης που οδηγεί την προσαρμογή μπορεί να επιλέξει να ελέγξει και να αναθεωρήσει την

εξαχθείσα οδηγία εφόσον απαιτείται κάτι τέτοιο.

Όπως συζητήθηκε προηγουμένως, η υλοποίηση της φάσης του χρόνου εκτέλεσης απαιτεί ως είσοδο την οδηγία της προσαρμογής, μαζί με ένα αρχείο διαμόρφωσης της υποδομής το οποίο προσδιορίζει παράγοντες του περιβάλλοντος εκτέλεσης του P2R προσαρμογέα. Ειδικότερα, η προσαρμοστική λογική εφαρμόζεται από μια γενικευμένη μηχανή προσαρμογής (*Runtime Adaptation Engine* στο Σχήμα 4.1), η οποία έχει υλοποιηθεί στα πλαίσια μιας SCA διασύνδεσης και σχηματίζει το δομοστοιχείο P2R προσαρμογής κατά τον χρόνον εκτέλεσης.

4.1.3 Συνοδευτικό παράδειγμα

Για την καλύτερη επίδειξη των τεχνικών και των αλγορίθμων του περιβάλλοντος-πλαισίου που παρουσιάζεται στο παρόν κεφάλαιο, χρησιμοποιούμε ένα συνοδευτικό παράδειγμα μιας απλοποιημένης, αλλά ρεαλιστική ΠΣΔ διεπαφής, ενός συστήματος διαχείρισης παραγγελιών, της υπηρεσίας *SimpleOMS*. Συνοπτικά, η υπηρεσία SimpleOMS περιέχει λειτουργίες οι οποίες μπορούν να χρησιμοποιηθούν ώστε κανείς να ψάξει έναν κατάλογο προϊόντων προς αγορά (*searchCatalog*), να δημιουργήσει και να διαχειριστεί παραγγελίες (*createOrder*, *getOrder*, *removeOrder*, *submitOrder*), να προσθέσει και να αφαιρέσει προϊόντα στις παραγγελίες αυτές (*addOrderItem*, *removeOrderItem*), να ανακτήσει τις υποβληθείσες παραγγελίες του (*getSubmittedOrders*) και να πληρώσει παραγγελίες (*checkout*). Επίσης, η υπηρεσία SimpleOMS υποστηρίζει ανεξάρτητη αποστολή των προϊόντων μιας παραγγελίας (π.χ. μόλις αυτά γίνουν διαθέσιμα, ή αποστελλόμενα μέσω διαφορετικών κατασκευαστών/προμηθευτών). Συνεπώς, χρησιμοποιώντας την υπηρεσία, ένας χρήστης μπορεί να ανακτήσει την κατάσταση αποστολής ενός προϊόντος (*getOrderItemShippingStatus*) όπως και της κατάστασης αποστολής συνολικά της παραγγελίας (δηλαδή του αν έχει αποσταλεί τμηματικά ή πλήρως) (*getOrderShippingStatus*). Και οι 11 λειτουργίες του SimpleOMS με τις υπογραφές τους απαριθμούνται στον Πίνακα 4.1. Επίσης, η WSDL περιγραφή του SimpleOMS παρέχεται στο Παράρτημα Α΄.

Πίνακας 4.1: Υπογραφές λειτουργιών του *SimpleOMS*

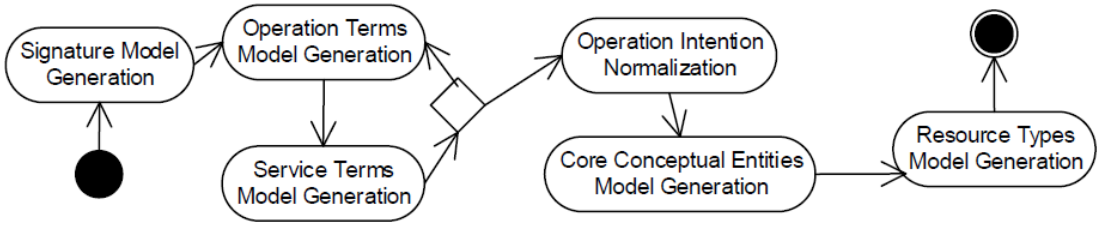
Λειτουργία	Είσοδος	Έξοδος
createOrder	auth:Auth, order:Order	result:string
getOrder	auth:Auth, orderId:string	result:Order
removeOrder	auth:Auth, orderId:string	-
submitOrder	auth:Auth, orderId:string	-
addOrderItem	auth:Auth, orderId:string, item:OrderItem	result:string
removeOrderItem	auth:Auth, orderId:string, itemId:string	-
getOrder ShippingStatus	auth:Auth, orderId:string	result:ShippingStatus
checkout	auth:Auth, orderId:string, payment:Payment	-
searchCatalog	auth:Auth, query:string	result:ArrayOfCatalogItem
getSubmittedOrders	auth:Auth	result:ArrayOfOrder
getOrderItem ShippingStatus	auth:Auth, orderId:string, itemId:string	result:ShippingStatus

4.2 Εξαγωγή πόρων

Όπως απεικονίζεται στο Σχήμα 4.1, το δομοστοιχείο Resource Model Extractor εφαρμόζει το πρώτο βήμα της προσαρμογής και υλοποιεί τη διαδικασία εξαγωγής πόρων, η οποία χρησιμοποιεί τις δοθείσες WSDL προδιαγραφές της διαδικαστικής διεπαφής για να παραγάγει ένα μοντέλο πόρων. Ωστόσο, εκτός από τον βασικό του ρόλο στην προσαρμογή υποδειγμάτων διεπαφών, η εξαγωγή πόρων μπορεί επίσης να χρησιμοποιηθεί ανεξάρτητα σε άλλα σενάρια. Για παράδειγμα, η εξαγωγή πόρων μπορεί να παρέχει ένα προβάδισμα κατά τη μετάπτωση υφιστάμενης λειτουργικότητας σε νέες υλοποιήσεις. Ακόμα και όταν η τελική υλοποίηση περιέχει επεκτάσεις ή αναδιαρθρώσεις της εκτιθέμενης λειτουργικότητας, η διαθεσιμότητα μιας οπτικής του συστήματος που μεταπίπτει η οποία μου παράγεται αυτόματα και η οποία μπορεί να αντληθεί νωρίς στη διάρκεια μιας διαδικασίας μετάπτωσης, μπορεί να βελτιώσει τη συνολική παραγωγικότητα (π.χ. διευκολύνοντας την top-down ανάπτυξη), όπως και να παρέχει μια καλύτερη εικόνα των λεπτομερειών των πιθανών τύπων πόρων που πρέπει να εξεταστούν, τόσο στους αρχιτέκτονες όσο και στους προγραμματιστές. Τέλος, καθώς η εξαγωγή πόρων παρέχει μια ανάλυση της εκτιθέμενης λειτουργικότητας μιας υπηρεσίας κυρίως μέσω πράξεων χαμηλού αφαιρετικού επιπέδου (π.χ. CRUD πράξεις) έναντι πληροφοριακών οντοτήτων, μπορεί να χρησιμοποιηθεί στις περιοχές ταξινόμησης υπηρεσιών, εντοπισμού υπηρεσιών, και σύνθεσης υπηρεσιών, στις οποίες η αναγνώριση των υποκείμενων εννοιολογικών οντοτήτων μιας διεπαφής ενός Web service αποτελεί μείζονα πρόκληση.

Η προσέγγιση εξαγωγής πόρων που προτείνεται στο παρόν κεφάλαιο χρησιμοποιεί τεχνικές επεξεργασίας φυσικής γλώσσας (Natural Language Processing (NLP)) για να αναλύσει στοιχεία περιγραφών διαδικαστικών διεπαφών και να παραγάγει ενδιάμεσα μοντέλα, στα οποία αναφερόμαστε ως *Μοντέλα Όρων* (*Term Models*). Αυτά τα Μοντέλα Όρων αναλύονται ώστε σταδιακά, χρησιμοποιώντας τεχνικές ανάλυσης γράφων και μετασχηματισμούς μοντέλων να σχηματίσουν ένα *Μοντέλο Πυρηνικών Εννοιολογικών Οντοτήτων* (*Core Conceptual Entities Model*) και τελικά ένα *Μοντέλο Τύπων Πόρων* (*Resource Types Model*), το οποίο αποτελεί τη βάση της ΠΣΠ διεπαφής. Το Μοντέλο Τύπων Πόρων μπορεί να χρησιμοποιηθεί στη συνέχεια ώστε να παραγάγει τον σκελετό μιας WADL προδιαγραφής [64], κάτι που δείχνεται στην εργασία [7].

4.2.1 Περίγραμμα διαδικασίας εξαγωγής πόρων



Σχήμα 4.2: Σύνοψη διαδικασίας εξαγωγής πόρων

Η διαδικασία εξαγωγής πόρων είναι μια διαδικασία πολλαπλών βημάτων η οποία καταναλώνει ένα WSDL έγγραφο και παράγει ένα μοντέλο πιθανών τύπων πόρων μαζί με σχέσεις συγκράτησης (containment relations) μεταξύ των τύπων αυτών. Η προτεινόμενη προσέγγιση βασίζεται στις αρχές της *Μηχανικής Οδηγούμενης από Μοντέλα* (*Model-Driven Engineering (MDE)*) και περιλαμβάνει βήματα τα οποία σταδιακά μετασχηματίζουν στοιχεία και πληροφορίες που περιέχονται στα μοντέλα προδιαγραφής ΠΣΔ διεπαφών σε στοιχεία μοντέλων πόρων. Το παραγόμενο μοντέλο πόρων αποτυπώνει όχι μόνο τους τύπους των πόρων αλλά και μια ιεραρχική διάταξη μεταξύ τους και παρέχει κατ' αυτόν τον τρόπο RESTful σημεία αλληλεπίδρασης με την υπηρεσία, εφόσον οι τύποι πόρων συνδυαστούν με κατάλληλες πράξεις (π.χ. μέσω HTTP μεθόδων). Το Σχήμα 4.2 απεικονίζει μια υψηλού αφαιρετικού επιπέδου οπτική της διαδικασία εξαγωγής πόρων η οποία αναλύεται σε πέντε βασικά βήματα-στάδια κατά τα οποία παράγονται αντίστοιχα μοντέλα.

Βήμα 1: Παραγωγή Μοντέλων Υπογραφών: Τα *Μοντέλα Υπογραφών* (*Signature Models*) εισάγονται ως ένα μέσο για την αναπαράσταση ΠΣΔ διεπαφών υπηρεσιών υπό μια κανονικοποιημένη μορφή, αποσυνδέοντας έτσι τη διαδικασία της ανάλυσης από συγκεκριμένες γλώσσες προδιαγραφής διεπαφών που χρησιμοποιούνται για την περιγραφή υπηρεσιών. Επίσης, το βήμα αυτό καθιστά την προσέγγιση εύκολα επεκτάσιμη και για άλλες IDL γλώσσες πέραν της WSDL.

Βήμα 2: Παραγωγή Μοντέλων 'Ορων Λειτουργιών και Μοντέλου 'Ορων Υπηρεσίας: Αφού δημιουργηθούν τα Μοντέλα Υπογραφών χρησιμοποιείται ένα σύνολο τεχνικών επεξεργασίας φυσικής γλώσσας και αλγορίθμων εξόρυξης πληροφορίας έτσι ώστε: *α'*) να αναγνωριστούν σημαντικοί όροι που υπάρχουν στα ονόματα των

λειτουργιών μιας υπηρεσίας, *β'*) να χαρακτηριστούν οι όροι που έχουν αναγνωριστεί βάσει του ρόλου τους σε κάθε λειτουργία και *γ'*) να αναγνωριστούν σχέσεις μεταξύ των όρων ως ένα πρώτο βήμα για τη δημιουργία ιεραρχιών. Το βήμα αυτό στοχεύει στον μετασχηματισμό των χαμηλής αφαίρεσης Μοντέλων Υπογραφών, τα οποία περιγράφουν τη διεπαφή μιας υπηρεσίας, σε υψηλότερου αφαιρετικού επιπέδου *Μοντέλα Όρων Λειτουργιών* (*Operation Terms Model* (*OTM*)) τα οποία περιέχουν όρους συγκεκριμένων τύπων και σχέσεις μεταξύ τους, αναπαριστώντας αφαιρετικές όψεις της σημασιολογίας κάθε λειτουργίας, όπως αυτές μπορούν να προσδιοριστούν μέσω του ονόματος της κάθε λειτουργίας. Οι πληροφορίες που αντλούνται από κάθε Μοντέλο Όρων Λειτουργιών για κάποια δεδομένη υπηρεσία συνδυάζονται στη συνέχεια έτσι ώστε να παραχθεί ένα μοναδικό, αθροιστικό και περισσότερο αφαιρετικό μοντέλο που αναφέρεται σε όλο το εύρος της υπηρεσίας, το *Μοντέλο Όρων Υπηρεσίας* (*Service Terms Model* (*STM*)).

Βήματα 3 και 4: Κανονικοποίηση Προθέσεων Λειτουργιών και Παραγωγή Μοντέλου Πυρηνικών Εννοιολογικών Οντοτήτων: Στο Βήμα 3, οι λειτουργίες μιας υπηρεσίας ταξινομούνται σε ένα σύνολο κανονικοποιημένων *κατηγοριών πρόθεσης* (*intention categories*) χρησιμοποιώντας πληροφορίες από τα αντίστοιχα Μοντέλα Υπογραφών και τα αντίστοιχα Μοντέλα Όρων Λειτουργίας. Οι κανονικοποιημένες κατηγορίες πρόθεσης υποδεικνύουν το κατά πόσο μια ΠΣΔ υπηρεσία μπορεί χαρακτηρισθεί ως: *Constructor*, *Destructor*, *Accessor*, *Mutator*, *Query*, *Investigator* και *Process* [6]. Η ταξινόμηση αυτή είναι σημαντική για τη συσχέτιση τύπων πόρων με μεθόδους του HTTP και άρα με πράξεις μεταχείρισης. Στο Βήμα 4, τα Μοντέλα Όρων Λειτουργίας και το αντίστοιχο συναθροιστικό Μοντέλο Όρων Υπηρεσίας χρησιμοποιούνται για να *α'*) κατασκευαστεί το Μοντέλο των Πυρηνικών Εννοιολογικών Οντοτήτων (Core Conceptual Entities Model) και *β'*) να προσδιοριστούν εξαρτήσεις μεταξύ των οντοτήτων αυτών, Οι οντότητες CCE αναπαριστούν τα θεμελιώδη στοιχεία των πόρων της REST διεπαφής-στόχου. Ομοίως, οι εξαρτήσεις μεταξύ των οντοτήτων αυτών αντανακλούν ιεραρχικές σχέσεις μεταξύ των πόρων. Υπό αυτή την άποψη, τα στοιχεία CCE και οι σχέσεις τους παρέχουν τη βασική δομή για το μοντέλο πόρων που εξάγεται.

Βήμα 5: Παραγωγή Μοντέλου Τύπων Πόρων: Οι εξαχθείσες οντότητες CCE μαζί με πληροφορίες ως προς συσχετισμένες με αυτές τις οντότητες, λειτουργίες, χρησιμοποιούνται για να σχηματίσουν ένα *Μοντέλο Τύπων Πόρων* (*Resource Types Model* (*RTM*)). Το παραγόμενο RTM αποτελεί το τελικό προϊόν της διαδικασίας

εξαγωγής πόρων.

Στις ενότητες που ακολουθούν παρουσιάζουμε κάθε ένα από τα βήματα της διαδικασίας λεπτομερώς.

4.2.2 Βήμα 1: Μοντέλα Υπογραφών

Τα Μοντέλα Υπογραφών εισάγονται ως ένας μηχανισμός μέσω του οποίου οι περιγραφές των υπηρεσιών μπορούν να αναπαρασταθούν με κανονικοποιημένο τρόπο. Αυτό συμβαίνει παρέχοντας μια αφαίρεση ως προς το συγκεκριμένο συντακτικό ή τις δομικές λεπτομέρειες της αρχικής περιγραφής της διεπαφής, όπως αυτές παρέχονται από την υποκείμενη γλώσσα προδιαγραφής η οποία αποτελεί αποδεκτό τύπο δεδομένων εισόδου. Στην τρέχουσα υλοποίηση του περιβάλλοντος πλαισίου θεωρούμε δεδομένα εισόδου υπό τη μορφή WSDL περιγραφών, τα οποία χρησιμοποιούνται για την παραγωγή αντίστοιχων, κανονικοποιημένων Μοντέλων Υπογραφών. Ωστόσο, οποιαδήποτε άλλη γλώσσα προδιαγραφής διεπαφών η οποία εκθέτει μια υπηρεσία ως συλλογή λειτουργιών θα μπορούσε εύκολα να υποστηριχθεί, εφόσον παρασχεθεί ένα κατάλληλο στοιχείο δημιουργίας Μοντέλων Υπογραφών για τη γλώσσα αυτή. Τα Μοντέλα Υπογραφών έχουν ένα απλό μεταμοντέλο, σύμφωνα με το οποίο κάθε υπογραφή λειτουργίας (s_i) έχει ένα όνομα ($s_i.name$) το οποίο ισούται με το όνομα της λειτουργίας, ένα σύνολο παραμέτρων εισόδου ($s_i.input$) και ένα σύνολο παραμέτρων εξόδου ($s_i.output$). Μια παράμετρος (p) έχει ένα όνομα ($p.name$), έναν τύπο ($p.type$) και μια ένδειξη πολλαπλότητας ($p.multiplicity$). Επίσης, ο τύπος μιας παραμέτρου μπορεί να είναι απλός (π.χ. βασικός τύπος) ή σύνθετος (μια δομή απλών ή σύνθετων τύπων). Τέλος, οι παράμετροι μπορούν να επισημειωθούν είτε ως *δεδομένα εφαρμογής* (*application data*) είτε ως *μεταδεδομένα* (metadata) ($p.class$), με βάση το αν αναπαριστούν πληροφορίες οι οποίες χρησιμοποιούνται άμεσα για την παροχή της λειτουργικότητας που προσφέρει η λειτουργία (π.χ. η παράμετρος *orderId* στη λειτουργία *getOrder* που υπάρχει στον Πίνακα 4.1), ή το αν αναπαριστούν πληροφορίες οι οποίες δε χρησιμοποιούνται άμεσα από τη λειτουργία (π.χ. κλειδιά αυθεντικοποίησης, χρονοσφραγίδες, κλπ). Η παραπάνω πληροφορία επισημείωσης των παραμέτρων χρησιμοποιείται κατά την εξαγωγή πόρων και συγκεκριμένα για την αποτίμηση ευρετικών κανόνων. Επειδή οι συνήθεις IDL γλώσσες όπως η WSDL δεν παρέχουν τέτοιου είδους πληροφορίες για τις παραμέτρους των λειτουργιών της υπηρεσίας, υπολογίζεται ένα *TF-IDF* [132]

σκορ κατηγοριοποίησης, όπου οι παράμετροι παίζουν το ρόλο των όρων και οι υπογραφές παίζουν το ρόλο των εγγράφων. Πιο συγκεκριμένα, για κάθε παράμετρο p κάθε υπογραφής s_i και για όλες τις υπογραφές λειτουργιών S, η βαθμολογία κατηγοριοποίησης υπολογίζεται από τη συνάρτηση C_{pf-isf} ως εξής:

$$C_{pf-isf}(p, s_i, S) = pf(p, s_i) \times isf(p, s_i, S)$$

όπου η *συχνότητα παραμέτρου parameter frequency* (pf) ορίζεται ως:

$$pf(p, s_i) = \begin{cases} 2, & (p \in s_i.input \cup s_i.output) \wedge (substring(p.name, s_i.name)) \\ 1, & (p \in s_i.input \cup s_i.output) \wedge (\neg substring(p.name, s_i.name)) \\ 0, & otherwise. \end{cases}$$

και η *αντίστροφη συχνότητα υπογραφής inverse signature frequency* (isf) υπολογίζεται ως εξής:

$$isf(p, s_i, S) = log_2 \frac{|S|}{|\{s_i \in S : pf(p, s_i) > 0\}|}$$

Χρησιμοποιώντας ένα κατώφλι T_{pf-isf}, μια παράμετρος p σε μια υπογραφή s_i κατηγοριοποιείται ως εξής:

$$p.class = \begin{cases} application\ data, & C_{pf-isf}(p, s_i, S) > T_{pf-isf} \\ metadata, & P_{pf-isf}(p, s_i, S) \leq T_{pf-isf} \end{cases}$$

Μια μηδενική τιμή για το $C_{pf-isf}(p, s_i, S)$ δείχνει ότι η παράμετρος p της s_i εμφανίζεται σε όλες τις υπογραφές (δηλαδή δεν μπορεί να θεωρηθεί ειδική για μια λειτουργία) και ως εκ τούτου μπορεί να γίνει η υπόθεση ότι πρόκειται για παράμετρο μεταδεδομένων. Ωστόσο, για να είναι επίσης δυνατή η κατηγοριοποίηση ως μεταδεδομένα παραμέτρων οι οποίες είναι ιδιαιτέρως συχνές αλλά οι οποίες ενδέχεται να μην υπάρχουν στις παραμέτρους εισόδου ή εξόδου σε ένα περιορισμένο υποσύνολο όλων των λειτουργιών, χρησιμοποιείται μια θετική τιμή T_{pf-isf}, κοντά στο 0, ως κατώφλι. Για τα πειράματα που παρουσιάζονται στο Κεφάλαιο 6, το T_{pf-isf} τέθηκε ίσο με 0.2. Για παράδειγμα, στην υπηρεσία SimpleOMS, η παράμετρος εισόδου *orderId* της λειτουργίας *getOrder* (Πίνακας 4.1) έχει επισημειωθεί ως δεδομένο εφαρμογής καθώς $C_{pf-isf}(orderId, getOrder, S) = 0.65$, ενώ η παράμετρος εισόδου *auth* της ίδιας λειτουργίας έχει επισημειωθεί ως μεταδεδομένα καθώς $C_{pf-isf}(auth, getOrder, S) = 0$.

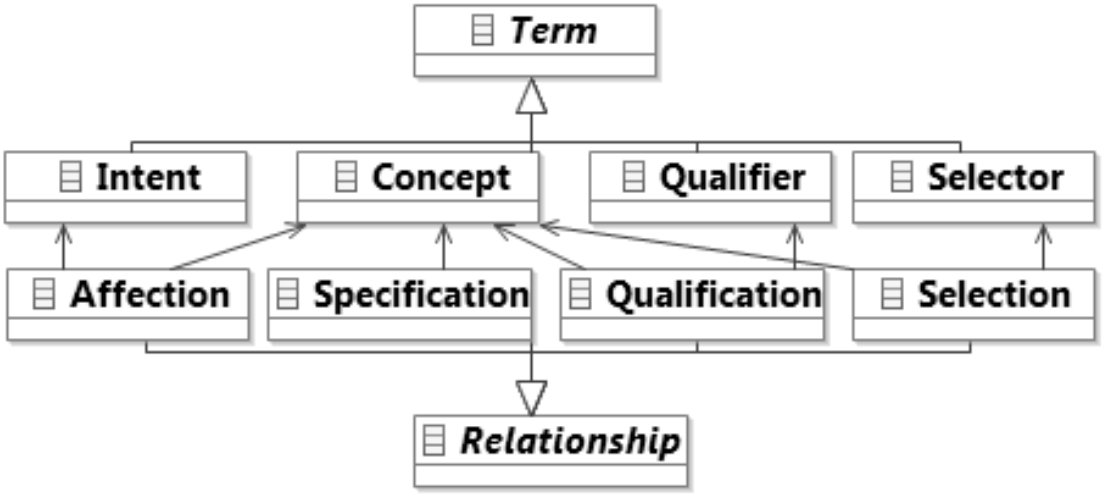
Η κατασκευή ενός Μοντέλου Υπογραφών για μια διεπαφή βασισμένη σε WSDL βασίζεται σε δύο στάδια: *α*) φόρτωση, και *β*) εξειδίκευση. Κατά τη διάρκεια του

σταδίου φόρτωσης, το WSDL έγγραφο υποβάλλεται σε επεξεργασία και δημιουργείται μια πρώτη έκδοση Μοντέλων Υπογραφών μέσω άμεσων αντιστοιχίσεων μεταξύ στοιχείων `PortType` ή `Interface` τα οποία υποδεικνύουν λειτουργίες υπηρεσίας και στοιχείων Μοντέλων Υπογραφών. Κατά τη διάρκεια του σταδίου εξειδίκευσης, τα Μοντέλα Υπογραφών ενημερώνονται βάσει του αν η διεπαφή που προσαρμόζεται ακολουθεί κάποιο συγκεκριμένο στυλ (π.χ. RPC, Document), μοτίβο (π.χ. document/literal wrapped), προφίλ (π.χ. WS-I Basic Profile [109]), ή άλλες συμβάσεις. Για παράδειγμα, για τις διεπαφές που ακολουθούν το μοτίβο document/literal wrapped [63] λαμβάνει χώρα το 'ξετύλιγμα' των δομών των αιτημάτων/απαντήσεων έτσι ώστε να μοντελοποιηθεί η πραγματική υπογραφή της λειτουργίας με μεγαλύτερη ακρίβεια. Σε κάθε περίπτωση, η παραγωγή των Μοντέλων Υπογραφών δεν επηρεάζει σημαντικά τα κύρια βήματα και τη λογική της διαδικασίας εξαγωγής πόρων. Τα κύρια αυτά βήματα και η λογική εξαγωγής συζητούνται λεπτομερώς στις ενότητες που ακολουθούν.

4.2.3 Βήμα 2: Μοντέλα Όρων Λειτουργίας και Μοντέλο Όρων Υπηρεσίας

Τα Μοντέλα Υπογραφών παρέχουν έναν κανονικοποιημένο τρόπο αναπαράστασης μιας ΠΣΔ διεπαφής. Παρ' όλα αυτά, η εξαγωγή πόρων απαιτεί έναν μηχανισμό μοντελοποίησης των λειτουργιών ο οποίος είναι σε θέση να υποδηλώσει τη σημασιολογία σους, αποτυπωμένη ως ένα σύνολο όρων οι οποίοι μπορούν να παρέχουν πληροφορίες ως προς το νόημα αλληλεπίδρασης με αυτή. Υπό αυτό το πρίσμα, εισάγουμε ένα σύνολο στοιχείων σχετικών με την ανάλυση όρων μιας λειτουργίας και συγκεκριμένα τους *τύπους όρων* (*term types*) και τους *τύπους σχέσεων* (*relationships types*). Οι όροι είναι λέξεις ή συμβολοσειρές οι οποίες μπορούν να εξαχθούν αυτόματα μέσω ανάλυσης ενός ονόματος μιας λειτουργίας και ενδεχομένως των παραμέτρων της, χρησιμοποιώντας NLP τεχνικές. Τα μοντέλα που αποτυπώνουν τις πληροφορίες αυτές είναι τα Μοντέλα Όρων Λειτουργιών (Operation Terms Model (OTM)) και το Μοντέλο Όρων Υπηρεσίας (Service Terms Model (STM)). Τα Μοντέλα Όρων Λειτουργιών (για συντομία OTM) στοχεύουν στην καταγραφή πληροφοριών στο επίπεδο της σημασιολογίας της λειτουργικότητας που προσφέρει μια λειτουργία. Το Σχήμα 4.3 απεικονίζει το μεταμοντέλο των OTM (δηλαδή τους τύπους όρων και τις σχέσεις όρων), ενώ ο Πίνακας 4.2 και ο Πίνακας 4.3 παρέχουν περιγραφές

των στοιχείων του μεταμοντέλου οι οποίες συνοδεύονται από παραδείγματα που αντλούνται από την υπηρεσία SimpleOMS.



Σχήμα 4.3: Στοιχεία του μεταμοντέλου του OTM

Πίνακας 4.2: Τύποι όρων στο OTM

Στοιχείο	Περιγραφή	Παραδείγματα
Intent	Δηλώνει την πρόθεση μιας αλληλεπίδρασης. Συνήθως, το στοιχείο Intent αποτελεί το ρηματικό τμήμα ενός ονόματος υπηρεσίας.	**add**OrderItem: Intent(**add**) **get**SubmittedOrders: Intent(**get**)
Concept	Δηλώνει ένα στοιχείο ή ένα χαρακτηριστικό με σημαντικό πληροφοριακό περιεχόμενο ως προς τη λογική που υλοποιεί η λειτουργία.	getSubmitted**Orders**: Concept(**orders**)
Qualifier	Προσαρμόζει ή επαυξάνει τα σημασιολογικά χαρακτηριστικά των στοιχείων Concept.	get**Submitted**Orders: Qualifier(**submitted**)
Selector	Μεσολαβεί για μια πράξη προβολής μεταξύ δύο στοιχείων Concept (π.χ. φιλτράρισμα, επιλογή, κλπ.).	getSubmittedOrders **By**Date: Selector(**by**)

Πίνακας 4.3: Τύποι σχέσεων στο OTM

Στοιχείο	Περιγραφή	Παραδείγματα
Affection	Συνδέει τον όρο Intent μιας λειτουργίας σε ένα από τα Concepts, υποδεικνύοντας σε πιο Concept εφαρμόζεται η πρόθεση της αλληλεπίδρασης.	getSubmittedOrders: get (Intent) affects orders (Concept)
Specification	Μπορεί να οριστεί μεταξύ δύο Concepts όπου το Concept που προσδιορίζει μειώνει το εύρος του τύπου του Concept που προσδιορίζεται.	addOrderItem: order (Concept) specifies item (Concept)
Qualification	Υποδεικνύει μια διαμόρφωση ή μια επαύξηση των σημασιολογικών χαρακτηριστικών ενός Concept από ένα Qualifier.	getSubmittedOrders: submitted (Qualifier) qualifies orders (Concept)
Selection	Υποδεικνύει μια πράξη προβολής μεταξύ δύο Concept η οποία ορίζεται από ένα Selector.	getSubmittedOrders ByDate: date (Concept) selects orders (Concept)

Το Σχήμα 4.4 απεικονίζει ένα παράδειγμα ενός OTM και συγκεκριμένα της λειτουργίας *getOrderItemShippingStatus* της υπηρεσίας SimpleOMS.

Ομοίως, το Μοντέλο Όρων Υπηρεσίας (για συντομία, STM) ορίζεται ως ένας επισημειωμένος πολυ-γράφος που συγχωνεύει σε ένα ενιαίο μοντέλο όλα τα OTM και τις πληροφορίες που αυτά αποτυπώνουν για κάθε λειτουργία της υπηρεσίας. Το STM περιέχει κόμβους οι οποίοι αντιστοιχούν σε όρους επιπέδου λειτουργιών, με σταθμισμένες επισημειώσεις. Οι σταθμισμένες αυτές επισημειώσεις αντιστοιχούν στους διάφορους τύπους με τους οποίους ένας όρος ενδέχεται να εμφανιστεί στα OTM. Επίσης, οι όροι συνδέονται μέσω σταθμισμένων ακμών οι οποίες προέρχονται

από τις σχέσεις των όρων στα OTM, ενώ τα βάρη τους αντιστοιχούν στους διαφόρους τύπους σχέσεων που εμφανίζονται στα OTM. Ο πρωταρχικός ρόλος ενός STM είναι να χρησιμεύσει ως μέσο για τη μετάβαση από όρους και σχέσεις, σε οντότητες και εξαρτήσεις. Ένα παράδειγμα STM για την υπηρεσία SimpleOMS απεικονίζεται στο Σχήμα 4.5.

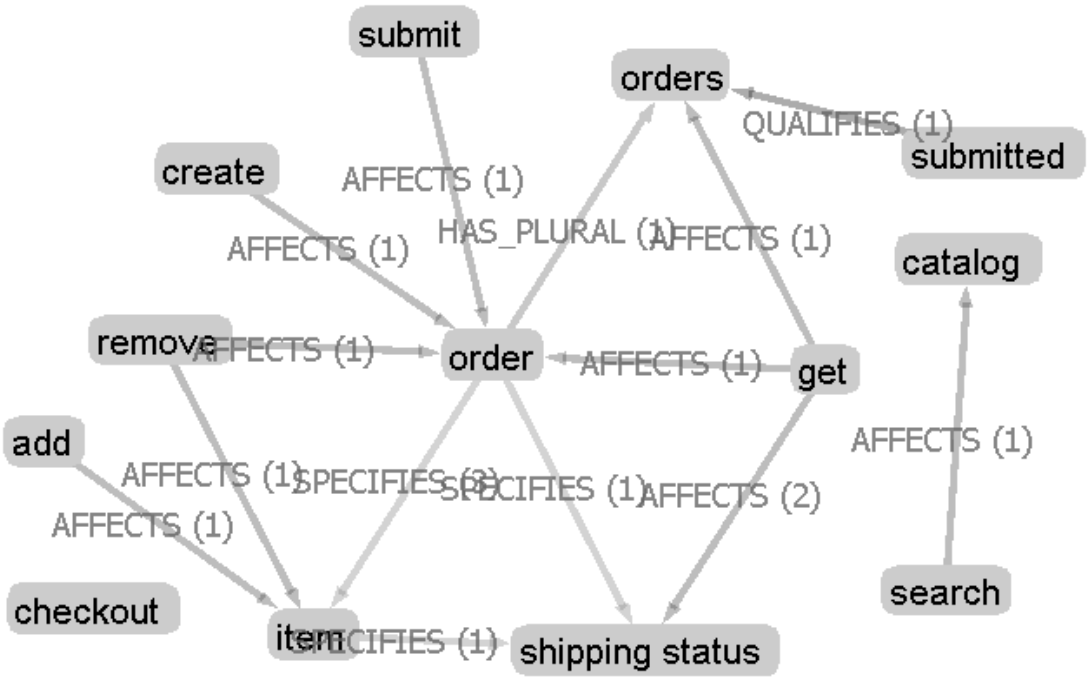
Operation Terms Model:
Operation: **getOrderItemShippingStatus**
Parsing/splitting: **get order item shipping status**
Intent: get(VB)
Concepts: order(NN), item(NN),
shipping(NN), status(NN)
Relationships:
- Affection: get AFFECTS status
- Specifications:
order SPECIFIES item
item SPECIFIES shipping
shipping SPECIFIES status

Σχήμα 4.4: Παράδειγμα μοντέλου OTM (Operation Terms Model)

Η παραγωγή των OTM και του STM συζητούνται παρακάτω σε μεγαλύτερη λεπτομέρεια.

Παραγωγή Μοντέλων Όρων Λειτουργίας

Α. Διαχωρισμός συμβολοσειρών (Tokenization): Εστιάζουμε την ανάλυσή μας στα ονόματα των λειτουργιών των υπηρεσιών, τα οποία τα αντιμετωπίζουμε ως αναγνωριστικά με ουσιώδες πληροφοριακό περιεχόμενο. Σε μια ΠΣΔ διεπαφή, τα ονόματα των λειτουργιών παρέχουν σημασιολογικά πλούσιες πληροφορίες ως προς την παρεχόμενη λειτουργικότητα από τη λειτουργία. Με άλλα λόγια, τα ονόματα των υπηρεσιών συνήθως ορίζονται με τέτοιο τρόπο ώστε να μπορούν να παρέχουν μια σύντομη απάντηση στο τι κάνει η λειτουργία. Τυπικά, μια εργασία διαχωρισμού συμβολοσειρών σε λεκτικές μονάδες (tokens) απαιτεί την εφαρμογή ενός ή περισσοτέρων κανόνων οριοθέτησης σε μια ακολουθία χαρακτήρων (π.χ. στη φυσική γλώσσα χρησιμοποιούνται συνήθως για τον σκοπό αυτό η ύπαρξη κενών διαστημάτων, σημείων στίξης, κλπ). Η ακρίβεια μιας τέτοιας εργασίας διαχωρι-



Σχήμα 4.5: Γράφος του Service Terms Model για την υπηρεσία *SimpleOMS*

σμού συμβολοσειρών μπορεί να βελτιωθεί περαιτέρω μέσω υπολογιστικών γλωσσολογικών τεχνικών οι οποίες ομαδοποιούν περισσότερες από μια λεκτικές μονάδες έτσι ώστε να αντιμετωπίζονται ως μια λεκτική μονάδα (για παράδειγμα, οι μονάδες "Los" και "Angeles" ως "Los Angeles"). Οι τεχνικές αυτές αναφέρονται συνήθως ως *collocation extraction techniques* [87]. Ένα γενικότερο αλλά στενά συνδεδεμένο πρόβλημα στην περιοχή της τεχνολογίας λογισμικού είναι αυτό του διαχωρισμού και της επέκτασης αναγνωριστικών [85], [34]. Οι τεχνικές που έχουν προταθεί στη βιβλιογραφία εφαρμόζουν συχνά σύνθετους αλγορίθμους διαχωρισμού μαζί με τη χρησιμοποίηση εξωτερικών πηγών πληροφορίας (π.χ. λεξικά), έτσι ώστε τα αναγνωριστικά και τα τμήματά τους να μπορούν να διαχωριστούν και στη συνέχεια να επεκταθούν σε πλήρης λέξεις.

Για την κατασκευή μιας αποτελεσματικής και ταυτόχρονα χαμηλών απαιτήσεων αλλά και αποδοτικής τεχνικής διαχωρισμού των ονομάτων λειτουργιών υπηρεσιών, μελετήσαμε τις συμβάσεις και τα μοτίβα που ακολουθούνται από τους προγραμματιστές κατά τον προσδιορισμό ονομάτων λειτουργιών. Ειδικότερα, αφού εξετάσαμε τη σχετική βιβλιογραφία [45], [31] και έγγραφα σχετικά με οδηγίες και συνιστώ-

μενες πρακτικές για την ονομοτοδοσία λειτουργιών υπηρεσιών [105], [130] [16] και μέσω της ανάλυσης ενός σημαντικού αριθμού περιγραφών υπηρεσιών για τον εντοπισμό συχνών μοτίβων, δημιουργήσαμε μια λίστα με συχνά χρησιμοποιούμενα μοτίβα που ακολουθούνται κατά την ονοματοδοσία λειτουργιών υπηρεσιών. Στη συνέχεια, υπολογίστηκε η συχνότητα κάθε μοτίβου μετά από επεξεργασία 867 υπηρεσιών οι οποίες περιείχαν 12.918 λειτουργίες (βλέπε Κεφάλαιο 6, Ενότητα 6.1.2 για περισσότερες πληροφορίες σχετικά με το σύνολο δεδομένων που χρησιμοποιήθηκε) με στόχο την αποτίμηση της σχετικής σημασίας των μοτίβων αυτών. Τα ευρήματα της ανάλυσης έδειξαν ότι τα μοτίβα upper Camel Case (π.χ. *GetOrders*) και lower Camel Case *getOrders* είναι τα πλέον ευρέως χρησιμοποιούμενα μοτίβα ονοματοδοσίας (77.7% των περιπτώσεων). Το μοτίβο μεικτών Camel Case με κεφαλαία (π.χ. *getClosestATMs*) είναι επίσης συχνό (13.4%) αν και οι αντίστοιχοι κανόνες οριοθέτησης είναι πιο σύνθετοι. Ένα άλλο συχνό μοτίβο είναι η ρητή οριοθέτηση (π.χ. *get_submitted_orders*) (9.3%), η οποία εμφανίζεται κυρίως βασισμένη σε διαχωρισμό μέσω κάτω παύλας (underscore-based) ή τελείας (dot-based). Τελικά, ο εντοπισμός προθημάτων (prefixes) και μεταθημάτων (postfixes) μπορεί επίσης να αποδειχθεί χρήσιμος, για τον ορθότερο διαχωρισμό του 1% των ονομάτων λειτουργιών. Σαφώς, τα παραπάνω μοτίβα χρησιμοποιούνται συχνά συνδυαστικά κατά τον προσδιορισμό των ονομάτων των λειτουργιών.

Χρησιμοποιώντας την εμπειρική γνώση που αντλήθηκε από την παραπάνω ανάλυση των ονομάτων λειτουργιών και των κατασκευαστικών μοτίβων τους, αναπτύξαμε έναν αλγόριθμο διαχωρισμού συμβολοσειρών σε λεκτικές μονάδες ο οποίος λειτουργεί σε δύο στάδια: *α*) αναγνωρίζει μοτίβα στα οποία συμμορφώνεται ένα όνομα λειτουργίας (π.χ. dot / underscore-based ρητή οριοθέτηση, ύπαρξη προθήματος/μεταθήματος, μεικτό / upper / lower Camel Case), και *β*) πραγματοποιεί το διαχωρισμό σύμφωνα με ειδικές ως προς τα μοτίβα ευρετικές οριοθέτησης. Ο αλγόριθμος εφαρμόζεται με αναδρομικό τρόπο έως ότου δεν εντοπίζεται άλλο μοτίβο, εξετάζοντας και διαχωρίζοντας πρώτα τα μοτίβα με ρητή οριοθέτηση. Για παράδειγμα, για τη λειτουργία *CreditCard_getExpirationDate* ο αλγόριθμος αρχικά αναγνωρίζει ότι το όνομα της λειτουργίας ακολουθεί το underscore-based μοτίβο ρητής οριοθέτησης και διαχωρίζει το όνομα της υπηρεσίας σε δύο τμήματα, όπως ορίζεται από τον χαρακτήρα οριοθέτησης. Έπειτα, η αναγνώριση μοτίβων εφαρμόζεται εκ νέου για κάθε τμήμα. Στο πρώτο τμήμα αναγνωρίζεται το μοτίβο upper CamelCase ενώ στο δεύτερο το μοτίβο lower Camel Case. Εφαρμόζονται οι κανόνες οριοθέτησης για Camel Case σε κάθε τμήμα και αντλούνται λεκτικές μονάδες οι

οποίες δεν ακολουθούν κάποιο περαιτέρω μοτίβο και άρα ο αλγόριθμος τερματίζει.

Αξιολογήσαμε την ακρίβεια του αλγορίθμου διαχωρισμού σε λεκτικές μονάδες χρησιμοποιώντας μια μέθοδο απλής τυχαίας δειγματοληψίας στο αρχικό σύνολο δεδομένων που περιελάμβανε 12.918 ονόματα λειτουργιών και για μέγεθος δείγματος $n = 388$ η ακρίβεια του διαχωρισμού αποτιμήθηκε σε 96.64%. Το σφάλμα διαχωρισμού αποτιμήθηκε σε 3.36% το οποίο αποδίδεται στους παρακάτω παράγοντες: *α*) προβληματική είσοδος, για παράδειγμα η λειτουργία *GetUnAuthorized* διαχωρίζεται ως *get*, *un*, *authorized*, *β*) απουσία μοτίβου, για παράδειγμα το όνομα λειτουργίας *additemtolist* δεν ακολουθεί καμία ρητή οριοθέτηση, ούτε κάποιο μοτίβο σχετικό με πεζούς ή κεφαλαίους χαρακτήρες, *γ*) λανθασμένη αναγνώριση μοτίβου, για παράδειγμα η λειτουργία *getATMbyLocation* αναγνωρίζεται ως περίπτωση μεικτού lower Camel Case με κεφαλαία

Επιστρέφοντας στο συνοδευτικό παράδειγμα, όλα τα ονόματα των λειτουργιών που περιέχονται στην υπηρεσία SimpleOMS διαχωρίζονται σε λεκτικές μονάδες μέσω του κανόνα lower Camel Case.

Β. Γραμματική επισημείωση: Οι λεκτικές μονάδες που αντλούνται μέσω του αλγορίθμου διαχωρισμού αποτελούν ακολουθίες χαρακτήρων χωρίς οποιαδήποτε μεταδεδομένα που να υποδεικνύουν το ρόλο κάθε λεκτικής μονάδας στην ακολουθία μονάδων στην οποία ανήκουν (δηλαδή, σε ένα όνομα λειτουργίας, παραμέτρου ή υπηρεσίας). Ένας χαμηλών απαιτήσεων και αποτελεσματικός τρόπος για να επιτευχθεί αυτό είναι η χρησιμοποίηση *Part-Of-Speech* (*POS*), ή αλλίως γραμματική, επισημείωση [154].

Η POS επισημείωση παρέχει πληροφορίες για τη γραμματική χρήση ενός όρου σε μια πρόταση. Για παράδειγμα, η POS επισημείωση υποδεικνύει αν ένας όρος χρησιμοποιείται ως ρήμα, ως γερούνδιο, ως πρόθεση, ως σύνδεσμος, κλπ. Στην παρούσα προσέγγιση χρησιμοποιούμε εργαλεία POS επισημείωσης ώστε να σχολιαστούν γραμματικά οι λεκτικές μονάδες και στη συνέχεια να μεταφραστούν οι αλληλουχίες των επισημειώσεων σε OTM. Ειδικότερα, κάθε αλληλουχία λεκτικών μονάδων ενός ονόματος λειτουργίας υποβάλλεται σε ένα εργαλείο επισημείωσης ως μια πρόταση φυσικής γλώσσας και το εργαλείο αυτό παρέχει επισημειώσεις στις λεκτικές μονάδες χρησιμοποιώντας το σύνολο επισημειώσεων Penn Treebank II (PTB) [25]. Επιπλέον, για να επιτευχθεί μια υψηλής ποιότητας POS επισημείωση, συνδυάζουμε τα αποτελέσματα διαφορετικών εργαλείων POS επισημείωσης και

συγκεκριμένα των OpenNLP, Lingpipe και Stanford POS, σε ένα αποτέλεσμα μέσω απλώς τεχνικών μετα-επισημείωσης. Συγκεκριμένα, η POS μετα-επισημείωση υλοποιείται εφαρμόζοντας *majority voting* ή αλλιώς τη μέθοδο *Basic Ensemble Method (BES)* [119] [131], σε δύο επίπεδα. Πρώτον, ανά εργαλείο επισημείωσης, οι αλληλουχίες επισημειώσεων που προκύπτουν χρησιμοποιώντας όλα τα διαθέσιμα μοντέλα για την Αγγλική γλώσσα -δηλαδή, δύο για το OpenNLP, τρία για το Lingpipe και τέσσερα για το Stanford- συγχωνεύονται με majority voting. Οι τρεις αλληλουχίες POS επισημειώσεων που προκύπτουν, μια από κάθε εργαλείο, συγχωνεύονται ξανά μέσω majority voting, ούτως ώστε να παραχθεί μια μοναδική αλληλουχία επισημειώσεων. Τα πειράματα που διενεργήσαμε δείξανε ότι η παραπάνω τεχνική POS μετα-επισημείωσης οδηγεί σε υψηλότερη συνολική ακρίβεια, σε σύγκριση με οποιοδήποτε εργαλείο επισημείωσης, μια βελτίωση η οποία αποτιμήθηκε κατά μέσο όρο σε 11.46%.

Η χρησιμοποίηση POS επισημείωσης αποτελεί αποδοτική εναλλακτική σε σχέση με την ενσωμάτωση τεχνικών μηχανικής μάθησης απευθείας στη διαδικασία εξαγωγής, κάτι που θα απαιτούσε ειδική ως προς το πεδίο διαδικασία εκπαίδευσης και αντίστοιχα σύνολα δεδομένων. Επίσης, υπάρχει διαθέσιμη μια ευρεία συλλογή μοντέλων για POS επισημείωση, σε μια ποικιλία φυσικών γλωσσών, κάτι που καθιστά την προσέγγιση περισσότερο ευέλικτη και λιγότερο συσχετισμένη με συγκεκριμένη γλώσσα.

Γ. Ταξινόμηση όρων και σχέσεων: Το επόμενο βήμα είναι η ταξινόμηση κάθε όρου επισημειωμένου με POS πληροφορίες σε έναν εκ των τεσσάρων τύπων όρων και ο εντοπισμός σχέσεων μεταξύ των όρων, οι οποίες αντλούνται από το μεταμοντέλο που απεικονίζεται στο Σχήμα 4.3. Για κάθε όνομα υπηρεσίας (δηλαδή αλληλουχία λεκτικών μονάδων), η αλληλουχία των γραμματικών επισημειώσεων υποβάλλεται σε ένα σύνολο κανόνων βασισμένων σε μοτίβα για να κατασκευαστεί ένας όρος ανά λεκτική μονάδα. Τα μοτίβα ορίζονται βάσει της θέσης κάθε επισημείωσης στην αλληλουχία. Στον Πίνακα 4.4 παρέχει τη λίστα των μοτίβων δημιουργίας OTM όρων μαζί με αντίστοιχα παραδείγματα. Έπειτα, ένα δεύτερο σύνολο κανόνων εφαρμόζεται για τη δημιουργία σχέσεων μεταξύ των όρων. Ο Πίνακας 4.5 παρέχει το σύνολο των κανόνων δημιουργίας OTM σχέσεων. Για παράδειγμα, όπως φαίνεται στον Πίνακα 4.5, αν ένας όρος *Concept* C_1 προηγείται άμεσα ενός άλλου όρου *Concept* C_2, τότε μια σχέση *Specification* δημιουργείται, η οποία ξεκινά από τον πρώτο όρο C_1 και οδηγεί στο δεύτερο όρο C_2. Περισσότερα παραδείγματα τέτοιων

σχέσεων φαίνονται στο Σχήμα 4.4 το οποίο απεικονίζει ένα OTM για τη λειτουργία *getOrderItemShippingStatus* της υπηρεσίας SimpleOMS.

Πίνακας 4.4: Παραδείγματα παραγωγής OTM όρων

Κατηγορία	Περιγραφή	Παράδειγμα
Μόνο ουσιαστικά	Θεωρείται "get" ως Intent	ItemInfo: Intent(get)
Μοναδικό ρήμα	Το ρήμα γίνεται Intent	getOrder: Intent(get)
Πολλαπλά ρήματα	Το αρχικό ρήμα γίνεται Intent	runPackageBuild: Intent(run)
Ουσιαστικό	Το ουσιαστικό γίνεται Concept	removeOrder: Concept(Order)
Μη κατηγοριοποιημένη λέξη	Η μη κατηγοριοποιημένη λέξη γίνεται Concept	createDMC: Concept(DMC)
Επίθετο	Το επίθετο γίνεται Qualifier	getTopSongs: Qualifier(Top)
Μετοχή	Η μετοχή γίνεται Qualifier	getSubmittedOrders: Qualifier(Submitted)
Πρόθεση (επιλογή)	Η πρόθεση γίνεται Selector	getSubmittedOrders ByDate: Selector(By)

Πίνακας 4.5: Παραδείγματα παραγωγής OTM σχέσεων

Μοτίβο	Σχέση	Παράδειγμα
$[*]C_1C_2[*]$	Concept C_1 specifies Concept C_2	removeOrderItem: Order specifies Item
$[*]QC[*]$	Qualifier Q qualifies Concept C	getTopSongs: Top qualifies Songs
$I[*\backslash S]C[S*]$	Intent I affects Concept C	getOrdersByDate: get affects Orders
$[*]C_1S[*]C_2[*]$	Concept C_2 selects Concept C_1	getOrdersByLocation: Location selects Orders

Παραγωγή του Service Terms Model

Η αρχική έκδοση των OTM χρησιμοποιείται για να κατασκευαστεί ένα αντίστοιχο STM. Αυτό επιτυγχάνεται ενσωματώνοντας τα ανεξάρτητα OTM μοντέλα σε ένα συναθροιστικό μοντέλο στο επίπεδο της υπηρεσίας. Ειδικότερα, το STM μοντέλο είναι ένας πολύ-γράφος ο οποίος συντίθεται από κόμβους που αναπαριστούν όρους του OTM και ακμές που αναπαριστούν σχέσεις μεταξύ των όρων αυτών. Οι όροι του STM επισημειώνονται με βάρη υπό τη μορφή $\langle$*term type: πλήθος εμφανίσεων του τύπου όρου* $\rangle$ δείχνοντας α) τους αντίστοιχους τύπους όρων για τον κόμβο και β) τον αριθμό των εμφανίσεων κάθε τύπου.

Για παράδειγμα, στο Σχήμα 4.5 στο οποίο απεικονίζεται το παραχθέν STM για το SimpleOMS, ο όρος *order* επισημειώνεται ως $\{\langle Concept : 8\rangle\}$ (δε φαίνεται στο σχήμα), πράγμα που σημαίνει ότι υπάρχουν 8 εμφανίσεις του όρου στις οποίες αποτελεί *Concept* σε όλα τα OTM για την υπηρεσία αυτή. Ομοίως, κάθε σχέση μεταξύ δύο όρων στο STM επισημειώνεται επίσης με τον αριθμό των εμφανίσεων της σχέσης, για το ίδιο ζεύγος όρων, σε όλα τα OTM της υπηρεσίας. Για παράδειγμα, στο Σχήμα 4.5, η *Specification* σχέση μεταξύ των όρων *order* και *item* επισημειώνεται με τον αριθμό 3, το οποίο υποδεικνύει ότι η σχέση αυτή υπάρχει

σε τρία OTM και συγκεκριμένα στα OTM που παράγονται για τις λειτουργίες *addOrderItem*, *removeOrderItem*, και *getOrderItemShippingStatus*. Χρησιμοποιώντας τους παραπάνω μετρητές εμφανίσεων, μπορούν να υπολογιστούν συχνότητες και να χρησιμοποιηθούν από μια διαδικασία αναδιάρθρωσης η οποία βελτιώνει την ακρίβεια των OTM και των STMs που δημιουργούνται. Τελικά, κατά τη διάρκεια δημιουργίας των STM προστίθεται ένας επιπλέον τύπος σχέσης που ονομάζεται *Has-Plural*, ο οποίος συνδέει κόμβους που αναπαριστούν την ίδια έννοια σε ενικό και πληθυντικό αριθμό.

Αναδιάρθρωση των Operation Terms Model

Το βήμα αναδιάρθρωσης είναι απαραίτητο διότι τα OTM παράγονται χωρίς να εξεταστεί η συνολική εικόνα όλων των λειτουργιών που περιέχουν έναν όρο, ενώ το παραχθέν STM μπορεί να παράσχει μια συνολική και συνεπή όψη όλων των όρων και των σχέσεων μεταξύ τους, επιτρέποντας τελικά την εκ νέου κατηγοριοποίηση των όρων εφαρμόζοντας ένα βρόγχο ανάδρασης. Συγκεκριμένα, το στάδιο της αναδιάρθρωσης αναλύει τις επισημειώσεις που υπάρχουν στο STM και αποδίδει ξανά τύπους στους όρους των OTM. Για παράδειγμα, υποθέτοντας ότι ο όρος *order* εμφανίζεται σε 8 OTM και ότι έχει κατηγοριοποιηθεί αρχικά ως *Concept* σε 7 από 8 OTM, τότε ο βρόγχος ανάδρασης μπορεί να κατηγοριοποιήσει εκ νέου τον όρο *order* ως *Concept* στη λειτουργία στην οποία διαφοροποιείται, παρέχοντας με αυτό τον τρόπο ένα αναθεωρημένο OTM για τη λειτουργία αυτή. Κατά συνέπεια, ο όρος *order* κατηγοριοποιείται πλέον αποκλειστικά ως *Concept*. Η νέα κατηγοριοποίηση βασίζεται σε έναν αλγόριθμο αναδιάρθρωσης ο οποίος λαμβάνει υπόψη τόσο κατώφλια συχνοτήτων όσο και το κατά πόσο είναι εφικτή μια εκ νέου κατηγοριοποίηση σε ένα OTM που εξετάζεται. Η διαδικασία αναδιάρθρωσης τερματίζει όταν δεν απαιτούνται περαιτέρω αναθεωρήσεις ή όταν έχει λάβει χώρα ο μέγιστος αριθμός δυνατών επαναλήψεων. Πιο συγκεκριμένα, κατά τη διάρκεια ενός βρόγχου αναδιάρθρωσης, εφαρμόζουμε τα παρακάτω κριτήρια έτσι ώστε να επιλέξουμε ένα υποσύνολο του συνόλου των OTM ως υποψήφια προς αναδιάρθρωσης. Πιο συγκεκριμένα, ένα OTM επιλέγεται όταν: *α*) οι επισημειώσεις του STM υποδεικνύουν ότι ο όρος *Intent* του OTM δεν κατηγοριοποιείται ως *Intent* στο 1/3 των OTM στα οποία υπάρχει ως όρος, *β*) οι επισημειώσεις του STM υποδεικνύουν ότι λιγότεροι από το 50% των όρων του OTM κατηγοριοποιούνται σε συμφωνία με την πλειονότητα των κατηγοριοποιήσεων των όρων αυτών. Για παράδειγμα, αν υποθέσουμε ότι

οι όροι της λειτουργίας *addOrderItem* κατηγοριοποιούνται ως *add: Intent, order: Qualifier, item: Selector* και ότι οι επισημειώσεις του STM του SimpleOMS είναι οι ακόλουθες: *add:* $\langle Concept : 7 \rangle$, *order:* $\langle Concept : 7 \rangle, \langle Qualifier : 1 \rangle$, *item:* $\langle Concept : 2 \rangle, \langle Selector : 1 \rangle$, τότε το OTM θα επιλεχθεί για αναδιάρθρωση καθώς δύο από τους τρεις όρους που περιλαμβάνει δεν κατηγοριοποιούνται σε συμφωνία με την πλειονότητα των κατηγοριοποιήσεων στο επίπεδο της υπηρεσίας, όπως φαίνεται από τις επισημειώσεις των κόμβων του STM.

Εφόσον επιλεχθούν τα υποψήφια OTM, εφαρμόζεται η ακόλουθη στρατηγική: αρχικά όλοι οι όροι του OTM εξετάζονται ως πιθανοί *Intent* όροι και με βάση τη σχετική συχνότητα εμφάνισης τους ως *Intent*, επιλέγεται ο καλύτερος ως όρος τύπου *Intent* για το OTM. Αφού επιλεχθεί ο όρος τύπου *Intent*, πιθανώς μετά από εκ νέου κατηγοριοποίηση, όλοι οι όροι ευθυγραμμίζονται με την πλειονότητα των κατηγοριοποιήσεων τους, βάσει των επισημειώσεων στο STM και των μεταξύ τους σχέσεων. Τελικά, εξετάζεται η δυνατότητα να ισχύσει η νέα κατηγοριοποίηση και δομή του OTM χρησιμοποιώντας δομικά κριτήρια συνέπειας (π.χ. ένα OTM δεν μπορεί να περιλαμβάνει περισσότερους από έναν όρους τύπου *Intent*). Ο παραπάνω βρόγχος ανάδρασης υπολογίστηκε ότι βελτιώνει την ακρίβεια της κατηγοριοποίησης κατά 7.83% (βλ. Κεφάλαιο 6).

Συγχώνευση κόμβων του STM

Μετά τη σύγκλιση του βρόγχου αναδιάρθρωσης των OTM και την παραγωγή του αντίστοιχου STM, η διαδικασία συνεχίζεται εφαρμόζοντας έναν αλγόριθμό επεξεργασίας γράφων ο οποίος συγχωνεύει κόμβους μεταξύ τους, εξετάζοντας μοτίβα επισημειώσεων κόμβων και ακμών, έτσι ώστε να αναγνωριστεί η στόχευση χρήσης των συγκεκριμένων όρων και τελικά να διευκολυνθεί η αναγνώριση οντοτήτων και πόρων. Ειδικότερα, όταν ένας κόμβος Α σε ένα STM είναι κατηγοριοποιημένος ως *Concept* και έχει μια *Specification* σχέση με ένα κόμβο Β ο οποίος έχει επίσης κατηγοριοποιηθεί ως *Concept*, τότε εφόσον δεν υπάρχει *Specification* σχέση που να ξεκινά από το Β και εφόσον δεν υπάρχουν άλλες σχέσεις που να ξεκινούν από τον κόμβο Α, τότε οι Α και Β συγχωνεύονται. Για παράδειγμα, αν και οι όροι shipping και status ενδέχεται να αποτελούν δόκιμους όρους στο πλαίσιο της διεπαφής του SimpleOMS, θα πρέπει να εξεταστούν από κοινού για να αναπαραστήσουν με καλύτερο τρόπο τη σημασιολογία των εννοιών και των οντοτήτων που καθορίζουν

τις δυνατότητες που παρέχει η υπηρεσία. Ο κόμβος που προκύπτει από συγχώνευση διατηρεί τις υπόλοιπες εισερχόμενες και εξερχόμενες ακμές από και προς τους υπόλοιπους κόμβους του STM. Το Σχήμα 4.5 απεικονίζει τον STM γράφο για την υπηρεσία SimpleOMS, στον οποίο οι κόμβοι για τους όρους shipping και status έχουν συγχωνευτεί σε ένα (βλ. κόμβο shipping status).

4.2.4 Βήμα 3: Κανονικοποίηση της πρόθεσης των λειτουργιών

Ένα σημαντικό θέμα που εξετάζεται σε αυτό το στάδιο είναι ο προσδιορισμός του ποια είναι η γενική πρόθεση ενός πελάτη που καλεί μια λειτουργία. Αυτό γίνεται με κατηγοριοποίηση κάθε λειτουργίας σε μια προκαθορισμένη συλλογή των κατηγοριών πρόθεσης (intention categories): *Constructor*, *Destructor*, *Accessor*, *Mutator*, *Query*, *Investigator*, *Process*. Στην τελευταία κατηγορία, εντάσσονται οι λειτουργίες που δεν κατηγοριοποιούνται σε καμία από τις προηγούμενες. Οι κατηγορίες αυτές χρησιμοποιούνται για να δηλώσουν τη σημασιολογία της πρόθεσης για αλληλεπίδραση. Έτσι, σε αυτό το στάδιο της διαδικασίας εξαγωγής πόρων, στόχος είναι να καθοριστεί η αντιστοιχία μεταξύ κάθε λειτουργίας (π.χ. *getOrderShippingStatus*) σε μια από αυτές τις κατηγορίες (π.χ. *Accessor*).

Η τεχνική της κανονικοποίησης βασίζεται σε πληροφορίες που πηγάζουν από τις παραμέτρους εισόδου και εξόδου, δηλαδή την υπογραφή μιας λειτουργίας όπως καταγράφεται στο αντίστοιχο Μοντέλο Υπογραφής, από το αντίστοιχο OTM και από προκαθορισμένες συσχετίσεις μεταξύ συγκεκριμένων *Intent* όρων και κατηγοριών κανονικοποίησης, π.χ. το *get* δείχνει πρόσβαση (*Accessor*), ενώ το *remove* υποδηλώνει καταστροφή (*Destructor*). Οι προκαθορισμένες σχέσεις που χρησιμοποιήσαμε υπολογίστηκαν χρησιμοποιώντας ένα σύνολο 12.918 λειτουργιών υπηρεσιών, μέσω της κατασκευής των αντίστοιχων OTM και της εξέτασης των 150 πιο συχνών *Intent* όρων. Αυτοί οι 150 *Intent* όροι αντιπροσωπεύουν το 90.55% του συνόλου λειτουργιών και κατατάχθηκαν σε κατηγορίες χειροκίνητα. Στη συνέχεια, χρησιμοποιώντας το WordNet [99] τα αρχικά σύνολα προθέσεων εμπλουτίστηκαν περαιτέρω μέσω των hyperonym synonym συνόλων κάθε επιλεγμένου όρου. Συγκεκριμένα, κάθε σύνολο hyperonym synonym, του οποίου η τομή με το σύνολο των επιλεγμένων όρων-εκπροσώπων αποτελούσε σύνολο διάφορο του κενού, επι-

λέχθηκε και εξετάστηκε με σκοπό των εμπλουτισμό της λίστας των επιλεγμένων όρων-εκπροσώπων κάθε κατηγορίας. Για την υποστήριξη γλωσσών πέραν της αγγλικής θα μπορούσαν να χρησιμοποιηθούν μεταφράσεις των όρων-εκπροσώπων ή να προσδιοριστούν ακολουθώντας μια παρόμοια διαδικασία.

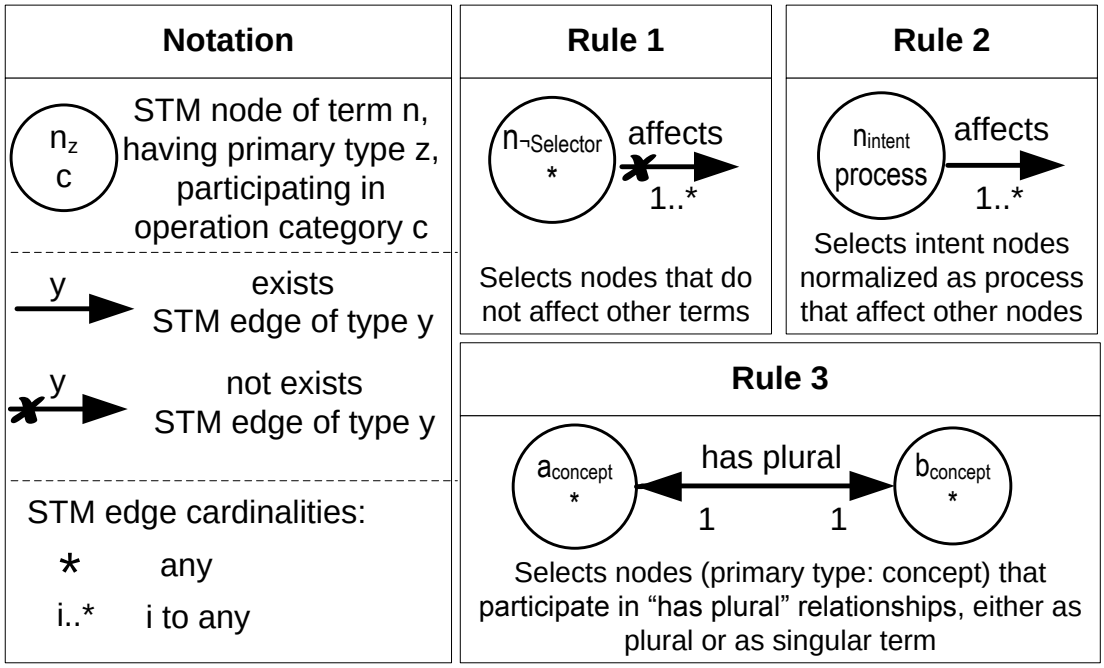
Η διαδικασία κανονικοποίησης παρέχει αποδεκτή ακρίβεια προσθέτοντας περιορισμένη υπολογιστική επιβάρυνση. Εξετάσαμε τα αποτελέσματα της τεχνικής τα οποία παρατίθενται στο Κεφάλαιο 6 και η ακρίβεια αποτιμήθηκε σε 88.02%. Η κύρια συνεισφορά των πληροφοριών που αποκτώνται μέσω της τεχνικής κανονικοποίησης των προθέσεων των λειτουργιών αφορούν στον ορισμό και στην αποτίμηση ευρετικών για την παραγωγή τύπων πόρων, οι οποίες συζητούνται στην Ενότητα 4.2.6.

4.2.5 Βήμα 4: Εξαγωγή του μοντέλου CCE

Παρόλο που οι όροι που περιέχονται στα OTM και το STM παρέχουν το απαραίτητο λεξιλόγιο για να υποδείξουν πόρους, βρίσκονται σε χαμηλότερο επίπεδο αφαίρεσης από εκείνο που χαρακτηρίζει τους πόρους ενός REST API. Η διαφορά αυτή μεταξύ των όρων και των πόρων είναι ορισμένες φορές δύσκολο να αναγνωριστεί. Ωστόσο, αποτελεί σημαντική διάσταση της διαδικασίας εξαγωγής πόρων και οδηγεί στην εισαγωγή ενός ενδιάμεσου επίπεδου, αυτό των *Conceptual Entities*. Ένας τρόπος για να αναδειχθεί η παραπάνω διαφορά είναι εξετάζοντας το γεγονός ότι η πληθικότητα μεταξύ όρων και πόρων ενδέχεται να είναι ένας-προς-πολλούς, καθώς το πλήθος των οντοτήτων που προέρχονται από έναν όρο εξαρτάται από το πλαίσιο στο οποίο χρησιμοποιείται κάθε όρος. Για παράδειγμα, στο SimpleOMS STM (Σχήμα 4.5) αν και ο όρος shipping status (κατάσταση αποστολής) εμφανίζεται μόνο μια φορά, τόσο οι παραγγελίες (*orders*) όσο και κάθε προϊόν παραγγελίας (*order item*) έχει κατάσταση αποστολής, οι οποίες αποτελούν σημασιολογικά διακριτές πληροφορίες και πρέπει κατά συνέπεια να σχετίζονται με δύο διαφορετικούς τύπους πόρων. Επιπρόσθετα, καθώς οι πόροι θεωρούνται διασυνδεδεμένα στοιχεία, είτε δομικά είτε μέσω υπερμέσων, μια σημαντική διάσταση της διαδικασίας εξαγωγής πόρων είναι η αναγνώριση μιας ιεραρχίας μεταξύ τους. Πιο συγκεκριμένα, η εξαγωγή πόρων εστιάζει σε μια από τις πιο σημαντικές σχέσεις μεταξύ πόρων, η οποία είναι η υπαρξιακή εξάρτηση. Ένας πόρος R_i ο οποίος είναι υπαρξιακά εξαρτώμενος από ένα πόρο R_j δε δύναται να υφίσταται χωρίς να υφίσταται ήδη ο πόρος R_j. Τέτοιου

είδους εξαρτήσεις δεν υπάρχουν στο STM, ωστόσο, πρέπει να υπάρχουν σε ένα μοντέλο τύπων πόρων.

Για τους λόγους που συζητήθηκαν παραπάνω, εισάγουμε στο σημείο αυτό το ενδιάμεσο μοντέλο των *Πυρηνικών Εννοιολογικών Οντοτήτων* (Core Conceptual Entities (CCE)) το οποίο θα αποτελέσει τη βάση στην οποία θα σχηματιστεί το τελικό μοντέλο τύπων πόρων. Το μοντέλο των CCE ορίζεται ως ένας επισημειωμένος κατευθυνόμενος ακυκλικός γράφος, κάθε κόμβος του οποίου έχει ένα υποχρεωτικό στοιχείο ετικέτας το οποίο αντιστοιχεί σε ένα όρο του STM. Επιπλέον, ένας CCE κόμβο μπορεί να είναι γονέας ενός ή περισσοτέρων CCE κόμβων μέσω κατευθυνόμενων ακμών που υποδηλώνουν εξάρτηση. Οι CCE κόμβοι προσδιορίζονται από μονοπάτια εξαρτήσεων τα οποία ξεκινούν από κόμβους-πηγές (δηλαδή κόμβους με μηδενικό βαθμό εισόδου) και κάθε μονοπάτι ορίζει μια διακριτή οντότητα. Τέλος, οι CCE κόμβοι χαρακτηρίζονται από ένα σύνολο μεταδεδομένων που υπολογίζονται με βάση τα χαρακτηριστικά των σχετικών όρων του STM. Για παράδειγμα, ένα CCE το οποίο προέρχεται από έναν όρο που έχει μια εισερχόμενη *Has-Plural* σχέση επισημαίνεται ως *plural*. Ανάλογα στοιχεία μεταδεδομένων είναι η ετικέτα *singular* και *filter*.



Σχήμα 4.6: Κανόνες επιλογής CCE στοιχείων

Η εξαγωγή των CCE αποτελείται από δύο βήματα: την *επιλογή στοιχείων* και την *ανάλυση εξαρτήσεων*. Το βήμα επιλογής στοιχείων καθορίζει το υποσύνολο των κόμβων του STM γράφου οι οποίοι θα χρησιμοποιηθούν ως κόμβοι του αντίστοιχου CCE μοντέλου. Το βήμα της ανάλυσης εξαρτήσεων παράγει σχέσεις εξάρτησης μεταξύ των επιλεγμένων κόμβων. Η επιλογή των στοιχείων υλοποιείται συνδυάζοντας ορισμένους κανόνες ανεξάρτητους από την υπηρεσία που προσαρμόζεται, οι οποίοι εφαρμόζονται στο Μοντέλο Όρων Υπηρεσίας. Η ανάλυση των εξαρτήσεων πραγματοποιείται χρησιμοποιώντας τις σχέσεις που υπάρχουν στον STM γράφο και αποσαφηνίζοντας με αλγοριθμικό τρόπο σχέσεις που υποδηλώνουν αντιφατικές εξαρτήσεις. Τόσο οι κανόνες επιλογής στοιχείων όσο και ο αλγόριθμος προσδιορισμού εξαρτήσεων αποτελούν τεχνικές ανεξάρτητες από τα χαρακτηριστικά των υπηρεσιών που εξετάζονται ή των χαρακτηριστικών των πεδίων εφαρμογής στα οποία ανήκουν.

Το Σχήμα 4.6 παρουσιάζει το σύνολο των κανόνων επιλογής που χρησιμοποιείται στο προτεινόμενο περιβάλλον-πλαίσιο. Για παράδειγμα, ο Κανόνας 1 στο Σχήμα 4.6 ορίζει ότι ένας STM κόμβος θα επιλεγεί ως CCE κόμβος, αν ο πρωταρχικός τύπος του -δηλαδή, ο τύπος όρου με τη μέγιστη συχνότητα για τον κόμβο- δεν είναι *Selector* και δεν έχει σχέση *Affects* με οποιοδήποτε άλλο CCE κόμβο. Ομοίως, ο Κανόνας 2 δείχνει ότι ένας STM κόμβος θα επιλεγεί ως CCE κόμβος εφόσον ο πρωταρχικός τύπος του είναι *Intent*, η αντίστοιχη κανονικοποιημένη κατηγορία πρόθεσης είναι *Process* και ο κόμβος έχει μια εξερχόμενη *Affection* σχέση με έναν ή περισσότερους STM κόμβους.

Αφού επιλεγούν οι CCE κόμβοι, ένας αλγόριθμος ανάλυσης εξαρτήσεων καθορίζει μια ιεραρχική δομή μεταξύ των CCE κόμβων. Ο Αλγόριθμος 1 περιγράφει τη διαδικασία αυτή.

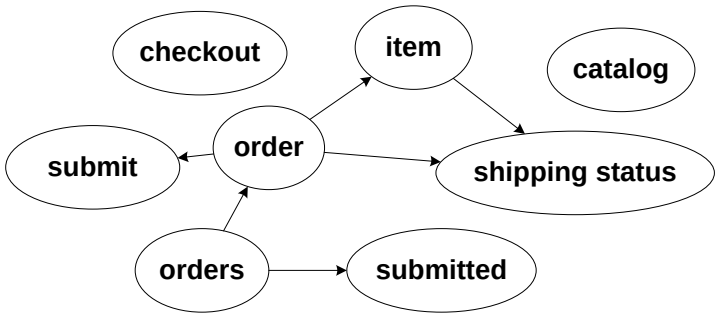
Στις γραμμές 1-8 του Αλγορίθμου 1 κάθε σχέση μεταξύ των επιλεγμένων στοιχείων του STM μεταφράζεται σε εξάρτηση μεταξύ των αντίστοιχων κόμβων του CCE γράφου. Έπειτα, ο γράφος ο οποίος αποτελείται από τα επιλεγμένα στοιχεία και τις εξαρτήσεις τους εξετάζεται για μη τετριμμένες ισχυρά συνεκτικές συνιστώσες (strongly connected components). Μια ισχυρά συνεκτική συνιστώσα ενός κατευθυνόμενου γράφου αποτελεί έναν υπογράφο στον οποίο κάθε κόμβος είναι προσβάσιμος από κάθε άλλο κόμβο του υπογράφου αυτού. Ειδικότερα, η συνάρτηση *stronglyConnectedComponents* αναγνωρίζει όλες τις ισχυρά συνεκτικές συνιστώσες ενός γράφου που ορίζεται από τα σύνολα N (κόμβοι) και D (ακμές), χρησιμοποιώ-

Αλγόριθμος 1: Αλγόριθμος ανάλυσης εξαρτήσεων μεταξύ των στοιχείων CCE

Require: T: STM nodes, E: STM edges, N: selected CCE nodes

$D \leftarrow \emptyset$ # *empty dependencies set* #
for all $e \in E$ **do**
  if $type(e) \in \{Has - Plural, Selects, Affects\}$ **then**
    $D \leftarrow D \cup \{\langle stmToCCENodeMap(e.from), stmToCCENodeMap(e.to)\rangle\}$
  else
    $D \leftarrow D \cup \{\langle stmToCCENodeMap(e.to), stmToCCENodeMap(e.from)\rangle\}$
  end if
end for
while $|SCC = stronglyConnectedComponents(N, D)| > 0$ **do**
  for all $s \in SCC$ **do**
    for all $d_i, d_j \in D_s$ **do**
      if $d_i.from \equiv d_j.to$ **and** $d_i.to \equiv d_j.from$ **then**
        $D_s \leftarrow D_s \backslash \{argmin(d_i.frequency, d_j.frequency)\}$
      end if
    end for
    while s is strongly connected **do**
      $P \leftarrow s.cyclicPaths$ # *a path is a set of dependencies* #
      $I \leftarrow \bigcap D_p, \forall p \in P$ # *intersection of dependencies in paths* #
      $D_s \leftarrow D_s \backslash \{i \in I : max(p \in P : i \in p)\}$
      $D' \leftarrow D' \cup D_s$
    end while
    $D \leftarrow D'$
  end for
end while
return $DAG(N, D)$

ντας τον αλγόριθμο του Tarjan [144] και συλλέγει όλες τις μη τετριμμένες συνιστώσες στο σύνολο SCC. Αν το SCC είναι μη κενό ($|SCC| > 0$) εφαρμόζεται η παρακάτω στρατηγική ανάλυσης. Πρώτον, για κάθε συνιστώσα που έχει αναγνωρισθεί και για κάθε ζεύγος αντίστροφων εξαρτήσεων το οποίο μπορεί αυτή να περιλαμβάνει, απομακρύνεται εκείνη με την ελάχιστη συχνότητα (γραμμές 10-15). Ο συγκεκριμένος υπογράφος ελέγχεται ξανά ως προς την ισχυρή συνεκτικότητα και υπό την προϋπόθεση ότι παραμένει ισχυρά συνεκτική, εφαρμόζεται μια επαναληπτική διαδικασία απομάκρυνσης εξαρτήσεων (γραμμές 16-21). Σε κάθε επανάληψη, αφαιρείται η εξάρτηση η οποία εμφανίζεται πιο συχνά στα κυκλικά μονοπάτια του γράφου. Ο στόχος του αλγόριθμου είναι να απομακρυνθούν όσο το δυνατόν λιγότερες εξαρτήσεις, σεβόμενοι την ίδια στιγμή την παρακάτω προτεραιοποίηση της σπουδαιότητας των εξαρτήσεων: *α*) στην περίπτωση ευθέως αντιφατικών εξαρτήσεων (δηλαδή εξαρτήσεων μεταξύ δύο κόμβων με αντίθετη κατεύθυνση) απομακρύνεται η λιγότερο συχνή εξάρτηση ως λιγότερο σημαντική, *β*) στην περίπτωση έμμεσων κύκλων εξαρτήσεων (ουσιαστικά, συνεκτικών συνιστωσών), όσο μεγαλύτερος είναι ο αριθμός των κυκλικών μονοπατιών στα οποία συμμετέχει μια εξάρτηση, τόσο λιγότερο σημαντική θεωρείται η εξάρτηση αυτή και επομένως απομακρύνεται με μεγαλύτερη προτεραιότητα. Το εξαχθέν CCE μοντέλο για το SimpleOMS απεικονίζεται στο Σχήμα 4.7 το οποίο περιέχει 9 CCE κόμβους.

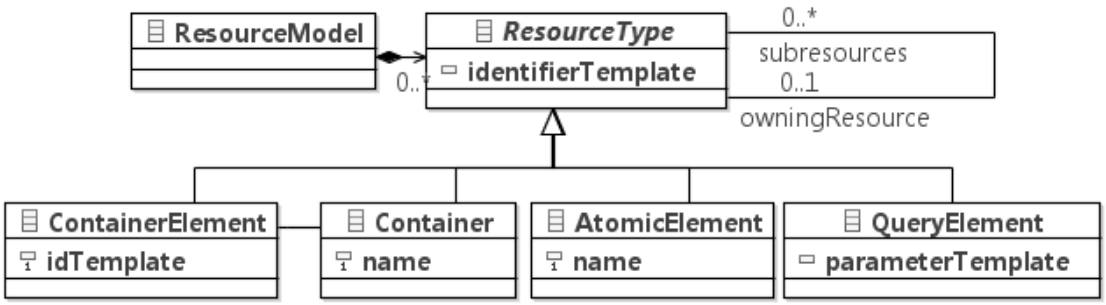


Σχήμα 4.7: Μοντέλο CCE για την υπηρεσία *SimpleOMS*

4.2.6 Βήμα 5: Προσδιορισμός του Μοντέλου Τύπων Πόρων

Μοντελοποίηση

Το τελικό βήμα της διαδικασίας εξαγωγής πόρων είναι η δημιουργία ενός *Μοντέλου Τύπων Πόρων* (*(Resource Types Model (RTM)*). Η φύση ενός πόρου του Παγκόσμιου Ιστού και το θέμα του τι αναπαριστά, αποτέλεσε αντικείμενο μακρύ διαλόγου στην κοινότητα της αρχιτεκτονικής του Παγκόσμιου Ιστού [22]. Ωστόσο, υπάρχουν ερευνητικές προσπάθειες που σχετίζονται με τα συστήματα παροχής υπηρεσιών που βασίζονται στο REST οι οποίες προσπαθούν να εξετάσουν και να αποσαφηνίσουν τις διάφορες πτυχές των RESTful συστημάτων υπηρεσιών μέσω της χρήσης περιβαλλόντων-πλαισίων μεταμοντελοποίησης [135], [133]. Ομοίως με αυτές τις προσεγγίσεις, στην προσέγγισή μας θεωρούμε τους πόρους ως στοιχεία πληροφοριών τα οποία χαρακτηρίζονται από κατάσταση, και τα οποία εκφράζουν διακριτή σημασιολογία. Στο πλαίσιο αυτό, εισάγουμε ένα απλό, ιεραρχικό μεταμοντέλο τύπων πόρων, το οποίο είναι ουδέτερο ως προ τις υπό προσαρμογή εφαρμογές. Στο Σχήμα 4.8 παρουσιάζεται μια διαγραμματική άποψη του προτεινόμενου μεταμοντέλου του RTM.



Σχήμα 4.8: Μεταμοντέλο του RTM

Ειδικότερα, στο μεταμοντέλο που προτείνεται περιλαμβάνεται μια αφαιρετική κλάση `ResourceType` η οποία αναπαριστά τύπους πόρων και η οποία μπορεί να ανήκει στο πολύ έναν τύπο πόρου, ενώ μπορεί να είναι κατέχει ένα συνόλο τύπων πόρων ως πόρους χαμηλότερης ιεραρχίας. Επίσης, τα χαρακτηριστικά στοιχεία της κλάσης αυτής περιλαμβάνουν ένα πρότυπο αναγνωριστικού πόρου το οποίο χρησιμοποιείται για να προσδιορίζει τις κλάσεις τύπων πόρων. Το πρότυπο του αναγνωριστικού τύπου πόρου αποτελείται από τμήματα τα οποία ενσαρκώνονται είτε

στατικά, είτε δυναμικά κατά τον χρόνο εκτέλεσης (τα δυναμικά τμήματα επισημαίνονται μέσω αγκυλών). Ορίζουμε τέσσερις συμπαγείς τύπους πόρων, όλοι εκ των οποίων προέρχονται από την αφαιρετική κλάση `ResourceType`. Οι τύποι πόρων αυτοί είναι οι ακόλουθοι:

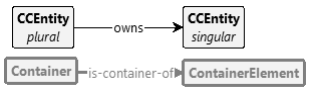
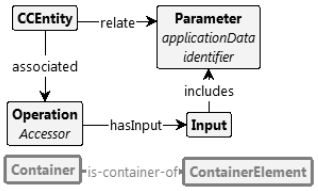
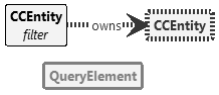
- `Container`: Ο τύπος `Container` υποδηλώνει μια συλλογή πόρων (ίδιου είδους) και φέρει στατική ονομασία. Για παράδειγμα, ο πόρος `/orders/` αναπαριστά μια συλλογή πόρων παραγγελίας.
- `ContainerElement`: Ο τύπος `ContainerElement` υποδηλώνει πολλαπλούς πόρους ίδιου είδους οι οποίοι ανήκουν στον ίδιο πόρο `Container` και έχουν ένα αναγνωριστικό το οποίο μπορεί να χρησιμοποιηθεί ως διακριτικό γνώρισμα μεταξύ των πόρων. Για παράδειγμα, ο τύπος πόρων `/orders/{order.id}/` αναπαριστά πόρους οι οποίοι είναι στοιχεία μιας συλλογής παραγγελιών, ή με άλλα λόγια, ατομικούς πόρους παραγγελίας.
- `AtomicElement`: Ο τύπος `AtomicElement` δηλώνει πόρους που αναπαριστούν πληροφοριακές οντότητες που φέρουν στατικά ονόματα, ή προκαθορισμένα σημεία αλληλεπίδρασης. Για παράδειγμα, ο πόρος `/orders/{order.id}/shipping-status/` αναπαριστά την κατάσταση αποστολής μιας παραγγελίας, η οποία είναι επίσης πόρος. Ένα `AtomicElement` μπορεί επίσης να χρησιμοποιηθεί για να αναπαραστήσει υψηλού αφαιρετικού επιπέδου διαδικασίες οι οποίες συνδέονται με τους πόρους-κατόχους τους, ή με άλλα λόγια, ενέργειες μεταχείρισης οι οποίες ξεφεύγουν από το προκαθορισμένο σύνολο πράξεων που παρέχει το χρησιμοποιούμενο πρωτόκολλο επικοινωνίας (π.χ. HTTP). Για παράδειγμα, ο τύπος πόρου `/vms/{vm.id}/reboot` μπορεί να χρησιμοποιηθεί για να αναπαραστήσει ένα σημείο αλληλεπίδρασης προσανατολισμένο σε πόρους για τη διαχείριση της δυνατότητας επανεκκίνησης ενός πόρου εικονικής μηχανής (virtual machine).
- `QueryElement`: Ο τύπος `QueryElement` υποδηλώνει πόρους οι οποίοι είναι επιλογές, προβολές και γενικότερα παραμετροποιημένες όψεις άλλων πόρων στη βάση συνδυασμών παραμέτρων. Για παράδειγμα, ο τύπος πόρου `/orders/{?status,date}` υποδηλώνει ένα υποσύνολο της συλλογής παραγγελιών το οποίο περιλαμβάνει παραγγελίες συγκεκριμένης κατάστασης και ημερομηνίας δημιουργίας.

Επιπλέον, υπάρχουν δύο τύποι σχέσεων πόρων οι οποίοι ορίζονται στο μεταμοντέλο τύπων πόρων, των οποίων η σημασιολογία εκτείνεται και στα στιγμιότυπα πόρων: *α*) *is owner of* που υποδηλώνει μια άμεση υπαρξιακή εξάρτηση μεταξύ δύο τύπων πόρων και *β*) *is container of* που υποδηλώνει μια σχέση συμπερίληψης μεταξύ ενός πόρου τύπου `Container` και ενός αντίστοιχου `ContainerElement`.

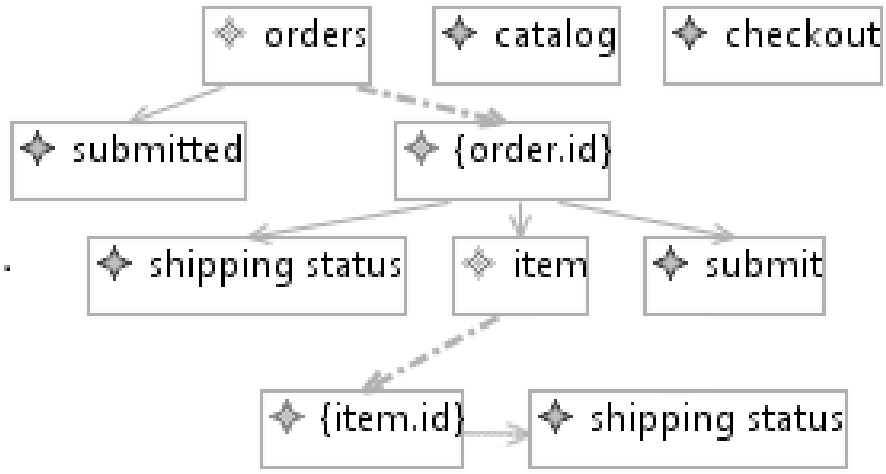
Τεχνική παραγωγής

Η δημιουργία του μοντέλου τύπων πόρων βασίζεται στα CCE μοντέλα και σε συμπληρωματικές πληροφορίες από τα Μοντέλα Υπογραφών και Μοντέλα Όρων. Ο αλγόριθμος παραγωγής του Μοντέλου Τύπων Πόρων (Resource Types Model - RTM) συσχετίζει κάθε λειτουργία σε μια CCE οντότητα, και στη συνέχεια εξετάζει τη συλλογή των CCE. Για κάθε CCE δημιουργείται ένας ή περισσότεροι τύποι πόρων ανάλογα με τα αποτελέσματα αποτίμησης ευρετικών παραγωγής, οι οποίες λαμβάνουν υπόψη το πλαίσιο κάθε οντότητας και τις λειτουργίες που έχουν συσχετιστεί με αυτή. Ο αλγόριθμος διατηρεί στο παραγόμενο μοντέλο την ιεραρχική δομή του CCE μοντέλου μέσω των τύπων σχέσεων πόρων που συζητήθηκαν παραπάνω. Η συσχέτιση μεταξύ CCE και λειτουργιών γίνεται εξετάζοντας OTM μοντέλα ως προς κάθε κομβο του CCE μοντέλου και του πλαισίου του. Στη συνέχεια, για κάθε CCE οι συσχετισμένες λειτουργίες εξετάζονται χρησιμοποιώντας το μοντέλο υπογραφής τους και την κατηγορία πρόθεσης στην οποία έχουν ανατεθεί. Πληροφορίες σχετικά με τις παραμέτρους εισόδου και εξόδου μιας συσχετισμένης λειτουργίας, μαζί με τα μεταδεδομένα που συλλέγονται για την αντίστοιχη CCE οντότητα (π.χ. μια επισημείωση *plural*), εξετάζονται ώστε να καθοριστεί ο τύπος ή οι τύποι του μεταμοντέλου του RTM που θα ενσαρκωθεί. Ο κατάλογος των ευρετικών παραγωγής του RTM που χρησιμοποιούνται, παρουσιάζονται στον Πίνακα 4.6 και απεικονίζονται όπως απλουστευμένοι κανόνες της γλώσσας GROOVE [126].

Πίνακας 4.6: Κανόνες παραγωγής του Resource Types Model

Κανόνας	Περιγραφή	Κανόνας
Κανόνας 1: C/CE pair 1	Μια CCE οντότητα σημασμένη ως plural και η αντίστοιχη οντότητα σημασμένη ως singular οδηγούν στην παραγωγή ενός ζεύγους Container - ContainerElement (C/CE).	
Κανόνας 2: C/CE pair 2	Μια CCE οντότητα συσχετισμένη με μια λειτουργία που έχει κανονικοποιηθεί ως Constructor αναλύεται σε ένα ζεύγος C/CE.	
Κανόνας 3: C/CE pair 3	Μια CCE οντότητα συσχετισμένη με μια λειτουργία που έχει κανονικοποιηθεί ως Accessor της οποίας η είσοδος περιλαμβάνει μια παράμετρο που υποδηλώνει αναγνωριστικό στοιχείο για την οντότητα αναλύεται σε ένα ζεύγος C/CE.	
Κανόνας 4: Query Element	Μια CCE οντότητα η οποία έχει επισημειωθεί ως filter οδηγεί στην παραγωγή ένας Query τύπου πόρου.	
Κανόνας 5: Atomic Element	Όλες οι CCE οντότητες που δεν καλύπτονται από τους προηγούμενους κανόνες.	

Για παράδειγμα, το Σχήμα 4.9 παρουσιάζει το RTM μοντέλο που παρήχθη αυτόματα για την υπηρεσία SimpleOMS. Στο CCE μοντέλο (Σχήμα 4.7), ο CCE κόμβος *orders* είναι επισημειωμένος ως plural και η CCE οντότητα *orders* → *order* είναι επισημειωμένη ως singular. Το πρότυπο αυτό οδηγεί στη δημιουργία ενός `Container` τύπου πόρου (`/orders/`) και ενός `ContainerElement` τύπου πόρου (`/orders/{order.id}/`) οι οποίοι συνδέ-



Σχήμα 4.9: Το εξαχθέν Resource Types Model για την υπηρεσία *SimpleOMS*

ονται με μια σχέση *is container of*, βάσει του πρώτου κανόνα του Πίνακα 4.6. Αντίθετα, στην περίπτωση των στοιχείων των παραγγελιών δεν υπάρχει "items" (plural) CCE κόμβος, αν και θα πρέπει να δημιουργηθεί ένα αντίστοιχο ζεύγος C/CE. Στην περίπτωση αυτήν εφαρμόζεται ο δεύτερος κανόνας του Πίνακα 4.6, καθώς η CCE οντότητα *orders* → *order* → *item* έχει συσχετιστεί με τη λειτουργία *addOrderItem* η οποία έχει κανονικοποιηθεί ως *Constructor*. Κατά συνέπεια δημιουργείται το ζεύγος C/CE: `/orders/{order.id}/item` και `/orders/{order.id}/item/{item.id}/`.

Ο διαχωρισμός μεταξύ εξαγωγής πληροφοριακών οντοτήτων και παραγωγής του τελικού μοντέλου πόρων απλοποιεί τη διαδικασία εξαγωγής πόρων και επιτρέπει στη διαδικασία να είναι ευέλικτη αλλά και να μπορεί να παραμετροποιηθεί. Ως προς την άρση του επιπέδου αφαίρεσης, το πρώτο βήμα εξαγωγής παράγει ένα μοντέλο εννοιολογικών οντοτήτων οι οποίες υποδηλώνουν σημασιολογία της εφαρμογής ή του πεδίου αυτής και το δεύτερο βήμα μετασχηματίζει το μοντέλο αυτό σε ένα λεπτομερές μοντέλο τύπων πόρων, το οποίο είναι σε θέση να αναπαραστήσει τη λειτουργικότητα και τα δεδομένα που παρέχονται από την υπηρεσία.

4.3 Σχέση βασικής αντιστοιχίας

Στο σημείο αυτό της διαδικασίας προσαρμογής, η εξαγωγή πόρων έχει ολοκληρωθεί και έχει εξαχθεί ένα μοντέλο που περιλαμβάνει τύπους πόρων. Τα βήματα εξαγωγής που ακολουθούν παράγουν αναπαραστάσεις οι οποίες αντιστοιχούνται στους τύπους πόρων που έχουν αναγνωριστεί, όπως και ένα σύνολο συσχετίσεων ή αντιστοιχιών μεταξύ στοιχείων της ΠΣΔ διεπαφής και της εξαχθείσας ΠΣΠ διεπαφής, οι οποίες επιτρέπουν την εφαρμογή της προσαρμογής κατά τον χρόνο εκτέλεσης.

Όπως συζητήθηκε νωρίτερα, ο ρόλος μιας διαδικασίας P2R προσαρμογής είναι η έκθεση υπάρχουσας λειτουργικότητας μέσω μιας διεπαφής προσανατολισμένης σε πόρους. Ως εκ τούτου, κάθε λειτουργία της υπηρεσίας η οποία αποτελεί ένα σημείο αλληλεπίδρασης της διεπαφής που δίνεται ως είσοδος, πρέπει να γίνει προσβάσιμη μέσω ενός αντίστοιχου σημείου αλληλεπίδρασης της ΠΣΠ διεπαφής. Ένα τέτοιο ΠΣΠ σημείο αλληλεπίδρασης προσδιορίζεται μέσω ενός κατάλληλου, ως προς τη σημασιολογία της αντίστοιχης λειτουργίας, τύπου πόρου και μιας πράξης μεταχείρισης και πιθανών δεδομένων και μεταδεδομένων ή αναπαραστάσεων που ορίζουν τα μηνύματα που ανταλλάσσονται. Αναφερόμαστε στη σχέση μεταξύ μιας λειτουργίας και ενός (κατάλληλου) τύπου πόρου ως *βασική αντιστοιχία base correspondence*.

4.3.1 Ρόλος και χρησιμοποίηση στη διαδικασία προσαρμογής

Στο Κεφάλαιο 3, Εντότητα 3.1 εισάγαμε δύο τύπους προσαρμογής διεπαφών, συγκεκριμένα τον One Way (OW) τύπο και τον Meet-In-The Middle (MITM) τύπο. Αν και το περιβάλλον-πλαίσιο που παρουσιάζεται στο κεφάλαιο αυτό υλοποιεί μια OW P2R διαδικασία προσαρμογής, σενάρια MITM προσαρμογής μπορούν επίσης να υποστηριχθούν από τα βήματα που ορίζονται σε μια OW προσέγγιση, με την προϋπόθεση ότι το μοντέλο εξαγωγής εφαρμόζεται με διακριτό τρόπο σε σχέση με την εξαγωγή αντιστοιχιών των διεπαφών (interface mappings extraction) και εφόσον δεν υπάρχουν εξαρτήσεις μεταξύ των τεχνικών εξαγωγής αντιστοιχιών διεπαφών και ενδιάμεσων μοντέλων ή πληροφοριών που παράγονται κατά τη διαδικασία εξαγωγής του μοντέλου πόρων. Πιο συγκεκριμένα, για να είναι εφαρμόσιμες οι τεχνικές εξαγωγής αντιστοιχιών διεπαφών σε ένα σενάριο MITM προσαρμογής θα πρέπει να έχουν σχεδιαστεί με τρόπο που να δέχεται ως δεδομένα εισόδου αποκλειστικά τις

προδιαγραφές των δύο διεπαφών. Την ίδια στιγμή, τόσο το βήμα εξαγωγής αναπαραστάσεων όσο και το βήμα εξαγωγής αντιστοιχιών απαιτούν ένα σύνολο σχέσεων βασικής αντιστοιχίας (base correspondence relations) μεταξύ των λειτουργιών της ΠΣΔ υπηρεσίας και των συστατικών του αντίστοιχου μοντέλου τύπων πόρων, καθώς οι σχέσεις αυτές περιορίζουν το εύρος της εξαγωγής στο οποίο εφαρμόζονται οι τεχνικές, καθώς μέσω της βασικής αντιστοιχίας περιορίζεται η γκάμα επιλογών και εναλλακτικών οι οποίες πρέπει να εξεταστούν από τους αλγορίθμους εξόρυξης. Ως εκ τούτου, εισάγουμε τις σχέσεις βασικής αντιστοιχίας ανεξάρτητα και παρουσιάζουμε μια τεχνική για την αναγνώρισή τους. Το αποτέλεσμα αυτής της τεχνικής είναι ένα σύνολο σχέσεων C_{base}, στο οποίο κάθε λειτουργία συσχετίζεται με το πολύ έναν τύπο πόρων.

Καθώς η τεχνική για την αναγνώριση του συνόλου σχέσεων C_{base} θα πρέπει να μπορεί να εφαρμοσθεί ανεξάρτητα από την εξαγωγή πόρων, οι πληροφορίες που που προέρχονται από τη συσχέτιση μιας λειτουργίας με ένα όρο και τελικά με έναν τύπο πόρου δε μπορούν να θεωρηθούν διαθέσιμες. Συνεπώς, γίνεται η θεώρηση ότι ορισμένες τεχνικές που ορίστηκαν στο πλαίσιο της εξαγωγής πόρων, όπως η κατασκευή του Μοντέλου Υπογραφής και η παραγωγή του OTM επαναχρησιμοποιούνται κατά την αναγνώριση των σχέσεων βασικής αντιστοιχίας με στόχο την αποτελεσματική εσωτερική μοντελοποίηση και προ-επεξεργασία της ΠΣΔ διεπαφής που χρησιμοποιείται ως είσοδος. Μια τέτοια απαίτηση επαναχρησιμοποίησης μπορεί να καλυφθεί με κατάλληλη δομοστοιχείωση της υλοποίησης του περιβάλλοντος-πλαισίου προσαρμογής και με κατάλληλη σχεδίαση και παραμετροποίηση μπορεί να εξασφαλιστεί ότι οι τεχνικές των οποίων τα αποτελέσματα επαναχρησιμοποιούνται εκτελούνται μόνο μια φορά κατά τη διαδικασία προσαρμογής.

4.3.2 Αναγνώριση σχέσεων βασικής αντιστοιχίας

Έστω ότι O είναι το σύνολο των λειτουργιών σε μια ΠΣΔ διεπαφή S_P και έστω R το σύνολο των τύπων πόρων σε μια ΠΣΠ διεπαφή S_R. Ορίζουμε ως σχέση *Δ-βασικής αντιστοίχισης* τη $c^{po}_{base} : O \rightarrow R$ η οποία συσχετίζει μια λειτουργία $o \in O$ με έναν τύπο πόρου $r \in R$, δηλαδή $c^{po}_{base}(o) = r$. Το σύνολο $C_{base} = \{(o, c^{po}_{base}(o)) \mid o \in O\}$ δηλώνει όλες τις σχέσεις βασικής αντιστοιχίας μεταξύ των S_P και S_R.

Αρχικά, για κάθε λειτουργία $o \in O$, υπολογίζεται ένα μοντέλο όρων OTM_o, του

οποίου το σύνολο των όρων που έχουν αναγνωριστεί, εξαιρώντας τον όρο τύπου *Intent* επισημειώνεται ως OT_o. Στη συνέχεια, το σύνολο των παραμέτρων εισόδου της λειτουργίας o εξετάζεται ως προς το αν υπάρχουν παράμετροι που υποδεικνύουν αναγνωριστικά στοιχείων χρησιμοποιώντας ευρετικές (π.χ. το όνομα της παραμέτρου να έχει μετάθημα το "id" και να περιλαμβάνει όρους του OT_o). Σε περίπτωση που υπάρχουν τέτοιες παράμετροι τα ονόματά τους προστίθενται στο OT_o. Έπειτα, για κάθε τύπο πόρου r στο RTM, το μονοπάτι το οποίο ξεκινά από το ριζικό στοιχείο του δένδρου στο οποίο ανήκει το r και τελειώνει στο r χρησιμοποιείται για να δημιουργηθεί ένα σύνολο *ονομάτων πόρων* για το r, το RN_r. Το RN_r αποτελείται από ετικέτες που φέρουν οι τύποι πόρων στο μονοπάτι που έχει υπολογιστεί για το r, όπου μια ετικέτα τύπου πόρου είναι η τιμή των χαρακτηριστικών ονομασίας του τύπου πόρου, δηλαδή του *idTemplate* για στοιχεία `ContainerElement`, του *name* για στοιχεία `Container`, του *name* για στοιχεία `AtomicElement`, του *parameterTemplate* για στοιχεία `QueryElement` αφού αφαιρεθούν ειδικοί χαρακτήρες του μοτίβου αναγνωριστικών εφόσον αυτοί υπάρχουν (π.χ. η ετικέτα για ένα *idTemplate* με τιμή *{order.id}* είναι *orderid*).

Χρησιμοποιώντας τα παραπάνω, ορίζουμε την Ομοιότητα Βασικής Αντιστοιχίας (*base correspondence similarity (BCS)*) $BCS : O \times R \rightarrow [0, 1]$ ως

$$BCS(o, r) = \frac{|OT_o \cap RN_r|}{|OT_o \cup RN_r|}$$

Ο ορισμός του BCS βασίζεται στην υπόθεση ότι όσο μεγαλύτερη είναι η επικάλυψη μεταξύ της ορολογίας που χρησιμοποιείται για τον προσδιορισμό μιας λειτουργίας και των ετικετών που χρησιμοποιούνται για έναν τύπο πόρου και τους ιεραρχικά ανώτερους τύπους πόρων του, τόσο μεγαλύτερη είναι η πιθανότητα ότι τα o και r σχετίζονται με την ίδια πληροφοριακή οντότητα.

Συνεπώς, η σχέση βασικής αντιστοιχίας για μια λειτουργία $o \in O$, $c^{po}_{base}(o)$, υπολογίζεται ως εξής:

$$c^{po}_{base}(o) = \arg\max_{r \in R} BCS(o, r)$$

Τελικά, χρησιμοποιώντας το σύνολο C_{base} μπορούμε να υπολογίσουμε εύκολα τη σχέση *Π-αντιστοίχισης* $c^{ro}_{base} : R \rightarrow \mathcal{P}(O)$ η οποία επιστρέφει ένα υποσύνολο του O το οποίο σχετίζεται με έναν τύπο πόρων $r \in R$, δηλαδή, $c^{ro}_{base}(r) = \{o\}, \forall o \in O \mid (o, r) \in C_{base}$.

Για παράδειγμα, έστω $s = addOrderItem$. Τότε $OT_s = \{order, item, orderid\}$. Δεδομένων των παραπάνω, τα ονόματα πόρου για κάθε τύπο πόρου και η αντίστοιχη τιμή BCS υπολογίζονται ως εξής:

- Το σύνολο RN για το `/orders/` είναι $RN_{r_1} = \{orders\}$, οπότε $BCS(s, r_1) = 0$.
- Το σύνολο RN για το `/orders/{?submitted}` είναι $RN_{r_2} = \{orders, submitted\}$, οπότε $BCS(s, r_2) = 0$.
- Το σύνολο RN για το `/orders/{order.id}/` είναι $RN_{r_3} = \{orders, orderid\}$, οπότε $BCS(s, r_3) = 1 / 4 = 0.25$.
- Το σύνολο RN για το `/orders/{order.id}/shipping-status/` είναι $RN_{r_4} = \{orders, orderid, shippingstatus\}$, οπότε $BCS(s, r_4) = 1 / 5 = 0.2$.
- Το σύνολο RN για το `/orders/{order.id}/submit/` είναι $RN_{r_5} = \{ orders, orderid, submit \}$, οπότε $BCS(s, r_5) = 1 / 5 = 0.2$.
- Το σύνολο RN για το `/orders/{order.id}/item/` είναι $RN_{r_6} = \{orders, orderid, item\}$, οπότε $BCS(s, r_6) = 2 / 4 = 0.5$.
- Το σύνολο RN για το `/orders/{order.id}/item/{item.id}/` είναι $RN_{r_7} = \{orders, orderid, item, itemid\}$, οπότε $BCS(s, r_7) = 2 / 5 = 0.4$.
- Το σύνολο RN για το `/orders/{order.id}/item/{item.id}/shipping-status/` είναι $RN_{r_8} = \{orders, orderid, item, itemid, shippingstatus\}$, οπότε $BCS(s, r_8) = 2 / 6 = 0.33$.
- Το σύνολο RN για το `/checkout/` είναι $RN_{r_9} = \{checkout\}$, οπότε $BCS(s, r_9) = 0$.
- Το σύνολο RN για το `/catalog/` είναι $RN_{r_{10}} = \{catalog\}$, οπότε $BCS(s, r_{10}) = 0$.

Κατά συνέπεια, $c^{po}_{base}(s) = r_6$ και ως εκ τούτου αναγνωρίζεται μια σχέση βασικής αντιστοιχίας μεταξύ της λειτουργίας *addOrderItem* και του τύπου πόρου `/orders/{order.id}/item/`.

Με βάση τα παραπάνω, ο Πίνακας 4.7 περιλαμβάνει το σύνολο των σχέσεων βασικής αντιστοιχίας οι οποίες αναγνωρίστηκαν για την ΠΣΔ διεπαφή του SimpleOMS (βλ. Πίνακα 4.1) και το RTM που έχει εξαχθεί (Σχήμα 4.9).

Πίνακας 4.7: Σχέσεις βασικών ανταποκρίσεων για το *SimpleOMS*

Λειτουργία	Τύπος πόρου (πρότυπο αναγνωριστικού)
createOrder	*/orders/*
getOrder	*/orders/{order.id}/*
removeOrder	*/orders/{order.id}/*
submitOrder	*/orders/{order.id}/submit/*
addOrderItem	*/orders/{order.id}/item/*
removeOrderItem	*/orders/{order.id}/item/{item.id}/*
getOrderShippingStatus	*/orders/{order.id}/shipping-status/*
checkout	*/checkout/*
searchCatalog	*/catalog/*
getSubmittedOrders	*/orders/{?submitted}*
getOrderItemShippingStatus	*/orders/{order.id}/item/{item.id}/shipping-status/*

.

4.4 Εξαγωγή αναπαραστάσεων

Το μοντέλο πόρων παρέχει τη βάση για την έκθεση μιας ΠΣΠ διεπαφής μέσω μιας P2R προσαρμογής της αντίστοιχης ΠΣΔ διεπαφής. Όπως συζητήθηκε στο Κεφάλαιο 2, Ενότητα 2.3.2, ένα μοντέλο πόρων περιλαμβάνει πόρους, σχέσεις μεταξύ των πόρων, αναγνωριστικά πόρων και αναπαραστάσεις πόρων. Το RTM που παράγεται από τη διαδικασία εξαγωγής πόρων η οποία περιγράφεται στην Ενότητα 4.2 προσδιορίζει τα πρώτα τρία. Στην ενότητα αυτή, παρουσιάζουμε μια τεχνική για την εξαγωγή αναπαραστάσεων πόρων για τους τύπους πόρων που περιλαμβάνονται στο RTM χρησιμοποιώντας πληροφορίες σχετικά με τις υπογραφές των λειτουργιών της υπηρεσίας οι οποίες μοντελοποιούνται στα αντίστοιχα Μοντέλα Υπογραφής.

Στο σημείο αυτό, θα πρέπει να σημειωθεί ότι η προτεινόμενη προσέγγιση εξαγωγής πόρων στοχεύει στην αναγνώριση ενός μοντέλου τύπων αναπαραστάσεων υψηλού αφαιρετικού επιπέδου και δεν αντιμετωπίζει ζητήματα αντιστοίχισης των τύπων αναπαραστάσεων που εξορύσσονται σε υφιστάμενους τύπους μέσων, κάτι

που θεωρείται ως ανεξάρτητο πρόβλημα το οποίο υπερβαίνει το υπό μελέτη πρόβλημα προσαρμογής. Με άλλα λόγια, στόχος της τεχνικής εξαγωγής αναπαραστάσεων που παρουσιάζεται στην παρούσα ενότητα είναι η αναγνώριση των στοιχείων που πρέπει να συνθέτουν μια αναπαράσταση, κατά πόσο αυτά φέρουν πληροφορίες σχετικά με την κατάσταση ενός πόρου ή μεταδεδομένα και υπό ποιες συνθήκες θα πρέπει να παρέχονται ή να αναμένονται σε μια αλληλεπίδραση.

Στις επόμενες παραγράφους, αρχικά εισάγουμε ορισμένες επεκτάσεις στο μεταμοντέλο του RTM έτσι ώστε να καλύπτει πληροφορίες σχετικές με αναπαραστάσεις των πόρων. Στη συνέχεια, παρουσιάζουμε μια τεχνική η οποία κατασκευάζει στοιχεία αναπαραστάσεων για κάθε τύπο πόρου που περιλαμβάνεται σε ένα RTM, μέσω ανάλυσης των μοντέλων υπογραφών των λειτουργιών με τις οποίες σχετίζεται κάθε τύπος πόρου.

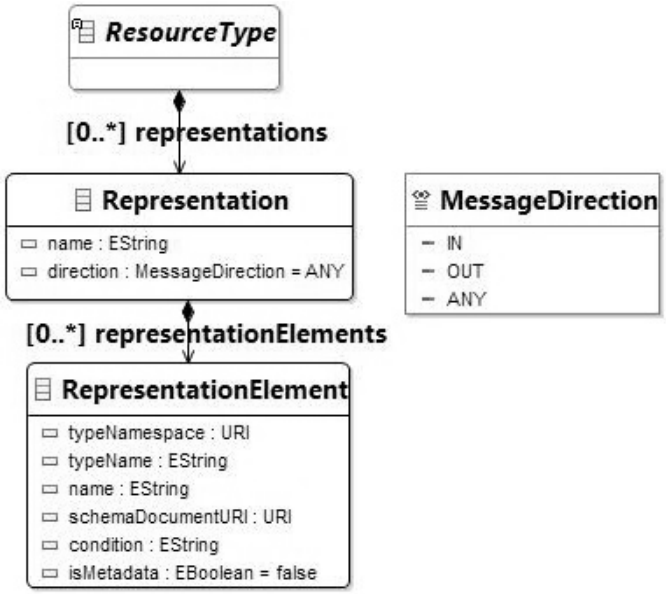
4.4.1 Τύπος αναπαράστασης

Το Σχήμα 4.10 απεικονίζει τις κλάσεις που προστίθενται στο μεταμοντέλο τύπων πόρων (Σχήμα 4.8) ώστε να περιλαμβάνονται πληροφορίες αναπαραστάσεων. Συγκεκριμένα, η κλάση `Representation` δηλώνει τύπους αναπαράστασης και σχετίζεται με την αφηρημένη κλάση `ResourceType` με μια σχέση συμπερίληψης, καθώς η πρόσβαση και η δυνατότητα μεταχείρισης ενός πόρου μπορεί να γίνει μέσω ενός πλήθους αναπαραστάσεων. Η κλάση `Representation` περιλαμβάνει ένα χαρακτηριστικό *name* το οποίο μπορεί να χρησιμοποιηθεί για την αναγνώριση του τύπου αναπαράστασης και ένα χαρακτηριστικό *direction* το οποίο δηλώνει αν η αναπαράσταση μπορεί να χρησιμοποιηθεί σε εισερχόμενα μηνύματα, σε εξερχόμενα μηνύματα ή και στα δύο. Επιπλέον, η κλάση `Representation` σχετίζεται με την κλάση `RepresentationElement` με μια σχέση συμπερίληψης (*representationElements*), με την πρώτη να είναι η κλάση συμπερίληψης της δεύτερης. Η κλάση `RepresentationElement` μπορεί να χρησιμοποιηθεί για τον προσδιορισμό τύπων δεδομένων και μεταδεδομένων οι οποίοι συνθέτουν μια αναπαράσταση. Πιο συγκεκριμένα, η κλάση `RepresentationElement` περιλαμβάνει τα παρακάτω χαρακτηριστικά:

- *name*, το οποίο προσδιορίζει ένα όνομα για το στοιχείο αναπαράστασης και το οποίο χρησιμοποιείται για τη μοναδική αναγνώριση του στοιχείου αναπα-

ράστασης στο πλαίσιο του `Representation` στο οποίο ανήκει,

- *typeName* και *typeNamespace*, το οποίο προσδιορίζει το τύπο του στοιχείου,
- *schemaDocumentURI*, το οποίο προσδιορίζει το URI του σχήματος του εγγράφου που περιλαμβάνει τον τύπο του στοιχείου,
- *condition*, το οποίο προσδιορίζει μια έκφραση συνθήκης για το αν και πότε το στοιχείο αναπαράστασης θα πρέπει να είναι τμήμα της αναπαράστασης -σαφώς, όταν δίνεται ως έκφραση η τιμή `true` το στοιχείο θεωρείται απαραίτητο, δηλαδή θα πρέπει να περιλαμβάνεται πάντα,
- *isMetadata*: το οποίο προσδιορίζει κατά πόσο το στοιχείο αποτελεί μεταδεδομένο αναπαράστασης (`true`) ή δεδομένο αναπαράστασης (`false`).



Σχήμα 4.10: Επεκτάσεις στο μεταμοντέλο του RTM (Σχήμα 4.8) που επιτρέπουν τη μοντελοποίηση αναπαραστάσεων

4.4.2 Εξαγωγή αναπαραστάσεων

Οι παράμετροι εισόδου και εξόδου των λειτουργιών μιας υπηρεσίας χρησιμοποιούνται για την κατασκευή των δυναμικών τμημάτων των μηνυμάτων που ανταλλάσσονται κατά τις κλήσεις και τα οποία μεταφέρουν ειδική ως προς την κλήση πληροφορία μέσω των τιμών που ανατίθεται στις παραμέτρους τους. Κατά συνέπεια,

οι παράμετροι εισόδου και εξόδου σχετίζονται εννοιολογικά με τις αναπαραστάσεις πόρων καθώς οι αναπαραστάσεις εξυπηρετούν ένα παρόμοιο σκοπό για τις ΠΣΠ αλληλεπιδράσεις. Στην παρούσα ενότητα χρησιμοποιούμε αυτή τη σημασιολογική σχέση μεταξύ των παραμέτρων και των αναπαραστάσεων και προτείνουμε μια τεχνική εξαγωγής αναπαραστάσεων η οποία επεξεργάζεται Μοντέλα Υπογραφών των λειτουργιών της υπηρεσίας (Ενότητα 4.2.2) έτσι ώστε να παράγει στοιχεία αναπαραστάσεων για κάθε τύπο πόρου ενός RTM.

Αλγόριθμος 2: Εξαγωγή αναπαραστάσεων για έναν τύπο πόρου r

Require: C_{base}: base correspondence relations, r: selected resource type.
1: $rep.elements \leftarrow \emptyset$
2: **for all** $o \in c^{ro}_{base}(r)$ **do**
3: **for all** $p \in o.input$ **do**
4: IncludeElement(rep.elements, o, p, IN)
5: **end for**
6: **for all** $p \in o.output$ **do**
7: IncludeElement(rep.elements, o, p, OUT)
8: **end for**
9: **end for**
10: **return** *rep*

Ο αλγόριθμος για την εξαγωγή αναπαραστάσεων ενός τύπου όρου $r \in R$ παρέχεται στον Αλγόριθμο 2. Όπως συζητήθηκε παραπάνω, η εξαγωγή αναπαραστάσεων απαιτεί να είναι διαθέσιμο ένα σύνολο σχέσεων βασική αντιστοιχίας C_{base}. Έπειτα, για κάθε λειτουργία που σχετίζεται με το r μέσω μιας σχέσης βασικής αντιστοιχίας, εξετάζονται οι παράμετροι εισόδου και εξόδου της έτσι ώστε να δημιουργηθεί ένα στοιχείο στην κατάλληλη αναπαράσταση, ή για την ενημέρωση ενός στοιχείου που είχε αναγνωριστεί προηγουμένως σχετικά με τα χαρακτηριστικά *condition* και *isMetadata*, όπως υλοποιείται από τη διαδικασία `IncludeElement` του Αλγόριθμου 3. Με βάση τον αλγόριθμο εξαγωγής αναπαραστάσεων, η τιμή του χαρακτηριστικού *condition* ενός στοιχείου αναπαράστασης είναι μια λογική διάζευξη περισσότερο ειδικών εκφράσεων αποτελούμενων από δύο κυριολεκτήματα (literals), εκ των οποίων το ένα προσδιορίζει μια συνθήκη ως προς την πράξη που εφαρμόζεται στον πόρο (`ACTION`) και το άλλο προσδιορίζει μια συνθήκη σχετική με την κατεύθυνση του μηνύματος (`DIRECTION`).

Αλγόριθμος 3: `IncludeElement`

Require: *elements*: set of representation elements, *o*: operation o, *p*: parameter, *dir*: direction (IN/OUT)

1: **if** $\nexists e \in elements$ such that $e.name = p.name$, $e.typeName = p.typeName$, $e.typeNamespace = p.typeNamespace$ **then**

2: $e \Leftarrow$ new representation element with

$e.name \Leftarrow p.name$

$e.typeName \Leftarrow p.typeName$

$e.typeNamespace \Leftarrow p.typeNamespace$

$e.schemaDocumentURI \Leftarrow p.schemaDocumentURI$

$e.condition \Leftarrow ACTION = ic2a(o.intentCategory) \wedge$

$DIRECTION = dir$

$e.isMetadata \Leftarrow (p.class = metadata)$

3: $elements \Leftarrow elements \cup \{e\}$

4: **else**

5: $e.condition \Leftarrow e.condition \vee ACTION = ic2a(o.intentCategory) \wedge DIRECTION = dir$

6: $e.isMetadata \Leftarrow (e.isMetadata \wedge p.class = metadata)$

7: **end if**

Πιο συγκεκριμένα, η έκφραση `ACTION = ic2a(o.intentCategory)` στη διαδικασία `IncludeElement` προσδιορίζει ότι η πράξη για την οποία το στοιχείο αναπαράστασης θα πρέπει να υπάρχει στην ανταλλασσόμενη αναπαράσταση, θα πρέπει να ισούται με την πράξη η οποία έχει αντιστοιχιστεί στην κατηγορία κανονικοποίησης του όρου *Intent* της λειτουργίας. Για παράδειγμα, έστω *o* η λειτουργία `addOrderItem` της οποίας ο *Intent* όρος (`add`) έχει κανονικοποιηθεί στην κατηγορία κανονικοποίησης *Constructor*. Η συνάρτηση `ic2a` (intentCategory-to-action), η οποία ορίζεται ως $ic2a : I \rightarrow A$, όπου I είναι το σύνολο των κατηγοριών κανονικοποίησης των όρων *Intent* (βλέπε Ενότητα 4.2.4) και A το σύνολο των πράξεων χειρισμού στην ΠΣΠ διεπαφή και η οποία αντιστοιχίζει την κατηγορία *Constructor* στην πράξη δημιουργίας, ή σε όρους HTTP, στη μέθοδο `POST`. Η συνάρτηση *ic2a* αντιστοιχίζει τις κατηγορίες πρόθεσης σε πράξεις όπως περιγράφεται στον Πίνακα 4.8.

Πίνακας 4.8: Ανταποκρίσεις κατηγοριών κανονικοποίησης πρόθεσης, πράξεων μεταχείρισης και μεθόδων του HTTP

Κατηγορία κανονικοποίησης	Πράξη μεταχείρισης	HTTP
Constructor	Create	POST
Destructor	Delete	DELETE
Accessor	Read	GET
Mutator	Update	PUT/PATCH
Query	Read	GET
Investigator	Read	GET
Process	Process	POST

Η έκφραση `DIRECTION = dir`, περιορίζει το ποια στοιχεία αναπαράστασης θα συμπεριληφθούν στην αναπαράσταση ανάλογα με την κατεύθυνση (`IN` ή `OUT`) του μηνύματος. Η τιμή `IN` δηλώνει κατεύθυνση από τον πελάτη προς τον εξυπηρετητή και η τιμή `OUT` δηλώνει κατεύθυνση μηνύματος από τον εξυπηρετητή προς τον πελάτη. Τέλος, στο χαρακτηριστικό `isMetadata` ενός στοιχείου αναπαράστασης ανατίθεται η τιμή `true` αν η κλάση όλων των παραμέτρων που αντιστοιχούν στο στοιχείο αναπαράσταση είναι `metadata`.

Για παράδειγμα έστω ότι για τον τύπο πόρου `/orders/order.id/item/` έχει εξορυχθεί ο παρακάτω τύπος αναπαράστασης:

```
<representations>
    <representationElements typeNamespace="http://simpleoms.sample.
        softeng.ece.ntua.gr/" typeName="OrderItem" name="orderItem"
        schemaDocumentURI="http://simpleoms.sample.softeng.ece.ntua.gr/
        SimpleOMS.wsdl" condition="ACTION = Create ^ DIR = IN"
        isMetadata="false"/>
    <representationElements typeNamespace="http://simpleoms.sample.
        softeng.ece.ntua.gr/" typeName="Auth" name="auth"
        schemaDocumentURI="http://simpleoms.sample.softeng.ece.ntua.gr/
        SimpleOMS.wsdl" condition="ACTION = Create ^ DIR = IN"
        isMetadata="true"/>
    <representationElements typeNamespace="http://www.w3.org/2001/
        XMLSchema" typeName="string" name="result" schemaDocumentURI="
        http://simpleoms.sample.softeng.ece.ntua.gr/SimpleOMS.wsdl"
        condition="ACTION = Create ^ DIR = OUT" isMetadata="false"/>
</representations>
```

Ο τύπος πόρου αυτός έχει μόνο μια σχέση βασικής αντιστοιχίας με τη λειτουργί-

α *addOrderItem* (Πίνακας 4.7) και συνεπώς τα στοιχεία αναπαράστασης που έχουν προσδιοριστεί αντιστοιχούν στις παραμέτρους εισόδου και εξόδου της λειτουργίας. Η λειτουργία *addOrderItem* κανονικοποιείται ως Constructor και επομένως όλα τα παραπάνω στοιχεία αναπαράστασης συμμετέχουν σε *Create* αλληλεπιδράσεις με τον πόρο, όπως προκύπτει από το χαρακτηριστικό *condition* (`ACTION = Create`) που έχουν. Επίσης, δύο από τα στοιχεία αναμένεται να υπάρχουν σε μηνύματα εισόδου (auth, orderItem), ενώ ένα θα πρέπει να εμφανίζεται σε μηνύματα εξόδου (result), όπως φαίνεται από τις αντίστοιχες συνθήκες `DIR`. Τέλος, ένα από τα στοιχεία αναπαράστασης (auth) αφορά σε μεταδεδομένα, όπως προσδιορίζεται από το χαρακτηριστικό *isMetadata*, ακολουθώντας την κατηγοριοποίηση της παραμέτρου στην οποία στηρίχθηκε η αναγνώρισή του.

4.5 Εξαγωγή αντιστοιχιών διεπαφών

Μετά την εξαγωγή και τη συσχέτιση των αναπαραστάσεων με τύπους πόρων, τα στοιχεία του εμπλουτισμένου μοντέλου πόρων πρέπει να συσχετιστούν με στοιχεία της ΠΣΔ διεπαφής μέσω κανόνων αντιστοίχισης, ή απλώς, *αντιστοιχιών* (*mappings*), έτσι ώστε RESTful αιτήματα και αποκρίσεις να μπορούν να μετασχηματιστούν με σωστό τρόπο σε κλήσεις και αποτελέσματα λειτουργιών κατά τον χρόνο εκτέλεσης από το P2R δομοστοιχείου προσαρμογής. Συνεπώς, στην ενότητα αυτή περιγράφουμε ορισμένες τεχνικές για τον προσδιορισμό ή την αναγνώριση τέτοιων αντιστοιχιών οι οποίες συσχετίζουν στοιχείων των δύο μοντέλων διεπαφής. Αφού αναγνωριστούν οι αντιστοιχίες, συγχωνεύονται με το εμπλουτισμένο μοντέλο πόρων συνθέτοντας ένα ενιαίο αρχείο προδιαγραφής της προσαρμογής (adaptation prescription file), το οποίο καταναλώνεται στη συνέχεια από τον προσαρμογέα του χρόνου εκτέλεσης. Επιπλέον, όπως συζητείται στην Ενότητα 4.3, οι τεχνικές εξαγωγής αντιστοιχιών διεπαφών που προτείνονται στην παρούσα ενότητα έχουν σχεδιαστεί ώστε να είναι εφαρμόσιμες και στην περίπτωση της MITM προσαρμογής.

Με στόχο τη διευκόλυνση της παρουσίασης των τεχνικών αναγνώρισης αντιστοιχιών, κατηγοριοποιούμε τις αντιστοιχίες στοιχείων διεπαφής στις παρακάτω κατηγορίες:

- *Πρόθεσης* (*Intention*), η οποία συσχετίζει τις ΠΣΔ λειτουργίες σε πράξεις χειρισμού των ΠΣΠ διεπαφών, με βάση την πρόθεση της αλληλεπίδρασης.

- *Τελικού Σημείου* (*Endpoint*), η οποία συσχετίζει ΠΣΔ και ΠΣΠ στοιχεία τελικών σημείων των διεπαφών τα οποία επιτρέπουν την αλληλεπίδραση με τις υπηρεσίες.
- *Μηνύματος* (*Message*), η οποία συσχετίζει τα συνθετικά στοιχεία των μηνυμάτων όπως προσδιορίζονται στα δύο υποδείγματα.

4.5.1 Αντιστοιχίες πρόθεσης

Ο στόχος των αντιστοιχιών πρόθεσης (intention mappings) είναι η συσχέτιση μιας λειτουργίας με μια πράξη χειρισμού πόρου με βάση τη σημασιολογία της πρόθεσης μιας λειτουργίας, θεωρώντας ότι η σημασιολογία των πράξεων χειρισμού πόρων είναι καλά ορισμένη και κατανοητή.

Αρχικά, η τεχνική κανονικοποίησης πρόθεσης που παρουσιάστηκε στην Ενότητα 4.2 εφαρμόζεται έτσι ώστε να επιστραφεί η κατηγορία της κανονικοποιημένης πρόθεσης κάθε λειτουργίας. Το βήμα αυτό μπορεί να μοντελοποιηθεί ως μια συνάρτηση $o2ic : O \rightarrow I$, όπου O το σύνολο των λειτουργιών της ΠΣΔ υπηρεσίας και I το σύνολο των κατηγοριών πρόθεσης. Στη συνέχεια, χρησιμοποιείται η συνάρτηση $ic2a : I \rightarrow A$ που εισηχθεί στην Ενότητα 4.4 και η οποία συσχετίζει κατηγορίες με πράξεις χειρισμού, έτσι ώστε να αντληθεί η πράξη χειρισμού με βάση την κατηγορία της κανονικοποιημένης πρόθεσης. Χρησιμοποιώντας τα παραπάνω, μπορούμε να ορίσουμε τη συνάρτηση $intention_{mp} : O \rightarrow A$, ως $intention_{mp}(o) = (ic2a \circ o2ic)(o)$. Για παράδειγμα, έστω ότι $s = addOrderItem$ μια λειτουργία του SimpleOMS. Τότε $o2ic(s) = Constructor$ και $ic2a(Constructor) = Create$. Συνεπώς, $intention_{mp}(s) = Create$.

Το σύνολο $Intention_{mp} = \{(o, intention_{mp}(o)) \mid o \in O\}$ αποτελεί το σύνολο των αντιστοιχιών προθεσης και χρησιμοποιείται για τον υπολογισμό του συνόλου των αντιστοιχιών τελικών σημείων που συζητούνται στην επόμενη υποενότητα.

4.5.2 Αντιστοιχίες τελικού σημείου και αναγνωριστικού

Οι αντιστοιχίες τελικών σημείων συσχετίζουν σημεία αλληλεπίδρασης μεταξύ των ΠΣΔ και των ΠΣΠ διεπαφών. Πιο συγκεκριμένα, ως σημεία αλληλεπίδρασης μιας

ΠΣΔ διεπαφής S_P θεωρούμε το σύνολο των λειτουργιών O της υπηρεσίας και ως σημεία αλληλεπίδρασης μιας ΠΣΠ διεπαφής S_R θεωρούμε τα ζεύγη τύπων πόρων και πράξεων χειρισμού, δηλαδή $(r, a), \forall r \in R, a \in A$. Με βάση τα παραπάνω, ορίζουμε μια συνάρτηση αντιστοίχισης τελικών σημείων $endpoint_{mp} : O \rightarrow R \times A$ η οποία λαμβάνει ως είσοδο μια λειτουργία $o \in O$ και επιστρέφει ως έξοδο ένα ζεύγος πόρου-πράξεις (r, a). Οι αντιστοιχίες τελικών σημείων μπορούν να υπολογιστούν χρησιμοποιώντας τη Δ-βασική αντιστοίχιση, c^{po}_{base} (Ενότητα 4.3), η οποία συσχετίζει μια λειτουργία o με έναν πόρο r και συνδυάζοντας το αποτέλεσμά της με $intention_{mp}(o)$ ώστε να αντληθεί μια κατάλληλη πράξη χειρισμού a. Συνεπώς, για το $endpoint_{mp}$ ισχύει: $endpoint_{mp}(o) = (c^{po}_{base}(o), intention_{mp}(o))$.

Για παράδειγμα, αν $s = addOrderItem$ του SimpleOMS, τότε $endpoint_{mp}(s) = (/orders/\{order.id\}/item/, Create)$, χρησιμοποιώντας τις τιμές που έχουν υπολογιστεί για τα $c^{po}_{base}(s)$ και $intention_{mp}(s)$ στην Ενότητα 4.3.2 και στην Ενότητα 4.5.1, αντίστοιχα.

Το σύνολο $Endpoint_{mp} = \{(o, endpoint_{mp}(o)) \mid o \in O\}$ δηλώνει το σύνολο των αντιστοιχιών τελικών σημείων που πορίζονται για τα S_P και S_R και χρησιμοποιείται από τον P2R προσαρμογέα χρόνου εκτέλεσης για να επιλεχθεί η λειτουργία του S_P η οποία θα πρέπει να κληθεί, ανάλογα με τον πόρο και την πράξη που αφορούσε το ΠΣΠ αίτημα.

Αφού αναγνωριστούν οι αντιστοιχίες τελικών σημείων, τα δυναμικά τμήματα των προτύπων των αναγνωριστικών που φέρουν οι τύποι πόρων πρέπει να συσχετιστούν με παραμέτρους εισόδου των συσχετισμένων λειτουργιών. Συγκεκριμένα, για κάθε $e = (o, (r, a)) \in Endpoint_{mp}$ το πρότυπο αναγνωριστικού $r.identifierTemplate$ που έχει προσδιοριστεί αναλύεται ως προς τα δυναμικά του τμήματα. Έστω ότι το DP_e δηλώνει το σύνολο των δυναμικών τμημάτων, τότε για κάθε $dp \in DP_e$ επιλέγεται η παράμετρος $p \in o.input$ της οποίας η κλάση $p.class$ είναι `application data` και της οποίας το όνομα έχει την ελάχιστη απόσταση τροποποίησης (edit distance) από την ετικέτα του dp, ορίζοντας με αυτόν τον τρόπο μια *αντιστοιχία αναγνωριστικού* (p, dp). Το σύνολο όλων των αντιστοιχιών αναγνωριστικού αναφέρεται ως $Identifier_{mp}$.

4.5.3 Αντιστοιχίες μηνύματος

Τόσο οι αντιστοιχίες πρόθεσης όσο και οι αντιστοιχίες τελικού σημείου υπολογίζονται χρησιμοποιώντας και συνδυάζοντας πληροφορίες που εξήχθησαν νωρίτερα (π.χ. κανονικοποίηση πρόθεσης, σχέσεις βασικής αντιστοιχίας) και ορίζουν τη βάση για την εξαγωγή αντιστοιχιών μηνύματος των οποίων ο ρόλος είναι η συσχέτιση των παραμέτρων εισόδου και εξόδου οι οποίες χρησιμοποιούνται σε κλήσεις λειτουργιών ΠΣΔ διεπαφών σε στοιχεία αναπαράστασης ή κεφαλίδες μηνυμάτων που χρησιμοποιούνται σε ΠΣΠ αλληλεπιδράσεις. Για να εξασφαλιστεί η κάλυψη σεναρίων MITM στα οποία οι αναπαραστάσεις ενδεχομένως να παρέχονται αντί να εξορύσσονται, ορίζουμε μια τεχνική εξόρυξης η οποία είναι ανεξάρτητη από τις τεχνικές εξόρυξης αναπαραστάσεων που παρουσιάστηκαν στην Ενότητα 4.4. Σε κάθε περίπτωση, καθώς η αναγνώριση αντιστοιχιών τελικού σημείου μπορεί να εφαρμοστεί σε σενάρια MITM με τον τρόπο που συζητήθηκε παραπάνω, τα σύνολα αντιστοιχιών $Endpoint_{mp}$ ανδ $Identifier_{mp}$ θεωρούνται διαθέσιμα.

Η είσοδος ή η έξοδος μιας λειτουργίας, όπως και η αναπαράσταση ενός πόρου περιορισμένη από την κατεύθυνση του μηνύματος και την πράξη χειρισμού που χρησιμοποιείται, ορίζουν ουσιαστικά σχήματα μηνυμάτων. Στη λογική αυτή, υπό την προϋπόθεση ότι το πρόβλημα τοποθετείται σε κατάλληλο επίπεδο αφαίρεσης, η εξαγωγή αντιστοιχιών μηνύματος μπορεί να θεωρηθεί ως ένα πρόβλημα ταιριάσματος σχημάτων ή οντολογιών (schema or ontology matching) [124], [48]. Παρ' όλα αυτά, καθώς το ταίριασμα σχημάτων είναι γενικά ένα δύσκολο πρόβλημα ως προς την αυτοματοποιημένη επίλυσή του, πριν ενσωματωθούν τέτοιες τεχνικές στη διαδικασία προσαρμογής, αναγνωρίζονται και εντοπίζονται τα τμήματα των σχημάτων που πρέπει να ταιριάξουν έτσι ώστε να διαιρεθεί το πρόβλημα σε μικρότερα και να επιτευχθούν αποτελέσματα εξαγωγής υψηλότερης ακρίβειας.

Όπως συζητήθηκε παραπάνω, η εξαγωγή αντιστοιχιών μηνύματος εφαρμόζεται θεωρώντας διαθέσιμο ένα σύνολο αντιστοιχιών τελικού σημείου $Endpoint_{mp}$. Το γεγονός αυτό επιτρέπει τον περιορισμό του εύρους της ανάλυσης που απαιτείται για το schema matching πρόβλημα, περιορίζοντας το σε συσχετισμένα σημεία αλληλεπίδρασης. Με άλλα λόγια, με βάση το γεγονός ότι μια αντιστοίχιση τελικού σημείου συσχετίζει μια λειτουργία $o \in O$ του S_P με έναν τύπο πόρου $r \in R$ και μια πράξη $a \in A$ του S_R, μπορούν να οριστούν δύο προβλήματα schema matching:

- το ταίριασμα παραμέτρων εισόδου του o με στοιχεία των αναπαραστάσεων

που έχουν προσδιοριστεί για το r με κατεύθυνση `IN` και πράξη a, εξαιρώντας ήδη αντιστοιχισμένες παραμέτρους που υπάρχουν στο σύνολο αντιστοιχιών $Identifier_{mp}$.

- το ταίριασμα παραμέτρων εξόδου του o σε στοιχεία των αναπαραστάσεων που έχουν προσδιοριστεί για το r με κατεύθυνση `OUT` και πράξη a.

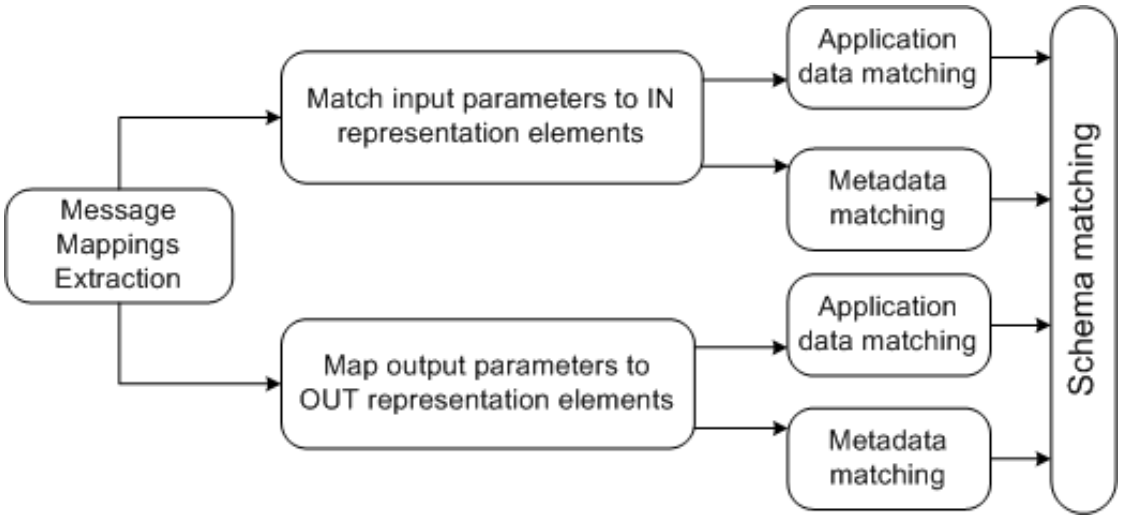
Επιπρόσθετα, καθώς τόσο οι παράμετροι όσο και τα στοιχεία αναπαραστάσεων χαρακτηρίζονται ως προς το αν φέρουν δεδομένα εφαρμογής ή μεταδεδομένα, κάθε schema matching πρόβλημα που ορίζεται παραπάνω μπορεί να διαιρεθεί περαιτέρω. Συγκεκριμένα, κάθε schema matching πρόβλημα που ορίζεται παραπάνω διαιρείται σε:

- ένα πρόβλημα για το ταίριασμα παραμέτρων των οποίων η κλάση της παραμέτρου είναι `metadata` σε στοιχεία αναπαράστασης των οποίων το χαρακτηριστικό *isMetadata* έχει τιμή `true` και
- ένα πρόβλημα για το ταίριασμα παραμέτρων των οποίων η κλάση της παραμέτρου είναι `application data` σε στοιχεία αναπαράστασης των οποίων το χαρακτηριστικό *isMetadata* έχει τιμή `false`

Τα παραπάνω βήματα ανάλυσης οδηγούν σε ζεύγη συνόλων παραμέτρων και συνόλων στοιχείων αναπαράστασης τα οποία πρέπει να ταιριάξουν ορίζοντας με αυτό τον τρόπο μια προσέγγιση ταιριάσματα *βασισμένη σε διχοτόμηση* (*partition-based matching*) [123]. Στη συνέχεια, για κάθε schema matching πρόβλημα, μπορεί να χρησιμοποιηθεί μια ποικιλία προσεγγίσεων [138], [139] έτσι ώστε να αναγνωριστούν οι αντιστοιχίες και δεδομένου του περιορισμένου εύρους κάθε προβλήματος, μπορούν να χρησιμοποιηθούν και περισσότερο ακριβές αλλά ταυτόχρονα και πιο ακριβείς τεχνικές. Εναλλακτικά, μπορούν να χρησιμοποιηθούν έτοιμα εργαλεία ταιριάσματος σχημάτων όπως τα COMA [93] και HARMONY [134] καθώς όχι μόνο υλοποιούν ή συνδυάζουν διαφορετικές μεθόδους ταιριάσματος αλλά διευκολύνουν την επιθεώρηση και διόρθωση των αντιστοιχιών που έχουν αναγνωριστεί μέσω ειδικών γραφικών διεπαφών αλληλεπίδρασης με τον χρήστη (GUI). Στο πρωτότυπο που έχει υλοποιηθεί, υπάρχει η υλοποίηση μιας απλής μεθόδου ταιριάσματος που βασίζεται στην απόσταση τροποποίησης (edit distance) των ονομάτων των παραμέτρων και των ετικετών των στοιχείων αναπαράστασης, ομοίως με την τεχνική που ορίστηκε για τον προσδιορισμό του συνόλου αντιστοιχιών $Identifier_{mp}$. Παρ' όλα

αυτά, το πρωτότυπο μπορεί να υποστηρίξει τη χρησιμοποίηση εξωτερικών εργαλείων ταιριάσματος κατά το τελικό βήμα εξαγωγής αντιστοιχιών μηνύματος, των οποίων τα αποτελέσματα μπορούν να ενσωματωθούν χρησιμοποιώντας το μορφότυπο EDOAL[1].

Το Σχήμα 4.11 παρουσιάζει μια ανάλυση ή *διχοτόμηση* της διαδικασία εξαγωγής αντιστοιχιών μηνύματος σε μικρότερα schema matching προβλήματα. Το σύνολο των αντιστοιχιών μεταξύ των στοιχείων μηνύματος αναφέρεται ως $Message_{mp}$.



Σχήμα 4.11: Ανάλυση της διαδικασία εξαγωγής αντιστοιχιών μηνύματος σε υποπροβλήματα

4.6 Προσαρμογή κατά τον χρόνο εκτέλεσης

Όπως απεικονίζεται στο Σχήμα 4.1 μετά τη φάση χρόνου σχεδίασης (design-time phase) λαμβάνει χώρα η φάση χρόνου εκτέλεσης (run-time phase). Η φάση χρόνου εκτέλεσης υλοποιείται από τον *μηχανισμό προσαρμογής χρόνου εκτέλεσης* (*run-time adaptation engine*) η οποία χρησιμοποιεί το αποτέλεσμα της φάσης του χρόνου σχεδίασης, δηλαδή το αρχείο προδιαγραφής της προσαρμογής έτσι ώστε να εφαρμόσει την P2R λογική προσαρμογής. Συνοπτικά, το αρχείο προδιαγραφής της προσαρμογής προσδιορίζει:

[1]EDOAL, `http://alignapi.gforge.inria.fr/edoal.html`

- τους τύπους πόρων και τις HTTP μεθόδους που πρέπει να εκτεθούν (RTM και συσχετισμένες πράξεις),
- το πως αυτά τα ΠΣΠ σημεία αλληλεπίδρασης σχετίζονται με ΠΣΔ κλήσεις λειτουργιών ($Endpoint_{mp}$),
- το πως τα URIs, HTTP κεφαλίδες και περιεχόμενα πρέπει να μετασχηματιστούν σε στοιχεία των υπογραφών των λειτουργιών ($Identifier_{mp}$, $Message_{mp}$) και
- πως το αποτέλεσμα των κλήσεων των λειτουργιών πρέπει να αντιστοιχιστεί σε HTTP μηνύματα αποκρίσεων ($Message_{mp}$).

Το δομοστοιχείο της P2R προσαρμογής κατά τον χρόνο εκτέλεσης παραλαμβάνει RESTful HTTP αιτήματα, αναγνωρίζει τη λειτουργία η οποία πρέπει να κληθεί με βάση τις αντιστοιχίες τελικού σημείου και συνεχίζει εφαρμόζοντας τη λογική μετασχηματισμού που προβλέπεται από τις αντιστοιχίες μηνυμάτων. Έπειτα, ο προσαρμογέας καλεί την ΠΣΔ υπηρεσία και αφού αυτή απαντήσει, μετασχηματίζει το αποτέλεσμα σε μια κατάλληλη HTTP απόκριση με βάση τις αντιστοιχίες μηνύματος που έχουν αναγνωριστεί.

Η προτεινόμενη διαδικασία προσαρμογής κατανέμει το μεγαλύτερο υπολογιστικό και επεξεργαστικό φορτίο στη φάση σχεδίασης έτσι ώστε η προσαρμογή κατά τον χρόνο εκτέλεσης να έχει χαμηλές απαιτήσεις και σημαντικές δυνατότητες κλιμάκωσης. Με βάση αυτή τη λογική, ο μηχανισμός προσαρμογής χρόνου εκτέλεσης δεν εισάγει ουσιαστική επιβάρυνση στις αλληλεπιδράσεις και η επίπτωση που έχει σχετικά με την επίδοση και τις δυνατότητες κλιμάκωσης του συστήματος είναι περιορισμένες. Επίσης, υλοποιώντας τον προσαρμογέας του χρόνου εκτέλεσης σε ένα περιβάλλον στο οποίο τα μέσα διεπαφής των υπηρεσιών είναι αποσυζευγμένα από τις υλοποιήσεις (π.χ. SCA) μπορεί να επιτρέψει την υποστήριξη πολλαπλών άλλων μέσων διεπαφών ΠΣΔ εκτός του SOAP, καθώς το αρχείο προδιαγραφής της προσαρμογής δεν είναι ειδικό ως προς συγκεκριμένες τεχνολογίες. Τέλος, ένα χαρακτηριστικό του προσαρμογέα χρόνου εκτέλεσης είναι ότι θα πρέπει να είναι σε θέση να παρέχει μια προδιαγραφή του REST API η οποία να εξυπηρετεί τη μηχανιστική της επεξεργασία (machine readable), καθώς κάτι τέτοιο θεωρείται συχνά χρήσιμο τόσο για τους καταναλωτές RESTful υπηρεσιών, όπως και για σκοπούς διενέργειας δοκιμών και ελέγχων.

4.7 Υλοποίηση πρωτοτύπου

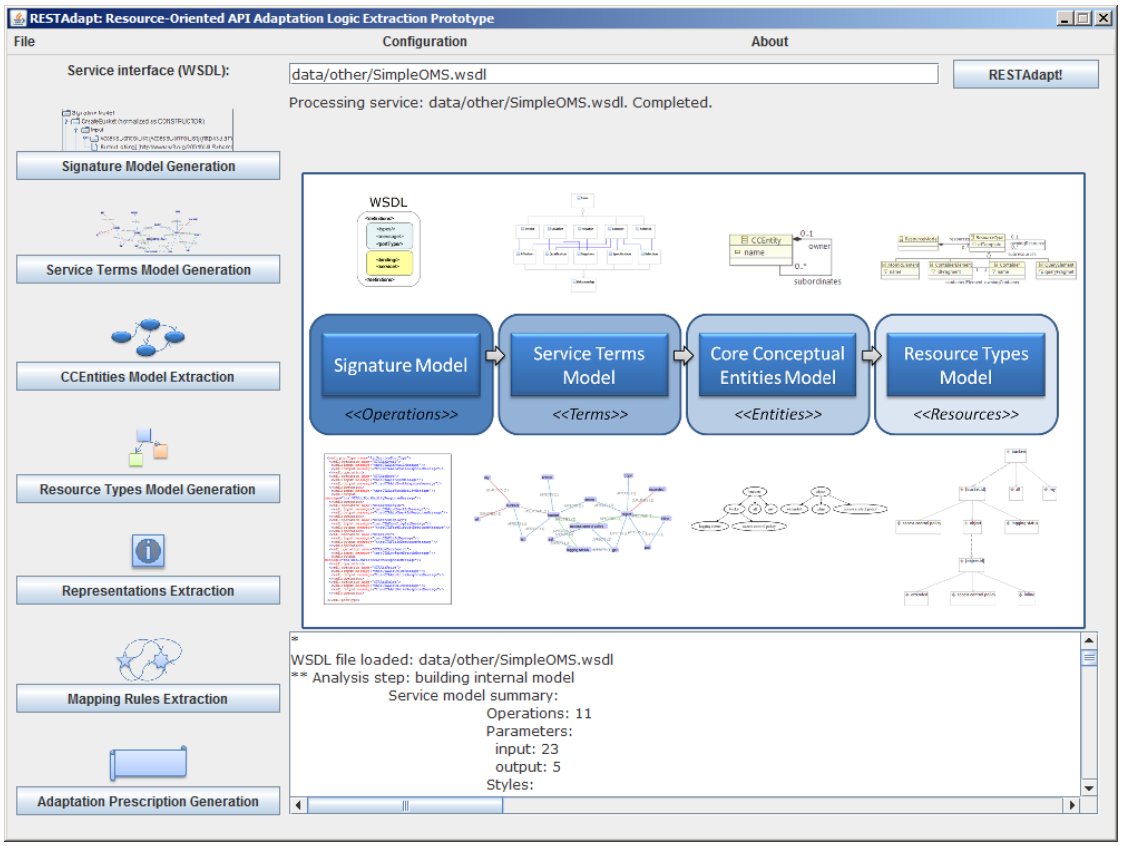
Η προσέγγιση P2R προσαρμογής που προτάθηκε έχει υλοποιηθεί στο πρωτότυπο *RESTAdapt* το οποίο αναλύεται σε δομοστοιχεία με τον τρόπο που απεικονίζεται στο Σχήμα 4.1.

Το δομοστοιχείο RESTAdapt Resource Model Extractor (RME) και το δομοστοιχείο RESTAdapt Interface Mappings Extractor (IME) έχουν υλοποιηθεί ως εφαρμογές Java. Το πρώτο δομοστοιχείο υλοποιεί τη λογική εξαγωγής του μοντέλου πόρων που παρουσιάστηκε στις Ενότητες 4.2 και 4.4 και το δεύτερο υλοποιεί τη λογική εξαγωγής αντιστοιχιών που παρουσιάστηκε στην Ενότητα 4.5. Οι τεχνικές για την παραγωγή μοντέλου υπογραφών, την κανονικοποίηση πρόθεσης λειτουργιών, την κατασκευή των OTM και του προσδιορισμού σχέσεων βασικής αντιστοιχίας έχουν υλοποιηθεί σε βιβλιοθήκες με δυνατότητες επαναχρησιμοποίησης. Συνεπώς, το RESTAdapt IME θα μπορούσε να χρησιμοποιηθεί σε MITM σενάρια, εφόσον ένα RTM μπορεί να παρασχεθεί για την ΠΣΠ υπηρεσία. Βέβαια, στο πλαίσιο μιας συνεδρίας OW P2R προσαρμογής, τα αποτελέσματα των παραπάνω τεχνικών επαναχρησιμοποιούνται αντί να εκτελούνται πολλαπλές φορές οι τεχνικές, έτσι ώστε να αποφεύγονται επιβαρύνσεις ως προς την επίδοση. Επίσης, υλοποιήθηκε και γραφική διεπαφή για τη διευκόλυνση της επίδειξης της λειτουργικότητας του πρωτότυπου και της μεταχείρισης των μοντέλων. Στο Σχήμα 4.12 απεικονίζεται ένα screenshot της γραφική διεπαφής του RESTAdapt που υλοποιήθηκε, στο οποίο ο χρήστης προσδιορίζει ένα αρχείο WSDL (τοπικό ή απομακρυσμένο) και εφαρμόζει τη διαδικασία αναγνώρισης της λογικής προσαρμογής. Τα αποτελέσματα κάθε βήματος όπως και οι περιγραφές των τεχνικών που εφαρμόζονται παρέχονται κάνοντας κλικ στα αντίστοιχα κουμπιά.

Για παράδειγμα, στο Σχήμα 4.13 το πρωτότυπο RESTAdapt χρησιμοποιεί μια βιβλιοθήκη απεικόνισης (Prefuse[2]) για να παρουσιάσει και να επιτρέψει την αλληλεπίδραση με το STM που υπολογίστηκε για το SimpleOMS.

Επιπρόσθετα, χρησιμοποιήθηκε μια ευρεία συλλογή βιβλιοθηκών και περιβαλλόντων-πλαισίων τρίτων μερών για την υλοποίηση υπολογιστικών δραστηριοτήτων όπως διαχείριση μοντέλων XSD, XML, WSDL και XMI/EMF, επεξεργασία κειμένου φυσικής γλώσσας (natural language processing), επεξεργασίας γραφη-

[2]Prefuse visualization toolkit, `http://prefuse.org/`



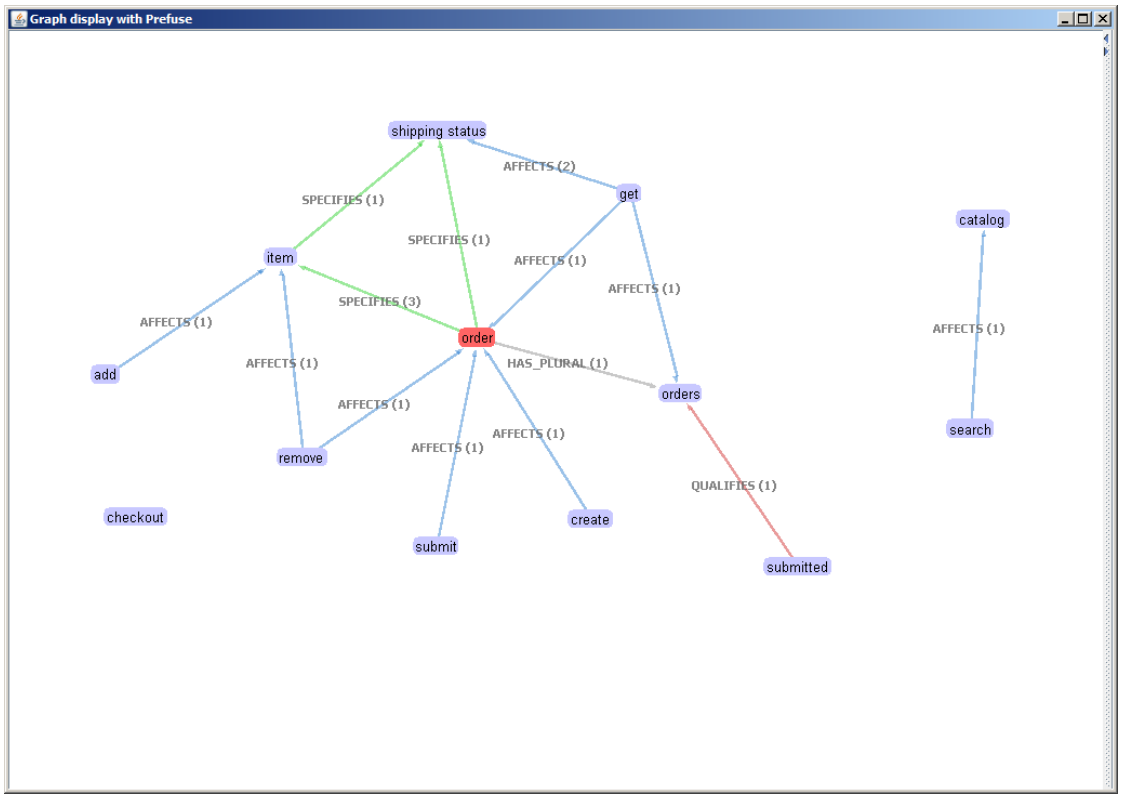
Σχήμα 4.12: Πρωτότυπο *RESTAdapt*

μάτων κλπ.

Οι δυνατότητες επισκόπησης και αναδιάρθρωσης από την πλευρά του χρήστη παρέχονται μέσω ενός γραφικού εργαλείου διαχείρισης το οποίο δημιουργήθηκε μέσω του GMF[3], στο οποίο ο χρήστης είναι σε θέση να αναθεωρήσει το RTM που έχει εξαχθεί όπως και ορισμένους τύπους αντιστοιχιών. Το Σχήμα 4.14 απεικονίζει ένα screenshot του εργαλείου με το RTM που εξήχθη για το SimpleOMS. Περαιτέρω παρεμβάσεις αναθεώρησης ή αναδιάρθρωσης μπορούν να υποστηριχθούν μέσω ενός αυτόματα παραχθέντος εργαλείου διαχείρισης, βασισμένο στο EMF, καθώς το μεταμοντέλο του αρχείου προδιαγραφής της προσαρμογής έχει οριστεί με το Ecore[4].

Μετά την οριστικοποίησή του, το αρχείο προδιαγραφής της προσαρμογής χρη-

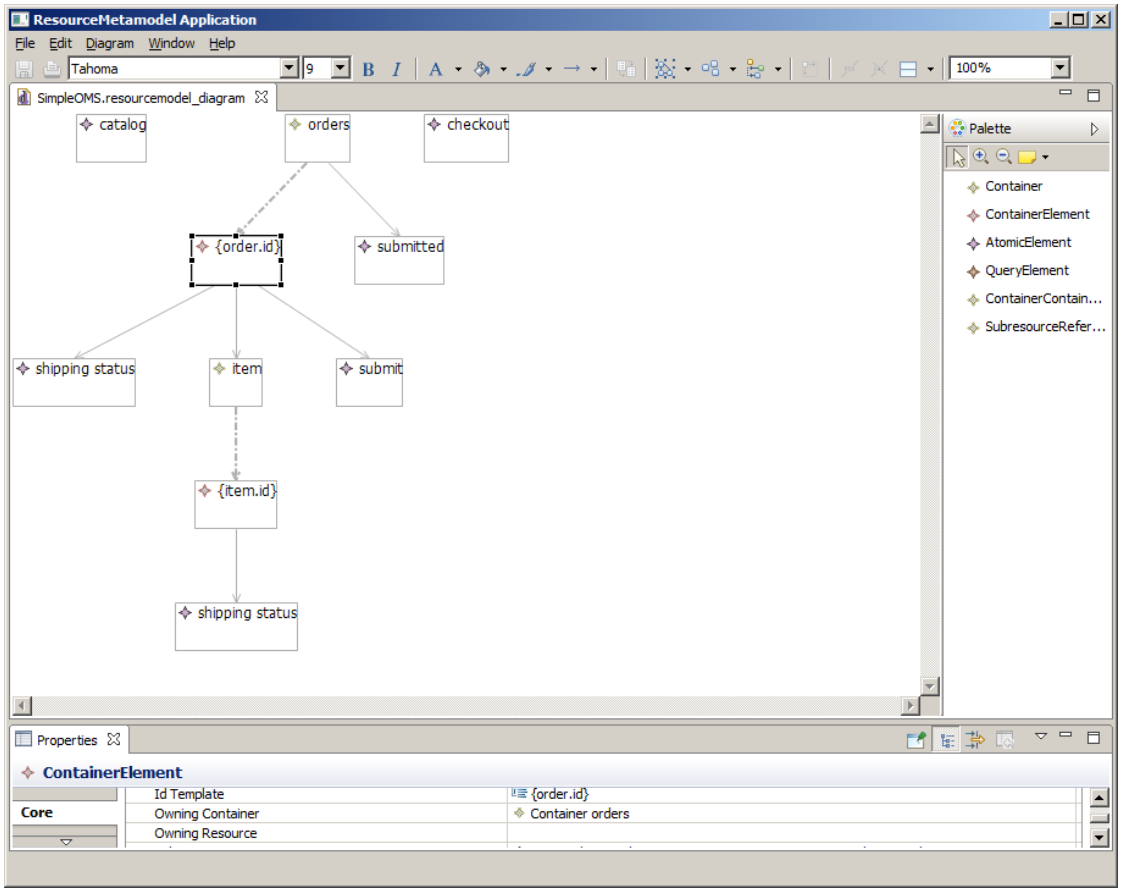
[3]GMP/GMF, `http://www.eclipse.org/modeling/gmp/`
[4]Ecore, `http://wiki.eclipse.org/Ecore`



Σχήμα 4.13: Απεικόνιση ενός μοντέλου STM στο RESTAdapt

σιμοποιείται από το δομοστοιχείο RESTAdapt Run-time Adaptation Engine (RAE) το οποίο υλοποιήθηκε ως ένα νέο binding στο Service Component Architecture (SCA). Συγκεκριμένα, υλοποιήθηκε το binding `restadapt` το οποίο μπορεί να χρησιμοποιηθεί στο πλαίσιο του SCA με στόχο να παρέχει ένα REST API για μια υφιστάμενη ΠΣΔ SOAP υπηρεσία, εφόσον έχει προσδιοριστεί ένα αρχείο προδιαγραφής προσαρμογής στην αντίστοιχη SCDL[5] περιγραφή της υπηρεσίας (Σχήμα 4.15).

[5]Σ"ΔΛ : Σεριςε ὄμπονεντ Δεφινιτιον Λανγυαγε



Σχήμα 4.14: Απεικόνιση του αποτελέσματος εξαγωγής πορών και του εργαλείου αναδιάρθρωσης στο *RESTAdapt*

```
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           xmlns:tuscany="http://tuscany.apache.org/xmlns/sca/1.1"
           targetNamespace="http://sample"
           name="simpleoms-contribution">
    <component name="SimpleOMSComponent">
        <implementation.java class="sample.SimpleOMSImpl"/>
        <service name="SimpleOMS">
            <binding.ws name="WS"/>
            <tuscany:binding.restadapt name="REST" prescription="SimpleOMSAdaptation.adp">
                <tuscany:operationSelector.adapt/>
            </tuscany:binding.restadapt>
        </service>
    </component>
</composite>
```

Σχήμα 4.15: SCDL περιγραφή στην οποία παρέχεται ένα REST API για την υπηρεσία *SimpleOMS* μέσω P2R προσαρμογής, παράλληλα με το υφιστάμενο WS binding

ΚΕΦΑΛΑΙΟ 5

ΥΠΟΚΑΤΑΣΤΑΣΙΜΟΤΗΤΑ ΚΑΙ ΕΥΘΥΓΡΑΜΜΙΑ ΥΠΗΡΕΣΙΩΝ ΔΙΑΦΟΡΕΤΙΚΩΝ ΥΠΟΔΕΙΓΜΑΤΩΝ

Λόγω επιχειρηματικών απαιτήσεων, τεχνολογικών κατευθύνσεων και στρατηγικών επιλογών που κάνουν οι οργανισμοί, οι πάροχοι υπηρεσιών συχνά απαιτείται να εξετάσουν και να σχεδιάσουν την παράλληλη έκθεση της λειτουργικότητας που παρέχουν τόσο μέσω ΠΣΔ διεπαφών υπηρεσιών όσο και ΠΣΠ διεπαφών υπηρεσιών. Επίσης, οι καταναλωτές υπηρεσιών συχνά χρειάζεται να αλλάξουν τις υπηρεσίες που καταναλώνουν και να χρησιμοποιήσουν εναλλακτικές, ακόμα και αν οι νέες αυτές υπηρεσίες ενδέχεται να ακολουθούν διαφορετικό υπόδειγμα διεπαφής. Συνεπώς, τα υπηρεσιοστρεφή περιβάλλοντα μετατρέπονται σε υβριδικά οικοσυστήματα ως προς τα υποδείγματα διεπαφών, στα οποία η ανάγκη για σύγκριση υπηρεσιών που ακολουθούν διαφορετικά υποδείγματα ως προς τη λειτουργική ευθυγραμμία τους και τις δυνατότητες υποκατάστασης εγείρεται ως ένα απαιτητικό ζήτημα. Ανάλογες εργασίας διερεύνησης της ευθυγραμμίας και της υποκαταστασιμότητας μεταξύ υπηρεσιών εφαρμόζονται προς το παρόν από αρχιτέκτονες λογισμικού και μηχανικούς μέσω άτυπων, εμπειρικών γνωματεύσεων, οι οποίες ποικίλουν ως προς το επίπεδο λεπτομέρειας, ακρίβειας και εύρους. Συνεπώς, η δυνατότητα να εξετάζεται και να αποδεικνύεται με τυπικό τρόπο το κατά πόσο μια ΠΣΠ υπηρεσία μπορεί να υποκαταστήσει μια ΠΣΔ υπηρεσία (και αντίστροφα), ως προς την παρεχόμενη λειτουργικότητα από τις υπηρεσίες αυτές όπως και ως προς τα συμπεριφορικά πρωτόκολλα που υποστηρίζονται από αυτές, κρίνεται σημαντική σε ένα πλήθος σεναρίων και περιπτώσεων. Πιο συγκεκριμένα, η δυνατότητα σύγκρισης των δυνατοτήτων που παρέχονται από υπηρεσίες που ακολουθούν διαφορετικά υποδείγματα επιτρέπει:

- στους παρόχους υπηρεσιών να διασφαλίζουν τη λειτουργική ισοδυναμία ανεξάρτητα ανεπτυγμένων υπηρεσιών οι οποίες ακολουθούν τα δύο βασικά υποδείγματα διεπαφών υπηρεσιών και παρέχονται ταυτόχρονα ως εναλλακτική επιλογή η μία της άλλης,
- στους καταναλωτές υπηρεσιών να εξετάζουν σχέδια μετάπτωσης και σχετικές επιλογές εστιάζοντας στη λειτουργική και συμπεριφορική συμβατότητα αντί στα αρχιτεκτονικά υποδείγματα των διεπαφών,
- στους παρόχους και στους συνθέτες υπηρεσιών να εξετάζουν και να αποτιμούν κατά πόσο μια διεπαφή που παρέχεται από προσαρμογή υποδείγματος

διατηρεί την ισοδυναμία.

Στο πλαίσιο των προκλήσεων ΔΥΥ και ΔΥΕ που ορίστηκαν στο Κεφάλαιο 1 και περιγράφονται παραπάνω, αναγνωρίζουμε δύο κύρια ερευνητικά ερωτήματα. Το πρώτο ερώτημα είναι κατά πόσο υπάρχουν κοινές αφαιρέσεις μεταξύ των ΠΣΠ και των ΠΣΔ αρχιτεκτονικών οι οποίες μπορούν να χρησιμοποιηθούν για την αναγνώριση εννοιολογικών αντιστοιχιών μεταξύ των δύο αρχιτεκτονικών στυλ. Το δεύτερο ερώτημα είναι κατά πόσο και πως, στα πλαίσια ενός συνόλου εννοιολογικών συσχετίσεων μεταξύ στοιχείων των διεπαφών των δύο υποδειγμάτων, μπορεί να δειχθεί ότι ένα ΠΣΠ API είναι δομικά και συμπεριφορικά ικανό να αντικαταστήσει ένα ΠΣΔ API και αντίστροφα.

Στο παρόν κεφάλαιο, εξετάζουμε τα ΠΣΔ και ΠΣΠ αρχιτεκτονικά στυλ στην υπολογιστική υπηρεσιών και προτείνουμε αντίστοιχα εννοιολογικά μεταμοντέλα υπηρεσιών. Στη συνέχεια, εξετάζουμε και προτείνουμε εννοιολογικές αντιστοιχίες μεταξύ των δύο μεταμοντέλων στην προσπάθεια αναγνώρισης κοινών αφαιρέσεων που θα μπορούσαν να ευθυγραμμίσουν ΠΣΠ και ΠΣΔ αρχιτεκτονικές υπηρεσιών. Δίνεται έμφαση στην εξέταση της ευθυγραμμίας μεταξύ WS-* υπηρεσιοστρεφών αρχιτεκτονικών και RESTful HTTP αρχιτεκτονικών, καθώς αυτές αποτελούν τις πιο ενδεικτικές και συγκεκριμένες υλοποιήσεις των ΠΣΔ και ΠΣΠ υποδειγμάτων, αντίστοιχα. Οι αντιστοιχίες μεταξύ των δύο αρχιτεκτονικών υποδηλώνονται στη μορφή των συσχετίσεων μεταξύ των αντίστοιχων μεταμοντέλων. Ορίζοντας αυτές τις συσχετίσεις μπορούν να προχωρήσουμε στον ορισμό ενός συνόλου μετασχηματισμών μοντέλων οι οποίοι επιτρέπουν την αναπαράσταση των ΠΣΔ και ΠΣΠ API ως open net τα οποία ένα μια εξειδίκευση των γενικών, χαμηλού αφαιρετικού επιπέδου δικτύων Petri. Στη συνέχεια, μπορεί να χρησιμοποιηθεί η θεωρία του *accordance* [148], [149] για να αποφασισθεί κατά πόσο τα δύο open net μπορούν να υποκαταστήσουν το ένα το άλλο, καθορίζοντας συνεπώς εάν τα δύο API βρίσκονται σε eujugramm'ia.

Το παρόν κεφάλαιο είναι οργανωμένο ως εξής: Η Ενότητα 5.1 αναλύει το εύρος και της συνεισφορές που παρουσιάζονται στο κεφάλαιο και περιγράφει τις αρχές και τη θεωρία εξέτασης της υποκαταστασιμότητας υπηρεσιών μέσω του accordance. Στην Ενότητα 5.2 εισάγουμε το περιβάλλον-πλαίσιο αποτίμηση υποκαταστασιμότητας υπηρεσιών διαφορετικών προτύπων. Αρχικά, παρέχουμε μια περιγραφή των μοντέλων και των δομοστοιχείων του. Έπειτα, εισάγονται δύο μεταμοντέλα υπηρε-

σιών τα οποία μοντελοποιούν εννοιολογικά στοιχεία των ΠΣΔ και ΠΣΠ υπηρεσιών. Έπειτα, προτείνεται ένα μεταμοντέλο συσχετίσεων το οποίο επιτρέπει τον ορισμό αντιστοιχιών υποδειγμάτων διεπαφών. Στη συνέχεια, παρουσιάζεται μια διαδικασία μετασχηματισμού οδηγούμενη από μοντέλα, μέσω της οποίας δημιουργούνται open net μοντέλα από μοντέλα υπηρεσιών και τέλος, συζητείται η διαδικασία εξέτασης δυνατότητας υποκατάστασης μεταξύ των υπηρεσιών. Στην Ενότητα 5.4 συζητούνται ορισμένα δυνατά και αδύναμα σημεία του περιβάλλοντος-πλαισίου που παρουσιάζεται στο κεφάλαιο.

Η εφαρμογή κάθε βήματος της διαδικασίας αποτίμησης υποκαταστασιμότητας παρουσιάζεται μέσω ενός συνοδευτικού παραδείγματος και στο Κεφάλαιο 6 συζητάμε μια πλήρη μελέτη περίπτωσης στην οποία το περιβάλλον-πλαίσιο και η μέθοδος που προτείνεται στο παρόν κεφάλαιο εφαρμόζεται σε μια υπηρεσία ηλεκτρονικού εμπορίου. Επίσης, κατευθύνσεις μελλοντικής έρευνας σχετικές με το περιβάλλον-πλαίσιο συζητούνται στο Κεφάλαιο 7.

5.1 Υπόβαθρο

5.1.1 Εύρος και συνεισφορές

Η ερευνητική εργασία που παρουσιάζεται στο παρόν κεφάλαιο στοχεύει να συνεισφέρει σε δύο περιοχές. Πρώτον, στοχεύει να ρίξει φως στη μακρά συζήτηση ως προς τη σχέση μεταξύ των ΠΣΠ και των ΠΣΔ αρχιτεκτονικών η οποία άπτεται πρακτικών ζητημάτων τα οποία φτάνουν πέρα από τις θεωρητικά ζητήματα της υπολογιστικής υπηρεσιών. Πιο συγκεκριμένα, καθώς τα μοντέρνα συστήματα παρέχουν δυικές διεπαφές, τόσο RESTful Web API όσο και παραδοσιακά Web Services, υφίσταται μια ανάγκη αναγνώρισης του αν υπάρχουν σημασιολογικές και συμπεριφορικές διαφορές μεταξύ των διεπαφών αυτών και κατά πόσο υπάρχουν ασυμφωνίες σχετικές με τη λειτουργικότητα που οι υπηρεσίες αυτές παρέχουν ανεξάρτητα στους καταναλωτές τους. Δεύτερον, υπάρχει η ανάγκη να αντιμετωπιστεί το ζήτημα σχεδίασης, συντήρησης και ταυτόχρονης εξέλιξης ΠΣΔ και ΠΣΠ διεπαφών υπηρεσιών. Πιο συγκεκριμένα, οι πάροχοι υπηρεσιών απαιτείται συχνά να παρέχουν η να συντηρούν την ίδια ή παρόμοια λειτουργικότητα μέσω τόσο μιας ΠΣΔ διεπαφής όσο και μιας ΠΣΠ διεπαφής. Η μετάπτωση από μια ΠΣΔ διεπαφή σε μια ΠΣΠ (και

αντίστροφα) όχι μόνο είναι μια σύνθετη εργασία αλλά επίσης απαιτεί την επιβεβαίωση ότι η νέα διεπαφή συμμορφώνεται με την αρχική. Επιπλέον, οι καταναλωτές υπηρεσιών επιθυμούν να επιβεβαιώνουν ότι API υπηρεσιών που παρέχονται από διαφορετικούς παρόχους βρίσκονται σε ευθυγραμμία (δηλαδή παρέχουν την ίδια λειτουργικότητα) ανεξάρτητα από το υπόδειγμα διεπαφής που ακολουθείται, έτσι ώστε οι εφαρμογές-πελάτες που έχουν θα είναι σε θέση να αλληλεπιδρούν με συνέπεια με τις διεπαφές αυτές, διευκολύνοντας με τον τρόπο αυτό τη μετάπτωση σε νέους παρόχους. Στο πλαίσιο αυτό, το παρόν κεφάλαιο στοχεύει αφενός να παρέχει εννοιολογικές αντιστοιχίες μεταξύ ΠΣΔ και ΠΣΠ αρχιτεκτονικών και αφετέρου να συνεισφέρει στο σημαντικό ζήτημα της συντήρησης και παράλληλης εξέλιξης δυικών διεπαφών υπηρεσιών.

5.1.2 Υποκαταστασιμότητα υπηρεσιών μέσω accordance

Όπως συζητήθηκε στο Κεφάλαιο 2, Ενότητα 2.5, η υποκαταστασιμότητα υπηρεσιών έχει εξεταστεί κυρίως ως προς τη σύγκριση ΠΣΔ υπηρεσιών. Στο πλαίσιο αυτό, έχει προταθεί η αποτίμηση του accordance [148], [149] ως μια μεθοδολογία για να αποφασίζεται η δυνατότητα υποκατάστασης μεταξύ ΠΣΔ υπηρεσιών που χρησιμοποιούν το μοντέλο επικοινωνίας ασύγχρονης μεταφορά μηνυμάτων. Συνοπτικά, η έννοια του accordance αναφέρεται στη δυνατότητα μιας υπηρεσίας N να εξυπηρετήσει τουλάχιστον όλους τους δυνατούς πελάτες μιας άλλης υπηρεσίας M. Στην περίπτωση αυτή λέμε ότι *η υπηρεσία N βρίσκεται σε accordance με την υπηρεσία M* [141]. Η έννοια του accordance έλαβε σημαντική προσοχή από την ερευνητική κοινότητα και η θεωρία του accordance θεωρείται μια εδραιωμένη μεθοδολογία για την αποτίμηση της υποκαταστασιμότητας [150].

Για να αποτιμήσει κανείς αν δύο API υπηρεσιών βρίσκονται σε accordance πρέπει να κάνει τα παρακάτω βήματα:

(α΄) Να αναπαραστήσει τις υπηρεσίες ως open net, τα οποία είναι ένα συγκεκριμένο είδος δικτύων Petri,

(β΄) Να συνθέσει δύο μοντέλα τα οποία υποδηλώνουν καταναλωτές του κάθε open net. Τα μοντέλα αυτά καλούνται Operating Guidelines (OG [94], [95],

(γ΄) Να αποτιμήσει κατά πόσο τα open net που μοντέλου την ΠΣΔ και την ΠΣΠ υπηρεσία μπορεί να αποδειχθεί ότι βρίσκονται σε accordance, χρησιμοποιώντας

τα αντίστοιχα OG μοντέλα και τον αλγόριθμο αποτίμησης του accordance.

Σε ό,τι αφορά στη σύγκριση ΠΣΔ υπηρεσιών, τα μοντέλα OG και η θεωρία του accordance έχουν χρησιμοποιηθεί με επιτυχία και έχουν εφαρμοστεί σε πλήθος σεναρίων [94], [95], [81], [94], [150], [141]. Παρ' όλα αυτά, οι υφιστάμενες προσεγγίσεις δεν αντιμετωπίζουν αρχιτεκτονικές ασυμφωνίες στο επίπεδο του υποδείγματος σχεδίασης των διεπαφών, αφήνοντας έτσι ανοιχτό το πρόβλημα της δια-υποδειγματικής υποκαταστασιμότητας υπηρεσιών. Σε μεικτά ως προς το υπόδειγμα διεπαφών, υπηρεσιοστρεφή περιβάλλοντα, για να είναι δυνατή η χρησιμοποίηση των θεωρητικών και αλγοριθμικών στοιχείων που είναι διαθέσιμα για την αποτίμηση της υποκαταστασιμότητας μέσω της θεωρίας του accordance, απαιτείται ένα περιβάλλον-πλαίσιο προσαρμογής για να αντιμετωπίσει τις εννοιολογικές αναντιστοιχίες των υποδειγμάτων, να παρέχει μια αρχιτεκτονική ευθυγράμμιση και να παρέχει συγκεκριμένους μηχανισμούς οι οποίοι χρησιμοποιούν την ευθυγράμμιση αυτή για να αποδώσουν συγκρίσιμα open net τα οποία αναπαριστούν τις υπηρεσίες που εξετάζονται. Αυτό ακριβώς αποτελεί την πρόκληση την οποία επιχειρεί να αντιμετωπίσει η προσέγγιση που προτείνεται στο παρόν κεφάλαιο και σύμφωνα με τη γνώση του πεδίου, αποτελεί την πρώτη του είδους της. Καθώς η θεωρία του accordance χρησιμοποιείται και στο πλαίσιο του προτεινόμενου περιβάλλοντος-πλαισίου, στις επόμενες παραγράφους συζητάμε τη διαδικασία αποτίμησης του αρρορδανσε και τα εμπλεκόμενα μοντέλα.

5.1.3 Μοντέλα και έννοιες για την αποτίμηση ύπαρξης accordance

Ένας ευνόητος και συνοπτικός, μηχανισμός τυπικής μοντελοποίησης υπηρεσιών και αλληλεπιδράσεων με υπηρεσίες με πέρασμα μηνυμάτων είναι τα δίκτυο Petri [103] και πιο συγκεκριμένα, μια εξειδίκευση των δικτύων αυτών, τα open net. Στο πλαίσιο αποτίμησης της υποκαταστασιμότητας, τα open net αναλύονται ώστε να δημιουργηθεί ένα ειδικού τύπου αυτόματο υπηρεσιών επισημειωμένο με Boolean εκφράσεις το οποίο ονομάζεται *operating guideline*. Χρησιμοποιώντας τα OG η αποτίμηση του accordance μπορεί να εφαρμοστεί για να αποφασιστεί αν η υπηρεσία N μπορεί να υποκαταστήσει μια άλλη υπηρεσία M, δηλαδή αν η N είναι σε θέση να εξυπηρετήσει τουλάχιστον όλους τους δυνατούς πελάτες της M. Στην παρούσα

ενότητα παρουσιάζουμε σε μεγαλύτερη λεπτομέρεια τα open net ως εξειδίκευση των δικτύων Petri, συζητάμε πως αυτά χρησιμοποιούνται για να αντλήσουμε προδιαγραφές μοντέλων OG και τελικά, δείχνουμε πως μπορεί να χρησιμοποιηθεί η μεθοδολογία αποτίμησης του accordance για να δειχθεί η υποκαταστασιμότητα μεταξύ υπηρεσιών.

Open Nets. Τα open net (ON) (*ανοιχτά δίκτυα*) ορίζονται ως μια ειδική κλάση χαμηλού επιπέδου δικτύων Petri. Ένα χαμηλού επιπέδου δίκτυο Petri είναι ένα σύστημα κατάστασης-μετάβασης (state-transition system) το οποίο αποτελείται από δύο τύπους κόμβων, *α*) *θέσεις* οι οποίες μπορούν να φέρουν ένα ή περισσότερα *token* και *β*) *μεταβάσεις*. Οι θέσεις χρησιμοποιούνται για να αποτυπώσουν την αιτιότητα (causality) μεταξύ των μεταβάσεων και για να μοντελοποιήσουν στατικές διαστάσεις κάποιου περιβάλλοντος, όπως για παράδειγμα προϋποθέσεις (preconditions) ή μετασυνθήκες (postconditions), δεδομένα εισόδου/εξόδου, απαιτούμενες δεσμεύσεις ή απελευθερώσεις υπολογιστικών πόρων κ.ά. Οι μεταβάσεις υποδηλώνουν δυναμικές διαστάσεις ενός περιβάλλοντος όπως γεγονότα, υπολογιστικά βήματα ή εργασίες. Οι θέσεις και οι μεταβάσεις συνδέονται μέσω ενός συνόλου τόξων. Η κατανομή των token στις θέσεις ενός δικτύου Petri, σε κάποιο χρονικό σημείο, αποτελεί το *μαρκάρισμα* (*marking*) του δικτύου. Για ένα marking m, το πλήθος των token το οποίο φέρει μια θέση p είναι ίσο με $m(p)$. Τα marking αποτελούν τις *καταστάσεις* των δικτύων Petri και κάθε δίκτυο Petri έχει ένα *αρχικό marking* (*initial marking*) m_0 το οποίο αποτελεί την αρχική κατανομή των token στις θέσεις. Τα marking μπορούν να μεταβάλλονται σύμφωνα με τον *κανόνα πυροδότησης* (*firing rule*) ο οποίος συζητείται παρακάτω. Σε απεικονίσεις των δικτύων Petri χρησιμοποιούνται συνήθως τετράγωνα ή παραλληλόγραμμα για την αναπαράσταση των μεταβάσεων, κύκλοι για την αναπαράσταση των θέσεων, κατευθυνόμενα τόξα για τις σχέσεις ροής μεταξύ των κόμβων και τελείες ή αριθμοί μέσα στις θέσεις για την αναπαράσταση των token.

Ορισμός 4 (Δίκτυο Petri)**.** *Ένα δίκτυο Petri N είναι μια πεντάδα* $[P, T, F, W, m_0]$ *όπου:*

- P *είναι ένα πεπερασμένο σύνολο θέσεων,*
- T *είναι ένα πεπερασμένο σύνολο μεταβάσεων,*
- $F \subseteq (P \times T) \cup (T \times P)$ *ένα πεπερασμένο σύνολο τόξων (σχέσεις ροής),*
- $W : F \to \mathbb{N}$ *είναι μια συνάρτηση απόδοσης επισημείωσης βάρους στα τόξα,*

- m_0 *είναι το αρχικό marking, όπου είναι marking είναι μια αντιστοίχιση* $m : P \rightarrow \mathbb{N}$, *και*
- $P \cap T = \emptyset$ *και* $P \cup T \neq \emptyset$.

Η συνάρτηση επισημείωσης βάρους W του Ορισμού 4 χρησιμοποιείται για να παρέχει μια συμπαγή αναπαράσταση ισοδύναμη με την κατάσταση στην οποία πολλαπλά τόξα συνδέουν τους ίδιους κόμβους. Δηλαδή, η τιμή βάρους που ανατίθεται είναι τυπικά ίση με το πλήθος των τόξων που θα χρησιμοποιούνταν για να συνδέσουν τους κόμβους. Παρ' όλα αυτά, για τα open net που εξετάζονται στο περιβάλλον-πλαίσιο υποκαταστασιμότητας ισχύει $W(f) = 1, \forall f \in F$, συνεπώς για λόγους απλότητας ο ορισμός της W παραλείπεται σε επόμενους ορισμούς.

Το *προ-σύνολο* (*pre-set*) ενός κόμβου n ενός δικτύου Petri, $\bullet n$, αποτελεί το σύνολο των κόμβων που προηγούνται του n, δηλαδή, $\bullet n = \{m \mid (m, n) \in F\}$. Ομοίως, το *μετα-σύνολο* (*post-set*) ενός κόμβου n ενός δικτύου Petri, $n\bullet$, είναι το σύνολο των κόμβων που ακολουθούν τον κόμβο n, δηλαδή $n\bullet = \{m \mid (n, m) \in F\}$.

Οι μεταβάσεις κατάστασης στα δίκτυα Petri περιγράφονται από τον κανόνα πυροδότησης ο οποίος αναφέρεται επίσης και ως *κανόνας μετάβασης* (*transition rule*). Μια μετάβαση t είναι ενεργοποιημένη (enabled) όταν μια θέση p_i στο προ-σύνολο $\bullet t$ φέρει τουλάχιστον $W(\{p_i, t\})$ token. Μια ενεργοποιημένη μετάβαση t μπορεί να *πυροδοτήσει* (*fire*) και όταν αυτό συμβεί $W(\{p_i, t\})$ token αφαιρούνται από κάθε θέση $p_i \in \bullet t$ και $W(\{t, p_j\})$ προστίθενται σε κάθε θέση $p_j \in t\bullet$.

Η πυροδότηση μιας μετάβασης οδηγεί σε νέο marking (κατάσταση), το οποίο δηλώνεται ως $m \rightarrow m'$, όπου m είναι το marking του δικτύου πριν την πυροδότηση της μετάβασης και m' το marking μετά από αυτή. Αλλεπάλληλες πυροδοτήσεις μεταβάσεων ονομάζονται *διαδρομές* (*runs*) και μπορούν να είναι μη πεπερασμένα. Ένα marking m' είναι *προσβάσιμο* (*reachable*) από ένα άλλο marking m ανν υπάρχει μια πεπερασμένη διαδρομή από το m στο m'. Το σύνολο όλων των δυνατών marking τα οποία είναι προσβάσιμα από το αρχικό marking m_0 ενός δικτύου Petri N δηλώνεται ως $R_N(m_0)$. Με βάση το $R_N(m_0)$ ορίζεται ο γράφος προσβασιμότητας του δικτύου ως ένας κατευθυνόμενος γράφος του οποίου οι κόμβοι αναπαριστούν τα προσβάσιμα marking και οι κατευθυνόμενες ακμές αναπαριστούν τις μεταβάσεις μεταξύ αυτών. Μια *source* μετάβαση (*μετάβαση αφετηρίας*) είναι μια μετάβαση η οποία δεν έχει θέσεις εισόδου και μια *sink* μετάβαση (*μετάβαση προορισμού*) είναι

μια μετάβαση η οποία δεν έχει θέσεις εξόδου. Συνεπώς, οι source μεταβάσεις δεν καταναλώνουν token όταν πυροδοτούνται και είναι συνεχώς ενεργοποιημένες, ενώ οι sink μεταβάσεις δεν παράγουν token όταν πυροδοτούνται.

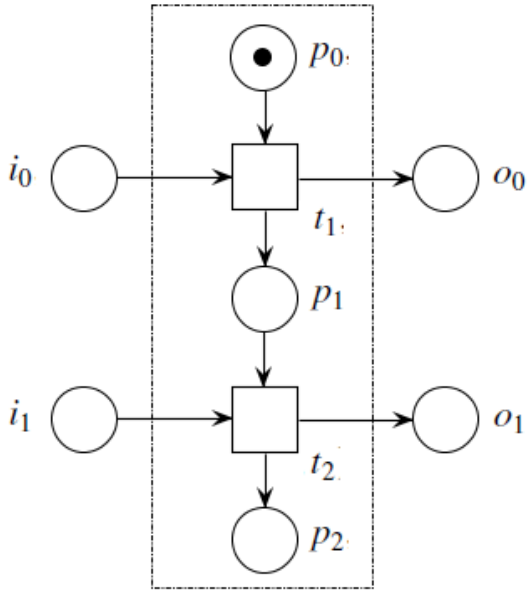
Τα Open Work Flow Net (OWFN) ή open net ορίζονται ως μια έκδοση των *workflow net* που προτάθηκαν στο [147] λιγότερων περιορισμών. Πιο συγκεκριμένα, τα workflow net αποτελούν εξειδίκευση δικτύων Petri τα οποία μοντελοποιούν τη συμπεριφορική δομή ελέγχου των υπηρεσιών. Τα open net επεκτείνουν τις δυνατότητες μοντελοποίησης που παρέχουν τα workflow net μέσω θέσεων επικοινωνίας (communication places) οι οποίες μοντελοποιούν κανάλια για την αποστολή και λήψη μηνυμάτων μεταξύ open net. Ένα open net μπορεί να προσδιορίσει το συμπεριφορικό πρωτόκολλο μιας υπηρεσίας και να ορίσει j'urec εισόδου και εξόδου οι οποίες αντιστοιχούν στα μηνύματα της διεπαφής της υπηρεσίας, η οποία επιτρέπει την επικοινωνία μεταξύ των παρόχων υπηρεσιών και των καταναλωτών υπηρεσιών. Τα open net σε σχέση με τα δίκτυα Petri εισάγουν το διαχωρισμό μεταξύ *θέσεων διεπαφής* (interface places) και *εσωτερικών θέσεων* (internal places). Επιπλέον, το σύνολο των θέσεων διεπαφής συντίθεται από δύο ξένα σύνολα θέσεων, ένα το οποίο περιέχει τις θύρες εισόδου και ένα το οποίο περιέχει τις θύρες εξόδου. Εξ' ορισμού, οι θέσεις εισόδου δεν έχει εισερχόμενα τόξα ($\bullet p_i = \emptyset$), ενώ οι θέσεις εξόδου έχουν μόνο εισερχόμενα τόξα συνδεδεμένα με αυτές ($p_o\bullet = \emptyset$). Ένα ακόμη χαρακτηριστικό που εισάγουν τα open net είναι ότι για να οριστεί ένα open net πρέπει να προσδιοριστεί ένα σύνολο *τελικών marking* (*final markings*) Ω, το οποίο υποδηλώνει τις συγκεκριμένες καταστάσεις στις οποίες μια υπηρεσία μπορεί να τερματίζει επιτυχώς.

Ορισμός 5 (Open Net)**.** *Ένα open net N είναι μια πεντάδα $[P, T, F, I, O, m_0, \Omega]$ η οποία αποτελείται από ένα δίκτυο Petri $[P, T, F, m_0]$ και τα εξής:*

- *$I \subseteq P$, ένα πεπερασμένο σύνολο θέσεων εισόδου τέτοιο ώστε $\bullet p = \emptyset, \forall p \in I$,*
- *$O \subseteq P$, ένα πεπερασμένο σύνολο θέσεων εξόδου τέτοιο ώστε $p\bullet = \emptyset, \forall p \in O$,*
- *Ω, ένα σύνολο συγκεκριμένων marking που ονομάζονται τελικά markings του N,*
- *το σύνολο $(I \cup O) \subseteq P$ ονομάζεται διεπαφή του N, και*
- *ισχύει ότι $I \cap O = \emptyset$, και $\forall m \in \Omega \cup M_0, m(p) = 0, \forall p \in I \cup O$.*

Εξ' ορισμού, το αρχικό marking m_0 όπως και σε όλα τα τελικά marking Ω, οι θέσεις διεπαφής δεν μπορούν να περιέχουν token. Γραφικά, ένα open net είναι παρόμοιο με ένα δίκτυο Petri. Παρ' όλα αυτά, οι θέσεις διεπαφής του συνήθως τοποθετούνται πάνω ή έξω από ένα παραλληλόγραμμο διακεκομμένης γραμμής για να παρέχει μια σαφή οπτική διαφοροποίηση μεταξύ των εσωτερικών θέσεων και των θέσεων διεπαφής. Στο Σχήμα 5.1 παρουσιάζεται ένα παράδειγμα ενός open net το οποίο ορίζεται ως:

- $P = \{p_0, p_1, p_2, i_0, i_1, o_0, o_1\}$,
- $T = \{t_1, t_2\}, F = \{\{p_0, t_1\}, \{t_1, p_1\}, \{p_1, t_2\}, \{t_2, p_2\}, \{i_0, t_1\}, \{t_1, o_0\}, \{i_1, t_2\}, \{t_2, o_1\}\}$,
- $I = \{i_0, i_1\}, O = \{o_0, o_1\}$,
- $m_0(p_0) = 1$ και $m_0(p_i) = 0$, $\forall p_i \in P$,
- $\Omega = \{[p_2]\}$



Σχήμα 5.1: Παράδειγμα απλού open net

Η έννοια του *open* (*ανοιχτό*) των open net σχετίζεται με την ύπαρξη θέσεων διεπαφής, δηλαδή $I \cup O \neq \emptyset$. Συνεπώς ένα open net με κενό σύνολο θέσεων διεπαφής καλείται *closed net* (*κλειστό δίκτυο*).

Σύνθεση open net. Μια υπηρεσία η οποία προσδιορίζεται από ένα open net N, μπορεί να συντεθεί μια ένα δομοστοιχείο καταναλωτή της υπηρεσίας το οποίο

μπορεί να περιγράφεται επίσης ως ένα open net M. Στο πλαίσιο του περιβάλλοντος-πλαισίου που περιγράφεται, τα N και M θεωρούνται ότι είναι ξένα, με μόνη εξαίρεση τις διεπαφές τους. Επίσης, γίνεται η υπόθεση ότι οι θέσεις εισόδου του ενός open net επικαλύπτει τις θέσεις εξόδου του άλλου και αντίστροφα. Η ιδιότητα αυτή μεταξύ των N και M καλείται *συμβατότητα διεπαφών* (*interface compatibility*). Συνεπώς, η σύνθεση δύο open net συμβατών διεπαφών ταυτίζεται με τη συγχώνευση θέσεων εισόδου και εξόδου των δύο δικτύων τα οποία έχουν ετικέτες που ταυτίζονται. Η σύνθεση των N και M, $N \oplus M$, αποτελεί επίσης open net και κατασκευάζεται με τις ενώσεις των συνιστωσών των N και M.

Ορισμός 6 (Σύνθεση open net)**.** *Μια σύνθεση $N \oplus M$ δύο open net συμβατών διεπαφών N, M, είναι ένα open net $[P_{N\oplus M}, T_{N\oplus M}, F_{N\oplus M}, I_{N\oplus M}, O_{N\oplus M}, M_{0,N\oplus M}, \Omega_{N\oplus M}]$ όπου:*

- $P_{N\oplus M} = P_N \cup P_M$,
- $T_{N\oplus M} = T_N \cup T_M$,
- $F_{N\oplus M} = F_N \cup F_M$,
- $I_{N\oplus M} = (I_N \cup I_M) \setminus (O_N \cup O_M)$,
- $O_{N\oplus M} = (O_N \cup O_M) \setminus (I_N \cup I_M)$,
- $m_{0,N\oplus M} = m_{0,N} \oplus m_{0,M}$, *και*
- $\Omega_{N\oplus M} = \{m_N \oplus m_M \mid m_N \in \Omega_N, m_M \in \Omega_M\}$.

Σχετικά με το $m_{0,N\oplus M}$ και το $\Omega_{N\oplus M}$, όπως συζητήθηκε παραπάνω, ούτε τα αρχικά marking $(m_{0,N}, m_{0,M})$, αλλά ούτε και το σύνολο των τελικών marking (Ω_N, Ω_M) μπορεί να περιλαμβάνει θέσεις διεπαφών με token. Συνεπώς, η σύνθεση $m = m_N \oplus m_M$ δύο τέτοιων marking ορίζεται ως $m(p) = m_N(p)$, αν $p \in P_N$, ή $m(p) = m_M(p)$, αν $p \in P_M$. Η σύνθεση open net είναι αντιμεταθετική και προσεταιριστική, συνεπώς $N \oplus M = M \oplus N$ ανδ $(N \oplus M) \oplus K = N \oplus (M \oplus K)$.

Υποθέτοντας ότι το M είναι συμβατό ως προς τις διεπαφές με το N και ότι $O_N = I_M$ και $I_M = O_N$, τότε το M καλείται partner του N και η σύνθεσή τους $N \oplus M$ είναι ένα closed net, όπου $I_{N\oplus M} = \emptyset$ και $O_{N\oplus M} = \emptyset$. Επίσης, λέμε ότι η σύνθεση $N \oplus M$ είναι *deadlock-free* (*ελεύθερη αδιεξόδων*) αν δεν υπάρχει marking m που να είναι προσβάσιμο από το αρχικό marking $m_{0,N\oplus M}$ έτσι ώστε καμιά μετάβαση $t \in T_{N\oplus M}$ να είναι ενεργοποιημένη και ταυτόχρονα να ισχύει $m \notin \Omega_{N\oplus M}$. Η απουσία αδιεξόδων αποτελεί σημαντική ιδιότητα για μια σύνθεση open net καθώς υποδηλώνει ότι οι

υπηρεσίες που συντίθενται μπορούν να αλληλεπιδρούν ορθώς. Ένα open net M του οποίου η σύνθεση με ένα δεύτερο open net N είναι deadlock-free καλείται *στρατηγική* (*strategy*) του N. Το σύνολο όλως των δυνατών στρατηγικών του N σημειώνεται ως *Strat(N)*.

Για μια υπηρεσία N, αν $Strat(N) \neq \emptyset$, τότε το N είναι *ελέγξιμο* (*controllable*). Η ιδιότητα αυτή είναι επίσης σημαντική καθώς εξασφαλίζει ότι υπάρχει τουλάχιστον μια partner υπηρεσία για τη N η οποία μπορεί να καταναλώσει τη λειτουργικότητα που παρέχεται χωρίς αδιέξοδα (deadlock-free).

Σχετικά με την πρόκληση διερεύνηση της υποκαταστασιμότητας, το να είναι κανείς σε θέσει να συγκρίνει σύνολα στρατηγικών δύο open net επιτρέπει να αποφασιστεί αν το ένα μπορεί να υποκαταστήσει το άλλο. Ως εκ τούτου, το ερώτημα της υποκαταστασιμότητας μεταξύ δύο open net N και N' ουσιαστικά αφορά στη σύγκριση του *Strat(N)* με το *Strat(N')*, τα οποία είναι δύο πιθανώς μη πεπερασμένα σύνολα. Για να είναι εφικτή μια τέτοια σύγκριση χρησιμοποιώντας μόνο τις προδιαγραφές των N και N' έχει προταθεί ο μηχανισμός των *operating guideline* [94], [82]. Τα μοντέλα αυτά είναι ειδικού τύπου αυτόματα τα οποία μοντελοποιούν τη συμπεριφορά όλων των πιθανών στρατηγικών ενός open net σε ένα, αθροιστικό και συμπαγές μοντέλο. Στην επόμενη ενότητα συζητάμε πως η συμπεριφορά ενός open net μπορεί να μοντελοποιηθεί μέσω ενός αυτόματου υπηρεσίας ή ενός επισημειωμένου συστήματος μεταβάσεων και πως το σύνολο των συμπεριφορών δικτύων μπορεί να αναπαρασταθεί με operating guideline μοντέλα.

Operating Guidelines. Όπως συζητήθηκε παραπάνω, το operating guideline ενός open net N, $OG(N)$, είναι ένα μοντέλο που προσδιορίζει τις συμπεριφορές των στρατηγικών του N με αθροιστικό και συμπαγή τρόπο. Οι *κανονικές* αλληλεπιδράσεις είναι αλληλεπιδράσεις οι οποίες δεν οδηγούν σε αδιέξοδα και την ίδια στιγμή σέβονται την ιδιότητα του limited communication [82].

Συνοπτικά, η ιδιότητα του limited communication απαιτεί ότι για δύο αλληλεπιδρώντα open net N και M: *α*) τα *εσωτερικά* υποδίκτυά τους $inner(N)$ και $inner(M)$ είναι *φραγμένα* (*bounded*) και *β*) ισχύει για κάποιο $k \in \mathbb{N}$, $m(p) \leq k$ για όλα τα marking m προσβάσιμα στο $N \oplus M$ και όλα τις θέσεις διεπαφής $p \in I_N \cup I_M$. Σχετικά με την πρώτη απαίτηση, το εσωτερικό υποδίκτυο ενός open net είναι ένα δίκτυο το οποίο αντλείται αφού αφαιρεθούν όλες οι θέσεις διεπαφής του αρχικού δικτύου και τα προσκείμενα τόξα. Επιπλέον, ένα open net είναι φραγμένο όταν

ο γράφος προσβασιμότητας του εσωτερικού υποδικτύου του είναι πεπερασμένος. Η δεύτερη απαίτηση ορίζει ότι οι αλληλεπιδρούσες υπηρεσίες δεν τοποθετούν περισσότερα από k token σε οποιαδήποτε θέση διεπαφής, σε οποιοδήποτε στάδιο εκτέλεσης, δηλαδή υπάρχουν το πολύ k μηνύματα σε εκκρεμότητα προς επεξεργασία. Η ιδιότητα του limited communication απαιτείται καθώς η ελεγξιμότητα δεν είναι αποφασισιμή για μη φραγμένα εσωτερικά υποδίκτυα των open net [96]. Παρ' όλα αυτά, η ιδιότητα αυτή δε θεωρείται περιοριστική για την προτεινόμενη προσέγγιση καθώς στην πράξη οι υπηρεσίες έχουν πεπερασμένες καταστάσεις και την ίδια στιγμή ενδέχεται να έχουν μη πεπερασμένο πλήθος στρατηγικών.

Η χρησιμοποίηση των open net για τη μοντελοποίηση υπηρεσιών παρέχει έναν καθαρό, εύκολα κατανοήσιμο και κομψό τρόπο μοντελοποίησης της επικοινωνίας με πέρασμα μηνυμάτων, κάτι που αποτελεί κατάλληλη υπόθεση για το περιβάλλον της υπολογιστικής υπηρεσιών. Συνεπώς, θεωρείται προτιμότερο κανείς να χρησιμοποιεί δίκτυα Petri στο επίπεδο σχεδίασης και προδιαγραφής. Παρ' όλα αυτά, για την ανάλυση των συμπεριφορών των δικτύων αυτών, έχουν χρησιμοποιηθεί μοντέλα που βασίζονται σε αυτόματα ως ένας περισσότερο βολικός και καλά κατανοητός μηχανισμός τυπικής μοντελοποίησης. Η συμπεριφορά ενός χαμηλού επιπέδου δικτύου Petri μπορεί να αποτυπωθεί στο γράφο προσβασιμότητάς του. Παρ' όλα αυτά, καθώς τα open net αλληλεπιδρούν με το περιβάλλον τους μέσω θέσεων διεπαφής, η συμπεριφορά ενός open net μοντελοποιείται με έναν ελαφρώς διαφορετικό τρόπο. Πιο συγκεκριμένα, η συμπεριφορά ενός open net N μπορεί να προσδιοριστεί από ένα επισημειωμένο σύστημα μεταβάσεων (labeled transition system (LTS)) βάσει του γράφου προσβασιμότητας του $inner(N)$. Για την απλοποίηση του ορισμό του LTS, θεωρούνται μόνο elementary communicating [82] δίκτυα, των οποίων οι μεταβάσεις είναι προσκείμενες μόνο το πολύ σε μια θέση διεπαφής (μέσω αντίστοιχου τόξου). Οποιοδήποτε open net μπορεί εύκολα να μετασχηματιστεί σε ένα ισοδύναμα elementary open net όπως συζητείται στο [82]. Επίσης, πριν την εισαγωγή των LTS για τον προσδιορισμό της συμπεριφοράς των open net, ορίζουμε μια συνάρτηση απόδοσης ετικέτας $l : T \to I \cup O \cup \{\tau\}$ έτσι ώστε το $l(t)$ να είναι ίσο είτε με την ετικέτα της μοναδικής προσκείμενης θέσης στη μετάβαση $t \in T$, εάν υπάρχει τέτοια θέση, είτε με τ σε διαφορετική περίπτωση. Επίσης, χρησιμοποιείται εξής σύμβαση: αν η προσκείμενη θέση διεπαφής είναι θέση εισόδου τότε προστίθεται ένα ερωτηματικό πριν την ετικέτα (π.χ. `?message`), ενώ αν είναι θέση εξόδου προστίθεται ένα θαυμαστικό (ε.γ. `!message`).

Για τον προσδιορισμό ενός LTS που μοντελοποιεί τη συμπεριφορά ενός open net, εξετάζεται το εσωτερικό δίκτυο αυτού. Συγκεκριμένα, για ένα open net N, ένα LTS $TS(N)$ μπορεί να οριστεί βάσει του γράφου προσβασιμότητας του $inner(N)$.

Ορισμός 7 (Labeled transition system (LTS)). *Ένα LTS $TS(N)$ ορίζεται για να μοντελοποιήσει τη συμπεριφορά ενός open net $N = [P, T, F, I, O, m_0, \Omega]$ ως μια πεντάδα $[Q, l, \delta, q_0, Q_F]$ όπου:*

- $Q = R_{inner(N)}(m_0)$, *το σύνολο των προσβάσιμων marking του* $inner(N)$,
- $l : T \to I \cup O \cup \tau$, *η συνάρτηση απόδοσης ετικέτας,*
- $\delta : Q \times I \cup O \cup \{\tau\} \times Q$, *μια σχέση μετάβασης όπου* $[m, l(t), m'] \in \delta$ *αυν* $m \xrightarrow{t} m'$, *για* $t \in T$, $m, m' \in Q$,
- $q_0 = m_0$, *η αρχική κατάσταση,*
- $Q_F = \Omega$, *το σύνολο των τελικών καταστάσεων.*

Μια χρήσιμο σχέση για τη συσχέτιση δύο παρόμοιων συστημάτων μεταβάσεων $TS(P)$ και $TS(R)$ είναι η sq'esh asjeno'uc prosomo'iwshc (*weak simulation relation*).

Ορισμός 8 (Σχέση ασθενούς προσομοίωσης). *Για δύο open net P και R, η ασθενής προσομοίωση του $TS(P)$ από το $TS(R)$ είναι μια σχέση $\rho_{P,R} \subseteq Q_P \times Q_R$ όπου $[q_P, q_R] \in \rho_{P,R}$, έτσι ώστε για κάθε μετάβαση $[q_P, a, q'_P] \in \delta_P$, υπάρχει μια μετάβαση $[q_R, a', q'_R] \in \delta_R$ και $[q'_P, q'_R] \in \rho_{P,R}$. Το open net R προσομοιώνει ασθενώς το P αυν μπορεί να οριστεί η σχέση $\rho_{P,R} \subseteq Q_P \times Q_R$ μεταξύ των δύο αντίστοιχων συστημάτων μεταβάσεων για τα οποία ισχύει $[q_{0,P}, q_{0,R}] \in \rho_{P,R}$.*

Όπως συζητήθηκε παραπάνω, το $OG(N)$ αποτελεί αθροιστική αναπαράσταση των συμπεριφορών όλων των open net που ανήκουν στο *Strat(N)*. Συνεπώς, το $OG(N)$ θα πρέπει να μπορεί να περιλαμβάνει την πληροφορία που περιέχεται σε οποιοδήποτε σύστημα μεταβάσεων $TS(M)$ το οποίο κατασκευάζεται χρησιμοποιώντας ένα open net $M \in$ *Strat(N)*. Επιπλέον, το $OG(N)$ θα πρέπει να μπορεί να περιλαμβάνει τις πληροφορίες που περιέχονται σε ένα σύστημα μετάβασης $TS(M^*)$ το οποίο προσομοιώνει ασθενώς κάθε $TS(M)$, όπου $M \in$ *Strat(N)*. Το M^* καλείται *most permissive partner* του N. Το $TS(M^*)$ παρέχει μια γενικευμένη αναπαράσταση συμπεριφορών open net τα οποία είναι στρατηγικές το N. Ωστόσο, το $TS(M^*)$

προσομοιάζει ασθενώς και συμπεριφορές open net που δεν αποτελούν στρατηγικές του N. Συνεπώς, υπάρχει μια ανάγκη προσδιορισμού του υποσυνόλου των ακμών του $TS(M^*)$ οι οποίες θα πρέπει να περιέχονται στην ασθενή προσομοίωση μεταξύ του $TS(M^*)$ και οποιουδήποτε $TS(M)$, για κάθε $M \in Strat(N)$. Για να επιτευχθεί αυτό, οι καταστάσεις του $TS(M^*)$ επισημειώνονται με μια Boolean έκφραση, οδηγώντας στον σχηματισμό ενός *λογικά επισημειωμένου αυτόματου υπηρεσιών* (*Boolean-annotated Service Automaton (BSA)*) όπως περιγράφεται στο [82].

Ορισμός 9 (Boolean-annotated service automaton (BSA))**.** *Ένα BSA B^{Φ} είναι μια πεντάδα $[Q, L, \delta, q_0, \Phi]$, όπου:*

- *Q, ένα μη κενό σύνολο καταστάσεων,*
- *L, ένα σύνολο ετικετών μεταβάσεων που αντιστοιχούν στις ετικέτες του M^*,*
- *$\delta \subseteq Q \times L \times Q$, μια ντετερμινιστική σχέση μετάβασης,*
- *q_0, μια αρχική κατάσταση, $q_0 \in Q$ και*
- *Φ, μια συνάρτηση Boolean επισημείωσης, η οποία αποδίδει μια λογική έκφραση η οποία συντίθεται από κυριολεκτήματα που ανήκουν στο σύνολο $L \cup \{final, \tau\}$, σε κάθε $q \in Q$ χρησιμοποιώντας μόνο $\wedge$ και $\vee$ Boolean τελεστές. Το $\ll \tau \gg$ αναπαριστά μια εσωτερική μετάβαση και το $\ll final \gg$ μια τελική κατάσταση.*

Όπως ορίστηκε παραπάνω, τα BSA αναπαριστούν αθροιστικές προδιαγραφές συμπεριφορών συνόλων open net. Ένα open net μπορεί αν εξεταστεί ως προς το αν ταιριάζει με ένα BSA B^{Φ}. Η εξέταση αυτή γίνεται χρησιμοποιώντας το $TS(M)$ σε συνδυασμό με μια αντιστοίχιση μεταξύ των κυριολεκτημάτων στο $L^+ = L \cup \{final, \tau\}$ και των Boolean τιμών ($true$, $false$). Συνεπώς, απαιτείται η εισαγωγή μιας συνάρτησης αντιστοίχισης $\beta : L^+ \rightarrow \{true, false\}$.

Ορισμός 10 (Assignment)**.** *Μια ανάθεση (assignment) $\beta : L^+ \rightarrow \{true, false\}$ είναι μια συνάρτηση αντιστοίχισης που ορίζεται ως:*

$$\beta_{TS(M)}(q)(x) = \begin{cases} true, & \text{ιφ } x \neq final \text{ αυδ } \exists q' \in Q : [q, x, q'] \in \delta, \\ true, & \text{ιφ } x = final \text{ αυδ } q \in Q_F, \\ false, & \text{οτιερωισε.} \end{cases}$$

Η βάση B ενός BSA B^{Φ} είναι ένα LTS το οποίο αντιστοιχεί στο BSA χωρίς τις Boolean ετικέτες. Έχοντας ορίσει την ανάθεση β, ένα LTS $TS(M)$ *ταιριάζει* σε ένα BSA B^{Φ} με το οποίο έχει τις ίδιες ετικέτες μεταβάσεων ανν το B προσομοιάζει ασθενώς το $TS(M)$ χρησιμοποιώντας μια σχέση $\rho \subseteq Q_{TS(M)} \times Q_B$ τέτοια ώστε $\forall [q_M, q_B] \in \rho : \beta_{TS(M)}(q_M) \models \Phi q_B$. Το σύνολο όλως των open net που ταιριάζουν στο B^{Φ} σημειώνεται ως $Match(B^{\Phi})$.

Χρησιμοποιώντας τις παραπάνω έννοιες, ο ορισμός του $OG(N)$ ενός open net N είναι ο ακόλουθος:

Ορισμός 11 (Operating guideline)**.** *Ένα BSA $OG(N) = B^{\Phi}$ είναι ένα operating guideline ενός open net N ανν $Match(OG(N)) = Strat(N)$.*

Αποτίμηση ύπαρξης accordance

Για να εξεταστεί αν μια υπηρεσία N μπορεί να υποκατασταθεί από μια υπηρεσία N' έχει εισαχθεί η έννοια του accordance. Μια υπηρεσία N' είναι σε θέση να υποκαταστήσει μια υπηρεσία N αν μπορεί να εξυπηρετήσει τουλάχιστον όλους τους δυνατούς καταναλωτές της N, δηλαδή αν *Strat(N)* $\subseteq$ *Strat(N')*. Συνεπώς, όταν μια υπηρεσία N' βρίσκεται σε accordance με μια υπηρεσία N, τότε κάθε συμβατός καταναλωτής M του N είναι επίσης συμβατός καταναλωτής της N'. Όντας μια σχέση preorder, η σχέση accordance είναι αυτοπαθής και μεταβατική.

Το accordance εξετάζεται σε όρους των αντίστοιχων open net ως ακολούθως. Αρχικά, εισάγουμε την έννοια της ισοδυναμίας διεπαφών μεταξύ open net μέσω του παρακάτω ορισμού.

Ορισμός 12 (Ισοδυναμία διεπαφών μεταξύ open net)**.** *Δύο open net N και N' τα οποία έχουν τις ίδιες θέσεις διεπαφής, $I_N \equiv I_{N'}$ και $O_N \equiv O_{N'}$, καλούνται ισοδύναμα ως προς τις διεπαφές (interface equivalent) open net.*

Μπορούμε να πούμε ότι *το N' μπορεί να υποκαταστήσει το N υπό accordance* ανν $Strat(N) \subseteq Strat(N')$. Παρ' όλα αυτά, η απευθείας σύγκριση μεταξύ *Strat(N)* και *Strat(N')* αποτελεί πρόκληση καθώς και τα δύο σύνολα στρατηγικών ενδέχεται να είναι μη πεπερασμένα.

Για τον έλεγχο το accordance έχουν προταθεί δύο προσεγγίσεις: *α*) *projection inheritance* [13] στη δομή των open net των υπηρεσιών και *β*) ανάλυση μέσω μοντέλων operating guideline που αντιστοιχούν στα open net των υπηρεσιών [141]. Η προσέγγιση μέσω *projection inheritance* παρέχει ένα ικανό κριτήριο για να αποφασιστεί η σχέση accordance, ενώ η εξέταση της σχέσης αυτής μέσω operating guideline μοντέλων παρέχει ένα ικανό και αναγκαίο κριτήριο. Στο περιβάλλον-πλαίσιο ΔΥΥ που προτείνεται στο κεφάλαιο αυτό, χρησιμοποιείται η δεύτερη προσέγγιση καθώς παρέχει πιο ισχυρά κριτήρια. Στη συνέχεια της ενότητας, συζητείται το πως μπορεί να δειχθεί η σχέση accordance χρησιμοποιώντας operating guideline μοντέλα.

Για να συγκρίνουμε τα operating guideline μοντέλα δύο ισοδύναμων ως προς τις διεπαφές open net N και N' ορίζεται η σχέση εξειδίκευσης $\sqsubseteq$ σύμφωνα με τον παρακάτω ορισμό:

Ορισμός 13 (Σχέση $\sqsubseteq$ μεταξύ operating guideline μοντέλων)**.** *Έστω N και N' δύο ισοδύναμα ως προς τις διεπαφές open net και έστω $OG(N) = [Q_N, L_N, \delta_N, q_{0,N}, \Phi_N]$ και $OG(N') = [Q_{N'}, L_{N'}, \delta_{N'}, q_{0,N'}, \Phi_{N'}]$ τα αντίστοιχα operating guideline μοντέλα. Τότε ισχύει $OG(N) \sqsubseteq OG(N')$ αν υπάρχει μια σχέση προσομοίωσης $\xi \subseteq Q_N \times Q_{N'}$ έτσι ώστε για κάθε $[q_N, q_{N'}] \in \xi$, η έκφραση $\Phi_N \Rightarrow \Phi_{N'}$ είναι ταυτολογία.*

Το $OG(N) \sqsubseteq OG(N')$ διαβάζεται ως το $OG(N')$ *εξειδικεύει* το $OG(N)$. Επιπλέον, όντας μια σχέση preorder, η σχέση εξειδίκευσης $\sqsubseteq$ είναι επίσης αυτοπαθής και μεταβατική.

Το Θεώρημα 1, το οποίο παρουσιάστηκε στο [141], δείχνει ότι η σχέση εξειδίκευσης $\sqsubseteq$ μεταξύ operating guideline μοντέλων μπορεί να χρησιμοποιηθεί για να αποφασιστεί η ύπαρξη accordance.

Θεώρημα 1 (Ἐlegqoc accordance [141])**.** *Έστω N και N' δύο open net και έστω $OG(N)$ και $OG(N')$ τα αντίστοιχα operating guideline μοντέλα. Τότε ισχύει $OG(N) \sqsubseteq OG(N')$ αν $Strat(N) \subseteq Strat(N')$.*

Το Θεώρημα 1 σημαίνει ότι η ύπαρξη accordance μπορεί να δειχθεί προσδιορίζοντας τη σχέση ξ. Καθώς τόσο το $OG(N)$ όσο και το $OG(N')$ είναι ντετερμινιστικά, η εύρεση της ξ μπορεί να πραγματοποιηθεί με μια κατά βάθος αναζήτηση στο $OG(N')$ η οποία προσομοιώνεται στο $OG(N)$. Ο αλγόριθμος για την εύρεση της

ξ κλιμακώνει γραμμικά ως προς το πλήθος των καταστάσεων και των ακμών του $OG(N')$, σχετικά με τις απαιτήσεις σε χρόνο και χώρο. Μια υλοποίηση της αποτίμησης accordance μεταξύ open net, η οποία χρησιμοποιεί αλγορίθμους τόσο για την παραγωγή OG όσο και για τον έλεγχο accordance παρέχεται από το εργαλείο Fiona [97]. Επίσης, ο συνδυασμός των εργαλείων Wendy και Cosme παρέχει την ίδια λειτουργικότητα μέσω πιο πρόσφατων και αποδοτικών υλοποιήσεων [83].

Χρησιμοποιώντας τη θεωρεία που παρουσιάστηκε παραπάνω, όπως αντίστοιχους αλγορίθμους για την κατασκευή operating guideline μοντέλων [82] και για τον προσδιορισμό της σχέσεις εξειδίκευσης $\sqsubseteq$ [98], [141], η ύπαρξη accordance και κατ' επέκταση η υποκαταστασιμότητα μεταξύ υπηρεσιών μπορεί να δειχθεί ανεξάρτητα από τα περιβάλλοντα στα οποία αυτές λειτουργικού (δηλαδή, τους δυνατούς καταναλωτές), χρησιμοποιώντας αποκλειστικά τις προδιαγραφές των open net των δομοστοιχείων που παρέχουν τις υπηρεσίες.

5.2 Αποτίμηση υποκαταστασιμότητας υπηρεσιών διαφορετικών υποδειγμάτων

Στο κεφάλαιο αυτό προτείνουμε ένα περιβάλλον-πλαίσιο αποτίμησης υποκαταστασιμότητα μεταξύ υπηρεσιών το οποίο στοχεύει στο να παρέχει τα απαραίτητα στοιχεία μοντελοποίησης και ανάλυσης ώστε για τον έλεγχο υποκαταστασιμότητας μεταξύ ΠΣΔ και ΠΣΠ υπηρεσιών.

Η παρουσίαση του περιβάλλοντος-πλαισίου γίνεται σε τέσσερα βήματα: Πρώτον, ορίζουμε μοντέλα πεδίου για τα API των υπηρεσιών, ένα για τα ΠΣΔ API και ένα για τα ΠΣΠ API. Δεύτερον, ορίζουμε ένα σύνολο κανόνων ευθυγραμμίας μεταξύ των δύο υποδειγμάτων διεπαφών υπηρεσιών. Τρίτον, ορίζουμε κανόνες μετασχηματισμού για την αντιστοίχιση των μοντέλων των υπηρεσιών σε αντίστοιχα open net. Τέλος, αφού έχουν παραχθεί συγκρίσιμα open net, μπορεί να χρησιμοποιηθεί η αποτίμηση της σχέσης accordance, όπως παρουσιάστηκε παραπάνω για να δειχθεί η υποκαταστασιμότητα. Μια σύνοψη του περιβάλλοντος που προτείνεται παρουσιάζεται στην επόμενη ενότητα.

5.2.1 Συνοπτική περιγραφή της ΔΥΥ

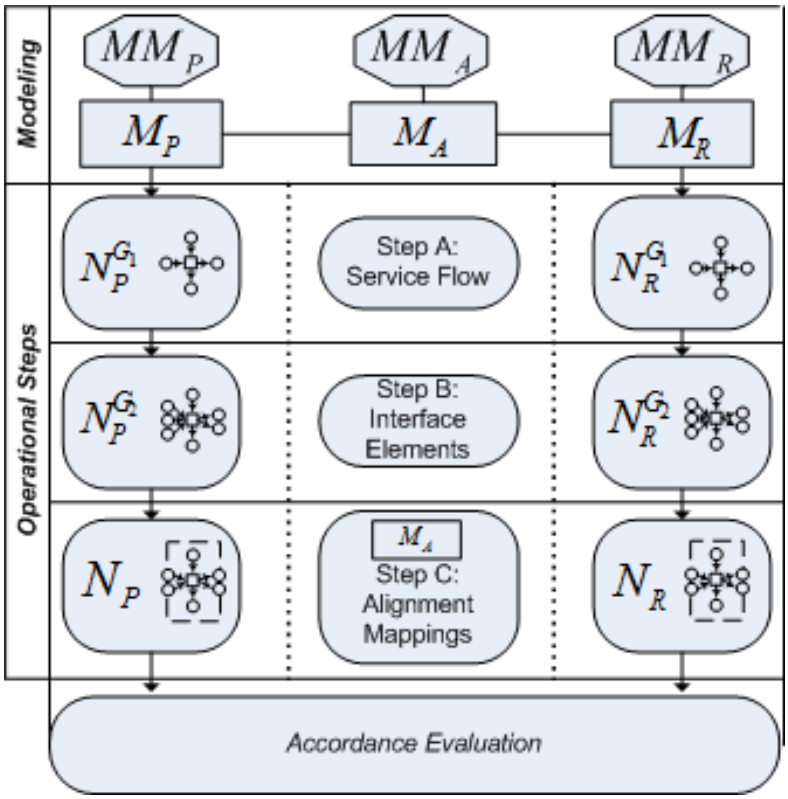
Στην παρούσα ενότητα εισάγουμε την αρχιτεκτονική του προτεινόμενου πλαισίου ελέγχου υποκαταστασιμότητα μεταξύ ΠΣΔ και ΠΣΠ υπηρεσιών, παρέχοντας μια περιγραφή των μοντέλων και των διαδικασιών που το συνθέτουν.

Ας υποθέσουμε ότι στόχος είναι να εξεταστεί το ερώτημα αν μια ΠΣΔ υπηρεσία S_P μπορεί να υποκατασταθεί από μια ΠΣΠ υπηρεσία S_R, στο πλαίσιο ενός συνόλου κανόνων ευθυγραμμίας (δηλαδή *αντιστοιχιών*), A. Όπως ορίστηκε σε προηγούμενες ενότητες, μια υπηρεσία S_R είναι σε θέση να υποκαταστήσει μια υπηρεσία S_P αν μπορεί να εξυπηρετεί τουλάχιστον όλους του πιθανούς καταναλωτές της S_P. Αναφερόμαστε στην πρόκληση αυτή ως *αποτίμηση υποκαταστασιμότητας υπηρεσιών διαφορετικών υποδειγμάτων* ή *αποτίμηση ΔΥΥ* για συντομία (*inter-paradigm service substitutability assessment (IPSS assessment)*). Συνεπώς, η *ευθυγραμμία υπηρεσιών διαφορετικών υποδειγμάτων* ή *δια-υποδειγματική ευθυγραμμία* (*inter-paradigm service alignment (IPSA)*) ορίζεται μεταξύ δύο υπηρεσιών S_R και S_P υπό την προϋπόθεση ότι μπορούν να υποκαταστήσουν η μια την άλλη, στο πλαίσιο των κανόνων A, δηλαδή υφίσταται ΔΥΥ δύο κατευθύνσεων μεταξύ των S_R και S_P.

Στο περιβάλλον-πλαίσιο που προτείνεται, οι υπηρεσίες και οι σχέσεις μεταξύ των υπηρεσιών διαφορετικών υποδειγμάτων μοντελοποιούνται χρησιμοποιώντας κατάλληλα μοντέλα. Τα μοντέλα αυτά μετασχηματίζονται στη συνέχεια σε open net ώστε να εξεταστούν ως προς τη σχέση accordance. Το Σχήμα 5.2 παρουσιάζει μια συνοπτική απεικόνιση του πλαισίου που προτείνεται. Το περιβάλλον-πλαίσιο επιτρέπει την εννοιολογική *μοντελοποίηση* των υπηρεσιών και των σχέσεων μεταξύ τους και υλοποιεί ένα σύνολο *διαδικαστικών βημάτων* τα οποία μετασχηματίζουν τα μοντέλα σε open net ώστε να συγκριθούν.

Μοντελοποίηση

Μεταμοντέλα Υπηρεσιών: Στον πυρήνα του περιβάλλοντος-πλαισίου ευθυγραμμίας βρίσκεται η αρχική μοντελοποίηση των API ΠΣΔ και ΠΣΠ υπηρεσιών. Για τον σκοπό αυτό ορίζουμε δύο μεταμοντέλα, MM_P και MM_R, τα οποία στοχεύουν να καταγράψουν στοιχεία διεπαφών και συμπεριφορικές ιδιότητες οι οποίες υποδηλώνονται από τα API των υπηρεσιών, για κάθε υπόδειγμα. Συνεπώς, η προδιαγραφή του API για μια συγκεκριμένη ΠΣΔ υπηρεσία S_P θα αναπαρίσταται από το μοντέλο



Σχήμα 5.2: Συνοπτική παρουσίαση του περιβάλλοντος-πλαισίου ΔΥΥ: πως χρησιμοποιούνται τα μοντέλα M_P, M_R και M_A για να παραχθούν συγκρίσιμα open net, N_P και N_R

M_P, το οποίο συμμορφώνεται στο αντίστοιχο μεταμοντέλο MM_P. Ομοίως, η προδιαγραφή του API για μια συγκεκριμένη ΠΣΠ υπηρεσία S_R θα αναπαρίσταται από το μοντέλο M_R, το οποίο συμμορφώνεται στο αντίστοιχο μεταμοντέλο MM_R. Τα μοντέλα M_P και M_R μπορούν να κατασκευαστούν με βάση τις διαθέσιμες προδιαγραφές και την τεκμηρίωση κάθε υπηρεσίας (π.χ. έγγραφα WSDL, WSCL για το S_P και WADL, τύποι μέσων ή σχέσεις υπερμέσων για το S_R).

Μεταμοντέλο Εννοιολογικής Ευθυγραμμίας: Το μεταμοντέλο εννοιολογικής ευθυγραμμίας, MM_A, συσχετίζει στοιχεία των μεταμοντέλων MM_P και MM_R. Το μεταμοντέλο MM_A, στοχεύει στον τυπικό ορισμό των συσχετίσεων μεταξύ δομικών και συμπεριφορικών στοιχείων των ΠΣΔ και των ΠΣΠ προδιαγραφών API. Συνεπώς, οι εφαρμόσιμες αντιστοιχίες μεταξύ ου μοντέλου υπηρεσίας M_P και του μοντέλου υπηρεσίας M_R θα προσδιορίζονται από το μοντέλο ευθυγραμμίας M_A, το οποίο συμμορφώνεται στο μεταμοντέλο MM_A.

Διαδικαστικά βήματα

Μοντελοποίηση ροής υπηρεσίας: Το πρώτο διαδικαστικό βήμα στο προτεινόμενο περιβάλλον-πλαίσιο είναι η παραγωγή δύο δικτύων Petri $N_P^{G_1}$ και $N_R^{G_1}$, από τα μοντέλα M_P και M_R αντίστοιχα. Τα δίκτυα $N_P^{G_1}$ και $N_R^{G_1}$ στοχεύουν στην αποτύπωση της ροής των αλληλεπιδράσεων που μπορούν να λάβουν χώρα μεταξύ των λειτουργιών που παρέχει ένα API υπηρεσίας. Για παράδειγμα, σε ένα ΠΣΔ API, όλες οι δυνατές ροές κλήσεων λειτουργιών μοντελοποιούνται από το $N_P^{G_1}$, ενώ οι μεταβάσεις που ορίζονται μέσω υπερμέσων μεταξύ πόρων σε μια RESTful υπηρεσία μοντελοποιούνται από το $N_R^{G_1}$.

Μοντελοποίηση δομής διεπαφών: Το δεύτερο διαδικαστικό βήμα στο προτεινόμενο περιβάλλον-πλαίσιο είναι η παραγωγή δύο δικτύων Petri $N_P^{G_2}$ και $N_R^{G_2}$, από το $N_P^{G_1}$ και το $N_R^{G_1}$, αντίστοιχα. Τα δίκτυα $N_P^{G_2}$ και $N_R^{G_2}$ στοχεύουν στη μοντελοποίηση των θεμελιωδών δομικών στοιχείων των αντίστοιχων διεπαφών των υπηρεσιών. Η διαδικασία παραγωγής βασίζεται στην αντικατάσταση και τον μετασχηματισμό των θέσεων και των μεταβάσεων του $N_P^{G_1}$ για τη δημιουργία του $N_P^{G_2}$, και του $N_R^{G_1}$ για τη δημιουργία του $N_R^{G_2}$.

Μοντελοποίηση open net: Το τρίτο διαδικαστικό βήμα είναι η παραγωγή δύο δικτύων Petri, N_P και N_R, τα οποία μοντελοποιούν τις απαραίτητες λειτουργικές και δομικές ιδιότητες των δύο API. Το βήμα μοντελοποίησης των open net εφαρμόζει περιορισμούς κατά τον μετασχηματισμό με στόχο τα δίκτυα N_P και N_R να είναι συγκρίσιμα με βάση το σύνολο κανόνων ευθυγραμμίας που περιλαμβάνεται στο M_A, να μπορούν να υπολογιστούν operating guideline μοντέλα για τα open net αυτά και να μπορεί να εξεταστεί η σχέση accordance.

Η αποτίμηση της σχέσης accordance βασίζεται σε ένα σύνολο εργαλείων ανοιχτού κώδικα και συγκεκριμένα, το εργαλείο Petri χρησιμοποιείται για να κανονικοποιήσει το N_P στο N_P^n και το N_R στο N_R^n, το εργαλείο Wendy για να υπολογίσει τα operating guideline μοντέλα για τα κανονικοποιημένα open net, $OG(N_P^n)$ και $OG(N_R^n)$, και τελικά το εργαλείο Cosme για να εξετάσει τη σχέση accordance μέσω της σχέσης εξειδίκευσης μεταξύ operating guideline μοντέλων (Ορισμός ρεφ-δεφ:ογΡεφινεμεντ, Θεώρημα 1).

Στις ενότητες που ακολουθούν εισάγουμε τις προδιαγραφές των μεταμοντέλων

υπηρεσιών MM_P και MM_R, όπως και ένα μοντέλο τύπων κανόνων ευθυγραμμίας MM_A, το οποίο συσχετίζει στοιχεία των δύο μεταμοντέλων υπηρεσιών βάσει σημασιολογικών σχέσεων και αρχιτεκτονικών αναλογιών. Έπειτα, παρουσιάζεται λεπτομερώς η διαδικασία μετασχηματισμού η οποία λαμβάνει ως είσοδο τα μοντέλα M_P, M_R και M_A και παράγει τα open net N_P και N_R. Τέλος, μέσω ενός παραδείγματος, παρουσιάζεται σε όλο το εύρος η διαδικασία ελέγχου ύπαρξης accordance μεταξύ των N_P και N_R.

5.2.2 Εννοιολογική μοντελοποίηση υπηρεσιών

Στην ενότητα αυτή, παρουσιάζουμε τα μεταμοντέλα MM_P και MM_R τα οποία περιλαμβάνουν εννοιολογικά στοιχεία των ΠΣΔ και ΠΣΠ API υπηρεσιών. Επιπλέον, παρουσιάζεται το μεταμοντέλο MM_A το οποίο περιλαμβάνει εννοιολογικούς κανόνες ευθυγραμμίας που συσχετίζουν στοιχεία των MM_P και MM_R. Το MM_A παίζει κυρίαρχο ρόλο στην αποτίμηση της υποκαταστασιμότητας καθώς παρέχει τις απαραίτητες πληροφορίες αντιστοιχίας ώστε να παραχθούν συγκρίσιμα open net που αντιστοιχούν στις υπηρεσίες S_P και S_R. Οι αντιστοιχίες αυτές μεταξύ των M_P και M_R ορίζονται σε ένα μοντέλο M_A που συμμορφώνεται στο MM_A.

Η αποτίμηση της ΔΥΥ βασίζεται σε πλήρη και σαφή μοντελοποίηση των υπό σύγκριση υπηρεσιών και των σχέσεών τους. Ως εκ τούτου τα MM_P, MM_R και MM_A λειτουργούν ως το πλαίσιο εννοιολογικής μοντελοποίησης της προτεινόμενης διαδικασίας αποτίμησης υποκαταστασιμότητας.

Μοντελοποίηση ΠΣΔ υπηρεσιών

Μια θεμελιώδης απαίτηση για ένα μεταμοντέλο ΠΣΔ υπηρεσιών είναι ότι θα πρέπει να είναι συμβατό με τις επικρατούσες αντιλήψεις γύρω από τις παραδοσιακές υπηρεσιοστρεφείς αρχιτεκτονικές. Πιο συγκεκριμένα, τα στοιχεία και οι σχέσεις που ορίζονται στο μεταμοντέλο πρέπει να είναι σύμφωνες με τις αρχές και τις έννοιες του αρχιτεκτονικού στυλ SOA όπως εκφράζονται σε σχετικά ευρέως αποδεκτά μοντέλα και πρότυπα. Επίσης, ένα τέτοιο μεταμοντέλο υπηρεσιών θα πρέπει να χαρακτηρίζεται από ένα κατάλληλο επίπεδο εκφραστικότητα και λεπτομέρειας έτσι ώστε να είναι δυνατό να προσδιοριστούν ΠΣΔ σημεία αλληλεπίδρασης με επαρκεί τρόπο

και ως διακριτά και αναγνωρίσιμα στοιχεία τα οποία εκθέτουν τις δυνατότητες της υπηρεσίας.

Λαμβάνοντας υπόψη τα παραπάνω ζητήματα, στις επόμενες παραγράφους γίνεται μια σύντομη συζήτηση των υφιστάμενων προτύπων και μοντέλων στην περιοχή της υπηρεσιοστρεφούς υπολογιστικής, εστιάζοντας στις λογικές αναπαραστάσεις που σχετίζονται με τη μοντελοποίηση υπηρεσιών, τη σχεδίασή τους και την προδιαγραφή τους. Έπειτα, εισάγεται και συζητείται λεπτομερώς το μεταμοντέλο MM_P.

Service-Oriented Architecture: αρχές και πρότυπα. Στην περιοχή της υπηρεσιοστρεφούς υπολογιστικής έχουν υπάρχει αρκετές προσπάθειες για τον προσδιορισμό προτυποποιημένων αρχιτεκτονικών περιγραφών και προδιαγραφών των συστημάτων που ακολουθούν τις αρχές του προσανατολισμού σε υπηρεσίες. Οι προσπάθειες αυτές αφορούν και αντιμετωπίζουν ένα σύνολο ζητημάτων, από οργανωτικές και επιχειρηματικές διαστάσεις για την πραγμάτωση και διαχείριση υπηρεσιοστρεφών αρχιτεκτονικών μέχρι θέματα σχεδίασης και υλοποίησης υπηρεσιοστρεφών δομοστοιχείων. Οι SOA προδιαγραφές σχετικά με ζητήματα σχεδίασης και υλοποίησης, όπως το OASIS Reference Model for SOA [84], το OASIS Reference Architecture for SOA Foundation [47], το The Open Group: SOA Reference Architecture Technical Standard [59], το The Open Group: SOA Ontology (version 2.0) [60] ή το OMG SoaML [58] επιχειρούν να δοκιμάσουν και να προτυποποιήσουν έννοιες και σχεδιαστικά υποδείγματα που σχετίζονται με τα υπηρεσιοστρεφή συστήματα. Σχετικά με τη σχεδίαση διεπαφών, τα περισσότερα πρότυπα, έμμεσα ή άμεσα, υποθέτουν την ύπαρξη λειτουργιών ως μέσω ανάλυσης της παρεχόμενης λειτουργικότητας σε αυτοτελής μονάδες. Ωστόσο, τα ζητήματα σχεδίασης διεπαφών τυπικά δεν καλύπτονται εκτενώς από υψηλού αφαιρετικού επιπέδου πρότυπα όπως τα παραπάνω καθώς θεωρούνται ζητήματα περισσότερο συγκεκριμένα και χαμηλότερου αφαιρετικού επιπέδου.

Τα σχεδιαστικά ζητήματα διεπαφών υπηρεσιών εξετάζονται σε προδιαγραφές που παρέχουν το αρχιτεκτονικό και τεχνολογικό πλαίσιο για την πραγμάτωση παραδοσιακών SOA συστημάτων. Στις προδιαγραφές αυτές περιλαμβάνονται το Web Service Architecture [156], το Web Service Description Language [44], [129], το Web Service Conversation Language [11] αλλά και έγγραφα όπως το Web Services Interoperability Basic Profile (WS-I BP) [160], [161] που προωθούν τη διαλειτουργικότητα και την προτυποποίηση.

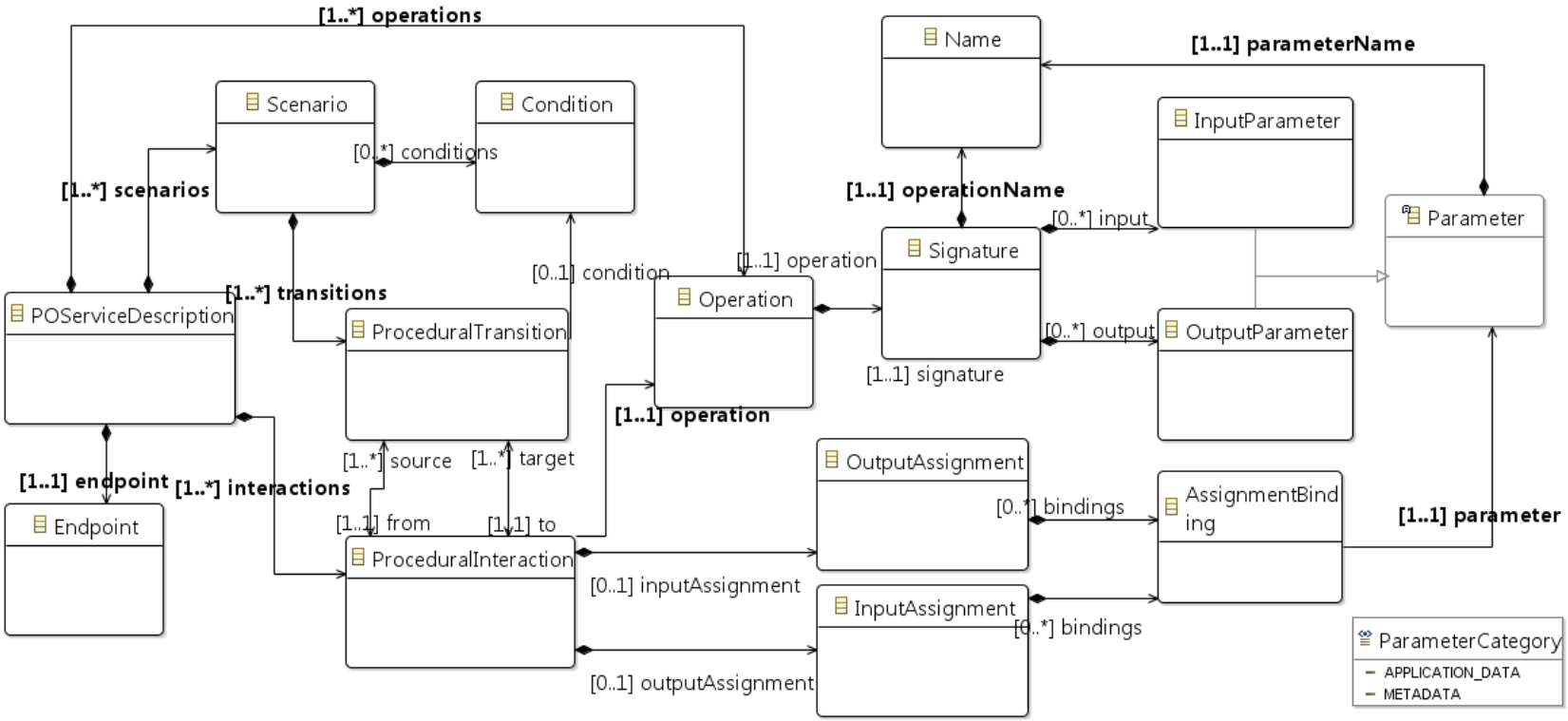
Με βάση την ανάλυση τόσο των υψηλής αφαίρεσης προτύπων όσο και τεχνολογικών προσεγγίσεων για την πραγμάτωση υπηρεσιοστρεφών συστημάτων, προτείνουμε το MM_P ως ένα γενικό, ουδέτερο ως προς τις υλοποιήσεις, εννοιολογικό μοντέλο για την αποτύπωση ΠΣΔ υπηρεσιών. Το MM_P βασίζεται στην αφαίρεση των κυρίαρχων τεχνολογιών υλοποίησης SOA συστημάτων όπως τα WSA Web services και είναι σχεδιασμένο να είναι συμβατό με τις αρχές του SOA. Επίσης, το MM_P παρέχει ένα κατάλληλο επίπεδο λεπτομέρειας ώστε να είναι σε θέση να επιτρέπει συσχετίσεις των στοιχείων των διεπαφών με στοιχεία του αντίστοιχου μεταμοντέλου ΠΣΠ υπηρεσιών.

Το μεταμοντέλο MM_P. Το μεταμοντέλο που προτείνεται στοχεύει στο να συμπεριλάβει πληροφορίες που αφορούν στη δομή των διεπαφών και στη ροή της υπηρεσίας. Για το λόγο αυτό μελετήθηκαν σχετικές προδιαγραφές και πρότυπα όπως τα WSDL [44], [129] και WSCL [11] ως πηγές πληροφορίας για τη σύνθεση του MM_P.

Το μεταμοντέλο MM_P απεικονίζεται στο Σχήμα 5.3, όπου τα χαρακτηριστικά των κλάσεων έχουν παραληφθεί για ευκρίνεια. Ωστόσο, το πλήρες μεταμοντέλο παρέχεται στο Παράρτημα Γ΄ ως ένα Ecore μοντέλο. Το στοιχείο που βρίσκεται στη ρίζα του MM_P είναι η κλάση `POServiceDescription` και αναπαριστά ένα αθροιστικό προσδιορισμό της ΠΣΔ υπηρεσίας S_P. Συγκεκριμένα, ένα `POServiceDescription` αποτελείται από ακριβώς ένα στοιχείο `Endpoint`, τουλάχιστον ένα στοιχείο `Scenario`, ένα σύνολο στοιχείων `Operation` και ένα σύνολο στοιχείων `ProceduralInteraction`.

Η κλάση `Endpoint` μοντελοποιεί ένα σημείο αναφοράς για την υπηρεσία. Ένα σημείο αναφοράς επιτρέπει στην υπηρεσία που περιγράφεται από το μοντέλο να είναι μοναδικά αναγνωρίσιμη και να μπορεί να είναι προσιτή σε καταναλωτές. Υπό αυτή τη λογική, περιλαμβάνει ένα χαρακτηριστικό `identifier` το οποίο αποτελεί ένα στοιχείο πληροφορίας αναγνώρισης (π.χ. ένα URL) το οποίο μπορεί να χρησιμοποιηθεί από ένα πελάτη για να αναγνωρίσει με μοναδικό τρόπο μια υπηρεσία, να την εντοπίσει και να αλληλεπιδράσει με αυτή.

Η κλάση `Operation` δηλώνει μια επαναλαμβανόμενη, αυτοτελή λειτουργία με συγκεκριμένη και διακριτή σημασιολογία. Στις ΠΣΔ SOA υπηρεσίες, η κλάση `Operation` αποτελεί μια βασική αφαίρεση για την ανάλυση της λειτουργικότητας. Δομικά, μια κλάση `Operation` περιέχει την κλάση `Signature`. Μια



Σχήμα 5.3: MM_P: ένα μεταμοντέλο για τον προσδιορισμό υπηρεσιών ΠΣΔ διεπαφών

υπογραφή (signature) χαρακτηρίζεται από ένα στοιχείο `Name` το οποίο συσχετίζεται με ένα στοιχείο `TermModel` για την ανάλυση και την περιγραφή των όρων που περιέχονται στο όνομα της λειτουργίας -η συγκεκριμένη αναφορά δεν απεικονίζεται στο Σχήμα 5.3, παρ' όλα αυτά αναφερόμαστε και επαναχρησιμοποιούμε το μεταμοντέλο όρων που ορίστηκε στο Κεφάλαιο 4 και απεικονίζεται στο Σχήμα 4.3. Επιπλέον, η κλάση `Signature` περιέχει τις κλάσεις `InputParameter` και `OutputParameter`, οι οποίες είναι υποκλάσεις της κλάσης `Parameter`. Ένα στοιχείο `Parameter` χρησιμοποιείται για να δηλώσει στοιχεία δεδομένων. Μπορεί να είναι προαιρετικό και μπορεί να επισημειωθεί ως είτε *application data* είτε ως *metadata*. Επίσης, το όνομά του μπορεί να περιγράφεται περαιτέρω μέσω μια κλάσης `Name`. Ένα στοιχείο `Parameter` μπορεί να συσχετιστεί με έναν τύπο δεδομένων που ορίζεται από την κλάση `DataType`. Ένα στοιχείο `DataType` μπορεί να είναι είτε `SimpleType` είτε `CompositeType`. Ένα στοιχείο `SimpleType` έχει μια άμεση αναφορά σε ένα βασικό τύπο, όπως integer, boolean, κλπ., ενώ η κλάση `CompositeType` δηλώνει συνθετικά στοιχεία `DataType` αποτελούμενα από `SimpleType` και `CompositeType` στοιχεία. Ο ορισμός της κλάσης `DataType` και των συνιστωσών τις δεν απεικονίζεται στο Σχήμα 5.3 για λόγους απλότητας. Παρ' όλα αυτά μπορούν να ανεβρεθούν στο Παράρτημα Γ΄ το οποίο περιλαμβάνει ένα γενικό μεταμοντέλο στοιχείων χαμηλού αφαιρετικού επιπέδου που χρησιμοποιούνται τόσο από το MM_P όσο και από το MM_R.

Ένα στοιχείο `ProceduralInteraction` συσχετίζεται με μια λειτουργία υπηρεσίας μέσω της σχέσης του της κλάσης του, με την κλάση `Operation`. Επίσης, περιέχει μια προδιαγραφή ανάθεσης στοιχείων εισόδου και εξόδου μέσω των κλάσεων `InputAssignment` και `OutputAssignment`, αντίστοιχα. Οι κλάσεις αυτές προδιαγράφουν μηνύματα που περιλαμβάνονται σε κλήσεις και απαντήσεις λειτουργιών. Μια ανάθεση εισόδου μπορεί να προδιαγράψει ως είσοδο ένα υποσύνολο των παραμέτρων εισόδου μιας λειτουργίας ως ουσιαστική είσοδο μιας αλληλεπίδρασης και μπορεί επίσης να αναθέσει συγκεκριμένες τιμές σε παραμέτρους μέσω του προσδιορισμού στοιχείων `AssignmentBinding`, έτσι ώστε να σχηματίσει συγκεκριμένες μορφές κλήσης της λειτουργίας. Ομοίως, μια ανάθεση εξόδου μπορεί να χρησιμοποιηθεί για να προσδιορίσει ένα υποσύνολο παραμέτρων εξόδου μιας λειτουργίας, το οποίο αποτελεί την ουσιαστική έξοδο μιας συγκεκριμένης αλληλεπίδρασης μεταξύ πελάτη και εξυπηρετητή. Οι αναθέσεις εισόδου και εξόδου περιλαμβάνονται στο MM_P ως μέσο αντιμετώπισης της συνήθους πρακτικής σημασιολογικής υπερφόρτωσης των λειτουργιών στις υπηρεσίες. Η πρακτική

υπερφόρτωσης της σημασιολογίας των λειτουργιών πραγματοποιείται μέσω συγκεκριμένων συμβάσεων στη χρήση συνδυασμών παραμέτρων ή τιμών παραμέτρων. Η πρακτική αυτή ορισμένες φορές φτάνει σε σημείο όπου η πρόθεση της αλληλεπίδρασης εξαρτάται από τον συνδυασμό των παραμέτρων ή προκαθορισμένες τιμές αυτών, ενώ η έξοδος της αλληλεπίδρασης αφορά συνήθως ένα υποσύνολο των παραμέτρων εξόδου. Χρησιμοποιώντας στοιχεία `ProceduralInteraction` τα οποία συσχετίζονται με αναθέσεις εισόδου και εξόδου, το MM_P μπορεί να αποτυπώσει διακριτές μονάδες λειτουργικότητας οι οποίες παρέχονται από την υπηρεσία ακόμα και στο πλαίσιο σημασιολογικά υπερφορτωμένων λειτουργιών των υπηρεσιών αυτών. Προφανώς, σε διεπαφές υπηρεσιών όπου η λειτουργικότητα αναλύεται μέσω των λειτουργιών της υπηρεσίας με σαφή τρόπο, τα στοιχεία ΠΣΔ αλληλεπιδράσεων (`ProceduralInteraction`) αντανακλούν τη σημασιολογία των συσχετισμένων λειτουργιών, απαιτώντας έτσι τετριμμένες ταυτολογικές αναθέσεις εισόδου και εξόδου.

Η κλάση `Scenario` υποδηλώνει τη χρησιμοποίηση σεναρίων χρήσης για την υπηρεσία που προσδιορίζεται από ένα σύνολο στοιχείων `Procedural Transition` και ένα σύνολο στοιχείων `Condition`. Μια ΠΣΔ μετάβαση (`ProceduralTransition`) ορίζει μια διμερή σχέση προτεραιότητας μεταξύ στοιχείων `ProceduralInteraction`, δηλώνοντας μια σημασιολογική εξάρτηση μεταξύ των εμπλεκόμενων μονάδων λειτουργικότητας. Οι ΠΣΔ αλληλεπιδράσεις μοντελοποιούν σημεία μιας διεπαφής υπηρεσίας τα οποία μπορούν να προσπελαστούν έτσι ώστε να κληθεί συγκεκριμένη λειτουργικότητα που παρέχεται από την υπηρεσία, επιτρέποντας έτσι την αλληλεπίδραση καταναλωτών της υπηρεσίας με τους παρόχους της. Διαφορετικές μονάδες λειτουργικότητας μιας υπηρεσίας μπορούν να συνδυαστούν για να καλύψουν πιο σύνθετες απαιτήσεις μέσω συντονισμού των ΠΣΔ αλληλεπιδράσεων. Ένας τέτοιος συντονισμός προσδιορίζεται από ένα σενάριο (`Scenario`) μέσω ΠΣΔ μεταβάσεων (`ProceduralTransition`). Κάθε μετάβαση προσδιορίζει μια αλληλεπίδραση `from` και οδηγεί σε μια αλληλεπίδραση `to`, ενώ μπορεί να ελέγχεται από μια συνθήκη (`Condition`) η οποία εξαρτάται από το πλαίσιο πραγμάτωσης της υπηρεσίας (π.χ. απαιτούνται επιπλέον βήματα σε μια διαδικασία πληρωμής παραγγελίας, ανάλογα με τοπικούς κανονιστικούς περιορισμούς). Ένας περιορισμός σχετικά με τα σενάρια είναι ότι δεν πρέπει να περιέχουν κύκλους, δηλαδή ο γράφος που συντίθεται από τις αλληλεπιδράσεις ως κόμβους και από τις μεταβάσεις ως κατευθυνόμενες ακμές θα πρέπει να είναι ακυκλικός (και κατευθυνόμενος). Ο προσδιορισμός σεναρίων μέσω ΠΣΔ αλληλεπιδράσεων

και μεταβάσεων για μια ΠΣΔ υπηρεσία S_P ακολουθεί παρόμοιους κανόνες με εκείνους που χρησιμοποιούνται στη γλώσσα WSCL [11] για τον ορισμό *conversation*. Με βάση τα παραπάνω, τα σενάρια μπορούν να προσδιορίσουν απλά εσωτερικά πρωτόκολλα για τις υπηρεσίες.

Μεταμοντέλο ΠΣΠ υπηρεσιών

Πηγές για το REST. Αντίθετα με την πλειάδα των προσπαθειών προτυποποίησης που έχουν αναπτυχθεί σχετικά με τις ΠΣΔ υπηρεσίες, για το REST και τις ΠΣΠ υπηρεσίες υπάρχει μόνο ένα μικρό σύνολο εγγράφων τα οποία θεωρούνται αξιόπιστες πηγές. Αυτό μπορεί να εξηγηθεί από το γεγονός ότι το REST περιγράφηκε λεπτομερώς στη διατριβή του Fielding [53] (και στη σχετική δημοσίευση [52]) και συνεπώς είναι διαθέσιμος ένας ορισμός από τον εμπνευστή του, αντίθετα με την ΠΣΔ υπηρεσιοστρεφή αρχιτεκτονική που αναπτύχθηκε ως αρχιτεκτονικό στυλ στη διάρκεια των χρόνων με τη συμμετοχή και τη δραστηριότητα μιας κοινότητας μηχανικών. Το [53] παρέχει έναν αναλυτικό ορισμό του REST ως αρχιτεκτονικό στυλ και αποτελεί άμεση και αξιόπιστη πηγή πληροφοριών για τον ορισμό αρχιτεκτονικών που συμμορφώνονται στο REST. Ωστόσο, υπάρχουν αρκετά αρχιτεκτονικά, σχεδιαστικά και τεχνολογικά ζητήματα για την κατασκευή RESTful υπηρεσιών τα οποία δεν εξετάζονται στο [53]. Για παράδειγμα, ο περιορισμός *Uniform Interface*, ο οποίος θεωρείται ως βασική έννοια στο REST παίζει ουσιαστικό ρόλο στον ορισμό ΠΣΠ διεπαφών υπηρεσιών. Παρ' όλα αυτά, η ανάλυσή του σε τέσσερις περαιτέρω περιορισμού παρουσιάζεται πολύ σύντομα και συζητείται έμμεσα μέσω της ενότητας *Data Elements* του Κεφαλαίου 5 στο [53]. Επιπρόσθετα, το REST προτάθηκε στο ευρύτερο πλαίσιο των δικτυακών εφαρμογών και κατά συνέπεια υπάρχουν ορισμένα ζητήματα τα οποία είναι ειδικά στην υπηρεσιοστρεφή υπολογιστική και τα οποία δεν συζητούνται απευθείας στον ορισμό του REST. Δεδομένων των παραπάνω, ορισμένα ζητήματα σχετικά με τη σχεδίαση RESTful υπηρεσιών συζητείται σε άρθρα, βιβλία και ερευνητικές συνεισφορές, εξετάζοντας πως οι περιορισμοί και οι αρχές του REST θα πρέπει να χρησιμοποιούνται στο πλαίσιο της ΠΣΠ υπηρεσιοστρεφούς υπολογιστικής. Μάλιστα, η σχεδίαση RESTful συστημάτων έχει αναγνωριστεί ως μια περιοχή έρευνας στην οποία έχει ως σήμερα γίνει μια σειρά συνεισφορών [116], [115], [3], [4], σχηματίζοντας κατά το πέρασμα των ετών μια αντίστοιχη κοινότητα ερευνητών. Επιπλέον, έχει δημοσιευτεί μια σειρά βιβλίων που εστιάζουν στη σχεδίαση και υλοποίηση RESTful υπηρεσιών, απευθυνόμενα τόσο στην ερευνητική

κοινότητα [159] όσο και στους μηχανικούς λογισμικού [128], [158], [46]. Τέλος, ορισμένα θέματα γύρο από το REST και τις διεπαφές RESTful υπηρεσιών συζητούνται από τον Fielding στο ιστολόγιό του (π.χ. [51]) όπως και σε διαδικτυακές ομάδες και λίστες ηλεκτρονικής αλληλογραφίας όπως η *rest-discuss*[1].

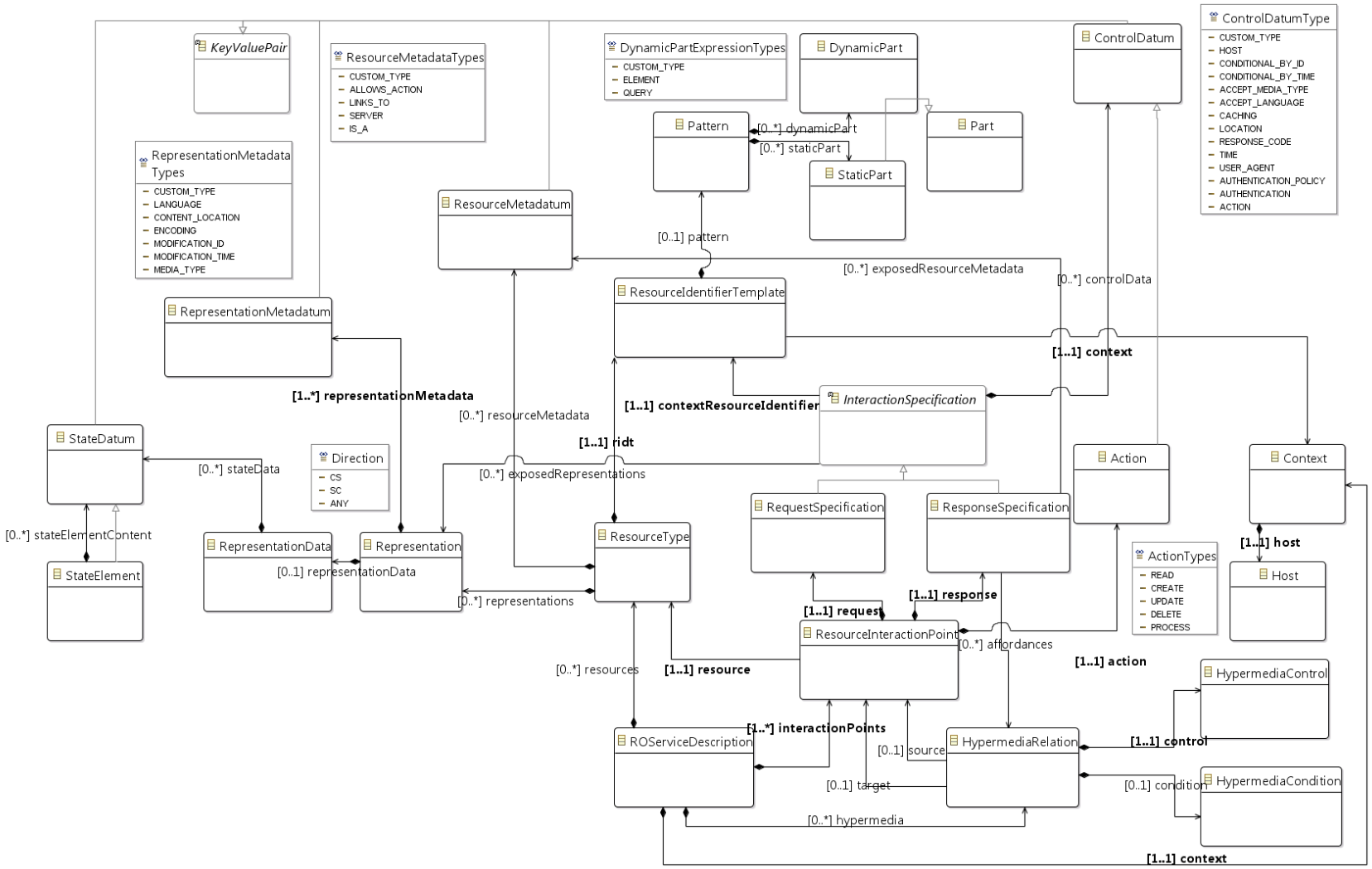
Σχετικά με τον ορισμό ενός κατάλληλου μεταμοντέλου για ΠΣΠ υπηρεσίες το οποίο θα χρησιμοποιηθεί στο πρόβλημα αποτίμησης υποκαταστασιμότητας, χρησιμοποιήσαμε τον ορισμό του REST ως βασική πηγή πληροφοριών. Επίσης, σε περιπτώσεις στις οποίες το [53] δεν παρείχε αρκετές πληροφορίες, εξετάσαμε πηγές όπως αυτές που συζητήθηκαν παραπάνω και την ανάλυσή μας για το UICF περιβάλλον-πλαίσιο που παρουσιάστηκε στο Κεφάλαιο 3, Ενότητα 3.3.2. Για παράδειγμα, τα *Data Element* του REST που ορίζονται στο [53] παρέχουν τη βάση για τη μοντελοποίηση διάφορων τύπων στοιχείων των διεπαφών που υπάρχουν στις ΠΣΠ αλληλεπιδράσεις. Παρ' όλα αυτά, για την κάλυψη του περιορισμού HATEOAS, εξετάσαμε εναλλακτικές πηγές όπως το [51] και το [158]. Τέλος, εξετάστηκαν ορισμένες άλλες προσπάθειες ορισμού μεταμοντέλων στην περιοχή RESTful σχεδίασης [133], [135]. Παρ' όλα αυτά, διαφέρει σημαντικά η οπτική των εργασιών αυτών ως προς την πρόκληση ορισμού ενός μεταμοντέλου ΠΣΠ υπηρεσιών, από την οπτική του προτεινόμενου περιβάλλοντος-πλαισίου.

Το μεταμοντέλο MM_R. Το προτεινόμενο μεταμοντέλο για υπηρεσίες ΠΣΠ διεπαφών απεικονίζεται στο Σχήμα 5.4 και παρέχεται ως Ecore μοντέλο στο Παράρτημα Γ'. Το στοιχείο που βρίσκεται στη ρίζα του MM_R είναι η κλάση `ROServiceDescription` και περιέχει ένα στοιχείο `Context`, ένα σύνολο στοιχείων `ResourceInteractionPoint`, ένα σύνολο στοιχείων `Hypermedia Relation` και ένα σύνολο στοιχείων `ResourceType`.

Η κλάση `Context` αναπαριστά μια αφαίρεση του περιβάλλοντος στο οποίο λειτουργεί η ΠΣΠ υπηρεσία. Πιο συγκεκριμένα, ένα στοιχείο `Context` περιέχει ένα στοιχείο `Host` το οποίο φέρει ένα χαρακτηριστικό αναγνώρισης (`identifier`), ένα όνομα για την εφαρμογή και ένα σύνολο παραμέτρων περιβάλλοντος για την υπηρεσία. Το χαρακτηριστικό `identifier` της κλάσης `Host` υποδεικνύει την τοποθεσία της υπηρεσίας και μπορεί επίσης να διατελεί το εξορισμού σημείο εισόδου για την ΠΣΠ διεπαφή που προσδιορίζεται.

Η κλάση `ResourceInteractionPoint` υποδηλώνει τα διακριτά ως προς

[1]rest-discuss mailing list, https://groups.yahoo.com/neo/groups/rest-discuss/info



Σχήμα 5.4: MM_R: ένα μεταμοντέλο για τον προσδιορισμό υπηρεσιών ΠΣΠ διεπαφών

τη σημασιολογία τους σημαία αλληλεπίδρασης σε μια ΠΣΠ διεπαφή. Ένα σημείο αλληλεπίδρασης σχετίζεται με ένα στοιχείο `ResourceType` το οποίο μοντελοποιεί τύπους REST πόρων και συσχετίζονται με ένα στοιχείο `Action` (πράξη) που εφαρμόζεται στον πόρο κατά την προσπέλαση του ΠΣΠ σημείου αλληλεπίδρασης. Η πράξη είναι ένας ειδικός τύπος δεδομένων ελέγχου και στο MM_R θεωρούμε πέντε τύπους για την απόδοση σημασιολογίας στα στοιχεία `Action`, συγκεκριμένα τους: `CREATE`, `READ`, `UPDATE`, `DELETE` και `PROCESS`. Οι πρώτοι τέσσερις τύποι πράξεων αντιστοιχούν στη συνήθη σημασιολογία CRUD πράξεων, ενώ ο τελευταίο τύπος πράξεις (`PROCESS`) φέρει σημασιολογία αυθαίρετης επεξεργασίας δεδομένων ώστε να επιτρέπει τη μοντελοποίηση πράξεων που δεν πραγματοποιούν τυπική CRUD μεταχείριση των πόρων. Επίσης, ένα ΠΣΠ σημείο αλληλεπίδρασης περιέχει μια προδιαγραφή των αιτημάτων τα οποία είναι σε θέση να υποδεχθεί μια υπηρεσία μέσω της κλάσης `RequestSpecification` και μια προδιαγραφή των δυνατών αποκρίσεων που μπορεί να επιστρέψει, μέσω της κλάσης `ResponseSpecification`. Οι κλάσεις `RequestSpecification` και `ResponseSpecification` μοντελοποιούν το πως πρέπει να σχηματίζονται τα μηνύματα αιτημάτων και αποκρίσεων. Πιο συγκεκριμένα, και οι δύο κλάσεις είναι επεκτάσεις της αφαιρετικής κλάσης `InteractionSpecification` η οποία σχετίζεται με ακριβώς ένα στοιχείο `ResourceIdentifierTemplate` και ένα σύνολο στοιχείων `Representation` (αναπαράσταση) που εκτίθενται. Ένα στοιχείο `ResourceIdentifierTemplate` αναγνωρίζει τον πόρο που προσπελάζεται ή τυγχάνει μεταχείρισης ενώ ένα στοιχείο `Representation` προσδιορίζει μια αναπαράσταση η οποία ανταλλάσσεται κατά την αλληλεπίδραση. Επίσης, η κλάση `InteractionSpecification` περιέχει ένα σύνολο δεδομένων ελέγχου, μέσω της κλάσης `ControlData`. Τα δεδομένα ελέγχου χρησιμοποιούνται σε μια προδιαγραφή αλληλεπίδρασης για να εξειδικεύσει τη σημασιολογία της πρόθεσης της αλληλεπίδρασης και την ερμηνεία της. Στο MM_R παρέχουμε μια λίστα δεκατεσσάρων δεδομένων ελέγχου που ορίζονται ως αφαιρέσεις των τύπων δεδομένων ελέγχου που χρησιμοποιούνται συνήθως σε RESTful API. Πρέπει από τις σχέσεις που κληρονομούνται από την κλάση `InteractionSpecification` η κλάση `ResponseSpecification` αναφέρεται επίσης σε ένα σύνολο στοιχείων `ResourceMetadatum` τα οποία αντιστοιχούν σε στοιχεία μεταδεδομένων πόρων του REST.

Η κλάση `ResourceType` μοντελοποιεί τους τύπους πόρων όπως αυτοί ορίζονται στο REST [53]. Ένας τύπος πόρου μπορεί να περιέχει στοιχεί-

α Representation σε αντιστοιχία με στοιχεία δεδομένων και δεδομένων αναπαραστάσεων στο REST. Συνεπώς, καθώς μια αναπαράσταση αναλύεται σε *representation data* και *representation metadata* (Πίνακας 2.3), η κλάση Representation περιέχει την κλάση RepresentationData και ένα σύνολο στοιχείων RepresentationMetadatum. Επίσης, μια αναπαράσταση μπορεί να χαρακτηριστεί από μια κατεύθυνση (Direction), υποδεικνύοντας αν παράγεται από τον πελάτη και παραλαμβάνεται από τον εξυπηρετητή (CS), αν παράγεται από τον εξυπηρετητή και παραλαμβάνεται από τον πελάτη (SC), ή και τα δύο (ANY). Τα δεδομένα αναπαράστασης προσδιορίζονται από τις κλάσεις StateDatum και StateElement οι οποίες αποτελούν γενικευμένα μέσα αποτύπωσης της κατάστασης ενός πόρου. Η κλάση RepresentationMetadatum μοντελοποιεί πληροφορίες οι οποίες περιγράφουν την αναπαράσταση (π.χ. τον τύπο δεδομένων της αναπαράστασης) και χαρακτηρίζεται από ένα type το οποίο προσδιορίζει την αντίστοιχη σημασιολογία του μεταδεδομένου. Στο MM_R έχουμε αναγνωρίσει επτά τύπου μεταδεδομένων αναπαράστασης που χρησιμοποιούνται σε RESTful υλοποιήσεις οι οποίοι περιλαμβάνονται στην απαρίθμηση RepresentationMetadataTypes. Επίσης, η κλάση ResourceType ενδέχεται να περιέχει ένα σύνολο στοιχείων ResourceMetadatum τα οποία μοντελοποιούν τα μεταδεδομένα πόρου του REST. Τα στοιχεία ResourceMetadatum είναι ζεύγη κλειδιού (αναγνωριστικού) - τιμής και η σημασιολογία τους χαρακτηρίζεται από μια ιδιότητα type. Στο MM_R έχουμε αναγνωρίσει πέντε τύπους μεταδεδομένων πόρων όπως ορίζεται από την απαρίθμηση ResourceMetadataTypes.

Ακολουθώντας τον ορισμό του REST, ένας πόρος μπορεί να έχει ένα ή περισσότερα αναγνωριστικά. Συνεπώς, ένα στοιχείο ResourceType μπορεί να συσχετιστεί με ένα ή περισσότερα στοιχεία ResourceIdentifierTemplate. Η κλάση ResourceIdentifierTemplate επιτρέπει τον προσδιορισμό προτύπων που οργανώνουν τους τύπους πόρων σε ιεραρχίες, οι οποίες μπορούν να χρησιμοποιηθούν για να κατασκευαστούν αναγνωριστικά κατά τον χρόνο εκτέλεσης. Ωστόσο, θα πρέπει να σημειωθεί το αν ένα πρότυπο αναγνωριστικού πόρου θα χρησιμοποιηθεί άμεσα για τη δημιουργία του πραγματικού αναγνωριστικού του πόρου κατά τον χρόνο εκτέλεσης, ή έμμεσα, μέσω εσωτερικών αντιστοιχίσεων τα οποία παράγουν URI αποτελεί ανεξάρτητο ζήτημα. Στο επίπεδο προδιαγραφής, επιλέξαμε την ιεραρχική μοντελοποίηση των προτύπων των αναγνωριστικών καθώς αποτελεί συμβατό μοτίβο με την ιεραρχική φύση που έχουν τα μοντέλα πόρων, χαρακτηριστικό εγγενές στο REST. Μέσω της ιεραρχικής δομής αποτυπώνονται

με άμεσο και σαφή τρόπο υπαρξιακές σχέσεις μεταξύ των πόρων. Ένα στοιχείο `ResourceIdentifierTemplate` προσδιορίζεται σε σχέση με το πλαίσιο μιας υπηρεσίας (κλάση `Context`) και περιλαμβάνει το πολύ ένα στοιχείο `Pattern`. Η κλάση `Pattern` ορίζει μια ανάλυση του προτύπου του αναγνωριστικού πόρου σε μεγαλύτερης ανάλυσης συνθετικά στοιχεία. Πιο συγκεκριμένα, η κλάση `Pattern` περιλαμβάνει δύο συλλογές τύπων `Part`, τον τύπο `StaticPart` και τον τόπο `DynamicPart`. Ένα `Part` μοντελοποιεί ένα τμήμα του προτύπου του αναγνωριστικού ενός στοιχείου `ResourceType`. Η κλάση `Part` περιέχει μια ιδιότητα `label` (ετικέτα) και μια ιδιότητα `position` (Θέση) η οποία προσδιορίζει το που τοποθετείται το τμήμα στην ιεραρχία. Τα στοιχεία `DynamicPart` αναπαριστούν τμήματα του προτύπου του αναγνωριστικού πόρου το οποίο λαμβάνει δυναμικές τιμές κατά τον χρόνο εκτέλεσης. Η αποτίμηση του τμήματος του αναγνωριστικού που ορίζεται από ένα δυναμικό τμήμα βασίζεται στην ιδιότητα `label` και στην έκφραση που περιλαμβάνεται στην ετικέτα. Στο MM_R ορίζουμε τρεις τύπους τέτοιες εκφράσεων, συγκεκριμένα `ELEMENT`, `QUERY` και `CUSTOM_TYPE`. Τα `ELEMENT` δυναμικά τμήματα αναπαριστούν αναγνωριστικά στοιχείων συλλογών (π.χ. ένα αναγνωριστικό εγγράφου σε μια συλλογή εγγράφων). Τα `QUERY` δυναμικά τμήματα αναπαριστούν μοναδικές ή πολλαπλές παραμέτρου του ίδιου επιπέδου οι οποίες χρησιμοποιούνται για να φιλτράρουν ή να εξειδικεύσουν σε σημασιολογικό επίπεδο τους πόρους επί τον οποίων ορίζονται, αναγνωρίζοντας ουσιαστικά ανεξάρτητους πόρους. Ο τύπος ετικέτας `CUSTOM_TYPE` χρησιμοποιείται ως ένας μηχανισμός επέκτασης έτσι ώστε να μπορούν να χρησιμοποιηθούν επιπλέον τύποι εκφράσεων. Ένα στοιχείο `StaticPart` αντιστοιχεί σε τμήματα των αναγνωριστικών τα οποία είναι στατικά, δηλαδή η ιδιότητα label είναι ίση με την τιμή του αντίστοιχου τμήματος κατά τον χρόνο εκτέλεσης. Ένα πρότυπο αναγνωριστικού πόρου το οποίο δεν περιέχει ένα στοιχείο `Pattern` αναπαριστά ένα πόρο - ρίζα στο μοντέλο πόρων, το οποίο είναι τυπικά και σημείο εισόδου στην ΠΣΠ υπηρεσία.

Τέλος, ένα στοιχείο `ROServiceDescription` ενδέχεται να περιέχει ένα σύνολο στοιχείων `HypermediaRelation`. Η κλάση `HypermediaRelation` επιτρέπει τον προσδιορισμό *υπερμέσων* (*hypermedia*). Όπως συζητήθηκε στο Κεφάλαιο 2 ο περιορισμός HATEOAS (hypermedia as the engine of application state) του REST δηλώνει ότι η υπηρεσία ή ο πάροχος της υπηρεσίας θα πρέπει να παρέχει πληροφορίες στον πελάτη ή καταναλωτή της υπηρεσίας μέσω υπερμέσων, ως προς το ποιες είναι οι επόμενες δυνατές αλληλεπιδράσεις που μπορούν να λάβουν χώρα και με ποιο τρόπο μπορούν να ακολουθηθούν. Ο πελάτης ενδέχεται να ακολουθή-

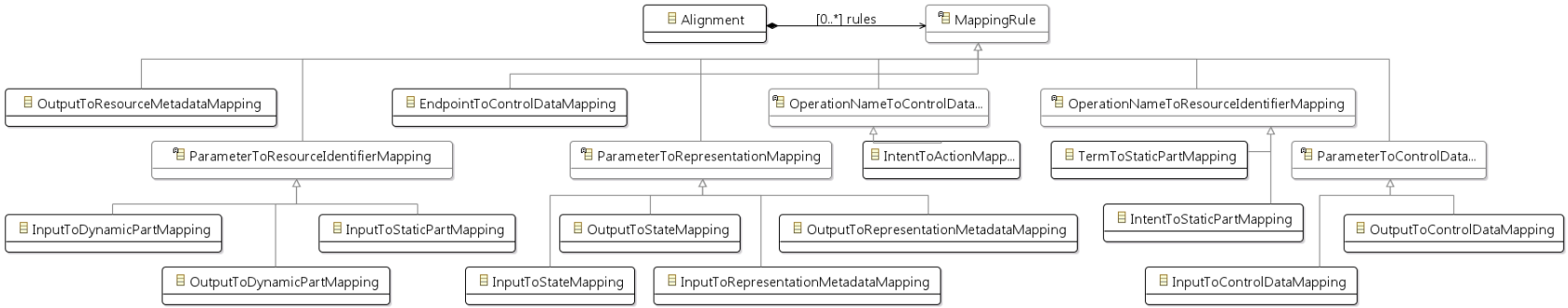
σει τα παρεχόμενα υπερμέσα, καθοδηγώντας με τον τρόπο αυτό την εξέλιξη της κατάστασης της εφαρμογής. Στο MM_R οι σχέσεις υπερμέσων, συνδέουν μεταξύ τους στοιχεία `ResourceInteractionPoint`. Πιο συγκεκριμένα, δύο ΠΣΠ σημεία αλληλεπίδρασης τα οποία σχετίζονται με ένα στοιχείο `HypermediaRelation` έχουν δύο διαφορετικούς ρόλους, ένα ως `source` (πηγή) και ένα ως `target` (στόχος). Επιπλέον, μια σχέση υπερμέσων ενδέχεται να ελέγχεται από ένα στοιχείο `HypermediaCondition`, που σημαίνει ότι μπορεί να ενεργοποιηθεί ανάλογα με τιμές των παραμέτρων της υπηρεσίας. Τέλος, μια σχέση υπερμέσων περιέχει ένα στοιχείο `HypermediaControl` το οποίο χρησιμοποιείται για να επεκτείνει τη σημασιολογία των υπερμέσων και να προδιαγράψει το πως ο πελάτης θα πρέπει να ερμηνεύσει τη σχέση υπερμέσων έτσι ώστε να πραγματοποιήσει επόμενες αλληλεπιδράσεις με την υπηρεσία. Μέσω της κλάσης `HypemediaRelation` και των συνιστωσών της, επιλέξαμε μια αφαιρετική αναπαράσταση των *αφφορδανσες* στο MM_R, έτσι ώστε διαφορετικοί τύποι υπερμέσων (π.χ. ορισμένοι σε τύπους μέσων, link relations, σημασιολογικοί, παρεχόμενοι από το πρωτόκολλο επικοινωνίας) να μπορούν να περιληφθούν στο μοντέλο.

Προδιαγραφή αντιστοιχιών υποδειγμάτων

Ένα από τα βασικά ζητήματα για την αποτίμηση της υποκαταστασιμότητας μεταξύ ΠΣΔ και ΠΣΠ API είναι η δυνατότητα μοντελοποίηση και ανάλυσης των εννοιολογικών και αρχιτεκτονικών συσχετίσεων μεταξύ υπηρεσιών που ορίζονται μέσω των μεταμοντέλων MM_P και MM_R. Για το λόγο αυτό, εισάγουμε το μοντέλο MM_A το οποίο ορίζει ένα σύνολο *αντιστοιχιών* υποδειγμάτων διεπαφών μεταξύ των δύο μεταμοντέλων υπηρεσιών. Το MM_A απεικονίζεται στο Σχήμα 5.5 και παρέχεται στο Παράρτημα Γ΄ ως Ecore μοντέλο. Το μεταμοντέλο MM_A ορίζει πως τα στοιχεία των διεπαφών των υπηρεσιών που μοντελοποιούνται με βάση τα MM_P και MM_R, μπορούν να συσχετιστούν σημασιολογικά και αρχιτεκτονικά.

Η κλάση που βρίσκεται στη ρίζα του μοντέλου MM_A είναι η κλάση `Alignment`. Ένα στοιχείο `Alignment` παίζει το ρόλο του φορέα των εννοιολογικών οντοτήτων που απαιτούνται για τον ορισμό του πως μια ΠΣΔ διεπαφή και μια ΠΣΠ διεπαφή σχετίζονται και περιέχει ένα σύνολο στοιχείων `MappingRule`.

Ένα `MappingRule` προσδιορίζει συσχετίσεις στοιχείων διεπαφών στο πλαίσιο μια ανταπόκρισης που υπάρχει μεταξύ ενός στοιχείου `Procedural`



Σχήμα 5.5: MM_A: ένα μεταμοντέλο για τον προσδιορισμό αρχιτεκτονικών αντιστοιχιών μεταξύ διεπαφών που ορίζονται με βάση τα μεταμοντέλα MM_P και MM_R

`Interaction` και ενός στοιχείου `ResourceInteractionPoint` και μπορεί να αναγνωριστεί μέσω του χαρακτηριστικού `label` που φέρει. Το μεταμοντέλο MM_A περιέχει δεκαεννιά υποκλάσεις της αφαιρετικής κλάσης `MappingRule`, οι οποίες διαιρούνται σε δύο επίπεδα αφαίρεσης. Στο πρώτο επίπεδο υπάρχουν επτά υποκλάσεις, πέντε εκ των οποίων είναι επίσης αφαιρετικές ενώ οι υπόλοιπες δύο είναι συμπαγείς. Η εισαγωγή των επιπέδων αφαίρεσης στο MM_A επιτρέπει τη συστηματική ανάλυση των θεμάτων που αφορούν στην ευθυγραμμία των διεπαφών, έτσι ώστε να μπορούν να προσδιορίζονται συσχετίσεις κατάλληλης λεπτομέρειας μεταξύ των στοιχείων των διεπαφών των δύο υποδειγμάτων. Υπό αυτή την έννοια, κάθε αφαιρετική υποκλάση της `MappingRule` παίζει το ρόλο μιας εννοιολογικής ομαδοποίησης συμπαγών κανόνων αντιστοίχισης.

Κάθε συμπαγής υποκλάση της `MappingRule` μοντελοποιεί μια συγκεκριμένη συσχέτιση, ή μια συμπαγή αντιστοίχιση μεταξύ στοιχείων του MM_P και του MM_R, στα πλαίσια πάντα ενός ζεύγους στοιχείων `ProceduralInteraction` και `ResourceInteractionPoint`. Η παρουσίαση των τύπων των κανόνων αντιστοίχισης οργανώνεται σύμφωνα με τη διάσταση στην οποία αφορούν. Για το λόγο αυτό θεωρούμε τις ακόλουθες διαστάσεις:

- *Intention* (Πρόθεσης), η οποία χαρακτηρίζει πληροφορίες που ανταλλάσσονται σχετικά με τη σημασιολογία των δυνατοτήτων μιας υπηρεσίας: τι/πως/αν κάτι χρειάζεται να γίνει/έχει γίνει,
- *Application Data* (Δεδομένων Εφαρμογής), η οποία χαρακτηρίζει πληροφορίες οι οποίες ανταλλάσσονται σχετικά με τιμές δεδομένων που απαιτούνται ή παράγονται από τη λογική της εφαρμογής (application logic) που εφαρμόζουν οι δυνατότητες της υπηρεσίας,
- *Metadata* (Μεταδεδομένων), η οποία χαρακτηρίζει πληροφορίες οι οποίες ανταλλάσσονται σχετικά με συγκεκριμένα στιγμιότυπα αλληλεπιδράσεων (π.χ. ποιος εξυπηρέτησε μια αλληλεπίδραση, πότε, κλπ).

Δεδομένων των παραπάνω, οι δεκατέσσερις συμπαγής τύποι κανόνων αντιστοίχισης που ορίζονται στο MM_A είναι οι ακόλουθοι:

Intention

1) `TermToStaticPartMapping`: συσχετίζει ένα στοιχείο `Term` ενός μοντέλου όρων το οποίο σχετίζεται με ένα στοιχείο `Name` ενός `Signature` σε μια ΠΣΔ διεπαφή, με ένα στοιχείο `StaticPart` μιας ΠΣΠ διεπαφής. Για παράδειγμα, θεωρώντας μια λειτουργία με όνομα *getOrders* και ένα πόρο */βασκετσ/*, το *ορδερς* `Concept` του μοντέλου όρων για το *getOrders*, μπορεί να συσχετιστεί με το *βασκετς* `StaticPart` του παραπάνω προτύπου αναγνωριστικού μέσω ενός στοιχείου `TermToStaticPartMapping`.

2) `IntentToStaticPartMapping`: συσχετίζει έναν τύπο όρου `Intent`, ο οποίος σχετίζεται με το `Name` ενός `Signature` σε μια ΠΣΔ διεπαφή, με ένα στοιχείο `StaticPart` που περιλαμβάνεται σε ένα πρότυπο αναγνωριστικού μιας ΠΣΠ διεπαφής. Για παράδειγμα, ο `Intent` όρος *reboot* της λειτουργίας *rebootVirtualMachine* σε μια ΠΣΔ διεπαφή μπορεί να αντιστοιχιστεί μέσω ενός `InputToStaticPartMapping` σε ένα *restart* `StaticPart` στοιχείο του συσχετισμένου πόρου */vm/vm.id/restart*.

3) *IntentToActionMapping*: συσχετίζει την κλάση `Intent` του μοντέλου όρων που σχετίζεται με την κλάση `Name` της κλάσης `Signature` των ΠΣΔ διεπαφών, και την κλάση `Action` των ΠΣΠ διεπαφών. Για παράδειγμα, ο `Intent` όρος *add* μιας λειτουργίας *addOrderItem* σε μια ΠΣΔ διεπαφή αντιστοιχίζεται μέσω ενός κανόνα `InputToActionMapping` σε ένα στοιχείο `Action` το οποίου η ιδιότητα `type` είναι `CREATE` (το οποίο τελικά μπορεί να ερμηνευτεί σε POST στο HTTP).

4) `InputToControlDataMapping`: συσχετίζει την κλάση `InputParameter` των ΠΣΔ διεπαφών, και την κλάση `ControlDatum` των ΠΣΠ διεπαφών. Για παράδειγμα, παράμετροι αυθεντικοποίηση που ονομάζονται *access-token*, και ορίζονται ως παράμετροι εισόδου σε υπογραφές λειτουργιών μιας ΠΣΔ διεπαφής, μπορούν να συσχετιστούν με ένα `ControlDatum` στοιχείο *Authentication* που ορίζεται σε μια ΠΣΠ διεπαφή.

5) `OutputToControlDataMapping`: συσχετίζει την κλάση `OutputParameter` των ΠΣΔ διεπαφών, και την κλάση `ControlDatum` των ΠΣΠ διεπαφών. Για παράδειγμα, μια *valid-until* παράμετρος εξόδου (`OutputParameter`) που επιστρέφεται από τη λειτουργία *getTicketPrice* μιας υπηρεσίας για να καθοδηγήσει τον πε-

λάτη σχετικά με την πολιτική επανάληψης του ερωτήματος για ανανεωμένη τιμή, αντιστοιχίζεται σε ένα `ControlDatum` στοιχείο *Cache-Control* που επιστρέφεται έπειτα από μια `READ` πράξη στον τύπο πόρου */tickets/{ticket.id}/*.

6) `OutputToResourceMetadataMapping`: συσχετίζει ένα στοιχείο `OutputParameter` μιας ΠΣΔ διεπαφής με ένα στοιχείο `ResourceMetadata` μιας ΠΣΠ διεπαφής. Για παράδειγμα, η παράμετρος εξόδου *next* (`OutputParameter`) της λειτουργίας *getReport* η οποία περιέχει το αναγνωριστικό της επόμενης αναφοράς από αυτή που έχει επιστραφεί, μπορεί να αντιστοιχιστεί σε ένα `ResourceMetadatum` στοιχείο *Link* το οποίο επιστρέφεται από μια πράξη `READ` στον τύπο πόρου */documents/{document.id}/*.

Application Data

7) `InputToDynamicPartMapping`: συσχετίζει την κλάση `InputParameter` των ΠΣΠ διεπαφών, και την κλάση `DynamicPart` των ΠΣΠ διεπαφών. Για παράδειγμα, η παράμετρο εισόδου *orderId* της λειτουργίας *getOrder* σε μια ΠΣΔ διεπαφή μπορεί να αντιστοιχιστεί μέσω του `InputToDynamicPartMapping` σε ένα δυναμικό τμήμα *{cart.id}* του προτύπου αναγνωριστικού πόρου */carts/{cart.id}/* το οποίο αναγνωρίζει πόρους καλαθιών αγορών σε μιας ΠΣΠ υπηρεσία.

8) `InputToStaticPartMapping`: συσχετίζει ένα στοιχείο `InputParameter` μιας ΠΣΔ διεπαφής, με ένα στοιχείο `StaticPart` το οποίο περιλαμβάνεται σε ένα πρότυπο αναγνωριστικού πόρου που περιλαμβάνεται σε μια ΠΣΠ διεπαφή, με βάση ένα προαιρετικό χαρακτηριστικό `value`. Για παράδειγμα ας υποθέσουμε ότι ένα WSDL περιέχει τη λειτουργία *RetrieveCustomerAddress* η οποία ανάμεσα στις παραμέτρους εισόδου της περιλαμβάνει την παράμετρο *addressType* η οποία προσδιορίζει το τύπο διεύθυνσης που θα ανακτηθεί (π.χ. *home* ή *work*). Επίσης, ας υποθέσουμε ότι η αντίστοιχο ΠΣΠ διεπαφή αναγνωρίζει τις διευθύνσεις πελάτη ως ορατούς και αυτόνομους πόρους μέσω των παρακάτω προτύπων αναγνωριστικών: */customers/{customer.id}/addresses/residence/*, */customers/{customer.id}/addresses/workplace/*. Τότε ένας κανόνας `InputToStaticPartMapping` μια την τιμή του `value` χαρακτηριστικού ίση με *home* μπορεί να χρησιμοποιηθεί για να συσχετίσει το στοιχείο *addressType* (`InputParameter`) με το στοιχείο *residence* (`StaticPart`), ενώ ένα δεύτε-

ρο στοιχείο InputToStaticPartMapping με τιμή value ίση με *work* θα μπορούσε να συσχετίζει το *addressType* (InputParameter) στο *workplace* (StaticPart). 9) OutputToDynamicPartMapping: συσχετίζει ένα στοιχείο OutputParameter μιας ΠΣΔ διεπαφής και ένα στοιχείο DynamicPart μιας ΠΣΠ διεπαφής, το οποίο περιέχεται σε ένα πρότυπο αναγνωριστικού τύπου πόρου. Για παράδειγμα, ας θεωρήσουμε τη λειτουργία *addOrder* η οποία επιστρέφει το OutputParameter *orderId*, και ένα πόρο */orders/* όπου η πράξη CREATE επιστρέφει το αναγνωριστικό της παραγγελίας που δημιουργείται, το οποίο έχει πρότυπο το */orders/{order.id}/*. Στην περίπτωση αυτή το *orderId* (OutputParameter) μπορεί να συσχετιστεί με το *{order.id}* (DynamicPart) μέσω ενός στοιχείου OutputToDynamicPartMapping.

10) InputToStateMapping: συσχετίζει ένα στοιχείο InputParameter μιας ΠΣΔ διεπαφής και ένα στοιχείο StateDatum μιας ΠΣΠ διεπαφής. Για παράδειγμα, η παράμετρος εισόδου *catalogItemId* της λειτουργίας *addOrderItem* σε μια ΠΣΔ διεπαφή μπορεί να αντιστοιχιστεί μέσω ενός κανόνα InputToStateMapping στο στοιχείο *product* (StateDatum) που περιλαμβάνεται στα δεδομένα αναπαράστασης ενός αιτήματος δημιουργίας που αποστέλλεται σε ένα πόρο καλαθιού αγαθών μιας ΠΣΠ διεπαφής με πρότυπο αναγνωριστικού */baskets/{basket.id}/goods/*.

11) OutputToStateMapping: συσχετίζει μια παράμετρο εξόδου OutputParameter μιας ΠΣΔ διεπαφής και ένα στοιχείο StateDatum μιας ΠΣΠ διεπαφής. Για παράδειγμα, η παράμετρος εξόδου *RH* (σχετική υγρασία) (OutputParameter) μιας λειτουργίας *getWeather* μπορεί να συσχετιστεί στο StateDatum στοιχείο *humidity-percentage* που περιλαμβάνεται στην αναπαράσταση του πόρου */weather/city/* μέσω ενός κανόνα OutputToStateMapping.

Metadata

12) EndpointToControlDataMapping: συσχετίζει την κλάση Endpoint των ΠΣΔ διεπαφών, με την κλάση ControlDatum των ΠΣΠ διεπαφών. Για παράδειγμα, το URL που ορίζεται για ως endpoint μιας ΠΣΔ διεπαφής μπορεί να συσχετιστεί στο ControlDatum στοιχείο *Host* αλληλεπιδράσεων της ΠΣΔ διεπαφής.

13) `InputToRepresentationMetadataMapping`: συσχετίζει την κλάση `InputParameter` των ΠΣΔ διεπαφών, με την κλάση `Representation Metadatum` των ΠΣΠ διεπαφών. Για παράδειγμα, ένα `InputParameter` στοιχείο *locale* της λειτουργίας *getReport*, μπορεί να συσχετιστεί σε ένα `RepresentationMetadata` στοιχείο *Language* το οποίο αποστέλλεται σε `READ` αιτήματα στον τύπο πόρου */reports/{report.id}/* μέσω ενός κανόνα `InputToRepresentationMetadataMapping`.

14) `OutputToRepresentationMetadataMapping`: συσχετίζει ένα στοιχείο `OutputParameter` μιας ΠΣΔ διεπαφής με ένα στοιχείο `Representation Metadatum` μιας ΠΣΠ διεπαφής. Για παράδειγμα, μια παράμετρος εξόδου *timestamp* (`OutputParameter`) που επιστρέφεται από τη λειτουργία *fetchCurrentSharePrice* μιας ΠΣΔ υπηρεσίας, μπορεί να συσχετιστεί μέσω ενός κανόνα `OutputToRepresentationMetadataMapping` με ένα `RepresentationMetadatum` στοιχείο *Last-Modified* το οποίο περιλαμβάνεται σε αναπαραστάσεις αποκρίσεων `READ` πράξεων, στον τύπο πόρου */stocks/{stock.id}/price/*.

5.2.3 Παραγωγή open net

Οι πληροφορίες που περιέχονται στα μοντέλα M_P, M_R και M_A, τα οποία συμμορφώνονται στα μεταμοντέλα MM_P, MM_R, και MM_A, αντίστοιχα, χρησιμοποιούνται για να ελεγχθεί αν η ΠΣΔ υπηρεσία S_P και η ΠΣΠ υπηρεσία S_R μπορεί να υποκαταστήσει η μια την άλλη. Ο έλεγχος αυτός γίνεται σε δύο φάσεις. Η πρώτη φάση είναι μια διαδικασία μετασχηματισμού η οποία μετατρέπει τα μοντέλα M_P και M_R σε δύο open net, N_P και N_R, τα οποία αντιστοιχούν στις υπηρεσίες S_P και S_R. Η δεύτερη φάση είναι η εξέταση της ύπαρξης accordance μεταξύ των open net που έχουν δημιουργηθεί, το αποτέλεσμα της οποίας δείχνει αν το N_R μπορεί να υποκαταστήσει το N_P και αντίστροφα.

Στις παραγράφους που ακολουθούν εστιάζουμε στην πρώτη φάση και συζητάμε τη διαδικασία μετασχηματισμού που εισηχθεί στην Ενότητα 5.2.1 και απεικονίζεται στο Σχήμα 5.2.1.

Τα βήματα του μετασχηματισμού αναλύονται μέσω δύο υπηρεσιών που χρη-

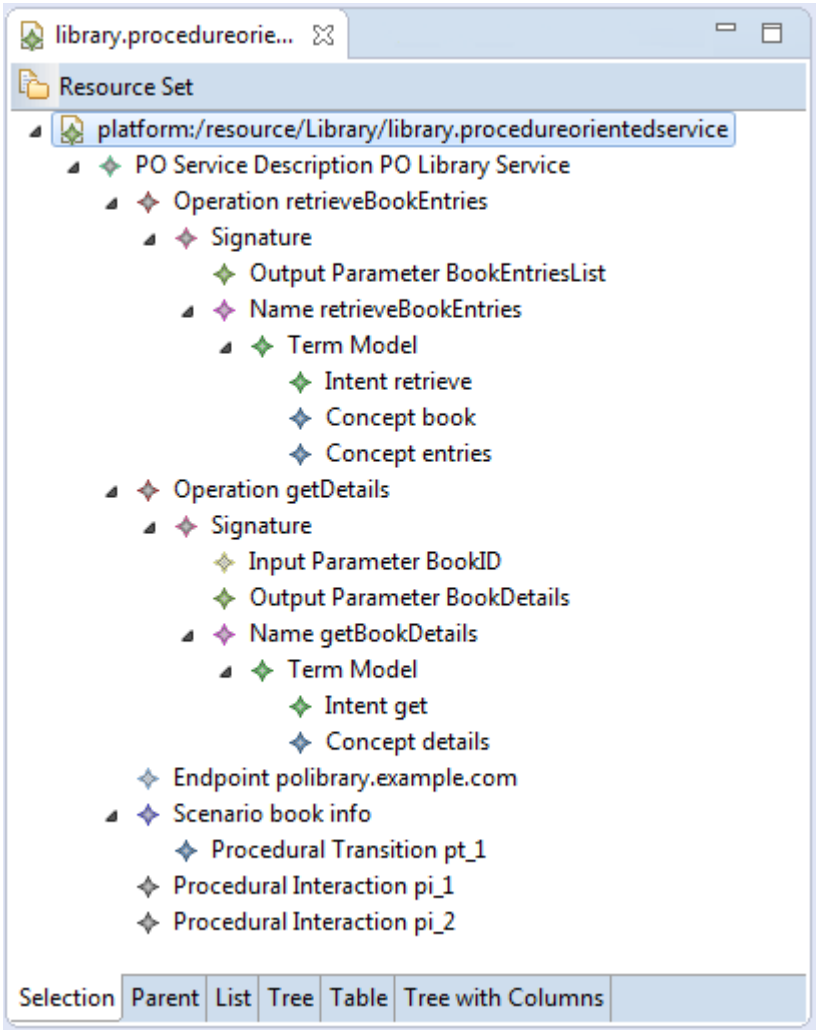
σιμοποιούνται ως παραδείγματα, την *PO-Library* και την *RO-Library*, οι οποίες απεικονίζονται στον Πίνακα 5.1 και στο Σχήμα 5.12, αντίστοιχα.

Παραγωγή open net για ΠΣΔ υπηρεσίες

Στην παρούσα ενότητα παρουσιάζουμε τα βήματα του μετασχηματισμού τα οποία παράγουν το open net N_P από την προδιαγραφή του API της ΠΣΔ υπηρεσίας S_P. Στη βιβλιογραφία έχει προταθεί ποικίλες προσεγγίσεις αντιστοίχισης ΠΣΔ προδιαγραφών υπηρεσιών σε δίκτυα Petri [81], [143] τα οποία εστιάζουν πρωτίστως στη μεταφορά της ροής ελέγχου και αλληλεπιδράσεων όπως εκφράζεται σε μοντέλα όπως η γλώσσα BPEL, αντιμετωπίζοντας αφαιρετικά την προδιαγραφή των μηνυμάτων. Στην ανάλυση που γίνεται στο παρόν κεφάλαιο, λαμβάνονται υπόψη τόσο η ροή ελέγχου και αλληλεπιδράσεων όσο και η ανάλυση και δομή των μηνυμάτων, έτσι ώστε να είναι δυνατή η εξέταση των υποκαταστασιμότητας υπηρεσιών διαφορετικών υποδειγμάτων. Συνεπώς, στη συνέχεια της ενότητας παρουσιάζουμε τρία βήματα μετασχηματισμού της ΠΣΔ διαδρομής μετασχηματισμού, τα οποία οδηγούν στην παραγωγή του ΠΣΔ open net N_P.

Συνοδευτικό παράδειγμα: *Procedure-Oriented Library*. Για τη διευκόλυνση της συζήτησης σχετικά με την ΠΣΔ διαδρομή μετασχηματισμού, χρησιμοποιούμε ένα απλό παράδειγμα ΠΣΔ υπηρεσίας που αφορά σε λειτουργικότητα μιας βιβλιοθήκης και στο οποίο αναφερόμαστε ως *PO-Library*. Η υπηρεσία περιλαμβάνει τις εξής δύο υπηρεσίες: `retrieveBookEntries` η οποία επιστρέφει μια λίστα εγγραφών οι οποίες αφορούν σε βιβλία που υπάρχουν στον κατάλογο της βιβλιοθήκης (`BookEntriesList`), και τη λειτουργία `getDetails` η οποία δέχεται ως είσοδο μια παράμετρο `BookID` και επιστρέφει μια παράμετρο εξόδου `BookDetails`, παρέχοντας πληροφορίες σχετικά με μια συγκεκριμένη εγγραφή βιβλίου. Με βάση τα παραπάνω και ακολουθώντας τις έννοιες του μεταμοντέλου MM_P, το μοντέλο M_P περιέχει δύο στοιχείο `Operation` μαζί με τις πληροφορίες των υπογραφών τους, δύο αντίστοιχα στοιχεία `ProceduralInteraction` pi_1 για την άντληση της λίστας των βιβλίων και pi_2 για την άντληση λεπτομερειών για μια εγγραφή βιβλίου και ένα σενάριο το οποίο περιέχει ένα στοιχείο `Procedural Transition` $pt_1 = \{pi_1, pi_2\}$. Το Σχήμα 5.6 απεικονίζει μια αναπαράσταση του μοντέλου της υπηρεσίας *PO-Library*, M_P, όπως απεικονίζεται στο εργαλείο μεταχείρισης μοντέλων που συμμορφώνονται στο MM_P. Επίσης, στον Πίνακα 5.1 υπάρχει

μια λίστα με τα στοιχεία που περιέχονται στο μοντέλο M_P για το *PO-Library*. Τέλος, το πλήρες μοντέλο M_P για το *PO-Library* παρέχεται στο Παράρτημα Β'.



Σχήμα 5.6: Συνοδευτικό παράδειγμα ΠΣΔ υπηρεσίας: το μοντέλο για την υπηρεσία *PO-Library* όπως εμφανίζεται στο εργαλείο μεταχείρισης M_P μοντέλων

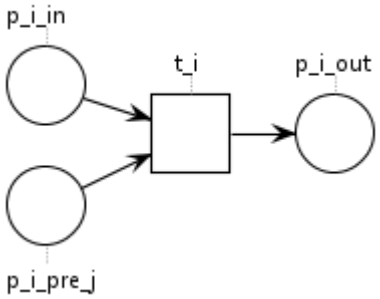
Βήμα Α: Ροή υπηρεσίας - $M_P \rightarrow N_P^{G_1}$**.** Το πρώτο βήμα είναι η αντιστοίχιση του M_P σε ένα δίκτυο Petri $N_P^{G_1}$. Πιο συγκεκριμένα, ο στόχος είναι να εισάγουμε αντιστοιχίες μεταξύ στοιχείων `ProdeceduralInteraction` και τμημάτων δικτύων Petri χαμηλού επιπέδου και ταυτόχρονα να διατηρηθούν οι εξαρτήσεις μεταξύ των αλληλεπιδράσεων όπως προβλέπονται στα στοιχεία `ProceduralTransition` του M_P.

Δεδομένων των παραπάνω, κάθε ΠΣΔ αλληλεπίδραση αντιστοιχίζεται σε ένα σύνολο στοιχείων δικτύων Petri το οποίο περιλαμβάνει μια μετάβαση, τρεις τύπους θέσεων και ένα σύνολο αντίστοιχων τόξων. Επίσης, οι διμερείς σχέσεις μεταξύ ΠΣΔ αλληλεπιδράσεων που ορίζονται στο M_P και περιέχονται σε στοιχεία `Procedural Transition` αντιστοιχούνται σε δομές δικτύων Petri που βασίζονται σε token. Εφαρμόζοντας τους κανόνες μετασχηματισμού που ορίζονται στο βήμα αυτό κατασκευάζεται η πρώτη γενιά δικτύων Petri, $N_P^{G_1}$.

Μετασχηματισμός 1 (Αντιστοιχία ΠΣΔ αλληλεπίδρασης)**.** *Έστω PI το σύνολο στοιχείων `ProceduralInteraction` στο M_P και PT το σύνολο στοιχείων `ProceduralTransition` στο M_P. Κάθε ΠΣΔ αλληλεπίδραση $pi_i \in PI$ αντιστοιχίζεται στα παρακάτω στοιχεί δικτύων Petri:*

- *μια μετάβαση t_i,*
- *μια θέση p_i^{in} και ένα τόξο $a_i^{in} = \{p_i^{in}, t_i\}$,*
- *μια θέση p_i^{out} και ένα τόξο $a_i^{out} = \{t_i, p_i^{out}\}$,*
- *μια θέση $p_{i\leftarrow j}^{pre}, \forall pi_j \in PI$ ώστε $\exists pt_k = \{pi_j, pi_i\} \in PT$, με ένα αντίστοιχο τόξο $a_{i\leftarrow j}^{pre} = \{p_{i\leftarrow j}^{pre}, t_i\}$.*

Στο Σχήμα 5.7 παρουσιάζονται τα στοιχεία δικτύων Petri που δημιουργούνται για ένα στοιχείο `ProceduralInteraction`, $pi_i \in PI$.



Σχήμα 5.7: Τμήματα δικτύων Petri που δημιουργούνται για ένα στοιχείο `ProceduralInteraction` με βάση τον Μετασχηματισμό 1

Αφου αντιστοιχιστούν όλες οι ΠΣΔ αλληλεπιδράσεις σε τμήματα δικτύων Petri θα πρέπει στο $N_P^{G_1}$ να προστεθεί ένα σύνολο τόξων έτσι ώστε τα τμήματα αυτά να συνδεθούν σύμφωνα με τη ροή και τις εξαρτήσεις μεταξύ των αλληλεπιδράσεων που αποτυπώνονται στις ΠΣΔ μεταβάσεις.

Μετασχηματισμός 2 (Αντιστοιχία ΠΣΔ μετάβασης). *Για κάθε στοιχείο* `Procedural Transition` $pt_k = \{pi_j, pi_i\} \in PT$, *δημιουργείται ένα τόξο* $a^{post}_{j \to i}$ *ώστε* $a^{post}_{j \to i} = \{t_j, p^{pre}_{i \leftarrow j}\}$.

Επιπλέον, ορίζονται ειδικές αντιστοιχίες για ΠΣΔ αλληλεπιδράσεις που είναι ελεύθερες εξαρτήσεων (δηλαδή αλληλεπιδράσεις που εκκινούν μια ροή της υπηρεσίας), όπως και για αλληλεπιδράσεις προορισμού (δηλαδή αλληλεπιδράσεις που τερματίζουν μια ροή).

Μια ΠΣΔ αλληλεπίδραση ελεύθερη εξάρτησης pi_i ορίζεται ως μια αλληλεπίδραση για την οποία ισχύει:

$$\nexists pt_k = \{pi_j, pi_i\} \in PT, \forall pi_j \in PI$$

Ομοίως, μια ΠΣΔ αλληλεπίδραση προορισμού pi_i ορίζεται ως μια αλληλεπίδραση για την οποία ισχύει:

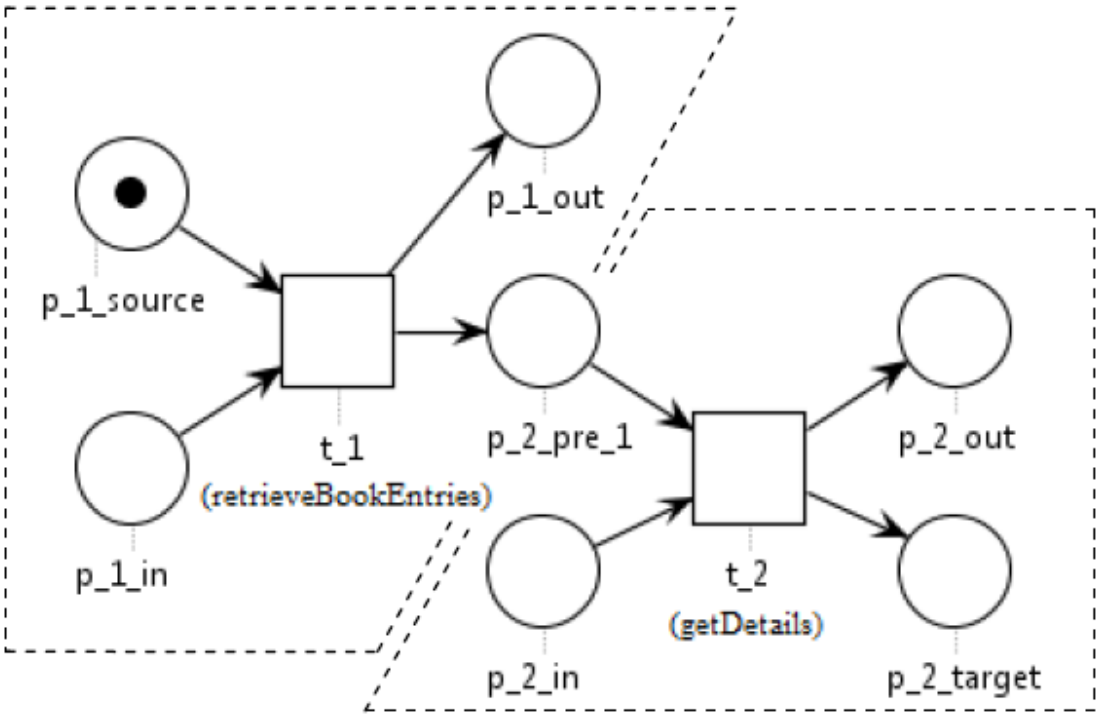
$$\nexists pt_k = \{pi_i, pi_j\} \in PT, \forall pi_j \in PI$$

Μετασχηματισμός 3 (Δημιουργία source θέσεων). *Για κάθε μετάβαση* t_i *που αντιστοιχεί σε μια ΠΣΔ αλληλεπίδραση ελεύθερη εξαρτήσεων* pi_i, *προστίθεται μια θέση* p^{source}_i *και ένα τόξο* $a^{source}_i = \{p^{source}_i, t_i\}$. *Η θέση* p^{source}_i *μαρκάρεται με ένα token στο αρχικό marking του* $N^{G_1}_P$, *δηλαδή,* $m_0(p^{source}_i) = 1$.

Μετασχηματισμός 4 (Δημιουργία target θέσεων). *Για κάθε μετάβαση* t_i *που αντιστοιχεί σε μια ΠΣΔ αλληλεπίδραση ελεύθερη εξαρτήσεων* pi_i, *προστίθεται μια θέση* p^{target}_i *και ένα τόξο* $a^{target}_i = \{t_i, p^{target}_i\}$.

Το Σχήμα 5.8 απεικονίζει το δίκτυο $N^{G_1}_P$ το οποίο παράγεται εφαρμόζοντας το Βήμα Α για την υπηρεσία *PO-Library* που παρουσιάστηκε παραπάνω (Σχήμα 5.6). Το $N^{G_1}_P$ για την υπηρεσία *PO-Library* ορίζεται από τις παρακάτω συνιστώσες:

- $T = \{t_1, t_2\}$,
- $P = \{p^{source}_1, p^{pre}_{2\leftarrow 1}, p^{target}_2, p^{in}_1, p^{in}_2, p^{out}_1, p^{out}_2\}$,
- $F = \{ \{p^{source}_1, t_1\}, \{p^{in}_1, t_1\}, \{t_1, p^{out}_1\}, \{t_1, p^{pre}_{2\leftarrow 1}\}, \{p^{in}_2, t_2\}, \{t_2, p^{out}_2\}, \{p^{pre}_{2\leftarrow 1}, t_2\}, \{t_2, p^{target}_2\} \}$,



Σχήμα 5.8: Παράδειγμα δικτύου $N_P^{G_1}$ που κατασκευάστηκε με βάση το M_P της υπηρεσίας *PO-Library*

- $m_0(p_1^{source}) = 1$, $m_0(p) = 0$, $\forall p \in P \setminus \{p_1^{source}\}$.

Βήμα Β: Στοιχεία διεπαφών - $N_P^{G_1} \rightarrow N_P^{G_2}$**.** Αφού αντιστοιχιστεί η ροή της υπηρεσίας από το M_P στο $N_P^{G_1}$, ο μετασχηματισμός προχωρά κατασκευάζοντας το δίκτυο Petri $N_P^{G_2}$ εξετάζοντας τις πληροφορίες που σχετίζονται με τα στοιχεία διεπαφής στο M_P. Το Βήμα Β εστιάζει στον εμπλουτισμό του μοντέλου $N_P^{G_2}$ μέσω περαιτέρω ανάλυσης των θέσεων *in* (p_i^{in}) και *out* (p_i^{out}) (βλ. Μετασχηματισμό 1).

Πιο συγκεκριμένα, σχετικά με την ανάλυση των θέσεων *in*, πρέπει να εξεταστούν όλα τα στοιχεία του M_P που παίζουν κάποιο ρόλο στην παροχή πληροφοριών σχετικά με την πρόσβαση και κλήση της λειτουργικότητας που εκτίθεται μέσω ενός στοιχείου `ProceduralInteraction`.

Μια ΠΣΔ αλληλεπίδραση $pi_i \in PI$ σχετίζεται με το `Endpoint` της υπηρεσίας και με ένα στοιχείο `Operation` op_i το οποίο αναγνωρίζεται από ένα στοιχείο `Name`. Επίσης, το op_i περιέχει ένα στοιχείο `Signature` το οποίο προσδιορίζει ένα σύνολο

στοιχείων `InputParameter`. Τέλος, η ΠΣΔ αλληλεπίδραση pi_i σχετίζεται με ένα στοιχείο `InputAssignment` το οποίο χρησιμοποιείται για να προδιαγράψει την είσοδο της pi_i.

Μετασχηματισμός 5 (Ανάλυση θέσεων *in* σε ΠΣΔ υπηρεσίες)**.** *Μια θέση p_i^{in} του $N_P^{G_1}$ αναλύεται στις ακόλουθες θέσεις:*

- *μια θέση $p_i^{in,endpoint}$ που αντιστοιχεί στο στοιχείο* `Endpoint` *του M_P,*
- *ένα σύνολο θέσεων $p_i^{in,term_m}$ για κάθε στοιχείο* `Term` *$term_m$ που ανήκει στο όνομα του op_i,*
- *ένα σύνολο θέσεων p_i^{in,par_k} για κάθε στοιχείο* `InputParameter` *par_k το οποίο ανήκει στην υπογραφή του op_i αν δεν υπάρχει στοιχείο* `InputAssignment` *που περιλαμβάνεται στο pi_i, ή διαφορετικά, μόνο για κάθε στοιχείο* `InputParameter` *του op_i που περιλαμβάνεται στην ανάθεσή εισόδου.*

Το τόξο a_i^{in} του $N_P^{G_1}$ αφαιρείται και προστίθενται τα τόξα $a_i^{in,endpoint}$, $a_i^{in,term_m}$ και a_i^{in,par_m} για τη σύνδεση των νέων θέσεων στη μετάβαση t_i.

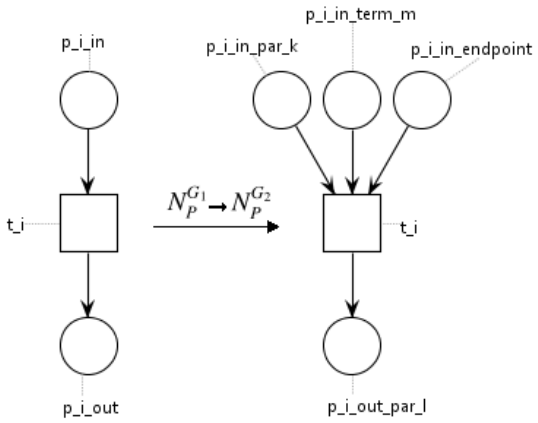
Σχετικά με την ανάλυση των *out* θέσεων, χρησιμοποιείται το σύνολο των στοιχείων `OutputParameter` του op_i, λαμβάνοντας υπόψη την ύπαρξη στοιχείων `OutputAssignment` εφόσον περιλαμβάνονται στην αντίστοιχη ΠΣΔ αλληλεπίδραση pi_i.

Μετασχηματισμός 6 (Ανάλυση θέσεων *ουτ* σε ΠΣΔ υπηρεσίες)**.** *Μια θέση p_i^{out} του $N_P^{G_1}$ αναλύεται σε ένα σύνολο θέσεων p_i^{out,par_l} για κάθε στοιχείο* `OutputParameter` *par_l το οποίο ανήκει στο στοιχείο* `OutputAssignment` *της pi_i, και αν μια τέτοια ανάθεση εξόδου δεν προσδιορίζεται στο M_P, τότε η θέση p_i^{out} αναλύεται σε ένα σύνολο θέσεων p_i^{out,par_l} για κάθε στοιχείο* `OutputParameter` *par_l που ανήκει στην υπογραφή του op_i.*

Ομοίως με το a_i^{in}, το τόξο a_i^{out} του $N_P^{G_1}$ αφαιρείται και προστίθενται ένα σύνολο τόξων a_i^{out,par_m} για τη σύνδεση της μετάβασης στις νέες θέσεις.

Το Σχήμα 5.9 απεικονίζει την ανάλυση των θέσεων p_i^{in} και p_i^{out} του $N_P^{G_1}$, σύμφωνα με τους παραπάνω μετασχηματισμούς

Το Σχήμα 5.10 απεικονίζει το δίκτυο Petri $N_P^{G_2}$ που παράγεται για την υπηρεσία *PO-Library* από το αντίστοιχο δίκτυο $N_P^{G_1}$ που απεικονίζεται στο Σχήμα 5.8 και



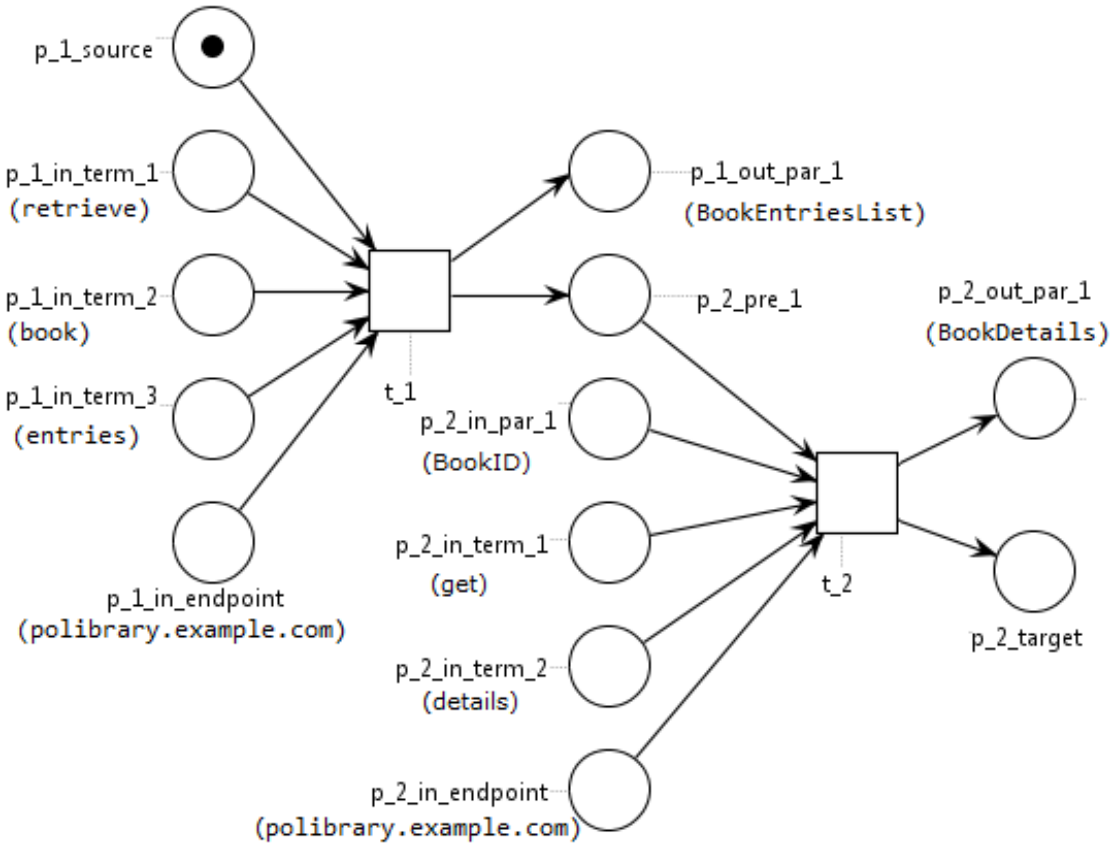
Σχήμα 5.9: Τμήματα δικτύων Petri που αφαιρούνται (αριστερά) και προστίθενται (δεξιά) κατά το Βήμα Β - $N_P^{G_1} \rightarrow N_P^{G_2}$

το M_P (Σχήμα 5.6 και Πίνακας 5.1) Η θέση p_1^{in} του $N_P^{G_1}$ αναλύεται στις $p_1^{in,term_1}$ (`get`), $p_1^{in,term_2}$ (`book`), $p_1^{in,term_2}$ (`entries`) και $p_1^{in,endpoint}$ στο $N_P^{G_2}$, και η θέση p_1^{out} του $N_P^{G_1}$ αντικαθίσταται από το p_1^{out,par_1} (`BookEntriesList`) στο $N_P^{G_2}$. Ομοίως, η λογική μετασχηματισμού του Βήματος Β εφαρμόζεται στις θέσεις *in* και *out* που αντιστοιχούν στην αλληλεπίδραση pi_2.

Το δίκτυο $N_P^{G_2}$ για την υπηρεσία *PO-Library* ορίζεται από τις παρακάτω συνιστώσες:

- $T = \{t_1, t_2\}$,
- $P = \{p_1^{source}, p_{2\leftarrow 1}^{pre}, p_2^{target}, p_1^{in,endpoint}, p_1^{in,term_1}, p_1^{in,term_2}, p_1^{in,term_3}, p_2^{in,endpoint}, p_2^{in,par_1}, p_2^{in,term_1}, p_2^{in,term_2}, p_1^{out,par_1}, p_2^{out,par_1}\}$,
- $F = \{ \{p_1^{source}, t_1\}, \{p_1^{in,endpoint}, t_1\}, \{p_1^{in,term_1}, t_1\}, \{p_1^{in,term_2}, t_1\}, \{p_1^{in,term_3}, t_1\}, \{t_1, p_1^{out,par_1}\}, \{t_1, p_{2\leftarrow 1}^{pre}\}, \{p_2^{in,endpoint}, t_2\}, \{p_2^{in,par_1}, t_2\}, \{p_2^{in,term_1}, t_2\}, \{p_2^{in,term_2}, t_2\}, \{t_2, p_2^{out,par_1}\}, \{p_{2\leftarrow 1}^{pre}, t_2\}, \{t_2, p_2^{target}\} \}$,
- $m_0(p_1^{source}) = 1$, $m_0(p) = 0$, $\forall p \in P \setminus \{p_1^{source}\}$.

Βήμα Γ: Αντιστοιχίες ευθυγραμμίας - $N_P^{G_2} \rightarrow N_P$**.** Στόχος του μετασχηματισμού στο Βήμα Γ είναι η χρησιμοποίηση του μοντέλου των κανόνων ευθυγραμμίας M_A έτσι ώστε το δίκτυο $N_P^{G_2}$ να μπορεί να μετασχηματιστεί σε ένα open net N_P, το οποίο μπορεί τελικά να συγκριθεί με το αντίστοιχο open net N_R που κατασκευάζε-



Σχήμα 5.10: Το δίκτυο $N_P^{G_2}$ που παρήχθη για την υπηρεσία *PO-Library* μετά την εφαρμογή του Βήματος Β της διαδικασίας μετασχηματισμού στο δίκτυο $N_P^{G_1}$ που απεικονίζεται στο Σχήμα 5.8

ται θα την ΠΣΠ υπηρεσία μέσω της αντίστοιχης διαδρομής μετασχηματισμού (βλ. Σχήμα 5.2)

Η κατασκευή του open net:

$$N_P = [P_{N_P}, T_{N_P}, F_{N_P}, I_{N_P}, O_{N_P}, m_{0N_P}, \Omega_{N_P}]$$

γίνεται εφαρμόζοντας τις παρακάτω ενέργειες:

1. *Κατηγοριοποίηση θέσεων* (*place categorization*): κατηγοριοποίηση των θέσεων του $N_P^{G_2}$ σε τρεις κατηγορίες που μοντελοποιούνται ως τρία ξένα σύνολο, το σύνολο εισόδου (I_{N_P}), το σύνολο εξόδου (O_{N_P}) και το σύνολο εσωτερικών θέσεων

($E_{N_P} = P_{N_P} \setminus (I_{N_P} \cup O_{N_P})$),

2. *Αναδιαμόρφωση διεπαφής* (*interface refactoring*): αναδιαμόρφωση των θέσεων διεπαφής ($I_{N_P} \cup O_{N_P}$ βάσει των κανόνων που περιέχονται στο M_A,
3. *Αναγνώριση τελικών καταστάσεων* (*final states identification*): προσδιορισμός του συνόλου Ω_{N_P} των τελικών καταστάσεων του N_P.

Οι υπόλοιπες συνιστώσες του N_P, συμπεριλαμβανομένου του συνόλου μεταβάσεων (T_{N_P}) και του συνόλου τόξων (F_{N_P} σχέσεις ροής) αντλούνται άμεσα από το $N_P^{G_2}$.

Η κατηγοριοποίηση θέσεων γίνεται ως εξής:

- Οι θέσεις εισόδου του N_P περιλαμβάνουν τις θέσεις *in* για τους πόρους, τις παραμέτρους εισόδου και τα endpoint: $I_{N_P} = \{p_i^{in,endpoint} \cup p_i^{in,term_j} \cup \{p_i^{in,par_k}, \forall i \in |PI|\}$,
- Οι θέσεις εξόδου του N_P περιλαμβάνουν τις θέσεις *out* για τις παραμέτρους εξόδου: $O_{N_P} = \{p_i^{out,par_j}, \forall i \in |PI|\}$,
- Οι εσωτερικές θέσεις του N_P περιλαμβάνουν τις θέσεις *source*, *target* και *pre*: $E_{N_P} = \{p_i^{source} \cup p_j^{target} \cup p_{n \leftarrow m}^{pre}, \forall i, j, n, m \in |PI|\}$.

Έπειτα, εφαρμόζεται η αναδιαμόρφωση της διεπαφής χρησιμοποιώντας τους κανόνες που περιέχονται στο M_A. Συγκεκριμένα, για κάθε κανόνα αντιστοιχίας του M_A, το στοιχείο της ΠΣΔ διεπαφής στο οποίο αναφέρεται (π.χ. μια παράμετρος εισόδου) χρησιμοποιείται για ένταξη των αντίστοιχων θέσεων του $I_{N_P} \cup O_{N_P}$ και είτε την ενημέρωση της ετικέτας του με εκείνη του κανόνα υπό την προϋπόθεση ότι δεν έχει ήδη ενημερωθεί ή με την προσθήκη μιας νέας θέσης με ετικέτα αυτή του κανόνα του M_A, στην περίπτωση που υπάρχουν περισσότερες από μια αντιστοιχίσεις που αναφέρονται στο ίδιο στοιχείο. Με βάση τα παραπάνω, η αναδιαμόρφωση της διεπαφής του δικτύου περιλαμβάνει τη μετονομασία των θέσεων και την προσθήκη θέσεων όπου είναι απαραίτητο.

Τέλος, στα open net οι τελικές καταστάσεις αναγνωρίζονται από μια Boolean έκφραση που ονομάζεται *τελική συνθήκη* (*final condition*), της οποίας οι μεταβλητές εκφράζουν συνθήκες σχετικά με την κατανομή των token στο P_{N_P}. Η τελική συνθήκη για το N_P θα πρέπει να εκφράζει την απαίτηση ότι όλες οι θέσεις *target*

φέρουν token ενώ όλες οι άλλες θέσεις, τόσο οι εσωτερικές όσο και οι εξωτερικές είναι κενές. Συνεπώς, οι τελικές καταστάσεις του N_P αναγνωρίζονται από τη σύζευξη των εκφράσεων $m(p_i^{target}) = 1, \forall i \in |PI|$ για τα οποία υπάρχει μια θέση target και $m(p) = 0$ για όλες τις άλλες θέσεις (ή η συνθήκη `ALL_OTHER_PLACES_EMPTY` όπως ορίζεται στη γλώσσα OWFN[2])

Για την επίδειξη της των παραπάνω τεχνικών, εφαρμόζουμε το Βήμα Γ στο παράδειγμα *PO-Library*. Για τον σκοπό αυτό προσδιορίστηκε ένα σύνολο κανόνων αντιστοιχίσεων σε ένα μοντέλο M_A για το *PO-Library* το οποίο περιγράφεται από τον Πίνακα 5.2.

Πίνακας 5.2: Κανόνες αντιστοιχίας για την υπηρεσία *PO-Library*

Κανόνας αντιστοιχίας	Στοιχείο διεπαφής
rl_1 (IntentToActionMapping)	retrieve (Intent)
rl_2 (TermToStaticPartMapping)	book (Concept)
rl_3 (TermToStaticPartMapping)	entries (Concept)
rl_4 (EndpointToControlDataMapping)	polibrary.example.com (Endpoint)
rl_5 (OutputToStateMapping)	BookEntriesList (OutputParameter)
rl_6 (IntentToActionMapping)	get (Intent)
rl_7 (TermToStaticPartMapping)	details (Concept)
rl_8 (InputToDynamicPartMapping)	BookID (InputParameter)
rl_9 (EndpointToControlDataMapping)	polibrary.example.com (Endpoint)
rl_{10} (OutputToStateMapping)	BookDetails (OutputParameter)

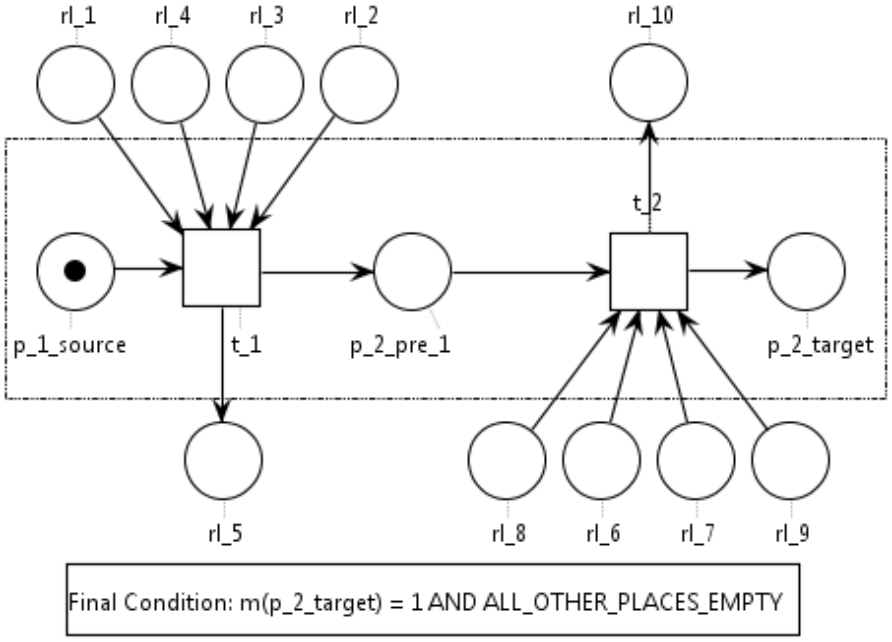
Το Σχήμα 5.11 παρουσιάζει το N_P για την υπηρεσία *PO-Library* που κατασκευάζεται στο Βήμα Γ από το δίκτυο $N_P^{G_2}$ (Σχήμα 5.10) χρησιμοποιώντας τους κανόνες στον Πίνακα 5.2. Με βάση τους κανόνες που ορίστηκαν για την κατηγοριοποίηση θέσεων, οι θέσεις $p_1^{in,endpoint}$, $p_1^{in,term_1}$, $p_1^{in,term_2}$, $p_1^{in,term_3}$, $p_2^{in,endpoint}$, p_2^{in,par_1}, $p_2^{in,term_1}$,

[2]FIONA manual (http://download.gna.org/service-tech/fiona/fiona.pdf), Section 4.1.3.2 Final conditions.

$p_2^{in,term_2}$ κατηγοριοποιούνται ως θέσεις εισόδου, οι θέσεις p_1^{out,par_1}, p_2^{out,par_1} κατηγοριοποιούνται ως θέσεις εξόδου και οι θέσεις p_1^{source}, $p_{2\leftarrow 1}^{pre}$, p_2^{target} κατηγοριοποιούνται ως εσωτερικές θέσεις.

Στη συνέχεια, χρησιμοποιώντας τον Πίνακα 5.2, οι θέσεις διεπαφής αναδιαμορφώνονται ως εξής: $p_1^{in,endpoint} \rightarrow rl_4$, $p_1^{in,term_1} \rightarrow rl_1$, $p_1^{in,term_2} \rightarrow rl_2$, $p_1^{in,term_3} \rightarrow rl_3$, $p_2^{in,endpoint} \rightarrow rl_9$, $p_2^{in,par_1} \rightarrow rl_8$, $p_2^{in,term_1} \rightarrow rl_6$, $p_2^{in,term_2} \rightarrow rl_7$, $p_1^{out,par_1} \rightarrow rl_5$ και $p_2^{out,par_1} \rightarrow rl_{10}$.

Έπειτα, η τελική συνθήκη του N_P για την υπηρεσία *PO-Library* προσδιορίζεται από την έκφραση: $m(p_2^{target}) = 1 \wedge (\nexists m(p) > 0, \forall p \in P_{N_P} \setminus \{p_2^{target}\})$.



Σχήμα 5.11: Το open net N_P που παράγεται για την υπηρεσία *PO-Library*.

Με βάση τα παραπάνω, το N_P που παράγεται για την υπηρεσία *PO-Library* και απεικονίζεται στο Σχήμα 5.11 ορίζεται από τις ακόλουθες συνιστώσες:

- $T = \{t_1, t_2\}$,
- $P = \{p_1^{source}, p_{2\leftarrow 1}^{pre}, p_2^{target}, rl_1, rl_2, rl_3, rl_4, rl_6, rl_7, rl_8, rl_9, rl_5, rl_{10}\}$,
- $F = \{ \{p_1^{source}, t_1\}, \{rl_1, t_1\}, \{rl_2, t_1\}, \{rl_3, t_1\}, \{rl_4, t_1\}, \{t_1, rl_5\}, \{t_1, p_{2\leftarrow 1}^{pre}\}, \{rl_6, t_2\}, \{rl_7, t_2\}, \{rl_8, t_2\}, \{rl_9, t_2\}, \{t_2, rl_{10}\}, \{p_{2\leftarrow 1}^{pre}, t_2\}, \{t_2, p_2^{target}\} \}$,
- $I_{N_P} = \{ rl_1, rl_2, rl_3, rl_4, rl_6, rl_7, rl_8, rl_9\}$,

- $O_{N_P} = \{ rl_5, rl_{10}\}$,
- $m_0(p_1^{source}) = 1$, $m_0(p) = 0$, $\forall p \in P \setminus \{p_1^{source}\}$,
- Τελική συνθήκη: $m(p_2^{target}) = 1 \wedge (\nexists m(p) > 0, \forall p \in P_{N_P} \setminus \{p_2^{target}\}$.

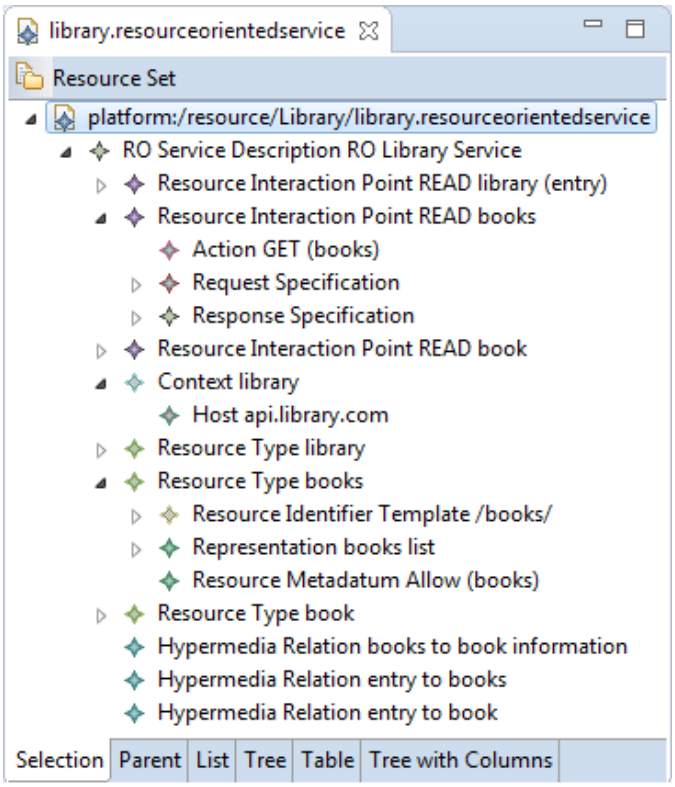
Παραγωγή open net για ΠΣΠ υπηρεσίες

Στην παρούσα ενότητα, συζητάμε τους μετασχηματισμούς που απαιτούνται για την κατασκευή του open net N_R για μια ΠΣΠ υπηρεσία S_R. Όπως και με τις ΠΣΔ υπηρεσίες, ο μετασχηματισμός γίνεται σε τρία βήματα. Στο πρώτο βήμα, η προδιαγραφή της υπηρεσίας όπως παρέχεται από το μοντέλο M_R μετασχηματίζεται σε ένα δίκτυο Petri το οποίο $N_R^{G_1}$ μοντελοποιεί τις λεπτομέρειες ροής της υπηρεσίας. Στο δεύτερο βήμα, το $N_R^{G_1}$ μετασχηματίζεται στο $N_R^{G_2}$ το οποίο μοντελοποιεί τις λεπτομέρειες εισόδου και εξόδου της διεπαφής της υπηρεσίας. Στο τρίτο βήμα, το $N_R^{G_2}$ μετασχηματίζεται στο μοντέλο N_R χρησιμοποιώντας τους κανόνες ευθυγραμμίας που ορίζονται στο μοντέλο M_A.

Αν και η διαδρομή μετασχηματισμού για τις ΠΣΠ υπηρεσίες απαιτεί πιο σύνθετους κανόνες μετασχηματισμού σε σχέση με τη διαδρομή της τις ΠΣΔ υπηρεσίες, οι απαιτήσεις και οι στόχοι κάθε βήματος είναι ίδιοι και στις δύο περιπτώσεις.

Συνοδευτικό παράδειγμα: *Resource-Oriented Library*. Ομοίως με την περίπτωση των ΠΣΔ υπηρεσιών που παρουσιάστηκε παραπάνω, παρέχουμε μια ΠΣΠ υπηρεσία ως συνοδευτικό παράδειγμα στην οποία αναφερόμαστε ως *RO-Library*. Η υπηρεσία *RO-Library* περιέχει δύο τύπους πόρων, τους `books` και `book`. Ένας καταναλωτής της υπηρεσίας μπορεί να αντλήσει τη λίστα όλων των βιβλίων στη βιβλιοθήκη όπως και επιπλέον πληροφορίες για κάθε βιβλίο αποστέλλοντας κατάλληλα READ αιτήματα. Το Σχήμα 5.12 παρέχει μια εικόνα του μοντέλου M_R για την υπηρεσία *RO-Library* όπως εμφανίζεται στο εργαλείο διαχείρισης M_R μοντέλων που βασίζεται στο EMF. Το πλήρες M_R μοντέλο για το *RO-Library* παρέχεται στο Παράρτημα Β′.

Βήμα Α: Ροή υπηρεσίας - $M_R \rightarrow N_R^{G_1}$. Το Βήμα Α στη διαδρομή μετασχηματισμού για ΠΣΠ υπηρεσίες περιλαμβάνει από τις παρακάτω ενέργειες: *α′*) αντιστοίχιση ΠΣΠ σημείων αλληλεπίδρασης σε τμήματα δικτύων Petri χαμηλού επιπέδου και *β′*) αντιστοίχιση στοιχείων σχέσεων υπερμέσων σε τμήματα δικτύων Petri τα οποία διατηρούν τη ροή αλληλεπιδράσεων της υπηρεσίας όπως προσδιορίζεται στο



Σχήμα 5.12: Συνοδευτικό παράδειγμα ΠΣΠ υπηρεσίας: το μοντέλο για την υπηρεσία *RO-Library* όπως εμφανίζεται στο εργαλείο μεταχείρισης M_R μοντέλων

M_R. Στις παραγράφους που ακολουθούν ορίζουμε τις παραπάνω αντιστοιχίες και παρουσιάζουμε τον τρόπο εφαρμογής τους μέσω της υπηρεσίας *RO-Library*.

Μετασχηματισμός 7 (Αντιστοιχία ΠΣΠ σημείων αλληλεπίδρασης)**.** *Έστω ROIP το σύνολο των ΠΣΠ σημείων αλληλεπίδρασης σε ένα μοντέλο υπηρεσίας M_R και HR το σύνολο σχέσεων υπερμέσων επίσης στο μοντέλο M_R. Ένα στοιχείο* `Resource InteractionPoint` *$roip_i \in ROIP$, αντιστοιχίζεται στα παρακάτω τμήματα δικτύων Petri χαμηλού επιπέδου:*

- *μια μετάβαση t_i,*
- *μια θέση p_i^{in} και ένα τόξο $a_i^{in} = \{p_i^{in}, t_i\}$,*
- *μια θέση p_i^{out} και ένα τόξο $a_i^{out} = \{t_i, p_i^{out}\}$,*
- *μια θέση p_i^{pre} και ένα τόξο $a_i^{pre} = \{p_i^{pre}, t_i\}$, αν $\exists roip_j \in ROIP$ ώστε $\exists hr_l \in HR = \{roip_j, roip_i\}$.*

Ομοίως με το Βήμα Α της ΠΣΔ διαδρομής, στο $N_R^{G_1}$ προστίθενται θέσεις source και target. Συγκεκριμένα, προστίθενται θέσεις source σε *ΠΣΠ σημεία αλληλεπίδρασης εισόδου*. Συνεπώς, όταν το χαρακτηριστικό `entry` ενός ΠΣΠ σημείου αλληλεπίδρασης $roip_i$ είναι `true` προστίθεται μια θέση εισόδου και ένα τόξο το οποίο τη συνδέει με τη μετάβαση t_i. Θέσεις target προστίθενται σε *ΠΣΠ σημεία αλληλεπίδρασης εξόδου* τα οποία είναι ένα το υποσύνολο των ΠΣΠ σημείων αλληλεπίδρασης, για τα οποία δεν υπάρχουν σχέσεις υπερμέσων οι οποίες να τα χρησιμοποιούν ως *source* σημεία αλληλεπίδραση.

Μετασχηματισμός 8 (Δημιουργία source θέσεων)**.** *Για κάθε μετάβαση t_i που αντιστοιχεί σε ένα σημείο αλληλεπίδρασης εισόδου $roip_i$, προστίθεται μια θέση p_i^{source} και ένα τόξο $a_i^{source} = \{p_i^{source}, t_i\}$. Η θέση p_i^{source} μαρκάρεται με ένα token στο αρχικό marking του $N_R^{G_1}$, δηλαδή $m_0(p_i^{source}) = 1$.*

Μετασχηματισμός 9 (Δημιουργία target θέσεων)**.** *Για κάθε μετάβαση t_i που αντιστοιχεί σε ένα σημείο αλληλεπίδρασης εξόδου $roip_i$, προστίθεται μια θέση p_i^{target} και ένα τόξο $a_i^{target} = \{t_i, p_i^{target}\}$.*

Οι αντιστοιχίες που προσδιορίστηκαν παραπάνω βασίζονται στην ίδια προσέγγιση που ακολουθήθηκε για την αντιστοίχιση στοιχείων `Procedural Interaction` σε τμήματα δικτύων Petri για τις ΠΣΔ υπηρεσίες. Αυτό οφείλεται στο γεγονός ότι τόσο οι ΠΣΠ αλληλεπιδράσεις όσο και τα ΠΣΠ σημεία αλληλεπίδρασης εξυπηρετούν σε επίπεδο μοντελοποίησης την ανάλυση της λειτουργικότητας σε μονάδες. Ωστόσο, ο μετασχηματισμός λίγο διαφορετικός σχετικά με τη ροή της υπηρεσίας. Πιο συγκεκριμένα, οι σχέσεις υπερμέσων μοντελοποιούν affordances (δηλαδή, τι μπορεί να κάνει ένας πελάτης με έναν πόρο) τα οποία θα πρέπει να παρέχονται κατά τον χρόνο εκτέλεσης από τον εξυπηρετητή στον πελάτη στα πλαίσια των μηνυμάτων απόκρισης. Τέτοια affordances ενδέχεται να χρησιμοποιούνται από τον πελάτη για να εξελίσσει την κατάσταση της εφαρμογής μέσω περαιτέρω αλληλεπιδράσεων με πόρους. Παρ' όλα αυτά, ακολουθώντας ένα συγκεκριμένο στοιχείο υπερμέσων αμέσως μετά την εμφάνισή του ενδεχομένως να μην είναι απαραίτητο για την εκπλήρωση κάποιου στόχου. Επίσης, δεν υπάρχει εγγύηση ότι ένα affordance το οποίο εμφανίζεται κάποια στιγμή, θα παραμείνει έγκυρο όταν ο πελάτης αποφασίσει να το καταναλώσει. Συνεπώς, οι σχέσεις υπερμέσων δε θέτουν ισχυρές χρονικές εξαρτήσεις μεταξύ των ΠΣΠ αλληλεπιδράσεων και φέρουν σχετικά χαλαρή σημασιολογία σχετικά με τη σειρά με την οποία καταναλώνεται η

λειτουργικότητα. Για την μετάφραση της σημασιολογίας που προσδιορίζεται μέσω των σχέσεων υπερμέσων στο M_R σε κατάλληλα στοιχεία δικτύων Petri, οι σχέσεις υπερμέσων αντιστοιχούνται σε ανεξάρτητε μεταβάσεις των οποίων η δυνατότητα να πυροδοτούνται ενδέχεται να ελέγχεται από κοινόχρηστες *guard* θέσεις (*θέσεις φρούρησης*).

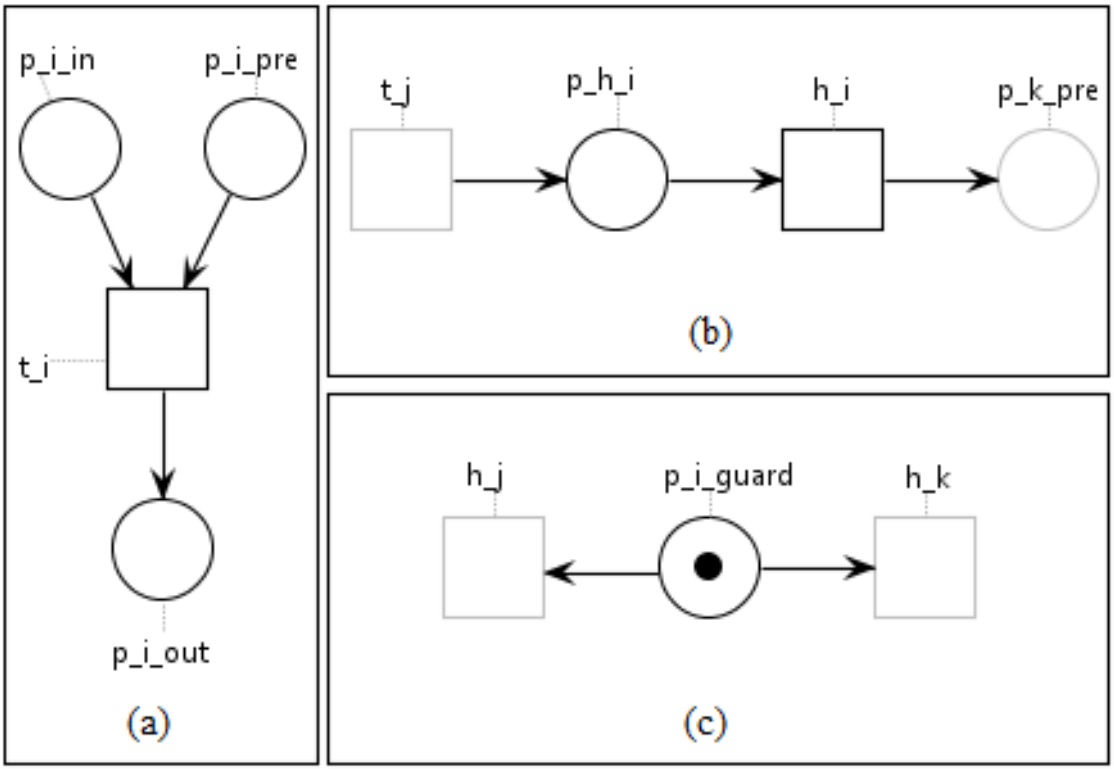
Μετασχηματισμός 10 (Αντιστοιχία σχέσεων υπερμέσων)**.** *Έστω HR το σύνολο των (απαιτούμενων) στοιχείων σχέσεων υπερμέσων σε ένα μοντέλο υπηρεσίας M_R. Ένα στοιχείο* `HypermediaRelation` *$hr_i \in HR$ αντιστοιχίζεται στα ακόλουθα τμήμα δικτύων Petri χαμηλού επιπέδου:*

- *μια μετάβαση h_i,*
- *μια θέση p^{h_i} και ένα τόξο $a^{h_i} = \{p^{h_i}, h_i\}$,*
- *ένα τόξο a_i^{from,t_j} που ξεκινά από τη μετάβαση t_j και τερματίζει στη θέση p_{h_i}, όπου η t_j αντιστοιχεί στο $roip_j = hr_i.source$,*
- *ένα τόξο a_i^{to,t_k} που ξεκινά από τη μετάβαση h_i και τερματίζει στη θέση p_k^{pre}, όπου η θέση p_k^{pre} αντιστοιχεί στο $roip_k = hr_i.target$.*

Για τη διαχείριση του πως οι προσφερόμενες δυνατότητες (affordances) μπορούν να ενεργοποιούνται ή να απενεργοποιούνται καθώς το σύστημα λειτουργεί, οι σχέσεις υπερμέσων που οδηγούν στα ίδια ΠΣΠ σημεία αλληλεπίδρασης ελέγχονται από κοινόχρηστες θέσεις οι οποίες φέρουν token στο αρχικό marking. Για λόγους απλότητας, περιορίζουμε το αρχικό marking των guard θέσεων σε 1, το οποίο επιτρέπει κάθε ΠΣΠ σημείο αλληλεπίδρασης στόχου να προσπελάζεται το πολύ μια φορά.

Μετασχηματισμός 11 (Έλεγχος υπερμέσων)**.** *Για κάθε ΠΣΠ σημείο αλληλεπίδρασης $roip_i$ το οποίο είναι στόχος ενός ή περισσότερων σχέσεων υπερμέσων, δημιουργείται μια θέση p_i^{guard} η οποία μαρκάρεται με ένα token ($m_0(p_i^{guard}) = 1$). Στη συνέχεια, για κάθε σχέση αλληλεπίδρασης $hr_j \in HR$ της οποίας ο στόχος είναι το $roip_i$, δημιουργείται ένα τόξο a_i^{guard,h_j} από τη θέση p_i^{guard} στην αντίστοιχη μετάβαση h_j.*

Το Σχήμα 5.13 (a) παρουσιάζει τα στοιχεία δικτύων Petri που δημιουργούνται κατά τη διάρκεια του Βήματος Α για ένα ΠΣΠ σημείο αλληλεπίδρασης $roip_i \in ROIP$ (Μετασχηματισμός 7), το Σχήμα 5.13 (b) απεικονίζει τα στοιχεία δικτύων Petri που δημιουργούνται για μια σχέση αλληλεπίδρασης $hr_i \in HR$ (Μετασχηματισμός 10)



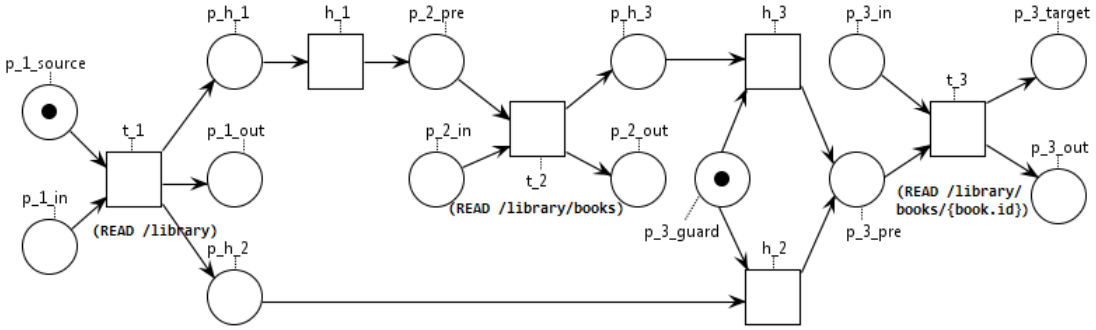
Σχήμα 5.13: Τμήματα δικτύων Petri που δημιουργούνται για: (a) ΠΣΠ σημεία αλληλεπίδρασης, (b) σχέσεις υπερμέσων και (c) έλεγχο υπερμέσων

και το Σχήμα 5.13 (c) απεικονίζει την προσθήκη guard θέσεων (Μετασχηματισμός 11).

Το Σχήμα 5.14 απεικονίζει ένα παράδειγμα δικτύου $N_R^{G_1}$ που παράγεται για την υπηρεσία *RO-Library*.

Βήμα Β: Στοιχεία διεπαφής - $N_R^{G_1} \rightarrow N_R^{G_2}$**.** Κατά το Βήμα Β, οι θέσεις p_i^{in} και p_i^{out} που συνδέονται με μια μετάβαση t_i η οποία αντιστοιχεί σε ένα ΠΣΠ σημείο αλληλεπίδρασης $roip_i \in ROIP$, αναλύονται σε θέσεις που αντιστοιχούν στα στοιχεία του M_R που είτε αναπαριστούν πληροφορίες εισόδου απαραίτητες για την πρόσβαση σε μια λειτουργική μονάδα ή πληροφορίες εξόδου που παρέχονται από τη δυνατότητα της υπηρεσίας ακολουθώντας το ΠΣΠ υπόδειγμα.

Συγκεκριμένα, με βάση το MM_R, ένα ΠΣΠ σημείο αλληλεπίδρασης συσχετίζεται με ένα πρότυπο αναγνωριστικού πόρου, έναν τύπου πόρου όπως και προδιαγραφές



Σχήμα 5.14: Παράδειγμα δικτύου $N_R^{G_1}$ που κατασκευάστηκε με βάση το M_R της υπηρεσίας *RO-Library*

αιτήματος και απόκρισης. Το πρότυπο αναγνωριστικού πόρου αναλύεται σε στατικά και δυναμικά τμήματα και ο τύπος πόρου χαρακτηρίζεται από μεταδεδομένα πόρου και συσχετίζεται με μια ή περισσότερες αναπαραστάσεις. Μια αναπαράσταση περιέχει δεδομένα και μεταδεδομένα. Επίσης, μια προδιαγραφή αιτήματος περιλαμβάνει δεδομένα ελέγχου και το σύνολο των αναπαραστάσεων ενός πόρου οι οποίες μπορούν να χρησιμοποιηθούν για την πρόσβαση στο ΠΣΠ σημείο αλληλεπίδρασης. Ομοίως, μια προδιαγραφή απόκρισης προδιαγράφει τα δεδομένα ελέγχου και τις αναπαραστάσεις που παρέχονται σε μηνύματα εξόδου όπως και το υποσύνολο των μεταδεδομένων πόρου που ενδεχομένως να περιλαμβάνονται.

Μετασχηματισμός 12 (Ανάλυση θέσεων *in* σε ΠΣΠ υπηρεσίες)**.** *Μια θέση p_i^{in} αναλύεται στις εξής θέσεις:*

- *μια θέση p_i^{in,dp_j} για κάθε δυναμικό τμήμα j που ανήκει στο πρότυπο αναγνωριστικού του τύπου πόρου που σχετίζεται με το $roip_i$,*
- *μια θέση p_i^{in,sp_k} για κάθε στατικό τμήμα k που ανήκει στο πρότυπο αναγνωριστικού του τύπου πόρου που σχετίζεται με το $roip_i$,*
- *μια θέση $p_i^{in,action}$ για την πράξη που έχει συσχετιστεί με το $roip_i$,*
- *μια θέση p_i^{in,cd_l} για κάθε δεδομένο ελέγχου που έχει συσχετιστεί με το $roip_i$ μέσω της σχετικής προδιαγραφής αιτήματος,*
- *μια θέση p_i^{in,rd_m} για κάθε στοιχείο δεδομένου m που περιέχεται στην αναπαράσταση αιτήματος που εκτίθεται και ανήκει στον τύπο πόρου που σχετίζεται με το $roip_i$*

- *μια θέση* p_i^{in,rmd_n} *για κάθε στοιχείο μεταδεδομένων* n *που περιέχεται στην αναπαράσταση αιτήματος που εκτίθεται και ανήκει στον τύπο πόρου που σχετίζεται με το* $roip_i$.

Μετασχηματισμός 13 (Ανάλυση θέσεων *out* σε ΠΣΠ υπηρεσίες)**.** *Μια θέση* p_i^{out} *αναλύεται στις εξής θέσεις:*

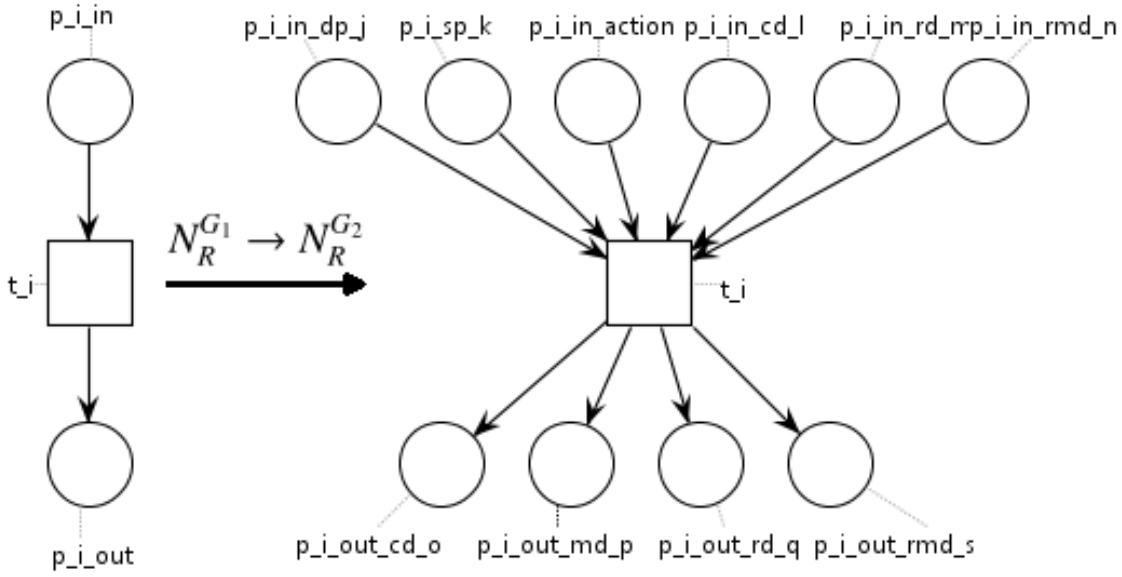
- *μια θέση* p_i^{out,cd_o} *για κάθε δεδομένο ελέγχου που σχετίζεται με το* $roip_i$ *μέσω της προδιαγραφής απόκρισης,*
- *μια θέση* p_i^{out,md_p} *για κάθε μεταδεδομένο πόρου που εκτίθεται και σχετίζεται με το* $roip_i$ *μέσω της προδιαγραφής απόκρισης,*
- *μια θέση* p_i^{out,rd_q} *για κάθε στοιχείο δεδομένων* q *που περιέχεται στην αναπαράσταση απόκρισης που εκτίθεται και ανήκει στον τύπο πόρου που σχετίζεται με το* $roip_i$.
- *μια θέση* p_i^{out,rmd_s} *για κάθε στοιχείο μεταδεδομένων* s *που περιέχεται στην αναπαράσταση απόκρισης που εκτίθεται και ανήκει στον τύπο πόρου που σχετίζεται με το* $roip_i$.

Μια γραφική αναπαράσταση της ανάλυσης των θέσεων *in* και *out* του $N_R^{G_1}$ όπως ορίζεται στους Ορισμούς 12 και 13 παρέχεται στο Σχήμα 5.15.

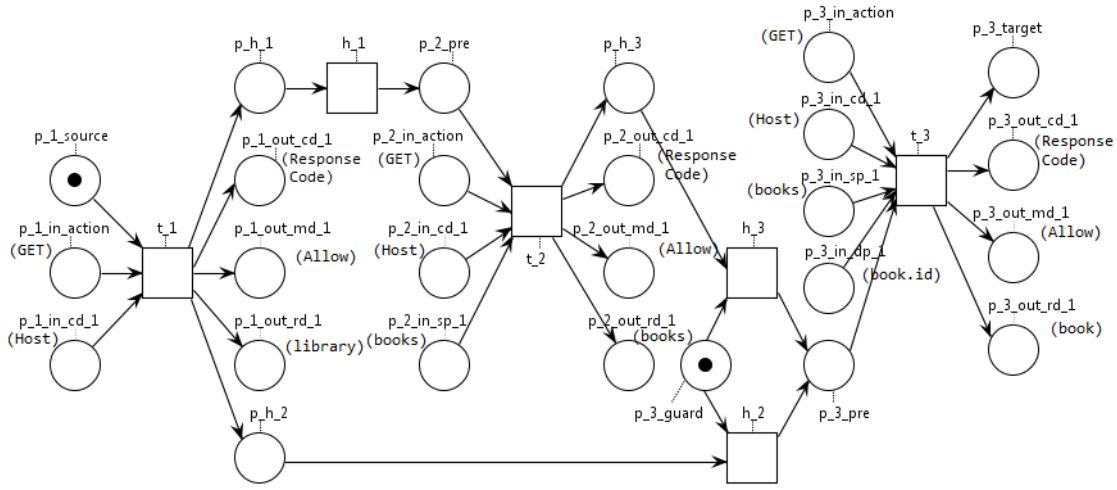
Το Σχήμα 5.16 παρουσιάζει το αποτέλεσμα του μετασχηματισμού $N_R^{G_1} \rightarrow N_R^{G_2}$ που εφαρμόζεται στην υπηρεσία *RO-Library* η οποία περιγράφεται από το μοντέλο M_R που απεικονίζεται στο Σχήμα 5.12.

Βήμα Γ: Αντιστοιχίες ευθυγραμμίας - $N_R^{G_2} \rightarrow N_R$**.** Το Βήμα Γ είναι το τελικό βήμα της διαδικασίας μετασχηματισμού, κατά το οποίο το δίκτυο $N_R^{G_2}$ μετασχηματίζεται σε ένα open net N_R. Στο βήμα αυτό, χρησιμοποιούνται οι πληροφορίες των κανόνων ευθυγραμμίας - αντιστοιχιών που περιλαμβάνονται στο μοντέλο M_A (βλ. Παράρτημα Γ΄ έτσι ώστε το N_R είναι είναι συγκρίσιμο με το N_P ως προς τις διεπαφές τους.

Ομοίως με το Βήμα Γ της ΠΣΔ διαδρομής, εφαρμόζονται οι τεχνικές *κατηγοριοποίησης θέσεων*, *αναδιαμόρφωσης διεπαφής* και *αναγνώρισης τελικών καταστάσεων*.



Σχήμα 5.15: Ανάλυση των θέσεων *in* και *out* του $N_R^{G_1}$ (Μετασχηματισμοί 12 και 13)



Σχήμα 5.16: Το δίκτυο $N_R^{G_2}$ που κατασκευάζεται μετά την εφαρμογή του Βήματος Β στο δίκτυο $N_R^{G_1}$ της υπηρεσίας *RO-Library* (Σχήμα 5.14)

Η κατηγοριοποίηση θέσεων υλοποιείται ως μια ανάλυση του συνόλου $P_{N_R^{G_2}}$ σε τρία ξένα σύνολα θέσεων και συγκεκριμένα το σύνολο θέσεων εισόδου (I_R), το σύνολο θέσεων εξόδου (O_R) και το σύνολο εσωτερικών θέσεων (E_R). Η ανάλυση ξεκινά επιλέγοντας σε ένα σύνολο $P_{N_R}^{selected}$ τις θέσεις που σχετίζονται με στοιχεία του M_R τα οποία δεν έχουν ανατεθειμένες τιμές σε αυτά στο μοντέλο M_R (π.χ. ένα στοιχείο `ControlDatum` του οποίου το χαρακτηριστικό `value` είναι κενό) και αντιστοιχούν σε ΠΣΠ σημεία αλληλεπίδρασης των οποίων το χαρακτηριστικό `entry` είναι `false`. Στη συνέχεια, χρησιμοποιώντας τις θέσεις εισόδου και εξόδου που περιλαμβάνονται στο $P_{N_R}^{selected}$, τα σύνολα I_R και O_R συντίθενται ως εξής:

- $I_R = \{p_i^{in,dp_j}, \ldots, p_i^{in,sp_k}, \ldots, p_i^{in,action}, \ldots, p_i^{in,cd_l}, \ldots, p_i^{in,rd_m}, \ldots, p_i^{in,rmd_n}\}$,
- $O_R = \{p_i^{out,cd_o}, \ldots, p_i^{out,md_p}, \ldots, p_i^{out,rd_q}, \ldots, p_i^{out,rmd_s}, \ldots\}$,

Στη συνέχεια, προσδιορίζεται το σύνολο $E_R = P_{N_R^{G_2}} \setminus (I_R \cup O_R)$. Το E_R περιέχει όλες τις θέσεις του $P_{N_R}^{selected}$ που δεν ανατέθηκα στο I_R ή στο O_R, όπως και τις υπόλοιπες θέσεις του δικτύου που δεν επιλέχθηκαν. Θα πρέπει να σημειωθεί ότι με βάση την παραπάνω κατηγοριοποίηση, είναι δυνατό θέσεις *in* που αντιστοιχούν σε ΠΣΠ σημεία αλληλεπίδρασης τα οποία δεν είναι σημεία εισόδου να ενταχθούν στο σύνολο E_R. Τέτοιες θέσεις μαρκάρονται με ένα token στο αρχικό marking του N_R έτσι ώστε το αρχικό marking να μην αποτελεί αδιέξοδο.

Έπειτα, εφαρμόζεται η αναδιάρθρωση διεπαφής με τον ίδιο τρόπο με τον οποίο γίνεται στο Βήμα Γ της ΠΣΔ διαδρομής μετασχηματισμού. Πιο συγκεκριμένα, για κάθε κανόνα αντιστοιχίας στο M_A, το ΠΣΠ στοιχείο διεπαφής στο οποίο αναφέρεται ο κανόνας χρησιμοποιείται για να αναγνωριστούν οι αντίστοιχες θέσεις στο $I_{N_R} \cup O_{N_R}$ και να ενημερωθούν οι ετικέτες τους.

Τέλος, το σύνολο των τελικών καταστάσεων Ω_{N_R} του open net N_R αναγνωρίζονται μέσω της κατασκευής της κατάλληλης τελικής συνθήκης. Συγκεκριμένα, η τελική συνθήκη για το N_R είναι η σύζευξη των παρακάτω συνθηκών:

- $m(p_i) = 1$, $\forall p_i \in E_R$ ώστε $p_i\bullet = \emptyset$: όλα οι εσωτερικές θέσεις προορισμού θα πρέπει να είναι μαρκαρισμένες με ένα token. Οι θέσεις αυτές περιλαμβάνουν τις *target* θέσεις και τις *out* που αντιστοιχούν σε σημεία αλληλεπίδρασης που δεν είναι σημεία εισόδου ή παρέχονται με προκαθορισμένες τιμές στο επίπεδο του M_R.

- εκφράσεις σύζευξης συνθηκών υπερμέσων που υπολογίζονται ως εξής: έστω SH όλα τα σύνολα των θέσεων υπερμέσων p^{h_i} των οποίων οι αντίστοιχες σχέσεις υπερμέσων έχουν κοινούς προορισμούς. Τότε, $\forall s \in SH$, δημιουργείται μια Boolean έκφραση σύζευξης η οποία επιτρέπει το πολύ μια θέση στο s να μην είναι μαρκαρισμένη. Για παράδειγμα, έστω $s = \{p^{h_1}, p^{h_2}, p^{h_3}\}$, τότε η έκφραση που παράγεται για το s είναι $(m(p^{h_1}) = 0 \wedge m(p^{h_2}) = 1 \wedge m(p^{h_3}) = 1) \vee (m(p^{h_1}) = 0 \wedge m(p^{h_2}) = 1 \wedge m(p^{h_3}) = 1) \wedge (m(p^{h_1}) = 1 \wedge m(p^{h_2}) = 1 \wedge m(p^{h_3}) = 0)$.
- όλες οι άλλες θέσεις θα πρέπει να είναι κενές για να διασφαλίζεται ότι το open net μοντελοποιεί ορθά και χρησιμοποιείται σύμφωνα με τα προβλεπόμενα σενάρια για την υπηρεσία.

Στις παραγράφους που ακολουθούν εφαρμόζονται οι τεχνικές του Βήματος Γ στην υπηρεσία *RO-Library*.

Αρχικά, καθώς το χαρακτηριστικό `entry` του $roip_1$ στο M_R έχει τιμή `true`, οι θέσεις των $roip_2$ και $roip_3$ επιλέγονται για ανάθεση στα σύνολα θέσεων εισόδου και εξόδου του N_R. Επίσης, δεδομένου ότι στο επίπεδο του M_R παρέχονται τιμές για τις θέσεις p_1^{out,cd_1}, p_1^{out,md_1}, p_1^{out,rd_1}, p_2^{out,md_1}, p_2^{out,cd_1}, p_2^{out,md_1}, p_2^{out,cd_1}, p_3^{out,cd_1} και p_3^{out,md_1}, δεν εντάσσονται στο σύνολο $P_{N_R}^{selected}$.

Στη συνέχεια χρησιμοποιώντας το $P_{N_R}^{selected}$ και με βάση του κανόνες κατηγοριοποίησης θέσεων που παρουσιάστηκαν παραπάνω οι θέσεις $p_2^{in,action}$, p_2^{in,sp_1}, p_2^{in,sp_1}, p_2^{in,cd_1}, $p_3^{in,action}$, p_3^{in,sp_1}, p_3^{in,dp_1}, p_3^{in,cd_1}, κατηγοριοποιούνται ως θέσεις εισόδου, οι θέσεις $p_2^{out_rd_1}$, $p_3^{out_rd_1}$ κατηγοριοποιούνται ως θέσεις εξόδου και οι υπόλοιπες θέσεις p_1^{source}, $p_1^{in,action}$, p_1^{in,cd_1}, p_1^{out,cd_1}, p_1^{out,md_1}, p_1^{out,rd_1}, p_2^{out,md_1}, p_2^{out,cd_1}, p_2^{out,md_1}, p_2^{out,cd_1}, p_3^{out,cd_1}, p_3^{out,md_1}, p^{h_1}, p^{h_2}, p^{h_3}, p_2^{pre}, p_3^{pre}, p_3^{guard}, p_3^{target} κατηγοριοποιούνται ως εσωτερικές θέσεις.

Πίνακας 5.3: Κανόνες αντιστοιχίας για την υπηρεσία *RO-Library*

Κανόνας αντιστοιχίας	Στοιχείο διεπαφής
rl_1 (IntentToActionMapping)	READ (Action)
rl_2 (TermToStaticPartMapping)	books (StaticPart)
rl_3 (TermToStaticPartMapping)	books (StaticPart)
rl_4 (EndpointToControlDataMapping)	rolibrary.example.com (Host)
rl_5 (OutputToStateMapping)	books (StateElement)
rl_6 (IntentToActionMapping)	READ (Action)
rl_7 (TermToStaticPartMapping)	books (StaticPart)
rl_8 (InputToDynamicPartMapping)	{book.id} (DynamicPart)
rl_9 (EndpointToControlDataMapping)	rolibrary.example.com (Host)
rl_{10} (OutputToStateMapping)	book (StateElement)

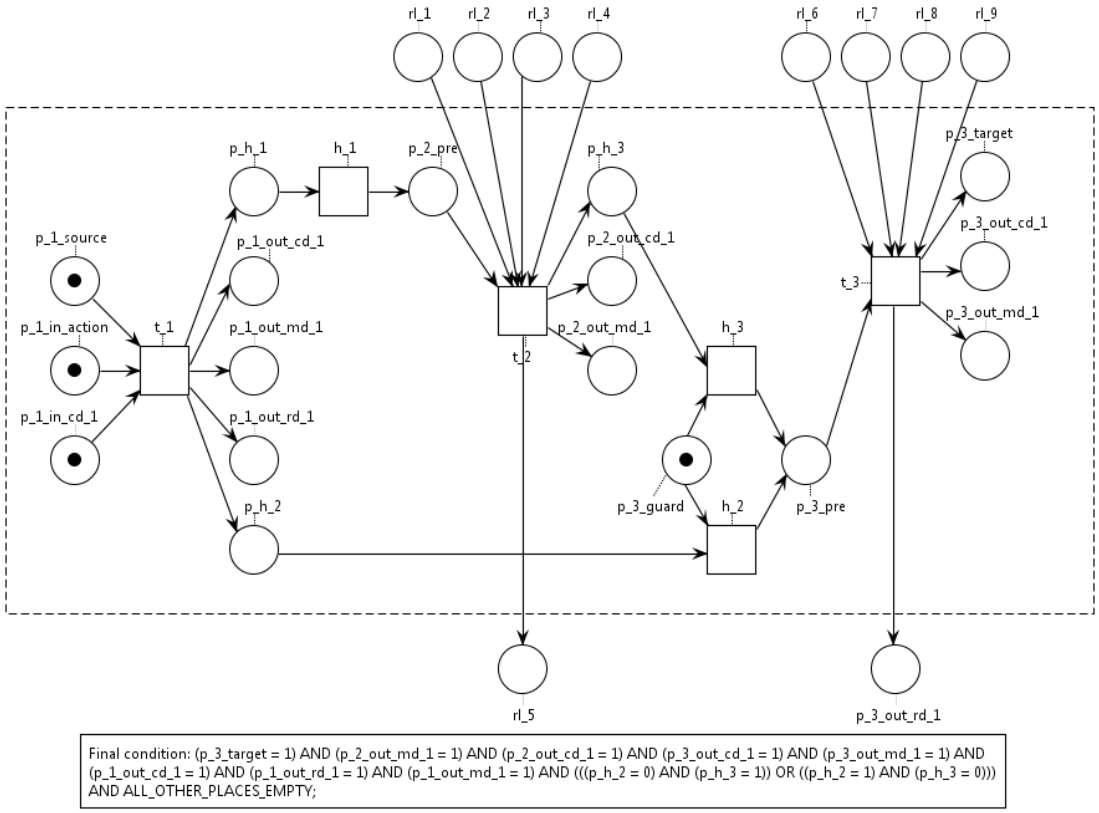
Σχετικά με την αναδιαμόρφωση διεπαφής, χρησιμοποιούνται οι συμπαγείς κανόνες αντιστοιχίας που ορίζονται στο M_A και περιλαμβάνονται στον Πίνακα 5.3. Συγκεκριμένα οι θέσεις διεπαφής αναδιαμορφώνονται ως εξής: $p_1^{in,endpoint} \rightarrow rl_4$, $p_1^{in,term_1} \rightarrow rl_1$, $p_1^{in,term_2} \rightarrow rl_2$, $p_1^{in,term_3} \rightarrow rl_3$, $p_2^{in,endpoint} \rightarrow rl_9$, $p_2^{in,par_1} \rightarrow rl_8$, $p_2^{in,term_1} \rightarrow rl_6$, $p_2^{in,term_2} \rightarrow rl_7$, $p_1^{out,par_1} \rightarrow rl_5$ και $p_2^{out,par_1} \rightarrow rl_{10}$.

Τέλος, η τελική συνθήκη N_R κατασκευάζεται χρησιμοποιώντας τους τρεις τύπους συνθηκών που συζητήθηκαν παραπάνω. Συγκεκριμένα, η τελική συνθήκη του παραγόμενου open net για την υπηρεσία *RO-Library* προσδιορίζεται από τη σύζευξη των παρακάτω συνθηκών:

- εσωτερικές θέσεις προορισμού: $m(p_3^{target}) = 1 \wedge m(p_2^{out,md_1}) = 1 \wedge m(p_2^{out,cd_1}) = 1 \wedge m(p_3^{out,cd_1}) = 1 \wedge m(p_3^{out,md_1}) = 1 \wedge m(p_1^{out,cd_1}) = 1 \wedge m(p_1^{out,rd_1}) = 1 \wedge m(p_1^{out,md_1}) = 1$
- εκφράσεις σύζευξης υπερμέσων: $((m(p^{h_2}) = 0 \wedge m(p^{h_3}) = 1) \vee (m(p^{h_2}) = 1 \wedge m(p^{h_3}) = 0))$,

- όλες οι άλλες θέσεις κενές, δηλαδή `ALL_OTHER_PLACES_EMPTY`.

Το Σχήμα 5.17 παρουσιάζει το open net N_R που δημιουργείται μετά την εφαρμογή της λογικής μετασχηματισμού του Βήματος Γ στο δίκτυο $N_R^{G_2}$ που απεικονίζεται στο Σχήμα 5.16, στο πλαίσιο των κανόνων ευθυγραμμίας που περιέχονται στον Πίνακα 5.3.



Σχήμα 5.17: Το open net N_R που παράγεται για την υπηρεσία *RO-Library*

Όπως συζητήθηκε παραπάνω, το αποτέλεσμα του Βήματος Γ είναι ένα open net το οποίο αποτελεί το τελικό προϊόν της διαδικασίας μετασχηματισμού για κάθε διαδρομή. Στην επόμενη ενότητα, συζητάμε πως τα open net αυτά χρησιμοποιούνται για την αντιμετώπιση της πρόκλησης ΔΥΥ μέσω της αποτίμησης της ύπαρξης accordance.

5.2.4 Έλεγχος accordance

Αφού προσδιοριστούν τα δίκτυα N_P και N_R, εφαρμόζεται η παρακάτω διαδικασία για τον έλεγχο ύπαρξης accordance και συνεπώς την αποτίμηση της υποκαταστασιμότητας και της ευθυγραμμίας.

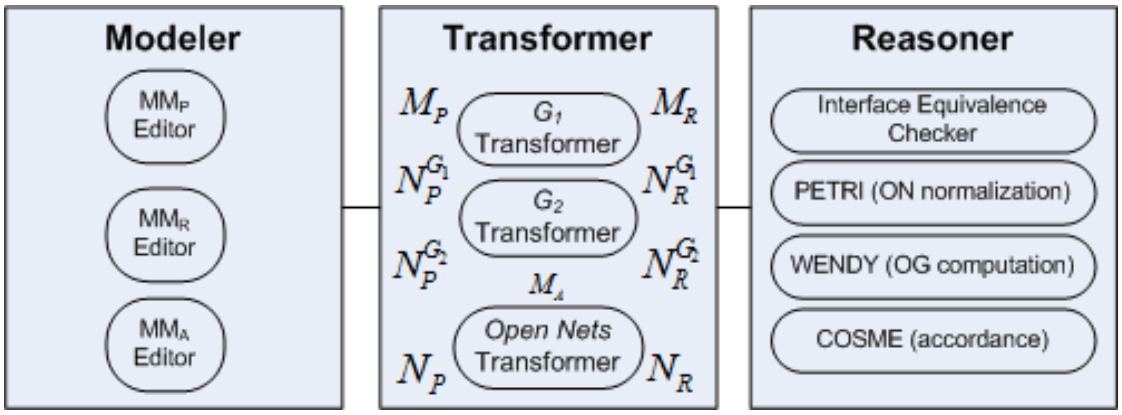
1. Το σύνολο I_{N_P} συγκρίνεται με το σύνολο I_{N_R} και το O_{N_P} συγκρίνεται με το O_{N_R} για τον έλεγχο της ισοδυναμίας διεπαφών των open net (βλ. Ορισμό 12), δηλαδή εξετάζεται αν $I_{N_P} \equiv I_{N_R} \wedge O_{N_P} \equiv O_{N_R}$.
2. Εφόσον τα N_P και N_R είναι ισοδύναμα ως προς τις διεπαφές τους, αντλούνται τα αντίστοιχα κανονικοποιημένα open net N_P^n και N_R^n (δηλαδή τα open net στα οποία οι μεταβάσεις συσχετίζονται με το πολύ μια θέση διεπαφής).
3. Υπολογίζονται τα operating guideline μοντέλα $OG(N_P^n)$ και $OG(N_R^n)$.
4. Ελέγχεται η ύπαρξη accordance εξετάζοντας την ύπαρξη της σχέσεις εξειδίκευσης μεταξύ των operating guideline μοντέλων, δηλαδή εξετάζεται αν $OG(N_P^n) \sqsubseteq OG(N_R^n)$, όπως συζητείται στην Ενότητα 5.1.3.

5.3 Υλοποίηση πρωτοτύπου

Το περιβάλλον-πλαίσιο ΔΥΥ που παρουσιάστηκε παραπάνω υλοποιήθηκε στα πλαίσια ενός πρωτότυπου με βάση το Eclipse Modeling Framework για τις προδιαγραφές των μεταμοντέλων και τη διαχείριση μοντέλων, μια Java υλοποίηση της διαδικασίας μετασχηματισμού των μοντέλων (Σχήμα 5.2) και τα εργαλεία PETRI, WENDY και COSME του SERVICE-TECHNOLOGY.ORG για την αποτίμηση ύπαρξης accordance.

Πιο συγκεκριμένα, τα MM_P, MM_R και MM_A ορίστηκαν ως Ecore μοντέλα καθώς τα εργαλεία παραγωγής κώδικα και εργαλείων τροποποίησης των μοντέλων αυτών που παρέχονται από το EMF διευκολύνει τον ορισμό μοντέλων στιγμιοτύπων (instances), M_P, M_R και M_A. Εναλλακτικά, τα ΠΣΔ μοντέλα θα μπορούσαν να οριστούν μετασχηματίζοντας WSDL και WSCL μοντέλα και τα ΠΣΠ μοντέλα θα μπορούσα να οριστούν χρησιμοποιώντας WADL περιγραφές και προδιαγραφές σχέσεων υπερμέσων. Το M_A μπορεί να προσδιοριστεί είτε από τον χρήστη, είτε με ημι-αυτοματοποιημένο τρόπο μέσω των τεχνικών εξαγωγής αντιστοιχίσεων.

Χρησιμοποιώντας τα M_P, M_R και M_A, η διαδικασία μετασχηματισμού παρέχει δύο open net N_P και N_R και ελέγχει την ισοδυναμία διεπαφών μεταξύ τους. Έπειτα, χρησιμοποιείται το εργαλείο Petri για να κατασκευάσει το N_P^n και το N_R^n. Στη συνέχεια, χρησιμοποιείται το εργαλείο Wendy για να υπολογίσει τα operating guideline μοντέλα για τα κανονικοποιημένα open net, $OG(N_P^n)$ και $OG(N_R^n)$. Τέλος, χρησιμοποιείται το εργαλείο Cosme για να εξετάσει την ύπαρξη accordance μέσω της σχέσης εξειδίκευσης των operating guideline μοντέλων.



Σχήμα 5.18: Σχηματική απεικόνιση του πρωτοτύπου αποτίμησης υποκαταστασιμότητας και ευθυγραμμίας υπηρεσιών διαφορετικών υποδειγμάτων (ΔΥΥ και ΔΥΕ)

Στο Σχήμα 5.18 παρέχεται μια υψηλού αφαιρετικού επιπέδου οπτική του πρωτοτύπου ΔΥΥ/ΔΥΕ. Τα κύρια δομοστοιχεία είναι τα ακόλουθα:

- Modeler: περιλαμβάνει ένα σύνολο εργαλείων διαχείρισης μοντέλων βασισμένων στο EMF για τη δημιουργία και τροποποίηση μοντέλων που συμμορφώνονται στα μεταμοντέλα MM_P, MM_R και MM_A.
- Extractor: υλοποιεί τεχνικές εξαγωγής που επιτρέπουν την ημιαυτόματη σύνθεση μοντέλων που συμμορφώνονται στα μεταμοντέλα MM_P, MM_R και MM_A.
- Transformer: υλοποιεί τα τρία βήματα της διαδικασίας μετασχηματισμού μοντέλων και για τις δυο διαδρομές του μετασχηματισμού (ΠΣΔ και ΠΣΠ) έτσι ώστε να αντληθούν συγκρίσιμα open net για κάθε υπηρεσία.
- Reasoner: υλοποιεί τη διαδικασία αποτίμησης ύπαρξης accordance εξετάζοντας την ισοδυναμία διεπαφών open net και χρησιμοποιώντας τα εργαλεία Petri, Wendy και Cosme.

5.4 Συζήτηση

Στο περιβάλλον-πλαίσιο που προτάθηκε, δόθηκε ιδιαίτερη προσοχή στα μεταμοντέλα MM_P και MM_R ούτως ώστε να οριστούν με γενικό τρόπο και να μην είναι περιορισμένα μόνο στα Web services και στο REST. Επιπλέον, η χρήση εργαλείων ανοιχτού κώδικα και μεθοδολογιών για την αποτίμηση της σχέσης accordance χρησιμοποιώντας open net κάνει την προσέγγιση άμεσα διαθέσιμη προς χρήση.

Παρ' όλα αυτά υπάρχει ένα σύνολο θεμάτων τα οποία είναι σκόπιμο να συζητηθούν σχετικά με τους περιορισμούς της προσέγγισης. Εστιάζουμε παρακάτω σε τρεις βασικές περιοχές.

Η πρώτη περιοχή σχετίζεται με τη διαθεσιμότητα πληροφοριών στις υπάρχουσες γλώσσες προδιαγραφής API (π.χ. WSDL, WADL) και το αν οι πληροφορίες αυτές μπορούν να χρησιμοποιηθούν για να δημιουργηθούν αυτόματα μοντέλα που συμμορφώνονται στα μεταμοντέλα MM_P και MM_R, δηλαδή M_P και M_R μοντέλα. Για παράδειγμα η κλάση `Name` του MM_P και η αναφορά της στην κλάση `TermModel` πάει ένα βήμα πέρα μιας τυπικής προδιαγραφής μιας ΠΣΔ υπηρεσίας (π.χ. μια περιγραφή βασισμένη σε WSDL). Ωστόσο, τμήματα της διαδικασίας αναγνώρισης της πληροφορίας από WSDL περιγραφές μπορούν να αυτοματοποιηθούν αναλύοντας τις περιγραφές αυτές όπως συζητείται στο Κεφάλαιο 4. Ένα δεύτερο παράδειγμα της περιορισμένης πληροφορίας που είναι διαθέσιμη στα ΠΣΠ API (π.χ. WADL[3] or WSDL 2.0[4], Swagger[5], RAML[6], API Blueprint[7]) είναι ότι δεν επιτρέπουν εγγενώς την έκφραση υπερμέσων, έτσι ώστε να είναι εφικτή η αυτόματη δημιουργία στοιχείων `HypermediaRelation` του μεταμοντέλου MM_R. Παρ' όλα αυτά, υπάρχουν διάφοροι τρόποι να προσεγγιστεί ο περιορισμός αυτός, εξετάζοντας πληροφορίες υπερμέσων που ανταλλάσσονται κατά τον χρόνο εκτέλεσης μεταξύ εξυπηρετητή και πελάτη σε ένα RESTful σύστημα. Για παράδειγμα, πληροφορία υπερμέσων μπορεί να παρασχεθεί μέσω κεφαλίδων αλληλεπίδρασης (π.χ. HTTP `Allow`, `Link`), ή μέσω στοιχείων των τύπων μέσων που περιλαμβάνονται σε αντίστοιχες αναπαραστάσεις και χρησιμοποιούν σημασιολογικά στοιχία ελέγχου όπως τα Link Relations[8],

[3] WADL, `https://wadl.java.net/`
[4] WSDL 2.0, `http://www.w3.org/TR/wsdl20/`
[5] Swagger, `http://swagger.io/`
[6] RAML, `http://raml.org/`
[7] API Blueprint, `https://apiblueprint.org/`
[8] Link Relations, `http://www.iana.org/assignments/link-relations/link-relations.xhtml`

τα οποία χρησιμοποιούνται κάθετα σε σχέση με τους τύπους μέσων. Επίσης, συνήθως ορίζονται με μη τυπικό τρόπο πληροφορίες υπερμέσων σε περιγραφές φυσικής γλώσσας των RESTful υπηρεσιών. Επιπλέον, στο πλαίσιο των RESTful υπηρεσιών και των Web API, η κοινότητα εργάζεται στην κατεύθυνση τυπικού ορισμού των στοιχείων υπερμέσων και των μορφών με τις οποίες εμφανίζονται κατά τον χρόνο εκτέλεσης, όπου μάλιστα ιδιαίτερη προσπάθεια καταβάλλεται στον ορισμό γενικευμένων τύπων μέσων με δυνατότητες υπερμέσων, όπως και στην επέκταση των γλωσσών προδιαγραφής διεπαφών ώστε να διευκολύνουν τον ορισμό υπερμέσων.

Με βάση τα παραπάνω, η ανάλυση των API υπηρεσιών και η δημιουργία των προαναφερθέντων μοντέλων, όπως ορίζονται στο προτεινόμενο περιβάλλον-πλαίσιο, συνεχίζει να αποτελεί κατά πολύ μια διαδικασία που ενεργεί ο μηχανικός λογισμικού, πριν ξεκινήσει ο έλεγχος υποκαταστασιμότητας.

Η δεύτερη περιοχή σχετίζεται με τις σχεδιαστικές υποθέσεις που έγιναν για την κατασκευή open net. Πιο συγκεκριμένα, στην τρέχουσα προσέγγιση γίνεται στατική αντιμετώπιση των ζητημάτων που σχετίζονται με την υπό συνθήκη ενεργοποίηση μεταβάσεων και σχέσεων υπερμέσων, όπως αυτές μοντελοποιούνται από τις κλάσεις `Condition` του MM_P και `HypermediaCondition` του MM_R. Αυτό σημαίνει ότι οι συνθήκες θα πρέπει να αποτιμηθούν πριν προχωρήσει η μετάφραση των μοντέλων των υπηρεσιών σε open net. Μια τέτοια υπόθεση ενδεχομένως να μην απαιτείται αν εισαχθούν κατάλληλες δομές ώστε να μοντελοποιούνται υπό συνθήκη μονοπάτια στα παραγόμενα μοντέλα. Επιπλέον, στην ΠΣΠ διαδρομή της διαδικασία μετασχηματισμού, η εισαγωγή των *guard* θέσεων, οι οποίες μαρκάρονται αρχικά με ένα token δηλώνει την υπόθεση μιας κατά το μέγιστο αλληλεπίδρασης οδηγούμενης από υπερμέσα ανά ΠΣΠ σημείο αλληλεπίδρασης. Η υπόθεση αυτή εξασφαλίζει την ελεγξιμότητα των open net που δημιουργούνται με απλό και άμεσο τρόπο. Ωστόσο, αν η παραπάνω υπόθεση δεν είναι συμβατή με την πρόκληση ελέγχου υποκαταστασιμότητας που πρέπει να εξεταστεί, το Βήμα Γ της διαδικασίας μετασχηματισμού, μπορεί να τροποποιηθεί έτσι ώστε να αναγνωρίζει με κατάλληλο τρόπο τις τελικές καταστάσεις του open net που κατασκευάζεται.

Τέλος, η τρίτη περιοχή σχετίζεται με τις δυνατότητες των εργαλείων που χρησιμοποιούνται για την αποτίμηση ύπαρξης accordance. Πιο συγκεκριμένα, αν και η έννοια του accordance και οι σχετικοί αλγόριθμοι και τα εργαλεία παρέχουν ένα καλά εδραιωμένο και ισχυρό πλαίσιο, υπάρχουν μειονεκτήματα που σχετίζονται με τις δυνατότητες κλιμάκωσης και την υπόθεση ασύγχρονης επικοινωνίας μεταξύ των

δομοστοιχείων. Για την αντιμετώπιση αυτών των περιορισμών, θα μπορούσαν να εξεταστούν εξειδικεύσεις της έννοιας της υποκαταστασιμότητας όπως παρουσιάζονται στο [141].

Παρά τους περιορισμούς που συζητούνται παραπάνω, πιστεύουμε ότι το προτεινόμενο περιβάλλον-πλαίσιο φωτίζει το ζήτημα της εννοιολογικής ευθυγραμμίας των ΠΣΔ και των ΠΣΠ υπηρεσιών και ανοίγει νέες κατευθύνσεις στη συντήρηση και στην παράλληλη εξέλιξη δυικών API υπηρεσιών, μέσω της υποκαταστασιμότητας και την ευθυγραμμίας.

Πίνακας 5.1: Στοιχεία διεπαφής υπηρεσίας *PO-Library*

Όνομα κλάσης: αναγνωριστικό στοιχείου	Περιγραφή
`POServiceDescription`: *sd*	PO Library Service
`Operation`: o_1	retrieveBookEntries
`Signature`: s_1	retrieveBookEntries signature
`OutputParameter`: op_1	BookEntriesList
`Name`: n_1	retrieveBookEntries name
`TermModel`: tm_1	retrieveBookEntries term model
`Intent`: $intent_1$	retrieve (retrieveBookEntries)
`Concept`: $concept_1$	book
`Concept`: $concept_2$	entries
`Operation`: o_2	getDetails
`Signature`: s_2	getDetails's signature
`InputParameter`: ip_1	BookID
`OutputParameter`: op_2	BookDetails
`Name`: n_2	getDetails's name
`TermModel`: tm_2	getDetails term model
`Intent`: $intent_2$	get (getDetails)
`Concept`: $concept_3$	details
`Endpoint`: *ep*	polibrary.examples.com
`ProceduralInteraction`: pi_1	retrieveBookEntries invocation
`ProceduralInteraction`: pi_2	getDetails invocation
`ProceduralTransition`: pt_1	source: pi_1, target: pi_2

ΚΕΦΑΛΑΙΟ 6

ΜΕΛΕΤΕΣ ΠΕΡΙΠΤΩΣΕΩΝ ΚΑΙ ΠΕΙΡΑΜΑΤΑ

6.1 Εξαγωγή ΠΣΠ διεπαφής

6.1.1 Case Study: Amazon S3

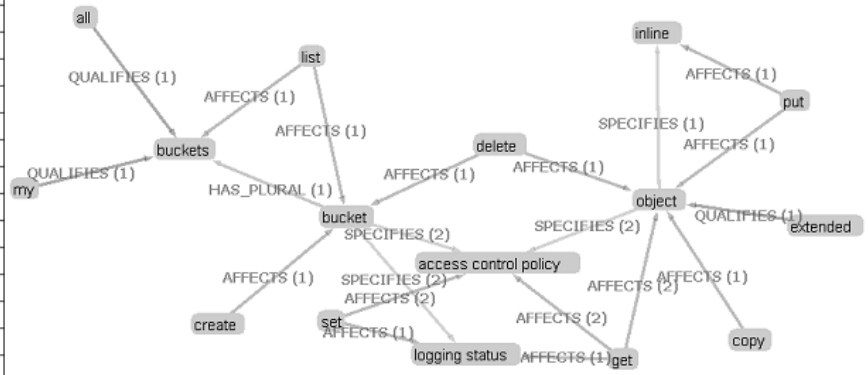
Στην παρούσα ενότητα, παρουσιάζουμε την εφαρμογή της διαδικασία εξαγωγής πόρων στο Amazon Simple Storage Service (S3) ως μια μελέτη περίπτωσης για τη μέθοδο εξόρυξης πόρων που προτάθηκε στο Κεφάλαιο 4. Το Amazon S3 είναι μια πραγματική υπηρεσία στο διαδίκτυο για την αποθήκευση δεδομένων και παρέχει δύο διεπαφές, μια ΠΣΔ και μια ΠΣΠ. Το γεγονός αυτό μας επιτρέπει να αποτιμήσουμε καλύτερα το εξαχθέν μοντέλο πόρων συγκρίνοντάς το με το υπάρχον. Επιπλέον, το Amazon S3 είναι επιτυχημένη υπηρεσία, η οποία χρησιμοποιείται ευρέως και η οποία παρέχει πλούσια τεκμηρίωση οπότε και οι σημαντικές έννοιες, οντότητες και τύποι πόρων μπορούν να αντληθούν αναλύοντας την τεκμηρίωση. Επίσης, το S3 παρέχει μια ΠΣΔ διεπαφή για την οποία εμφανίζονται αρκετές από της προκλήσεις κατά τη δημιουργία του RTM (π.χ. σύμπτυξη κόμβων, όροι για που πρέπει να αντιστοιχιστούν σε πολλές οντότητες). Για την ανάλυσή μας χρησιμοποιήσαμε την τελευταία έκδοση του WSDL εγγράφου (2006-03-01), η οποία συντίθεται από 16 λειτουργίες όπως καταγράφονται στο Σχήμα 6.1 (a).

Μοντέλα υπογραφών και όρων

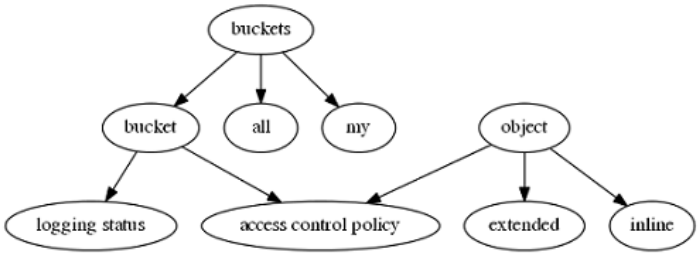
Μετά τη δημιουργία του μοντέλου υπογραφών για το S3, εφαρμόζεται ο αλγόριθμος διαχωρισμού συμβολοσειρών (tokenization) για κάθε λειτουργία έτσι ώστε να διαχωριστεί το όνομα της λειτουργίας σε λεκτικές μονάδες (token). Στη συνέχεια τα token επισημειώνονται με POS ετικέτες χρησιμοποιώντας μια meta POS τεχνική, στην οποία συνδυάζονται τα σύνολα ετικετών που αντλούνται από τρία εργαλεία POS επισημείωσης, συγκεκριμένα τα OpenNLP, Lingpipe και Stanford POS. Στη συνέχεια, για κάθε λειτουργία δημιουργείται ένα OTM το οποίο περιλαμβάνει όρους και σχέσεις όρων. Τα αποτελέσματα του tokenization και της POS επισημείωσης για το S3 παρουσιάζονται στο Σχήμα 6.1 (a). Παράλληλα με τα παραπάνω το εργαλείο κανονικοποιεί τις προθέσεις των λειτουργιών και τα αποτελέσματα αυτής της

Operation	Tokens	POS tags	Category
CreateBucket	create bucket	VB NN	Constructor
DeleteBucket	delete bucket	VB NN	Destructor
GetObjectAccessControlPolicy	get object access control policy	VB NN NN NN NN	Accessor
GetBucketAccessControlPolicy	get bucket access control policy	VB NN NN NN NN	Accessor
SetObjectAccessControlPolicy	set object access control policy	VB NN NN NN NN	Mutator
SetBucketAccessControlPolicy	set bucket access control policy	VB NN NN NN NN	Mutator
GetObject	get object	VB NN	Accessor
GetObjectExtended	get object extended	VB NN JJ	Accessor
PutObject	put object	VBN NN	Mutator
PutObjectInline	put object inline	VB NN NN	Mutator
DeleteObject	delete object	VB NN	Destructor
ListBucket	list bucket	VB NN	Accessor
ListAllMyBuckets	list all my buckets	VB DT PRP$ NNS	Accessor
GetBucketLoggingStatus	get bucket logging status	VB NN NN NN	Accessor
SetBucketLoggingStatus	set bucket logging status	VB NN NN NN	Mutator
CopyObject	copy object	VB NN	Constructor

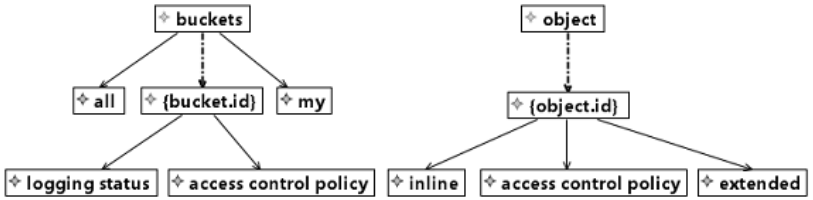
(a)



(b)



(c)



(d)

Σχήμα 6.1: Εξαγωγή πόρων για το Amazon S3: (a) λειτουργίες υπηρεσίας με αντίστοιχα token, POS επισημειώσεων και κανονικοποίηση πρόθεσης, (b) STM μοντέλο, (c) CCE μοντέλο, (d) αυτόματα παραχθέν RTM

διαδικασίας παρουσιάζονται επίσης στο 6.1 (a) (τελευταία στήλη). Έπειτα, τα 16 OTM υποβάλλονται σε επεξεργασία και συναθροίζονται σε ένα συνολικό STM το οποίο απεικονίζεται στο Σχήμα 6.1 (b).

Μοντέλο CCE και Resource Types Model

Το επόμενο βήμα περιλαμβάνει τη μετακίνηση από όρους σε σημαντικές για την υπηρεσία εννοιολογικές οντότητες. Αυτό πραγματοποιείται μέσω εξαγωγής ενός μοντέλου CCE. Το μοντέλο που παράγεται για το S3 απεικονίζεται στο Σχήμα 6.1 (c). Το μοντέλο CCE χρησιμοποιείται στη συνέχεια μαζί με ευρετικές παραγωγής τύπων πόρων για την κατασκευή του RTM για την υπηρεσία όπως απεικονίζεται στο Σχήμα 6.1 (d). Τέλος, το Σχήμα 6.2 παρουσιάζει τον σκελετό μιας περιγραφής WADL η οποία εξάγεται αυτόματα από το RTM που κατασκευάστηκε.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:Application xmlns="http://wadl.dev.java.net/2009/02" xmlns:ns2="http://wadl.dev.java.net/2009/02/"
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <grammars>
        <include href="RTMM.xsd"/>
    </grammars>
    <resources>
        <resource type="Container" path="/buckets/">
            <method name="POST">
                <request>
                    <doc title="Normalization for &quot;CreateBucket&quot; back-end operation"/>
                </request>
            </method>
            <method name="GET">...</method>
        </resource>
        <resource type="ContainerElement" path="/buckets/{bucket.id}/">
            <param name="bucket.id" style="template" type="xs:string"/>
            <method name="DELETE">
                <request>
                    <doc title="Normalization for &quot;DeleteBucket&quot; back-end operation"/>
                </request>
            </method>
            ...
        </resource>
        <resource type="AtomicElement" path="/buckets/{bucket.id}/logging-status/">...</resource>
        <resource type="AtomicElement" path="/buckets/{bucket.id}/access-control-policy/">...</resource>
        <resource type="Container" path="/buckets/{bucket.id}/object/">...</resource>
        <resource type="ContainerElement" path="/buckets/{bucket.id}/object/{object.id}/">...</resource>
        ...
    </resources>
</ns2:Application>
```

Σχήμα 6.2: Σκελετός περιγραφής WADL που παρήχθη από το RTM της υπηρεσίας Amazon S3

Συμμετοχή χρήστη και αποτίμηση

Το παραχθέν RTM παρέχει ένα σκελετό για το μοντέλο πόρων του REST και περιλαμβάνεται στην προδιαγραφή προσαρμογής η οποία παράγεται από τη φάση χρόνου σχεδίασης της P2R διαδικασίας προσαρμογής, μαζί με τις αντιστοιχίες των στοιχείων διεπαφής. Ο χρήστης που καθοδηγεί τη διαδικασία προσαρμογής μπορεί είτε να αποδεχθεί το RTM ως έχει, είτε να το επεξεργαστεί. Για παράδειγμα, στο RTM που κατασκευάστηκε αυτόματα για το Amazon S3 υπάρχει μια σχέση κτήσης η οποία μπορεί να προστεθεί μεταξύ του */buckets/{bucket.id}* ContainerElement και του */object/* Container καθώς, σύμφωνα με την τεκμηρίωση της υπηρεσίας, κάθε bucket ενδέχεται να περιλαμβάνει ένα ή περισσότερα object. Οι αυτόματες τεχνικές δεν ήταν σε θέση να εξάγουν τη σχέση αυτή η οποία μπορεί να προστεθεί απευθείας από τον χρήστη. Επίσης, δύο από τους τύπους πόρων που εξήχθησαν (*/buckets/all/* και */buckets/my/*) μπορούν να παραληφθούν καθώς δεν μπορούν να χρησιμοποιηθούν για να αντιστοιχιστεί κάποια ΠΣΔ αλληλεπίδραση, ούτε τους ανήκει κάποιος χαμηλότερης ιεραρχίας τύπος πόρου ο οποίος θα μπορούσε να χρησιμοποιηθεί για τον σκοπό αυτό, επομένως μπορούν να αφαιρεθούν. Συνοπτικά, για την υπηρεσία αυτή οι 9 από τους 11 τύπους πόρων ήταν ορθοί (0.818 πιστότητα (precision)) και όλοι οι 9 αναμενόμενοι τύποι πόρων αναγνωρίστηκαν (1.0 ανάκληση ((recall)). Τέλος, η τιμή της RTM_{sim} μετρικής, η οποία είναι μια μετρική ομοιότητας βασισμένη σε οντολογίες και η οποία θα παρουσιαστεί στην επόμενη ενότητα υπολογίστηκε και 0.636. Το Amazon S3 παρέχει μια καλά ορισμένη διεπαφή προσανατολισμένη σε περιεχόμενο η οποία αποτελεί καλό υποψήφιο για την επίδειξη της διαδικασίας. Παρ' όλα αυτά, μια αποτίμηση της προτεινόμενης διαδικασία με δυνατότητες γενίκευσης και η οποία αφορά ένα ευρύ σύνολο υπηρεσιών συζητείται στην επόμενη ενότητα.

6.1.2 Πειραματική αποτίμηση

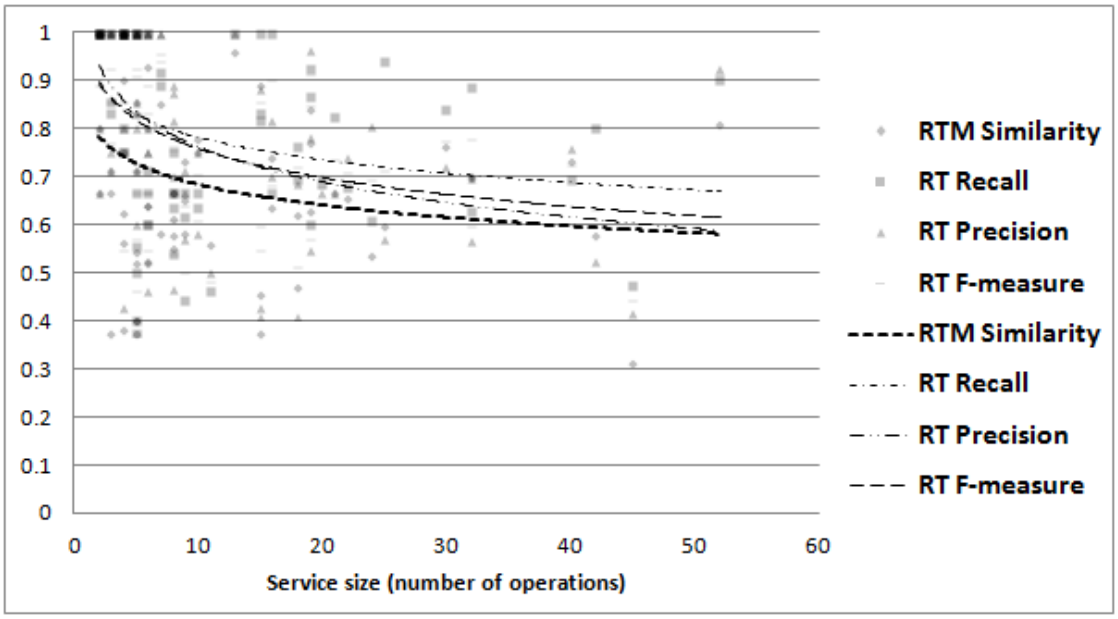
Στην ενότητα αυτή παρουσιάζουμε τα αποτελέσματα πειραμάτων που εκτελέσαμε έτσι ώστε να αποτιμηθεί η ακρίβεια των αποτελεσμάτων και η επίδοση της διαδικασία εξαγωγής πόρων χρησιμοποιώντας υπηρεσίες από το ανοιχτό ευρετήριο υπηρεσιών ProgrammableWeb[1]. Τα πειράματα αποτίμησης εστιάζουν σε τέσσερις περιοχές

[1]ProgrammableWeb.com, http://www.programmableweb.com/

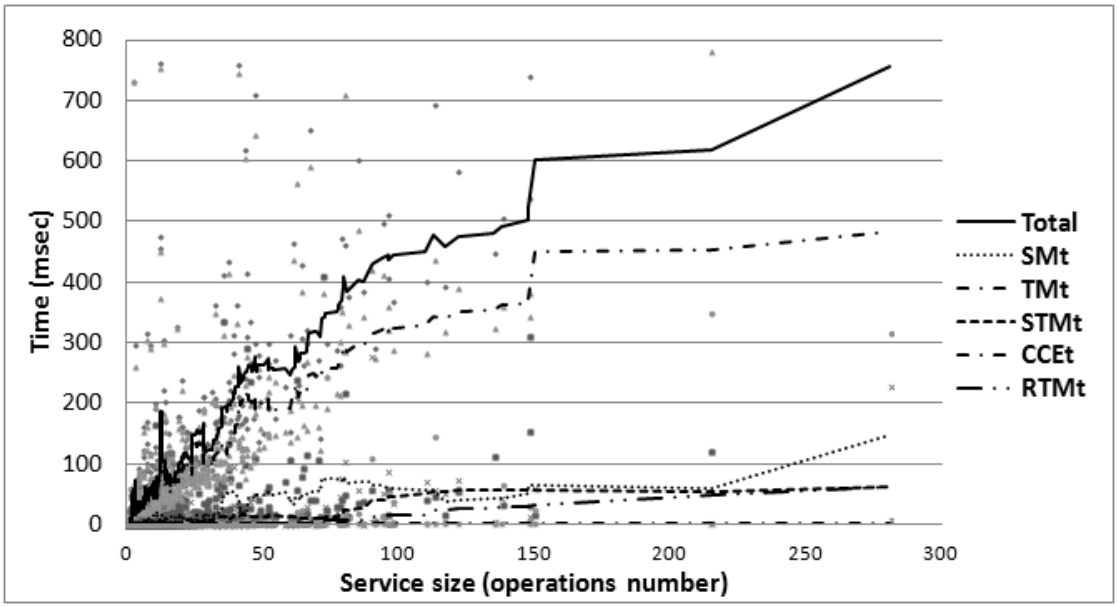
και τα αντίστοιχα αποτελέσματα συζητούνται στις τέσσερις επόμενες υποενότητες. Πιο συγκεκριμένα, η πρώτη περιοχή αποτίμησης σχετίζεται με την ακρίβεια των ενδιάμεσων βημάτων της διαδικασίας εξαγωγής πόρων. Τα αποτελέσματα απεικονίζονται στον Πίνακα 6.1. Η δεύτερη περιοχή αφορά στην ποιότητα των εξαχθέντων μοντέλων πόρων σε σύγκριση με μοντέλα που δημιουργήθηκαν από ειδικούς. Για τον σκοπό αυτό, αντλήσαμε ένα δείγμα 70 υπηρεσιών το οποίο χρησιμοποιήθηκε για τη σύνθεση 70 μοντέλων RTM από ειδικούς. Τα μοντέλα αυτά θεωρήθηκαν στη συνέχεια ως golden standard για τους σκοπούς της ανάλυσης. Η αποτίμηση αντιμετώπισε τη διαδικασία ως ένα πρόβλημα ανάκτησης πληροφορίας και υπολόγισε μετρικές όπως *ανάκληση*, *πιστότητα*, *F-measure* και οντολογική ομοιότητα μεταξύ των μοντέλων πόρων που δημιουργήθηκαν από τη διαδικασία και των μοντέλων που δημιούργησαν οι ειδικοί. Τα αποτελέσματα απεικονίζονται στον Πίνακα 6.2 και στο Σχήμα 6.3. Η τρίτη περιοχή σχετίζεται με την προαγωγή της παραγωγικότητας μέσω της χρήσης της διαδικασίας εξαγωγής έναντι της δημιουργίας συγκρίσιμων μοντέλων πόρων από ανθρώπους. Τα αποτελέσματα εμφανίζονται στον Πίνακα 6.3. Η τέταρτη περιοχή αφορά στην επίδοση ως προς τον χρόνο των διάφορων βημάτων της διαδικασίας, σε σχέση με το μέγεθος των υπηρεσιών. Τα αποτελέσματα απεικονίζονται στο Σχήμα 6.4. Στις επόμενες ενότητες συζητούνται τα αποτελέσματα αυτά με μεγαλύτερη λεπτομέρεια.

Πίνακας 6.1: Αποτίμηση των ενδιάμεσων βημάτων εξόρυξης

Βήμα	Σύνολο Δεδομένων	Μέθοδος Δειγματοληψίας	Μέγεθος Δείγματος	Ακρίβεια
Διαχωρισμός συμβολοσειρών	12918 λειτ.	Simple Random	388	96.6%
Δημιουργία OTM	12918 λειτ. από 867 υπ.	PPS & Systematic	370	80.54%, 88.37% (ανατροφ.)
Κανονικοποίηση Πρόθεσης	12918 οπ.	Σιμπλε Ρανδομ	388	88.02%



Σχήμα 6.3: Αποτίμηση ακρίβειας ως προς το μεγέθος των υπηρεσιών



Σχήμα 6.4: Αποτίμηση επίδοσης: χρόνος εξαγωγής πόρων

Πίνακας 6.2: Πειραματικά αποτελέσματα: αποτίμηση ακρίβειας

Δείγμα 70 υπηρεσιών	Μ.Ο.	Ελαχιστο	Μέγιστο	Τυπ. Απόκλιση
RT_{recall}	0.796	0.375	1	0.186
$RT_{precision}$	0.785	0.409	1	0.191
$RT_{F-Measure}$	0.777	0.442	1	0.167
RTM_{sim}	0.697	0.313	1	0.187

Πίνακας 6.3: Αποτίμηση επίπτωσης στην παραγωγικότητα

Υπηρεσία	R_e	R_t (sec)	C_e	C_t (sec)	I_e	I_t
#1	9	142.1	36	568.5	75.0%	75.0%
#2	13	340.5	30	736.3	56.7%	53.8%
#3	2	8.5	12	50.8	83.3%	83.3%
#4	6	56.5	15	98.2	60.0%	42.4%
#5	12	114.8	20	191.0	40.0%	39.9%
AVG					*63.0%*	*58.9%*

Σχεδίαση πειραμάτων και σύνολα δεδομένων

Οι περισσότερες από τις προσεγγίσεις που συζητήθηκαν στις σχετικές εργασίας (Κεφάλαιο 2, Ενότητα 2.4.3), απαιτούν είτε πληροφορίες επιπλέον του machine readable ορισμού μιας διεπαφής ή απαιτούν σημαντική συμμετοχή του χρήστη στη διαδικασία εντοπισμού πόρων. Επιπλέον, δεν υπάρχουν σύνολα δεδομένων τα οποία θα μπορούσαν να χρησιμοποιηθούν ως σύνολα δεδομένων αναφοράς για τη διαδικασία εξαγωγής πόρων και τα οποία θα επέτρεπαν τη συγκριτική αποτίμηση της προτεινόμενης προσέγγισης. Με βάση τα παραπάνω, επιλέξαμε να αντλήσουμε ένα σύνολο δεδομένων από το ευρετήριο υπηρεσιών ProgrammableWeb και συγκεκριμένα SOAP υπηρεσιών που παρείχαν έγκυρες WSDL περιγραφές. Η συλλογή των δεδομένων έγινε χρησιμοποιώντας ένα εργαλείο που δημιουργήθηκε για τον

σκοπό αυτό. Το εργαλείο χρησιμοποίησε το API του ευρετηρίου για να συλλέξει εγγραφές που αφορούσαν σε SOAP Web services, διατηρώντας αυτές που περιελάμβαναν WSDL URI στις περιγραφές τους. Έπειτα, τα URI που είχαν συλλεχθεί χρησιμοποιήθηκαν για να αντληθούν τα WSDL έγγραφα, καθένα από τα οποία εξετάστηκε ως προς την εγκυρότητά του πριν την προσθήκη του στο σύνολο δεδομένων. Με τον τρόπο αυτό συλλέχθηκαν 867 έγκυρα WSDL έγγραφα τα οποία περιείχαν 12.918 λειτουργίες και τα οποία χρησιμοποιήθηκαν για τη διεξαγωγή των πειραμάτων. Η λίστα αυτών των 867 WSDL URI όπως και περαιτέρω σύνολα δεδομένων που χρησιμοποιήθηκαν σε φάσεις αποτίμησης είναι διαθέσιμα online[2].

Ακρίβεια της διαδικασίας διαχωρισμού συμβολοσειρών

Το πρώτο σύνολο των μελετών αποτίμησης σχετίζεται με την ακρίβεια των ανεξάρτητων βημάτων της διαδικασίας. Τα αποτελέσματα για την αποτίμηση αυτή εμφανίζονται στον Πίνακα 6.1. Πιο συγκεκριμένα, χρησιμοποιώντας κατάλληλες μεθόδους δειγματοληψίας για κάθε βήμα (Simple Random, PPS, Systematic), επιλέγονται σύνολα λειτουργιών από το πλήθος των 12.918 λειτουργιών. Τα αποτελέσματα κάθε βήματος της αυτοματοποιημένης διαδικασίας εξετάστηκαν από ειδικούς ως προς το πόσο κοντά ήταν στα στοιχεία τα οποία θα μπορούσαν να είχαν δημιουργηθεί από τους ίδιους. Τα αποτελέσματα υποδεικνύουν ότι τα ενδιάμεσα μοντέλα παρέχουν υψηλή ακρίβεια με τιμές που κινούνται στο διάστημα 88% - 96%.

Ακρίβεια της διαδικασίας εξαγωγής πόρων

Η αποτίμηση της ακρίβειας των μοντέλων πόρων που εξορύχτηκαν πραγματοποιήθηκε χρησιμοποιώντας ένα δείγμα 70 τυχαίως επιλεχθέντων WSDL εγγράφων, μέσα από το αρχικό σύνολο των WSDL που αντλήθηκαν, αφού εξαιρέθηκαν οι υπηρεσίες με μια λειτουργία. Το μέσο μέγεθος υπηρεσίας για το σύνολο των επιλεγμένων υπηρεσιών ήταν 11.73 λειτουργίες ανά υπηρεσίας (ελάχ.: 2, μεγ.: 52, ΤΑ: 11.2) και το σύνολο αυτό περιελάμβανε υπηρεσίες από διάφορα πεδία όπως ηλεκτρονικό εμπόριο, φιλοξενία στο cloud, επεξεργασίας εικόνας, τηλεπικοινωνίες, επενδύσεις κ.ά. Κάθε υπηρεσία που επιλέχθηκε εξετάστηκε από δύο μηχανικούς λογισμικού ανεξάρτητα, οι οποίοι δημιούργησαν αντίστοιχα RTM. Τα δύο μοντέλα που δη-

[2] http://www.softlab.ntua.gr/~athanm/resourceExtraction/

μιουργήθηκαν για κάθε υπηρεσία εξετάστηκαν συγκριτικά και ενοποιήθηκαν σε ένα τελικό μοντέλο. Οι ειδικοί που συμμετείχαν στη διαδικασία αυτή είχαν σημαντική εμπειρία με το αρχιτεκτονικό στυλ REST και με RESTful HTTP υπηρεσίες. Εξαιτίας περιορισμένων ανθρώπινων πόρων ειδικών που είχαμε στη διάθεσή μας, περιορίσαμε το πλήθος των διεπαφών που αναλύθηκαν από ειδικούς σε συνολικά 70 διεπαφές. Παρ' όλα αυτά, ένα μεγαλύτερο σύνολο διεπαφών μπορεί να βελτιώσει περαιτέρω τη στατιστική σημαντικότητα των αποτελεσμάτων που αντλούνται.

Καθώς η προτεινόμενη προσέγγιση αποτελεί διαδικασία εξόρυξης πληροφορίας, αποτιμήθηκαν μετρικές οι οποίες σχετίζονται με την πιστότητα, την ανάκληση και το F-measure για τους τύπους πόρων που εξορύσσονται, αναφέροντας τις τιμές του μέσου όρου, τις ελάχιστες, τις μέγιστες όπως και των τυπικών αποκλίσεων. Επιπρόσθετα, θεωρώντας το RTM ως μια οντολογία πόρων, χρησιμοποιήθηκε η μετρική της απόστασης ελάχιστου βάρους μέγιστου ταιριάσματος γραφήματος minimum weight maximum graph matching distance (MWMGM) [38] η οποία ορίζεται για αποστάσεις οντολογιών, για να μετρήσει την ορθότητα της εξόρυξης συγκρίνοντας τα μοντέλα που εξορύχτηκαν και εκείνα που δημιουργήθηκαν από τους ειδικούς. Η απόσταση MWMGM ορίζεται ως μια μετρική απόστασης με βάση μια συνάντηση ανομοιότητας δ μεταξύ οντολογικών οντοτήτων, όπου στην περίπτωση των RTM είναι οι τύποι πόρων. Συγκεκριμένα, ορίζουμε την ομοιότητα RTM, $RTM_{sim} : E \times S \rightarrow [0, 1]$ ως μια μετρική αποτίμησης των τεχνικών εξόρυξης πόρων, όπου E είναι το σύνολο των RTM που έχουν εξορυχθεί και S το σύνολο των RTM που έχουν δημιουργηθεί από ειδικούς, ως εξής:

$$RTM_{sim} = 1 - \sum_{i}^{n} \frac{\Delta_{mwmgm}(e_i, s_i)}{n}$$

όπου n είναι το μέγεθος του δείγματος $e_i \in E$, $s_i \in S$ και Δ_{mwmgm} είναι η απόσταση MWMGM ενός ζεύγους RTM. Η συνάντηση ανομοιότητας $\delta : e_i \times s_i \rightarrow [0, 1]$ που χρησιμοποιείται για την ομοιότητα RTM_{sim} βασίζεται στην απόσταση Levenshtein μεταξύ του προτύπων αναγνωριστικών των τύπων πόρων, λαμβάνοντας υπόψη τα τμήματα των αναγνωριστικών. Επίσης, το ελάχιστο βάρος του μέγιστου ταιριάσματος M μπορεί να υπολογιστεί χρησιμοποιώντας επέκταση του αλγόριθμου Hungarian [74], ώστε να καλύπτει πίνακες κόστους $n \times m$. Η μετρική RTM_{sim} όχι μόνο αποτιμά το επίπεδο ταιριάσματος μεταξύ ανεξάρτητων τύπων πόρων αλλά

λαμβάνει επίσης υπόψη την οργάνωση των RTM, παρέχοντας με τον τρόπο αυτό μια αξιόπιστη αποτίμηση της ακρίβειας.

Ανάκληση, Πιστότητα, F-measure, Ομοιότητα. Η ενότητα συζητά αποτελέσματα που σχετίζονται με την ακρίβεια των παραγόμενων μοντέλων τύπων πόρων. Πιο συγκεκριμένα, αποτιμήσαμε την πιστότητα τύπων πόρων ($RT_{precision}$), την ανάκτηση τύπων πόρων (RT_{recall}), το F-measure τύπων πόρων ($RT_{F-measure}$) και την ομοιότητα μοντέλων τύπων πόρων (RTM_{sim}) μεταξύ των μοντέλων που αντλήθηκαν αυτόματα και των μοντέλων που δημιουργήθηκαν από ειδικούς για το δείγμα των 70 υπηρεσιών, υπολογίζονταν τις μέσες τιμές, τις μέγιστες τιμές, τις ελάχιστες τιμές και τις τιμές τυπικής απόκλισης. Ο Πίνακας 6.2 απεικονίζει τα αποτελέσματα που λήφθηκαν από τα πειράματα συγκρίνοντας τα δύο σύνολα μοντέλων. Συνοπτικά, το μέσο RT_{recall} ήταν ίσο με 0.796, το μέσο $RT_{precision}$ με 0.785, το μέσο $RT_{F-measure}$ με 0.777 και το μέσο RTM_{sim} σε 0.697. Δεδομένης της ποικιλομορφίας των υπηρεσιών που επιλέχθηκαν τυχαία και της πολυπλοκότητας της εργασίας, η ακρίβεια της προτεινόμενης προσέγγισης θεωρείται πολύ θετική. Μια άλλη παρατήρηση της ανάλυσής μας είναι ότι καθώς το μέγεθος της υπηρεσίας αυξάνει, οι τιμές όλων των μετρικών που εκφράζουν την ακρίβεια της εξόρυξης μειώνονται (Σχήμα 6.3). Παρ' όλα αυτά, το RT_{recall} μειώνεται με χαμηλότερο ρυθμό σε σχέση με τις υπόλοιπες μετρικές. Αυτό σχετίζεται με το γεγονός ότι καθώς αναλύονται περισσότερες λειτουργίες ανά υπηρεσία, προστίθενται περισσότεροι όροι και σχέσεις στον STM γράφο. Αντίθετα, το RT_{recall} εξαρτάται κυρίως από το βήμα επιλογής των CCE του οποίου η επίδοση είναι αρκετά σταθερή. Όμως, καθώς προστίθενται σχέσεις στο STM, δημιουργούνται επιπλέον εξαρτήσεις μεταξύ των CCE κάτι που ενδέχεται να οδήγησε στη δημιουργία άστοχων ιεραρχιών τύπων πόρων. Οι επιπλέον αυτές ιεραρχίες, παρ' όλα αυτά, είναι συνήθως εύκολο να αναγνωριστούν και ο χρήστης μπορεί να αναθεωρήσει το RTM αφαιρώντας τις ιεραρχίες αυτές χωρίς ιδιαίτερη προσπάθεια. Τέλος, σημειώνεται ότι το RTM_{sim}, το οποίο λαμβάνει υπόψη την ιεραρχική διάσταση των μοντέλων τύπων πόρων, εμφανίζει με συνέπεια σημαντικές τιμές και μπορεί να χρησιμοποιηθεί για να δείξει πως το επίπεδο ευστοχίας της διαδικασίας εξόρυξης πόρων μπορεί να ποικίλει σε σχέση με το μέγεθος των υπηρεσιών.

Υπολογιστική επίδοση και δυνατότητες κλιμάκωσης. Η αποτίμηση της υπολογιστικής επίδοσης στηρίχθηκε στην εφαρμογή της προσέγγισης εξαγωγής πόρων στο πλήρες σύνολο των 867 υπηρεσιών (12.918 λειτουργιών). Η υπολογι-

στική επίδοση και οι δυνατότητες κλιμάκωσης εξετάστηκαν μετρώντας τον χρόνο εξαγωγής πόρων (REX_t) ως προς το μέγεθος της υπηρεσίας και την ανάλυσή του σε χρόνο κατασκευής μοντέλου υπογραφών (SM_t), σε χρόνο παραγωγής μοντέλων όρων (TM_t), σε χρόνο αναδιάρθρωσης του STM (STM_t), σε χρόνο εξαγωγής των CCE (CCE_t) και σε χρόνο παραγωγής του RTM (RTM_t). Το εργαλείο αποτίμησης έτρεξε σε υπολογιστή με επεξεργαστή δύο πυρήνων στα 2.8GHz και μνήμης 4GB RAM. Στο Σχήμα 6.4 απεικονίζεται ο κινούμενος μέσος όρος του χρόνου εξαγωγής πόρων που απαιτήθηκε, όπως και η ανάλυσή του σε SM_t, TM_t, STM_t, CCE_t και RTM_t, σύμφωνα με την παραπάνω περιγραφή. Η διαδικασία, χωρίς κάποια ιδιαίτερη βελτιστοποίηση τρέχει σε λιγότερο από ένα δευτερόλεπτο για κάθε υπηρεσία, για την πλειονότητα των υπηρεσιών, κλιμακώνοντας σχεδόν γραμμικά ως προς το μέγεθος υπηρεσίας. Επίσης, ο περισσότερος χρόνος αναλώνεται στην παραγωγή των OTM και του STM (μοντέλα όρων), ο οποίος περιλαμβάνει τον χρόνο που απαιτείται ώστε να αντληθούν οι αλληλουχίες POS ετικετών από τα αντίστοιχα εργαλεία. Συνεπώς, μπορεί να σημειωθεί ότι η διαδικασία απαιτεί αρκετά χαμηλούς πόρους ώστε να μπορεί να ενσωματωθεί εύκολα σε μια διαδραστική διαδικασία πραγματικού χρόνου.

Αναθεώρηση αντί κατασκευής: επίπτωση στην παραγωγικότητα

Όπως συζητήθηκε παραπάνω, αφού παραχθεί το RTM ο χρήστης μπορεί να επιθεωρήσει και να αναθεωρήσει το μοντέλο ώστε να διασφαλίσει ότι αυτό αντιπροσωπεύει πραγματικά τη διεπαφή της υπηρεσίας που αναλύθηκε. Για την εκτίμηση της χρησιμότητας της προσέγγισης στο πλαίσιο ενός πραγματικού επιχειρησιακού περιβάλλοντος, προσκαλέσαμε έναν αρχιτέκτονα λογισμικού και την ομάδα του αποτελούμενη από μηχανικούς λογισμικού να εφαρμόσουν την προτεινόμενη διαδικασία εξόρυξης για ένα σύνολο ΠΣΔ υπηρεσιών. Το πρωτότυπο εξαγωγής πόρων χρησιμοποιήθηκε για να αναλυθούν πέντε υπηρεσίες και να εφαρμοστεί η διαδικασία εξαγωγής πόρων. Μετά την εξαγωγή κάθε μοντέλου τύπων πόρων, ο μηχανικός που εκτελούσε τη διαδικασία εξέταζε και αναθεωρούσε το μοντέλο έτσι ώστε να αντανακλά το νοητικό μοντέλο που είχε για την υπηρεσία. Κατά τη διάρκεια της διαδικασίας αναθεώρησης το εργαλείο τροποποίησης του RTM κατέγραφε τις πράξεις δημιουργίας, διαγραφής και τροποποίησης των στοιχείων του μοντέλου, όπως και μια χρονοσφραγίδα για κάθε πράξη που λάμβανε χώρα. Χρησιμοποιώντας τα αρχεία καταγραφής του εργαλείου τροποποίησης RTM μοντέλων και τα

αναθεωρημένα RTM μοντέλα, εφαρμόσαμε μια ανάλυση αποτίμησης έτσι ώστε να συγκρίνουμε την προσπάθεια και τον χρόνο που απαιτήθηκε για την αναθεώρηση κάθε μοντέλου (R_e και R_t αντίστοιχα) σε σχέση με μια εκτίμηση της προσπάθειας και του χρόνου που θα απαιτούνταν για τη δημιουργία των αναθεωρημένων μοντέλων από την αρχή (C_e και C_t). Το R_e ισούται με το πλήθος των πράξεων δημιουργίας, διαγραφής και τροποποίησης κατά τη διάρκεια μιας συνεδρίας αναθεώρησης μοντέλου. Το R_t ισούται με τον συνολικό χρόνο που απαιτήθηκε για τη συνεδρία αναθεώρησης σε δευτερόλεπτα. Για τον υπολογισμό του C_e χρησιμοποιήσαμε την αυστηρότερη δυνατή εκτίμηση θεωρώντας τον ελάχιστον αριθμό πράξεων που θα απαιτούνται για την κατασκευή του αναθεωρημένου μοντέλου από το μηδέν, το οποίο είναι ίσο με τον πλήθος των πράξεων δημιουργίας για όλους τους κόμβους και τις σχέσεις που περιλαμβάνονται στο αναθεωρημένο μοντέλο. Το C_t υπολογίστηκε με παρόμοιο τρόπο πολλαπλασιάζοντας τον μέσο χρόνο που απαιτήθηκε για τις πράξεις δημιουργίας και τροποποίησης κατά τη διάρκεια της αναθεώρησης με το C_e. Θα πρέπει να σημειωθεί ότι κατά τη διάρκεια της δημιουργίας ενός μοντέλου πόρων από το μηδέν, είναι σύνηθες για ένα χρήστη να προσθέτει στοιχεία του μοντέλου τα οποία στη συνέχεια τροποποιεί ή και αφαιρεί, προτού φτάσει στο τελικό, επιθυμητό αποτέλεσμα. Παρ' όλα αυτά, οι μετρικές C_e και C_t αντανακλούν το βέλτιστο δυνατό σενάριο για την περίπτωση κατασκευής του RTM, θεωρώντας ότι δεν απαιτούνται τέτοιου είδους διορθώσεις, ελαχιστοποιώντας ως εκ τούτου τις αντίστοιχες τιμές.

Χρησιμοποιώντας τα R_e, R_t, C_e και C_t μπορούμε πλέον να υπολογίσουμε την επίπτωση προσπάθειας I_e και την επίπτωση χρόνου I_t ως: $I_e = (1 - \frac{R_e}{C_e}) \times 100\%$ και $I_t = (1 - \frac{R_t}{C_t}) \times 100\%$. Οι τιμές επίπτωσης αντανακλούν το ποσοστό της προσπάθειας και του χρόνου που αποφεύγεται (οι θετικές τιμές υποδηλώνουν όφελος) ή προστίθεται (οι αρνητικές τιμές υποδηλώνουν επιβάρυνση) χρησιμοποιώντας την προτεινόμενη τεχνική εξαγωγής και στη συνέχεια την αναθεώρηση των μοντέλων αντί της κατασκευής τους από το μηδέν. Ο Πίνακας 6.3 παρουσιάζει τα αποτελέσματα των πειραμάτων για την αποτίμηση της επίπτωσης στην παραγωγικότητα. Συνοπτικά, χρησιμοποιώντας το πρωτότυπο η ομάδα των μηχανικών μπόρεσε να αντλήσει μοντέλα πόρων για τις ΠΣΔ υπηρεσίες τους με κατά μέσο όρο 63% λιγότερες πράξεις και σε 58.9% λιγότερο χρόνο, σε σύγκριση με τις πράξεις και τον χρόνο που θα απαιτούνταν για την εκ του μηδενός κατασκευή.

Απειλές εγκυρότητας και περιορισμοί

Εστιάζοντας στη μελέτη αποτίμησης της ακρίβειας της προσέγγισης, σημειώνουμε παρακάτω ένα απειλών εγκυρότητας και περιορισμών.

Εσωτερική Εγκυρότητα (Internal Validity): Όπως περιγράφηκε στην Ενότητα 6.1.2, λόγω έλλειψης διαθέσιμων συνόλων δεδομένων τα οποία θα μπορούσαν να χρησιμοποιηθούν ως σύνολα δεδομένων αναφοράς, για την αποτίμηση της ακρίβειας της προσέγγισης, επιλέξαμε να προσκαλέσουμε ειδικούς μηχανικούς για την εκ του μηδενός κατασκευή RTM μοντέλων για τις 70 υπηρεσίες που επιλέχθηκαν τυχαία, οι οποίοι στην πλειονότητα των περιπτώσεων δεν είχαν προηγούμενη γνώση ή εμπειρία με τις διεπαφές που εξέταζαν. Επιπλέον, μόνο ένα σύνολο των διεπαφών που εξετάστηκαν παρείχε τεκμηρίωση και για τις υπηρεσίες για τις οποίες υπήρχε τεκμηρίωση, υπήρχε ποικιλία ως προς το μέγεθος της τεκμηρίωσης και της ποιότητάς της. Συνεπώς, μια απειλή αποτελεί το κατά πόσο οι μηχανικοί που δημιούργησαν τα μοντέλα κατάφεραν να κατανοήσουν τις αρχικές διεπαφές υπηρεσιών και να τις μεταφράσουν σε ορθά ΠΣΠ μοντέλα.

Εξωτερική Εγκυρότητα (External Validity): Οι υπηρεσίες Ιστού που δημοσιεύονται από διάφορους παρόχους υπηρεσιών εμφανίζουν ποικιλομορφία ως προς τις συμβάσεις ονοματολογίας που χρησιμοποιούν. Αυτό συνδέεται με το γεγονός ότι δεν υπάρχει κάποια κεντρική αρχή η οποία να επιβάλει οριζόντια τέτοιες σχεδιαστικές αποφάσεις. Ταυτόχρονα, οι προδιαγραφές διαλειτουργικότητας γενικά δεν εξετάζουν αυτού του είδους ζητήματα. Επιπρόσθετα, αποτελεί συνήθη πρακτική να οι περιγραφές διεπαφών υπηρεσιών να παράγονται αυτόματα *από τα κάτω προς τα πάνω* (bottom-up). Στις περιπτώσεις αυτές, υπάρχουν εργαλεία που υποστηρίζουν τους προγραμματιστές στο να εκθέσουν υλοποιήσεις υπηρεσιών, παράγοντας αυτόματα τις περιγραφές των διεπαφών των υπηρεσιών αυτών. Συνεπώς, τα μοτίβα ονοματολογίας και δομής που ακολουθούνται σε διεπαφές υπηρεσιών που παράγονται με bottom-up τρόπο επηρεάζονται από τις υλοποιήσεις των υπηρεσιών, όπως και από τα εργαλεία που χρησιμοποιούνται για την παραγωγή των προδιαγραφών των διεπαφών. Η τεχνική που προτάθηκε επιχειρεί να αντιμετωπίσει το πρόβλημα ποικιλομορφίας διαχωρίζοντας τα διαφορετικά ζητήματα που εμπλέκονται στην πρόκληση της εξαγωγής πόρων σε ανεξάρτητα, καλώς ορισμένα βήματα. Σε κάθε βήμα, προτείνονται συγκεκριμένες τεχνικές και κανόνες, τα οποία, όπως εξετάστηκε στην Ενότητα 6.1.2 παρέχουν θετικά αποτελέσματα σε σχέση με την ακρίβεια και

την επίδοση. Σύμφωνα με τα παραπάνω, οι τεχνικές που προτάθηκαν σχεδιάστηκαν να λειτουργούν σε περιβάλλον ποικιλομορφίας και επιχειρούν να αποδώσουν όσο το δυνατόν περισσότερο ακριβή μοντέλα με βάση ορισμένες θεμελιώδεις υποθέσεις συνεπών και αναγνώσιμων/κατανοητών από ανθρώπους συμβάσεων ονοματολογίας στο όλο το εύρος των προδιαγραφών των διεπαφών. Ωστόσο, ένα ποικιλόμορφο περιβάλλον δεν αποτελεί πάντα το περιβάλλον λειτουργίας μιας διαδικασίας εξαγωγής πόρων. Για παράδειγμα, ενδέχεται να υπάρχουν περίπτωση ιδιαιτέρως ειδικών ή φτωχά σχεδιασμένων διεπαφών, στις οποίες θα πρέπει κάποιος να βασιστεί σε εξωτερικά τεχνουργήματα (μοντέλα/τεκμηρίωση/άλλα), ή ακόμα και σε επεξηγήσεις από ανθρώπους, έτσι ώστε να γίνει αποσαφηνιστεί η σημασιολογία των στοιχείων των διεπαφών. Σε περιβάλλοντα σαν και αυτά που περιγράφηκαν παραπάνω, η ακρίβεια της προσέγγισης αναμένεται να είναι χαμηλότερη, εκτός αν τουλάχιστον για τα βήματα διαχωρισμού συμβολοσειρών και παραγωγής μοντέλων όρων, υπάρξει κατάλληλη προσαρμογή ως προς τα ειδικά χαρακτηριστικά του περιβάλλοντος. Το κατά πόσο μια τέτοια προσπάθεια προσαρμογής μπορεί να δικαιολογηθεί εξαρτάται από το πόσο μεγάλο είναι το πλήθος και ο όγκος των υπηρεσιών που πρέπει να αναλυθούν, κάτι που ενδέχεται στα πλαίσια ενός οργανισμού να ποικίλει από μια και μοναδική υπηρεσία ως αποθετήρια εκατοντάδων υπηρεσιών. Σαφώς, για μεγαλύτερα αποθετήρια υπηρεσιών, μια αυτοματοποιημένη διαδικασία εξαγωγής φαίνεται να είναι προτιμότερη προσέγγιση.

Επίσης, όντας μια διαδικασία πολλαπλών βημάτων, υπάρχουν ορισμένα δυνατά σενάρια αποτυχίας για κάθε βήμα. Παρουσιάζουμε παρακάτω ορισμένα ενδεικτικά σενάρια, με αντίστοιχες πρακτικές αντιμετώπισης που μπορούν να ακολουθηθούν:

- Ο λανθασμένος διαχωρισμός συμβολοσειρών έχει άμεση επίπτωση στην παραγωγή των OTM και του STM. Οι λανθασμένες λεκτικές μονάδες είναι συνήθως άγνωστε λέξεις για τα εργαλεία POS επισημείωσης και συνεπώς ενδέχεται να επηρεαστεί η ακρίβεια της αλληλουχίας POS ετικετών. Τυπικές περιπτώσεις λανθασμένου διαχωρισμού συμβολοσειρών συζητούνται στο Κεφάλαιο 4, Ενότητα 4.2.3. Ωστόσο, με βάση τα πειράματα που διενεργήθηκαν, το σφάλμα διαχωρισμού συμβολοσειρών είναι χαμηλό και ακόμα και όταν συμβαίνει λανθασμένος διαχωρισμός, αυτό επηρεάζει μόνο ένα υποσύνολο των όρων οδηγώντας μεν σε μειωμένη ακρίβεια αλλά όχι σε πλήρη αποτυχία της συνολικής διαδικασίας.
- Το περιορισμένο γραμματολογικό και ορολογικό εύρος των ονομάτων των λει-

τουργιών όπως και η *μορφο-συντακτική αμφισημία* (*morpho-syntactic ambiguity*) η οποία είναι γνωστό ότι επηρεάζει εγγενώς την απόδοση POS ετικετών, ενδέχεται να οδηγήσει σε ατελή OTM μοντέλα, ακόμη και αν το βήμα διαχωρισμού συμβολοσειρών είναι ακριβές. Το πρόβλημα αντιμετωπίζεται μερικώς χρησιμοποιώντας τεχνικές μετα-επισημείωσης POS (meta POS tagging) όπως περιγράφεται στο Κεφάλαιο 4, Ενότητα 4.2.3, στις οποίες χρησιμοποιούνται διαφορετικά εργαλεία απόδοσης ετικετών για τον συνδυασμό των αποτελεσμάτων απόδοσης POS ετικετών κάθε εργαλείου μέσω majority voting συνάθροισης. Επιπλέον, ο κύκλος αναδιάρθρωσης OTM-STM βελτιώνει την ακρίβεια της παραγωγής OTM και εν συνεχεία και του STM. Τα σχετικά πειράματα υποδεικνύουν βελτίωση ακρίβειας 7.83% λόγω του κύκλου αναθεώρησης.

- Τα σφάλματα κατά την παραγωγή του RTM μπορούν να αποδοθούν πρωτίστως στο σφάλμα κανονικοποίησης πρόθεσης και στο σφάλμα επισημείωσης των παραμέτρων. Τέτοια σφάλματα ενδέχεται να οδηγήσουν σε ατελή RTM και συγκεκριμένα σε μοντέλα τα οποία έχουν μη ακριβείς ή ελλιπείς ιεραρχίες πόρων. Παρ' όλα αυτά, κατά τη διάρκεια του τελευταίου βήματος της διαδικασίας εξαγωγής πόρων, ο χρήστης μπορεί να επιθεωρήσει τα αποτελέσματα και να αποκαταστήσει αστοχίες του βήματος αυτού.

Τέλος, η προσέγγιση είναι επεκτάσιμη έτσι ώστε να μπορούν να προστεθούν νέες ευρετικές με αυξητικό τρόπο εφόσον κάτι τέτοιο χρειάζεται σε οποιοδήποτε βήμα της διαδικασίας, για τη βελτίωση της ακρίβειας.

6.2 Περιβάλλον-πλαίσιο αποτίμησης υποκαταστασιμότητας υπηρεσιών διαφορετικών υποδειγμάτων

6.2.1 Μελέτη περίπτωσης: DemoShop

Στην παρούσα ενότητα, παρουσιάζουμε μια μελέτη περίπτωσης στην οποία δύο υπηρεσίας, μια ΠΣΔ και μια ΠΣΠ, εξετάζονται ως προς την ευθυγραμμία και τη δυνατότητα υποκατάστασης χρησιμοποιώντας το περιβάλλον-πλαίσιο ΔΥΥ που προτάθηκε. Οι υπηρεσίες που χρησιμοποιούνται στη μελέτη αφορούν σε απλά συστήματα online παραγγελιών τα οποία επιτρέπουν την επιλογή ενός δείγματος

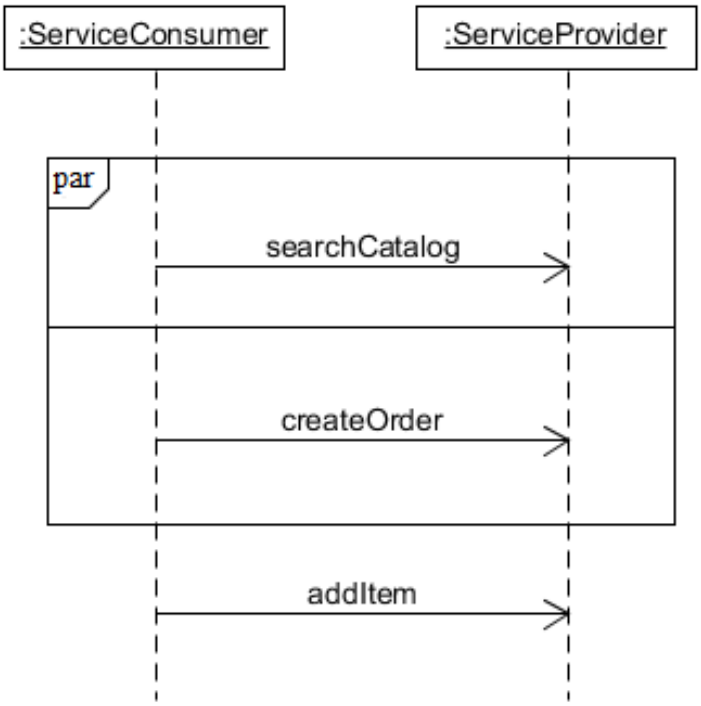
προϊόντος από μια συλλογή προϊόντων, έτσι ώστε αυτό να αποσταλεί σε δυνατούς νέους πελάτες, ως μέρος μιας προωθητικής ενέργειας ή στρατηγικής μάρκετινγκ. Η υπηρεσία, που ονομάζεται *DemoShop*, παρέχεται ως μια υπηρεσία τρίτου μέρους (third-party) σε πολλαπλούς καταναλωτές της υπηρεσίας. Με στόχο την αύξηση της ορατότητας και της υιοθέτησης της υπηρεσίας, ο πάροχος της υπηρεσίας αποφάσισε να εκθέσει τις δυνατότητες που παρέχει αυτή όχι μόνο μέσω ενός μιας υφιστάμενης ΠΣΔ διεπαφής, η οποία ονομάζεται *PO-DemoShop* αλλά και μέσω μιας νέας ΠΣΠ διεπαφής ονομαζόμενη *RO-DemoShop*. Δεδομένων των παραπάνω, ο πάροχος της υπηρεσίας απαιτείται να διασφαλίζει ότι οι δύο εκθέσεις της υπηρεσίας βρίσκονται σε ευθυγραμμία.

Μοντελοποίηση στα πλαισια της ΔΥΥ

Η υπηρεσία *PO-DemoShop* - M_P. Ας υποθέσουμε ότι η ΠΣΔ υπηρεσίας περιέχει τις εξής λειτουργίες: `searchCatalog`, `createOrder` και `addItem`. Το σενάριο χρήσης της υπηρεσίας είναι ότι μια εφαρμογή-πελάτης μπορεί να εκτελεί αναζητήσεις σε έναν κατάλογο με δείγματα - προϊόντα επίδειξης (`searchCatalog`) και στη συνέχεια μπορεί να δημιουργεί μια παραγγελία (`createOrder`) παρέχοντας πληροφορίες σχετικά με το πως και που θα σταλεί ό,τι επιλεχθεί. Έπειτα, η εφαρμογή-πελάτης μπορεί να προσθέσει το επιλεγμένο προϊόν στην παραγγελία χρησιμοποιώντας τη λειτουργία `addItem`. Ο Πίνακας 6.4 παρέχει τις υπογραφές των λειτουργιών της υπηρεσίας και το Σχήμα 6.5 απεικονίζει ένα UML ακολουθιακό διάγραμμα το οποίο προσδιορίζει το πρωτόκολλο αλληλεπίδρασης μεταξύ πελάτη και εξυπηρετητή σε υψηλό αφαιρετικό επίπεδο, ή το *εσωτερικό πρωτόκολλο της υπηρεσίας* (*intra-service protocol*) το οποίο απαιτείται από την υπηρεσία *PO-DemoShop*. Η πλήρης προδιαγραφή του μοντέλου για την υπηρεσία *PO-DemoShop*, M_P, παρέχεται στο Παράρτημα Β΄.

Πίνακας 6.4: Υπογραφές λειτουργιών της υπηρεσίας *PO-DemoShop*

Λειτουργία	Παράμετροι Εισόδου	Παράμετροι Εξόδου
searchCatelog	SearchCriteria	CatalogItems
createOrder	Authentication, Order	OrderID
addItem	Authentication, OrderID, Item	-



Σχήμα 6.5: Ακολουθιακό διάγραμμα που περιγράφει το εσωτερικό πρωτόκολλο της υπηρεσίας *PO-DemoShop*

Όπως απεικονίζεται στο Σχήμα 6.5 ένας καταναλωτής της υπηρεσίας *PO-DemoShop* μπορεί να ξεκινήσει να αλληλεπιδρά με αυτήν είτε δημιουργώντας μια παραγγελία μέσω της λειτουργίας `createOrder` είτε ψάχνοντας τον κατάλογο μέσω της λειτουργίας `searchCatalog` (βλ. `par` combined fragment). Έπειτα, ο καταναλωτής της υπηρεσίας μπορεί να συνεχίσει επιλέγοντας ένα στοιχείο του καταλόγου και προσθέτοντάς το ως στοιχείο στην παραγγελία μέσω της λειτουργίας `addItem`. Δεδομένων των παραπάνω, ανάλογα με το μοντέλο αλληλεπίδρασης που θέλει να εφαρμόσει η εφαρμογή-πελάτης, αυτή μπορεί να ξεκινήσει είτε επιτρέποντας τον χρήστη να δει το περιεχόμενο του καταλόγου και στη συνέχεια να δημιουργήσει μια παραγγελία στην οποία θα πρέπει να παρέχει ορισμένες πληροφορίες, είτε μπορεί να ξεκινήσει δημιουργώντας μια παραγγελία και στη συνέχεια να παρουσιάσει το περιεχόμενο του καταλόγου στον χρήστη.

Το μοντέλο M_P της υπηρεσίας *PO-DemoShop* περιέχει τρία `Procedural Interaction` στοιχεία, συγκεκριμένα το pi_1 για την κλήση της *createOrder*, το

pi_2 για την κλήση της *searchCatalog* και το pi_3 για την κλήση της *addItem*. Επίσης, το M_P περιέχει δύο `ProceduralTransition` στοιχεία: το $pt_1 = \{pi_1, pi_3\}$ το οποίο υποδηλώνει ότι ένα στοιχείο παραγγελίας μπορεί να προστεθεί εφόσον έχει δημιουργηθεί μια παραγγελία, και το $pt_2 = \{pi_2, pi_3\}$ το οποίο υποδηλώνει ότι ένα στοιχείο μπορεί να προστεθεί σε μια παραγγελία εφόσον έχει αναγνωριστεί και επιλεχθεί ένα αντίστοιχο στοιχείο καταλόγου μετά από αναζήτηση στον κατάλογο, δηλαδή μετά από κλήση της `searchCatalog`.

Η υπηρεσία *RO-DemoShop* - M_R. Η ΠΣΠ έκδοση της υπηρεσίας *DemoShop* περιέχει τους ακόλουθους τύπους πόρων: *store*, *products*, *product*, *carts*, *cart*, *items*, *item*. Ο πόρος εισόδου για την υπηρεσία είναι ο πόρος *store*, η αναπαράσταση του οποίου παρέχει συνδέσμους υπερμέσων στους πόρους *baskets* και *catalog*. Ένας καταναλωτής της υπηρεσίας μπορεί να δημιουργήσει ένα νέο καλάθι (basket) και να προσθέσει στοιχεία (item) σε αυτό. Ο Πίνακας 6.5 παρέχει μια λίστα των τύπων πόρων της διεπαφής της υπηρεσίας, τα πρότυπα αναγνωριστικών τους και τις επιτρεπόμενες πράξεις μεταχείρισης όπως προσδιορίζονται στο M_R. Επίσης, ο Πίνακας 6.6 παρέχει μια λίστα με τις σχέσεις υπερμέσων που υπάρχουν στο M_R, ορισμένες μεταξύ των ΠΣΠ σημείων αλληλεπίδρασης. Περιορίσαμε το πλήθος των επιτρεπόμενων πράξεων καθώς και των σχέσεων υπερμέσων έτσι ώστε το παράδειγμα να έχει εύλογο μέγεθος για τους σκοπούς της παρούσας παρουσίασης. Η πλήρης προδιαγραφή του μοντέλου για την υπηρεσία *RO-DemoShop*, M_R, παρέχεται στο Παράρτημα Β′.

Πίνακας 6.5: *RO-DemoShop*: μοντέλο πόρων

Τύπος πόρου	Πρότυπο αναγνωριστικού	Πράξεις
store	/store/	READ
products	/store/products/{?query}	READ
carts	/store/carts/	CREATE
cart	/store/carts/{cart.id}	-
items	/store/carts/{cart.id}/items/	-
item	/store/carts/{cart.id}/items/{cart.id}	CREATE

Πίνακας 6.6: *RO-DemoShop*: σχέσεις υπερμέσων

Σχέση υπερμέσων	Αλληλεπίδραση πηγής	Αλληλεπίδραση στόχου
hr_1	READ store	READ products
hr_2	READ store	CREATE carts
hr_3	CREATE carts	CREATE item
hr_4	READ products	CREATE carts

Κανόνες ευθυγραμμίας - M_A. Όπως συζητήθηκε στο Κεφάλαιο 5, για την αποτίμηση της υποκαταστασιμότητας μεταξύ M_P και M_R, απαιτείται ένα μοντέλο κανόνων ευθυγραμμίας M_A το οποίο συμφωνεί με το MM_A μεταμοντέλο. Μια εικόνα του μοντέλου των κανόνων ευθυγραμμίας μεταξύ των *PO-DemoShop* και *RO-DemoShop* παρέχεται στον Πίνακα 6.7, ενώ το αντίστοιχο πλήρες μοντέλο M_A παρέχεται στο Παράρτημα Β΄.

Πίνακας 6.7: Κανόνες ευθυγραμμίας *PO-DemoShop* και *RO-DemoShop*

Κανόνας	Τύπος Κανόνα	Στοιχείο του M_P	Στοιχείο του M_R
rl_{in2dp_1}	`InputToDynamicPart`	`OrderID` ⟨*InputParameter*⟩	`cart` ⟨*DynamicPart*⟩
rl_{in2st_1}	`InputToState`	`Item` ⟨*InputParameter*⟩	`item` ⟨*StateElement*⟩
$rl_{int2act_1}$	`IntentToAction`	`add` ⟨*Intent*⟩	`CREATE` ⟨*Action*⟩
rl_{in2cd_1}	`InputToControlData`	`Authentication` ⟨*InputParameter*⟩	`Authentication` ⟨*ControlDatum*⟩
rl_{in2dp_2}	`InputToDynamicPart`	`Item` ⟨*InputParameter*⟩	`item` ⟨*DynamicPart*⟩
rl_{t2sp_1}	`TermToStaticPart`	`item` ⟨*Concept*⟩	`carts` ⟨*StaticPart*⟩
rl_{t2sp_2}	`TermToStaticPart`	`item` ⟨*Concept*⟩	`items` ⟨*StaticPart*⟩
rl_{t2sp_3}	`TermToStaticPart`	`catalog` ⟨*Concept*⟩	`products` ⟨*StaticPart*⟩
rl_{t2sp_4}	`TermToStaticPart`	`order` ⟨*Concept*⟩	`carts` ⟨*StaticPart*⟩
rl_{in2dp_3}	`InputToDynamicPart`	`SearchCriteria` ⟨*InputParameter*⟩	`query` ⟨*DynamicPart*⟩
$rl_{int2act_2}$	`IntentToAction`	`search` ⟨*Intent*⟩	`READ` ⟨*Action*⟩
rl_{out2st_1}	`OutputToState`	`CatalogItems` ⟨*InputParameter*⟩	`products` ⟨*StateElement*⟩
rl_{in2cd_2}	`InputToControlData`	`Authentication` ⟨*InputParameter*⟩	`Authentication` ⟨*ControlDatum*⟩
rl_{in2st_2}	`InputToState`	`Order` ⟨*InputParameter*⟩	`cart` ⟨*StateElement*⟩
$rl_{int2act_3}$	`IntentToAction`	`create` ⟨*Intent*⟩	`CREATE` ⟨*Action*⟩
rl_{out2cd_1}	`OutputToControlData`	`OrderID` ⟨*OutputParameter*⟩	`Location` ⟨*ControlDatum*⟩

Διερεύνηση της ΔΥΥ

Παραγωγή των Open Nets - M_P, M_R, $M_A \rightarrow N_R$, N_P. Αφού αποδοθούν τιμές στα στοιχεία των μοντέλων M_P, M_R και M_A, η διαδικασία προχωρά στην εφαρμογή του μετασχηματισμού των μοντέλων η οποία αποτελείται από τρία βήματα όπως συζητείται στο Κεφάλαιο 5.

Στο Βήμα Α αντλείται η πρώτη γενιά των δικτύων Petri (G_1) για κάθε υπόδειγμα. Τα G_1 δίκτυα Petri αποτυπώνουν τη ροή αλληλεπίδρασης και ελέγχου για τη διεπαφή υπηρεσίας που μετασχηματίζεται.

Σχετικά με την υπηρεσία *PO-DemoShop*, χρησιμοποιώντας τον Μετασχηματισμό 1, οι ΠΣΔ αλληλεπιδράσεις στο M_P αντιστοιχίζονται σε στοιχεία δικτύων Petri ως ακολούθως:

- pi_1 (`searchCatalog`) $\rightarrow t_1$, p_1^{in}, a_1^{in}, p_1^{out}, a_1^{out},
- pi_2 (`createOrder`) $\rightarrow t_2$, p_2^{in}, a_2^{in}, p_2^{out}, a_2^{out},
- pi_3 (`addItem`) $\rightarrow t_3$, p_3^{in}, a_3^{in}, p_3^{out}, a_3^{out}, $p_{3\leftarrow 1}^{pre}$, $a_{3\leftarrow 1}^{pre}$, $p_{3\leftarrow 2}^{pre}$, $a_{3\leftarrow 2}^{pre}$

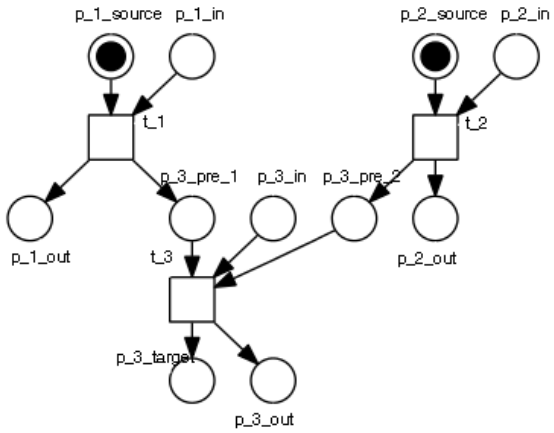
Στη συνέχεια, με βάση τον Μετασχηματισμό 2 οι ΠΣΔ μεταβάσεις του M_P μεταφράζονται σε στοιχεία δικτύων Petri ως εξής:

- $pt_1 \rightarrow a_{1\rightarrow 3}^{post}$,
- $pt_2 \rightarrow a_{2\rightarrow 3}^{post}$.

Έπειτα, χρησιμοποιώντας τον Μετασχηματισμό 3 τα στοιχεία p_1^{source}, a_1^{source}, p_2^{source}, και a_2^{source} προστίθενται και οι θέσεις p_1^{source}, p_2^{source} μαρκάρονται με ένα token στο αρχικό μαρκάρισμα. Τέλος, με βάση τον Μετασχηματισμό 4, τα στοιχεία p_3^{target} και a_1^{target} προστίθενται στο παραχθέν open net. Το Σχήμα 6.6 παρουσιάζει το $N_P^{G_1}$ όπως παράχθηκε για την υπηρεσία *ΠΟ-ΔεμοΣηοπ*, σύμφωνα με τις παραπάνω μεταφράσεις.

Σχετικά με την υπηρεσία *RO-DemoShop*, χρησιμοποιώντας τον Μετασχηματισμό 7, τα ΠΣΠ σημεία αλληλεπίδρασης στο M_R μεταφράζονται σε στοιχεία δικτύων Petri ως ακολούθως:

- rip_1 (`READ store`) $\rightarrow t_1$, p_1^{in}, a_1^{in}, p_1^{out}, a_1^{out}



Σχήμα 6.6: $N_P^{G_1}$: Η πρώτη γενιά δικτύων Petri που κατασκευάζεται στο Βήμα Α για την υπηρεσία *PO-DemoShop*

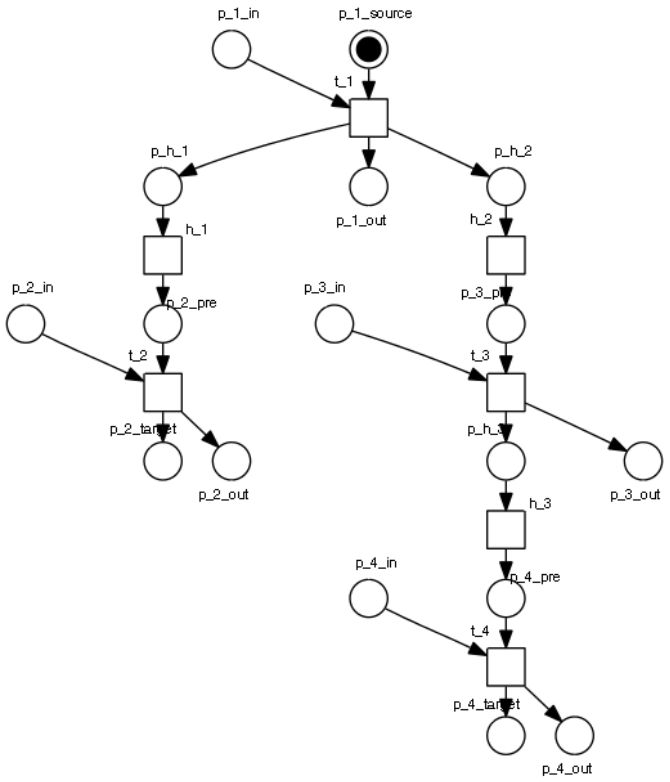
- rip_2 (`READ products`) $\rightarrow t_2$, p_2^{in}, a_2^{in}, p_2^{out}, a_2^{out}, p_2^{pre}, a_2^{pre}
- rip_3 (`CREATE carts`) $\rightarrow t_3$, p_3^{in}, a_3^{in}, p_3^{out}, a_3^{out}, p_3^{pre}, a_3^{pre}
- rip_4 (`CREATE item`) $\rightarrow t_4$, p_4^{in}, a_4^{in}, p_4^{out}, a_4^{out}, p_4^{pre}, a_4^{pre}

Έπειτα, με βάση τον Μετασχηματισμό 10, οι σχέσεις υπερμέσων στο M_R (παρέχονται στον Πίνακα 6.6) αντιστοιχίζονται σε στοιχεία δικτύων Petri ως εξής:

- $hr_1 \rightarrow h_1$, p^{h_1}, a^{h_1}, a_1^{from,t_1}, a_1^{to,t_2}
- $hr_2 \rightarrow h_2$, p^{h_2}, a^{h_2}, a_2^{from,t_1}, a_2^{to,t_3}
- $hr_3 \rightarrow h_3$, p^{h_3}, a^{h_3}, a_3^{from,t_3}, a_3^{to,t_4}
- $hr_4 \rightarrow h_4$, p^{h_4}, a^{h_4}, a_4^{from,t_1}, a_3^{to,t_3}

Τέλος, χρησιμοποιώντας τον Μετασχηματισμό 8, προστίθεται η *source* θέση p_1^{source} και το τόξο a_1^{source}, καθώς το rip_1 αποτελεί σημείο αλληλεπίδρασης *εισόδου* (*entry*). Σύμφωνα με τον ίδιο μετασχηματισμό, η θέση p_1^{source} μαρκάρεται με ένα token στο αρχικό μαρκάρισμα. Επίσης, χρησιμοποιώντας τον Μετασχηματισμό 9 και με βάση το γεγονός ότι το rip_4 είναι ένα σημείο αλληλεπίδρασης *εξόδου* (*exit*), προστίθεται η *target* θέση p_4^{target} και το αντίστοιχο σε αυτό τόξο a_4^{target}.

Στην υπηρεσία *RO-DemoShop* υπάρχουν δύο σχέσεις υπερμέσων, συγκεκριμένα οι hr_2 και hr_4, οι οποίες έχουν και οι δύο ως target ΠΣΠ σημείο αλληλεπίδρασης το rip_3. Συνεπώς, όπως προσδιορίζεται από τον Μετασχηματισμό 11, προστίθεται μια θέση, p_3^{guard}, στο παραχθέν δίκτυο Petri μαρκαρισμένο με ένα token στο αρχικό μαρκάρισμα, δηλαδή, $m_0(p_i^{guard}) = 1$, όπως και δύο τόξα $a_3^{guard,h_2} = \{p_3^{guard}, h_2\}$ και $a_3^{guard,h_4} = \{p_3^{guard}, h_4\}$. Στο Σχήμα 6.7 απεικονίζεται το $N_R^{G_1}$ που δημιουργήθηκε για την υπηρεσία *RO-DemoShop* με βάση τις παραπάνω μεταφράσεις.



Σχήμα 6.7: $N_R^{G_1}$: Η πρώτη γενιά δικτύων Petri που κατασκευάζεται στο Βήμα Α για την υπηρεσία *RO-DemoShop*

Η δεύτερη γενιά δικτύων Petri, $N_P^{G_2}$ και $N_R^{G_2}$, αντλείται εφαρμόζοντας το Βήμα Β στα δίκτυα $N_P^{G_1}$ και $N_R^{G_1}$, αντίστοιχα. Κατά το βήμα αυτό, οι *ιν* και *ουτ* θέσεις αναλύονται σύμφωνα με τους κανόνες ανάλυσης που παρουσιάστηκαν στο Κεφάλαιο 5, χρησιμοποιώντας πληροφορίες από τις προδιαγραφές των μοντέλων M_P και M_R.

Σχετικά με την ΠΣΔ έκδοση του Βήματος Β, χρησιμοποιώντας τον Μετασχημα-

τισμό 5, οι *in* θέσεις του $N_P^{G_1}$ αναλύονται σε θέσεις του $N_P^{G_2}$ ως εξής:

- $p_1^{in} \rightarrow \{\ p_1^{in,term_1}$ (`search` $\langle Intent \rangle$), $p_1^{in,term_2}$ (`Catalog` $\langle Concept \rangle$), p_1^{in,par_1} (`SearchCriteria` $\langle InputParameter \rangle$), $p_1^{in,endpoint}$ (`ServiceEndpoint` $\langle Endpoint \rangle$) $\}$
- $p_2^{in} \rightarrow \{\ p_2^{in,term_1}$ (`create` $\langle Intent \rangle$), $p_2^{in,term_2}$ (`Order` $\langle Concept \rangle$), p_2^{in,par_1} (`Authentication` $\langle InputParameter \rangle$), p_2^{in,par_2} (`Order` $\langle InputParameter \rangle$), $p_2^{in,endpoint}$ (`ServiceEndpoint` $\langle Endpoint \rangle$) $\}$
- $p_3^{in} \rightarrow \{\ p_3^{in,term_1}$ (`add` $\langle Intent \rangle$), $p_3^{in,term_2}$ (`Item` $\langle Concept \rangle$), p_3^{in,par_1} (`Authentication` $\langle InputParameter \rangle$), p_3^{in,par_2} (`OrderID` $\langle InputParameter \rangle$), p_3^{in,par_3} (`Item` $\langle InputParameter \rangle$), $p_3^{in,endpoint}$ (`ServiceEndpoint` $\langle Endpoint \rangle$) $\}$

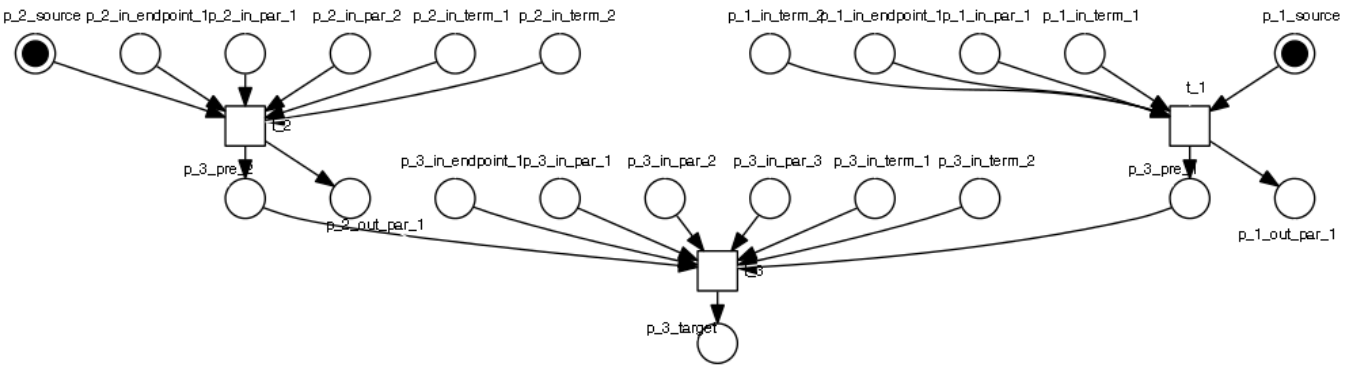
Χρησιμοποιώντας τον Μετασχηματισμό 6, οι *out* θέσεις του $N_P^{G_1}$ αναλύονται σε θέσεις του $N_P^{G_2}$ ως εξής:

- $p_1^{out} \rightarrow \{\ p_1^{out,par_1}$ (`CatalogItems` $\langle OutputParameter \rangle$) $\}$
- $p_2^{out} \rightarrow \{\ p_2^{out,par_1}$ (`OrderID` $\langle OutputParameter \rangle$) $\}$
- $p_3^{out} \rightarrow \emptyset$

Το Σχήμα 6.8 απεικονίζει το $N_P^{G_2}$ που δημιουργήθηκε $N_P^{G_1}$ με βάση τους παραπάνω μετασχηματισμούς.

Σχετικά με το ΠΣΠ μονοπάτι μετασχηματισμού, το Βήμα Β υλοποιείται αναλύοντας τις *in* και τις *out* θέσεις χρησιμοποιώντας τους Ορισμούς 12 και 13. Πιο συγκεκριμένα, χρησιμοποιώντας τον Μετασχηματισμό 12, *in* θέσεις του $N_R^{G_1}$ αναλύονται σε θέσεις του $N_R^{G_2}$ ως ακολούθως:

- $p_1^{in} \rightarrow \{\ p_1^{in,cd_1}$ (`Host` $\langle ControlDatum \rangle$), $p_1^{in,action}$ (`READ` $\langle Action \rangle$) $\}$
- $p_2^{in} \rightarrow \{\ p_2^{in,sp_1}$ (`products` $\langle StaticPart \rangle$), p_2^{in,dp_1} (`query` $\langle DynamicPart \rangle$), p_2^{in,cd_1} (`Host` $\langle ControlDatum \rangle$), $p_2^{in,action}$ (`READ` $\langle Action \rangle$) $\}$
- $p_3^{in} \rightarrow \{\ p_3^{in,sp_1}$ (`carts` $\langle StaticPart \rangle$), p_3^{in,cd_1} (`Host` $\langle ControlDatum \rangle$), p_3^{in,cd_2} (`Authentication` $\langle ControlDatum \rangle$), p_3^{in,rmd_1} (`Content-Type` $\langle RepresentationMetadatum \rangle$), p_3^{in,rd_1} (`cart` $\langle StateElement \rangle$), $p_3^{in,action}$ (`CREATE` $\langle Action \rangle$) $\}$



Σχήμα 6.8: $N_P^{G_2}$: Το δίκτυο Petri δεύτερης γενιάς που κατασκευάστηκε στο Βήμα Β για την υπηρεσία *PO-DemoShop*.

- $p_4^{in} \rightarrow \{\ p_4^{in,sp_1}$ (carts ⟨*StaticPart*⟩), p_4^{in,cd_1} (Host ⟨*ControlDatum*⟩), p_4^{in,cd_2} (Authentication ⟨*ControlDatum*⟩), p_4^{in,rmd_1} (Content-Type ⟨*RepresentationMetadatum*⟩), p_4^{in,rd_1} (cart ⟨*StateElement*⟩), p_4^{in,dp_1} (cart ⟨*DynamicPart*⟩), p_4^{in,dp_2} (item ⟨*DynamicPart*⟩), $p_4^{in,action}$ (CREATE ⟨*Action*⟩) }

Χρησιμοποιώντας τον Μετασχηματισμό 13, οι *out* θέσεις του $N_R^{G_1}$ αναλύονται σε θέσεις του $N_R^{G_2}$ ως ακολούθως:

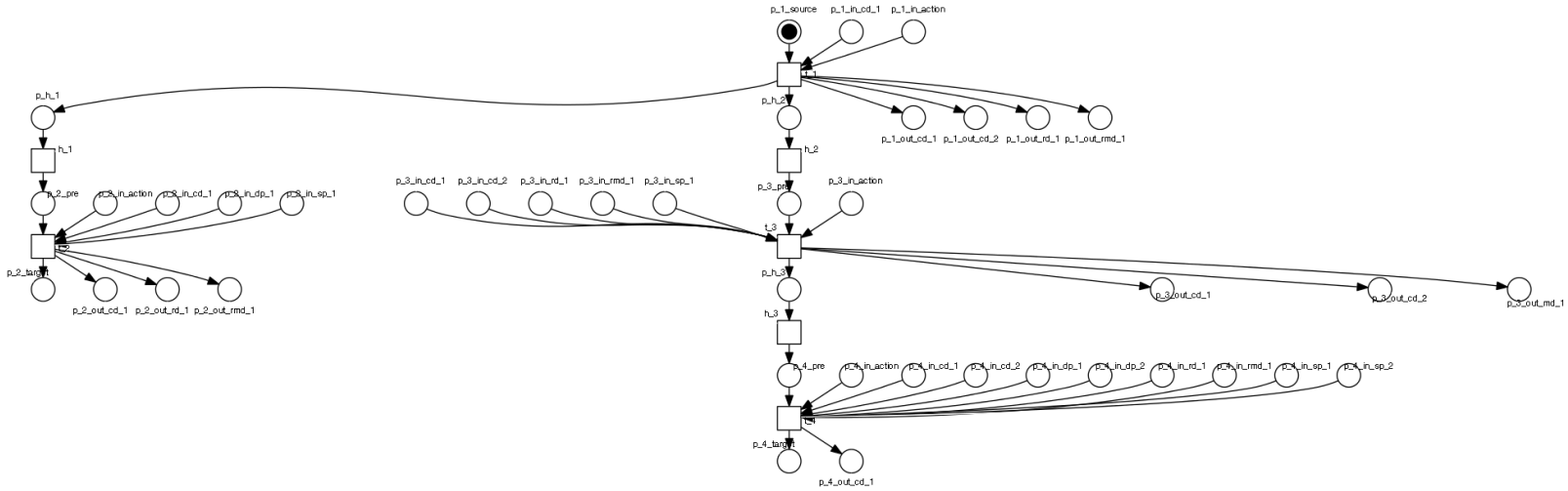
- $p_1^{out} \rightarrow \{\ p_1^{out,cd_1}$ (WWW-Authenticate ⟨*ControlDatum*⟩), p_1^{out,cd_2} (Response-Code ⟨*ControlDatum*⟩), p_1^{out,rmd_1} (Content-Type ⟨*RepresentationMetadatum*⟩), p_1^{out,rd_1} (status ⟨*StateDatum*⟩) }
- $p_2^{out} \rightarrow \{\ p_2^{out,cd_1}$ (Response-Code ⟨*ControlDatum*⟩), p_2^{out,rmd_1} (Content-Type ⟨*RepresentationMetadatum*⟩), p_2^{out,rd_1} (products ⟨*StateElement*⟩) }
- $p_3^{out} \rightarrow \{\ p_3^{out,cd_1}$ (Response-Code ⟨*ControlDatum*⟩), p_3^{out,cd_2} (Location ⟨*ControlDatum*⟩), p_3^{out,md_1} (Link ⟨*ResourceMetadatum*⟩) }
- $p_4^{out} \rightarrow \{\ p_4^{out,cd_1}$ (Response-Code ⟨*ControlDatum*⟩) }

Με βάση τις μεταφράσεις που παρουσιάστηκαν παραπάνω, παράγεται το δίκτυο $N_R^{G_2}$ από το $N_R^{G_1}$, το οποίο απεικονίζεται στο Σχήμα 6.9.

Τέλος, κατά το Βήμα Γ και κάνοντας χρήση των κανόνων ευθυγραμμίας που περιλαμβάνονται στο M_A (βλέπε Πίνακα 6.7), τα G_2 δίκτυα μετασχηματίζονται σε open net. Όπως συζητείται στο Κεφάλαιο 5, κατά το Βήμα Γ εφαρμόζονται οι τεχνικές κατηγοριοποίησης θέσεων (*place categorization*), αναδιαμόρφωσης διεπαφής (*interface refactoring*) και αναγνώρισης τελικών καταστάσεων (*final states identification*) για την παραγωγή των open net N_P και N_R. Η εφαρμογή των τεχνικών αυτών στις υπηρεσίες *PO-DemoShop* και *RO-DemoShop* παρουσιάζονται λεπτομερώς στις επόμενες παραγράφους.

Για τη δημιουργία του N_P, αρχικά η place categorization τεχνική αναθέτει τις θέσεις του $N_P^{G_2}$ στα I_{N_P}, O_{N_P} και E_{N_P} ως εξής:

- $I_{N_P} = \{\ p_1^{in,par_1},\ p_1^{in,term_1},\ p_1^{in,term_2},\ p_2^{in,par_1},\ p_2^{in,term_2},\ p_2^{in,par_2},\ p_2^{in,term_1},\ p_3^{in,par_2},\ p_3^{in,par_3},\ p_3^{in,term_1},\ p_3^{in,par_1},\ p_3^{in,par_3},\ p_3^{in,term_2},\ p_3^{in,term_2}\}$
- $O_{N_P} = \{\ p_1^{out,par_1},\ p_2^{out,par_1}\ \}$



Σχήμα 6.9: $N_R^{G_2}$: Το δίκτυο Petri δεύτερης γενιάς που κατασκευάστηκε στο Βήμα Β για την υπηρεσία *RO-DemoShop*.

- $E_{N_P} = \{ p^{pre}_{3\leftarrow 1}, p^{pre}_{3\leftarrow 2}, p^{source}_{1}, p^{source}_{2}, p^{target}_{3}, p^{in,endpoint}_{1}, p^{in,endpoint}_{2}, p^{in,endpoint}_{3} \}$

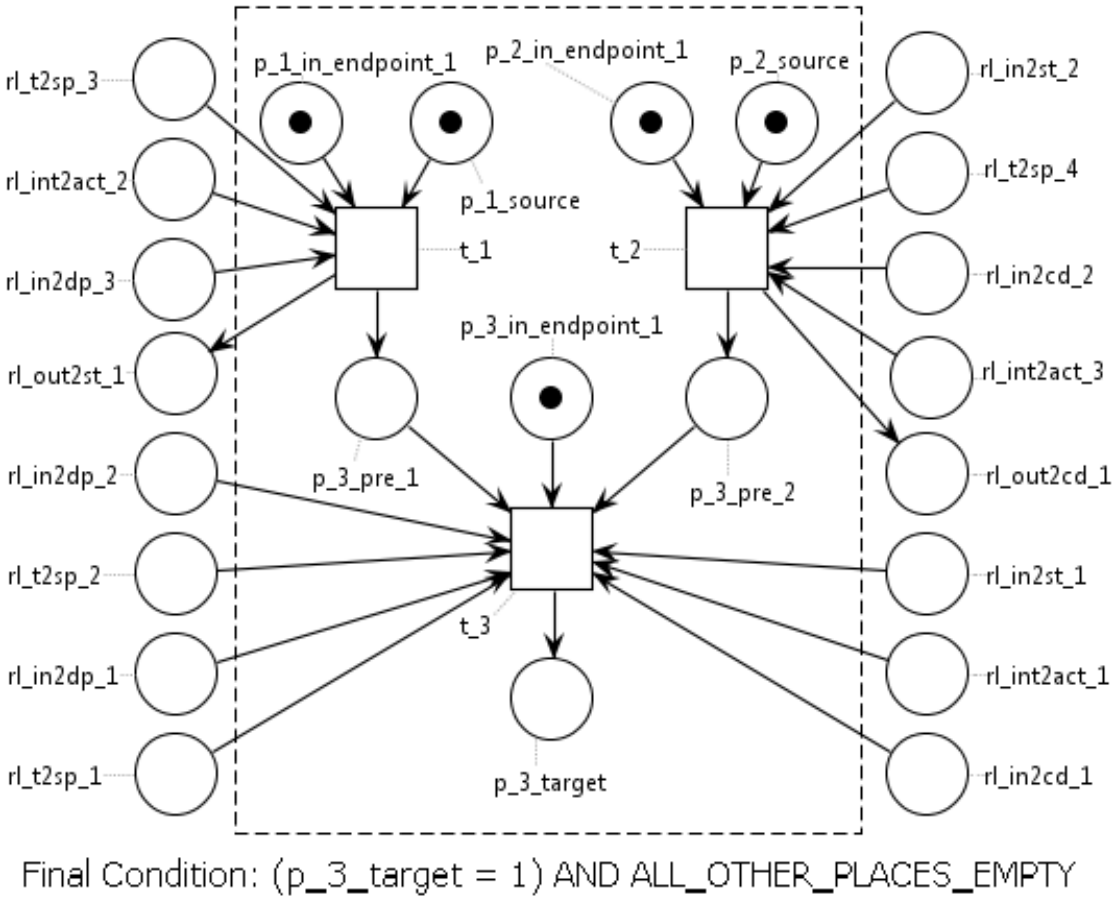
Έπειτα, η τεχνική αναδιαμόρφωσης διεπαφής χρησιμοποιεί τους συγκεκριμένους κανόνες αντιστοίχισης στο M_A είτε για να αναθέσει νέες ετικέτες, είτε για να προσθέσει νέες θέσεις στη διεπαφή του δικτύου ($I_{N_P} \cup O_{N_P}$). Σύμφωνα με τα παραπάνω, εφαρμόζονται οι παρακάτω πράξεις αναδιαμόρφωσης:

- $rl^{in2dp_3} \rightarrow$ `SearchCriteria` $\langle InputParameter \rangle \rightarrow p^{in,par_1}_{1}$ (αλλαγή ετικέτας)
- $rl^{int2act_2} \rightarrow$ `search` $\langle Intent \rangle \rightarrow p^{in,term_1}_{1}$ (αλλαγή ετικέτας)
- $rl^{t2sp_3} \rightarrow$ `Catalog` $\langle Concept \rangle \rightarrow p^{in,term_2}_{1}$ (αλλαγή ετικέτας)
- $rl^{in2cd_2} \rightarrow$ `Authentication` $\langle InputParameter \rangle \rightarrow p^{in,par_1}_{2}$ (αλλαγή ετικέτας)
- $rl^{t2sp_4} \rightarrow$ `Order` $\langle Concept \rangle \rightarrow p^{in,term_2}_{2}$ (αλλαγή ετικέτας)
- $rl^{in2st_2} \rightarrow$ `Order` $\langle InputParameter \rangle \rightarrow p^{in,par_2}_{2}$ (αλλαγή ετικέτας)
- $rl^{int2act_3} \rightarrow$ `create` $\langle Intent \rangle \rightarrow p^{in,term_1}_{2}$ (αλλαγή ετικέτας)
- $rl^{in2dp_1} \rightarrow$ `OrderID` $\langle InputParameter \rangle \rightarrow p^{in,par_2}_{3}$ (αλλαγή ετικέτας)
- $rl^{in2st_1} \rightarrow$ `Item` $\langle InputParameter \rangle \rightarrow p^{in,par_3}_{3}$ (αλλαγή ετικέτας)
- $rl^{int2act_1} \rightarrow$ `add` $\langle Intent \rangle \rightarrow p^{in,term_1}_{3}$ (αλλαγή ετικέτας)
- $rl^{in2cd_1} \rightarrow$ `Authentication` $\langle InputParameter \rangle \rightarrow p^{in,par_1}_{3}$ (αλλαγή ετικέτας)
- $rl^{in2dp_2} \rightarrow$ `Item` $\langle InputParameter \rangle \rightarrow p^{in,par_3}_{3}$ (προσθήκη θέσης)
- $rl^{t2sp_1} \rightarrow$ `Item` $\langle Concept \rangle \rightarrow p^{in,term_2}_{3}$ (αλλαγή ετικέτας)
- $rl^{t2sp_2} \rightarrow$ `Item` $\langle Concept \rangle \rightarrow p^{in,term_2}_{3}$ (προσθήκη θέσης)
- $rl^{out2st_1} \rightarrow$ `CatalogItems` $\langle OutputParameter \rangle \rightarrow p^{out,par_1}_{1}$ (αλλαγή ετικέτας)
- $rl^{out2cd_1} \rightarrow$ `OrderID` $\langle OutputParameter \rangle \rightarrow p^{out,par_1}_{2}$ (αλλαγή ετικέτας)

Όπως συζητήθηκε στο Κεφάλαιο 5, το Ω_{N_P} προσδιορίζεται μέσω μιας τελικής συνθήκης η οποία απαιτεί ότι κάθε *target* θέση μαρκάρεται με ένα token, ενώ όλες οι άλλες θέσεις πρέπει να είναι πρόσθετα. Βάσει των παραπάνω, η τελική συνθήκη του N_P είναι $m(p^{target}_{3}) = 1 \wedge ALL\_OTHER\_PLACES\_EMPTY$.

Το Σχήμα 6.10 απεικονίζει το N_P όπως κατασκευάστηκε στο Βήμα Γ για την υπηρεσία *PO-DemoShop*.

Για την κατασκευή του N_R, αρχικά, η εφαρμογή της τεχνικής κατηγοριοποίησης θέσεων οδηγεί στα παρακάτω σύνολα θέσεων:



Σχήμα 6.10: N_P: Το open net που κατασκευάστηκε στο Βήμα Γ για την υπηρεσία *PO-DemoShop*

- $I_{N_R} = \{ p_2^{in,sp_1}, p_2^{in,dp_1}, p_2^{in,action}, p_3^{in,cd_2}, p_3^{in,sp_1}, p_3^{in,rd_1}, p_3^{in,action}, p_4^{in,dp_1}, p_4^{in,rd_1}, p_4^{in,action}, p_4^{in,cd_2}, p_4^{in,dp_2}, p_4^{in,sp_1}, p_4^{in,sp_2} \}$
- $O_{N_R} = \{ p_2^{out,rd_1}, p_3^{out,cd_2} \}$
- $E_{N_R} = \{ p_2^{pre}, p_3^{pre}, p_4^{pre}, p^{h_1}, p^{h_4}, p^{h_2}, p^{h_3}, p_1^{source}, p_4^{target}, p_3^{guard}, p_3^{in,rmd_1}, p_3^{in,cd_1}, p_2^{in,cd_1}, p_1^{in,cd_1}, p_4^{in,cd_1}, p_4^{in,rmd_1}, p_3^{out,cd_1}, p_3^{out,md_1}, p_2^{out,cd_1}, p_2^{out,rmd_1}, p_1^{out,cd_2}, p_1^{out,rmd_1}, p_1^{out,cd_1}, p_1^{out,rd_1}, p_4^{out,cd_1} \}$

Η τεχνική αναδιαμόρφωσης διεπαφής για το N_R εφαρμόζεται ως εξής:

- $rl^{in2dp_1} \rightarrow$ `cart` $\langle DynamicPart \rangle \rightarrow p_4^{in,dp_1}$
- $rl^{in2st_1} \rightarrow$ `item` $\langle StateElement \rangle \rightarrow p_4^{in,rd_1}$

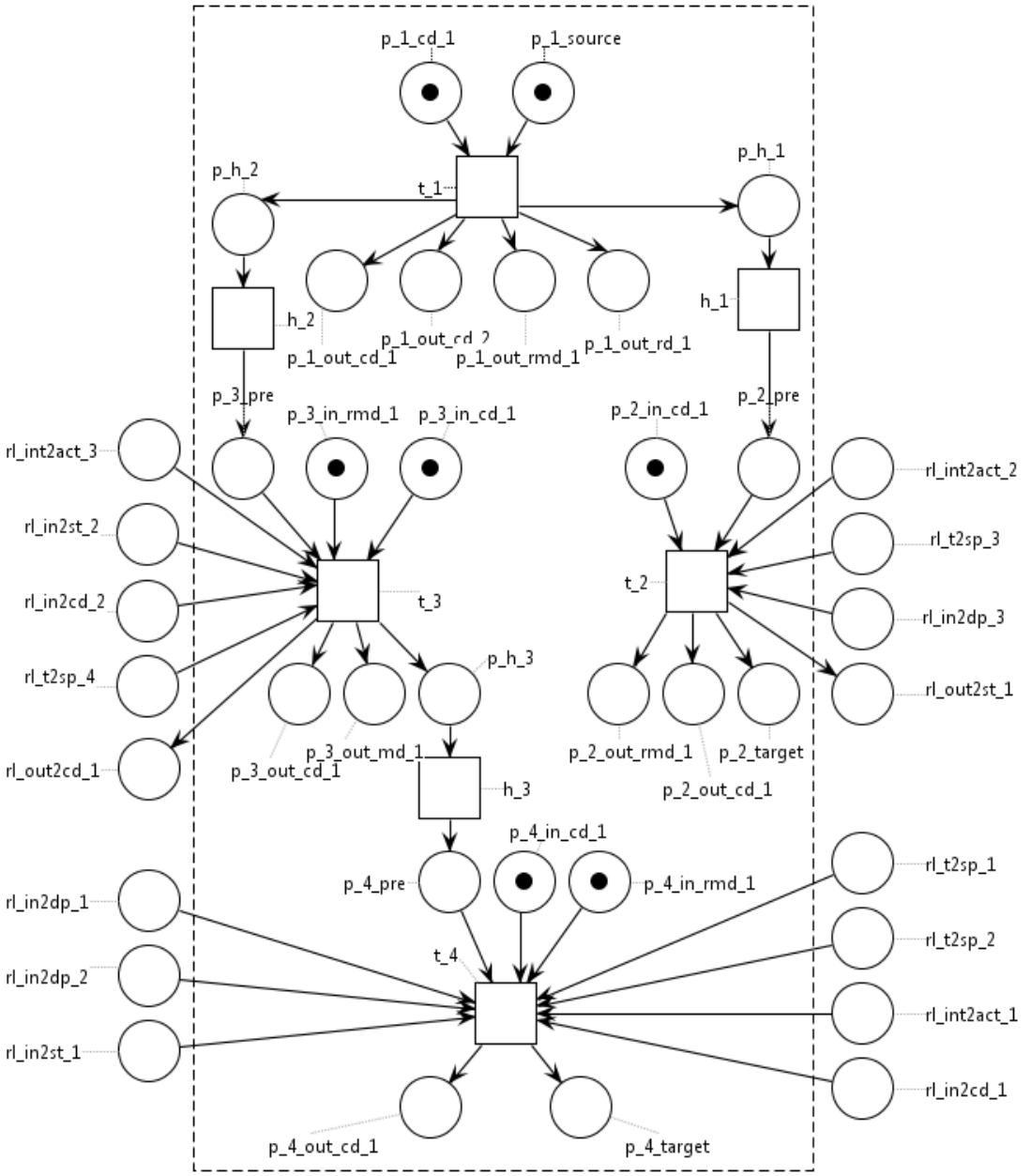
- $rl^{int2act_1} \rightarrow$ `CREATE` $\langle Action \rangle \rightarrow p_4^{in,action}$
- $rl^{in2cd_1} \rightarrow$ `Authentication` $\langle ControlDatum \rangle \rightarrow p_4^{in,cd_2}$
- $rl^{in2dp_2} \rightarrow$ `item` $\langle DynamicPart \rangle \rightarrow p_4^{in,dp_2}$
- $rl^{t2sp_1} \rightarrow$ `carts` $\langle StaticPart \rangle \rightarrow p_4^{in,sp_1}$
- $rl^{t2sp_2} \rightarrow$ `items` $\langle StaticPart \rangle \rightarrow p_4^{in,sp_2}$
- $rl^{t2sp_3} \rightarrow$ `products` $\langle StaticPart \rangle \rightarrow p_2^{in,sp_1}$
- $rl^{in2dp_3} \rightarrow$ `query` $\langle DynamicPart \rangle \rightarrow p_2^{in,dp_1}$
- $rl^{int2act_2} \rightarrow$ `READ` $\langle Action \rangle \rightarrow p_2^{in,action}$
- $rl^{out2st_1} \rightarrow$ `products` $\langle StateElement \rangle \rightarrow p_2^{out,rd_1}$
- $rl^{in2cd_2} \rightarrow$ `Authentication` $\langle ControlDatum \rangle \rightarrow p_3^{in,cd_2}$
- $rl^{t2sp_4} \rightarrow$ `carts` $\langle StaticPart \rangle \rightarrow p_3^{in,sp_1}$
- $rl^{in2st_2} \rightarrow$ `cart` $\langle StateElement \rangle \rightarrow p_3^{in,rd_1}$
- $rl^{int2act_3} \rightarrow$ `CREATE` $\langle Action \rangle \rightarrow p_3^{in,action}$
- $rl^{out2cd_1} \rightarrow$ `Location` $\langle ControlDatum \rangle \rightarrow p_3^{out,cd_2}$

Η αναγνώριση τελικών καταστάσεων για το N_R εφαρμόζεται μέσω σύζευξης των παρακάτω συνθηκών:

- $m(p_4^{target}) = 1 \wedge m(p_3^{out,cd_1}) = 1 \wedge m(p_3^{out,md_1}) = 1 \wedge m(p_2^{out,cd_1}) = 1 \wedge m(p_2^{out,rmd_1}) = 1 \wedge m(p_1^{out,cd_2}) = 1 \wedge m(p_1^{out,rmd_1}) = 1 \wedge m(p_1^{out,cd_1}) = 1 \wedge m(p_1^{out,rd_1}) = 1 \wedge m(p_4^{out,cd_1}) = 1$
- $m(p^{h_2}) = 0 \wedge m(p^{h_4}) = 1 \vee m(p^{h_2}) = 1 \wedge m(p^{h_4}) = 0$
- $ALL\_OTHER\_PLACES\_EMPTY$

Στο Σχήμα 6.11 απεικονίζεται το N_R όπως κατασκευάστηκε κατά το Βήμα Γ για την υπηρεσία *RO-DemoShop*.

Εξέταση accordance. Αφού παραχθούν τα open net για τις υπηρεσίες *PO-DemoShop* και *RO-DemoShop*, αρχικά εξετάζουμε το κατά πόσο τα N_P και N_R είναι ισοδύναμα ως προς τις διεπαφές τους (interface equivalence). Ο έλεγχος της ισοδυναμίας διεπαφών πραγματοποιείται συγκρίνοντας το I_{N_P} με το I_{N_R} και το O_{N_P} με το O_{N_R}. Εάν υπάρχει ταύτιση μεταξύ αυτών τότε τα N_P και N_R είναι ισοδύναμα



Final Condition: (p_2_target = 1) AND (p_4_target = 1) AND (p_4_out_cd_1 = 1) AND (p_2_out_rmd_1 = 1) AND (p_2_out_cd_1 = 1) AND (p_3_out_md_1 = 1) AND (p_3_out_cd_1 = 1) AND (p_1_out_cd_2 = 1) AND (p_1_out_rmd_1 = 1) AND (p_1_out_rd_1 = 1) AND (p_1_out_cd_1 = 1) AND ALL_OTHER_PLACES_EMPTY

Σχήμα 6.11: N_R: Το open net που κατασκευάστηκε στο Βήμα Γ για την υπηρεσία *RO-DemoShop*

ως προς τις διεπαφές τους. Στο περιβάλλον-πλαίσιο ΔΥΥ η ισοδυναμία διεπαφών εξετάζεται κατά τη διάρκεια της τελευταίας φάσης του μετασχηματισμού από το δομοστοιχείο IPSS transformer. Παρ' όλα αυτά, τα open net που παράγονται από τον μετασχηματισμό είναι δυνατό να ελεγχθούν και με χρήση εξωτερικών εργαλείων.

Για τη μελέτη περίπτωσης των *PO-DemoShop* και *RO-DemoShop* και τα N_P και N_R που παρήχθησαν, το εργαλείο μετασχηματισμού έδειξε ότι τα open net είναι ισοδύναμα ως προς τις διεπαφές τους. Τα δίκτυα N_P και N_R εξάγονται από το εργαλείο σε δύο αρχεία `po-demoshop.owfn` και `ro-demoshop.owfn`, αντίστοιχα.

Όπως περιγράφηκε στο Κεφάλαιο 5, αφού διασφαλιστεί η ισοδυναμία διεπαφών, χρησιμοποιείται το εργαλείο Petri για να παρέχει τις κανονικοποιημένες εκδόσεις των δικτύων, N_P^c και N_R^c -δηλαδή, η δομή κάθε open net θα αλλάξει έτσι ώστε κάθε μετάβαση να είναι συνδεδεμένη με το πολύ μια θέση διεπαφής.

```
1 $ petri -i "owfn" -n po-demoshop.owfn -o "owfn" --removePorts -v
2 pnapi: not using a configuration file
3 pnapi: po-demoshop.owfn: |P|= 24 |P_in|= 14 |P_out|= 2 |T|= 3 |F|= 26
4 pnapi: normalizing reducing Petri net 'po-demoshop.owfn'...
5 pnapi: writing owfn output to file 'po-demoshop.owfn.normalized.owfn'
6 pnapi: closed file 'po-demoshop.owfn.normalized.owfn'
```

```
1 $ petri -i "owfn" -n ro-demoshop.owfn -o "owfn" --removePorts -v
2 pnapi: not using a configuration file
3 pnapi: ro-demoshop.owfn: |P|= 41 |P_in|= 14 |P_out|= 2 |T|= 8 |F|= 50
4 pnapi: normalizing reducing Petri net 'ro-demoshop.owfn'...
5 pnapi: writing owfn output to file 'ro-demoshop.owfn.normalized.owfn'
6 pnapi: closed file 'ro-demoshop.owfn.normalized.owfn'
```

Η παράμετρος `-i` στο εργαλείο Petri προσδιορίζει τη μορφοδομή του αρχείου εισόδου και η παράμετρος `-o` προσδιορίζει τη μορφοδομή την οποία θα ακολουθεί το αρχείο εξόδου. Η παράμετρος `-n` προσδιορίζει ότι η πράξη τροποποίησης που θα εφαρμοστεί στο open net που παρέχεται ως είσοδος θα είναι αυτή της κανονικοποίησης. Στη μελέτη περίπτωσης που συζητείται στον παρόν κεφάλαιο, τόσο η μορφοδομή εισόδου όσο και η μορφοδομή εξόδου ακολουθούν το συντακτικό `OWFN`[3]. Η παράμετρος `--removePorts` αφαιρεί τις OWFN θύρες καθώς δεν απαιτούνται για την ανάλυσή μας και η παράμετρος `-v` αποτελεί επιλογή αναλυτικών δεδομένων εξόδου σχετικών με τη λειτουργία

[3]http://download.gna.org/service-tech/fiona/fiona.pdf, Section 4.

του εργαλείου, όπως εμφανίζεται παραπάνω. Μετά την εκτέλεση του εργαλείου Petri δύο φορές, μιά για κάθε open net, τα κανονικοποιημένα open net N_P^c και N_R^c αποθηκεύονται στα αρχεία `po-demoshop.owfn.normalized.owfn` και `ro-demoshop.owfn.normalized.owfn`, αντίστοιχα.

Στη συνέχεια, τα κανονικοποιημένα δίκτυα N_P^c και N_R^c υποβάλλονται στο εργαλείο Wendy για να αντληθούν τα αντίστοιχα operating guideline μοντέλα, $OG(N_P^c)$ και $OG(N_R^c)$.

```
1  $ wendy po-demoshop.owfn.normalized.owfn --og -v
2  wendy: not using a configuration file
3  wendy: read net: |P|= 50 |P_in|= 14 |P_out|= 2 |T|= 16 |F|= 78
4  wendy: message bound set to 1 (2 bytes/interface marking, 1 bits/event)
5  wendy: send event detection requires 2 bytes per inner marking
6  wendy: writing to temporary file '/tmp/wendy-gwjKdi'
7  wendy: calling LoLA: 'lola-statespace /tmp/wendy-gwjKdi -M'
8  lola: 16 significant places
9  lola: st: 16326 edg: 100000
10 lola: st: 31229 edg: 200000
11 wendy: LoLA is done (2 sec)
12 wendy: closed and deleted temporary file '/tmp/wendy-gwjKdi'
13 wendy: stored 40960 inner markings (1 final, 1 bad, 31166 inevitable deadlocks)
14 wendy: 12281 knowledges, 100000 edges
15 wendy: stored 19584 knowledges, 170688 edges (143 sec)
16 wendy: 19584 knowledges reachable
17 wendy: net is controllable: YES
18 wendy: writing operating guidelines to file 'po-demoshop.owfn.normalized.og'
19 wendy: closed file 'po-demoshop.owfn.normalized.og'
```

```
1  $ wendy ro-demoshop.owfn.normalized.owfn --og -v
2  wendy: not using a configuration file
3  wendy: read net: |P|= 67 |P_in|= 14 |P_out|= 2 |T|= 21 |F|= 102
4  wendy: message bound set to 1 (2 bytes/interface marking, 1 bits/event)
5  wendy: send event detection requires 2 bytes per inner marking
6  wendy: writing to temporary file '/tmp/wendy-HUr7PX'
7  wendy: calling LoLA: 'lola-statespace /tmp/wendy-HUr7PX -M'
8  lola: 21 significant places
9  lola: st: 15835 edg: 100000
10 lola: st: 32118 edg: 200000
11 lola: st: 47136 edg: 300000
12 wendy: 50000 inner markings
13 lola: st: 61444 edg: 400000
14 lola: st: 74706 edg: 500000
15 lola: st: 88485 edg: 600000
```

```
wendy: 100000 inner markings
lola: st: 102451 edg: 700000
lola: st: 115495 edg: 800000
lola: st: 127756 edg: 900000
lola: st: 141320 edg: 1000000
wendy: 150000 inner markings
lola: st: 153576 edg: 1100000
wendy: LoLA is done (7 sec)
wendy: closed and deleted temporary file '/tmp/wendy-HUr7PX'
wendy: stored 163840 inner markings (2 final, 2 bad, 117484 inevitable deadlocks)
wendy: 12370 knowledges, 100000 edges
wendy: stored 19840 knowledges, 172096 edges (702 sec)
wendy: 19840 knowledges reachable
wendy: net is controllable: YES
wendy: writing operating guidelines to file 'ro-demoshop.owfn.normalized.og'
wendy: closed file 'ro-demoshop.owfn.normalized.og'
```

Όπως καταγράφεται παραπάνω, το εργαλείο WENDY εκτελείται δύο φορές, μια φορά για κάθε κανονικοποιημένο open net. Επίσης, το WENDY καλείται με την παράμετρο `--og` η οποία προσδιορίζει ότι η έξοδος θα είναι ένα operating guideline μοντέλο. Ομοίως με το PETRI, η παράμετρος `-v` χρησιμοποιείται για να εμφανιστούν αναλυτικά δεδομένα εκτέλεσης. Το WENDY αρχικά χρησιμοποιεί το LOLA, ένα άλλο εργαλείο του SERVICETECHNOLOGY, για να υπολογίζει τα marking του εσωτερικού δικτύου Petri και στη συνέχεια εξετάζει την ελεγξιμότητα (controllability) και υπολογίζει το operating guideline μοντέλο εξόδου. Τα δύο operating guideline μοντέλα που παράγονται, $OG(N_P^c)$ και $OG(N_R^c)$, αποθηκεύονται στα αρχεία `po-demoshop.owfn.normalized.og` και `ro-demoshop.owfn.normalized.og`, αντίστοιχα.

Δεδομένου ότι το WENDY παράγει operating guideline μοντέλα σε διαφορετική μορφοδομή από αυτή που απαιτείται από το εργαλείο COSME, απαιτείται ένα βήμα μετασχηματισμού αρχείων. Κατά συνέπεια, χρησιμοποιούμε ένα εργαλείο που ονομάζεται WENDYFORMULA2BITS και πάλι από το SERVICETECHNOLOGY, για την προσαρμογή των αρχείων με μορφοδομή *Wendy formula* σε αρχεία με μορφοδομή *bits*.

```
$ wendyFormula2bits -i po-demoshop.owfn.normalized.og -o po-demoshop.owfn.normalized.bits.og
$ wendyFormula2bits -i ro-demoshop.owfn.normalized.og -o ro-demoshop.owfn.normalized.bits.og
```

Στο WENDYFORMULA2BITS, η παράμετρος `-i` χρησιμοποιείται για να προσ-

διορίσει το αρχείο εισόδου ενώ η παράμετρος `-o` χρησιμοποιείται για να προσδιορίσει το αρχείο στο οποίο θα αποθηκευτεί η έξοδος του μετασχηματισμού. Συνεπώς, το $OG(N_P^c)$ αποθηκεύεται σε μορφοδομή bits στο αρχείο `po-demoshop.owfn.normalized.bits.og` και το $OG(N_R^c)$ αποθηκεύεται σε μορφοδομή bits στο αρχείο `ro-demoshop.owfn.normalized.bits.og`.

Τέλος, χρησιμοποιώντας το εργαλείο Cosme μπορούμε να αποφανθούμε αν το N_R^c συμφωνεί με το N_P^c και αντίστροφα, εξετάζοντας τη σχέση εξειδίκευσης $\sqsubseteq$ μεταξύ $OG(N_P^c)$ και $OG(N_R^c)$, όπως περιγράφεται στον Ορισμό 13 (Κεφάλαιο 5). Για παράδειγμα, για να αποφασιστεί αν το S_R μπορεί να υποκαταστήσει το S_P θα πρέπει να εξεταστεί αν $OG(N_P^c) \sqsubseteq OG(N_R^c)$ καθώς από αυτό προκύπτει το αν το N_R^c βρίσκεται σε accordance με το N_P^c.

```
1 $ cosme -a po-demoshop.owfn.normalized.bits.og -b ro-demoshop.owfn.normalized.bits.og --simulation -v
2 cosme: Operating guideline A contains 19585 markings
3 cosme: Operating guideline B contains 19841 markings
4 cosme: compared 20161 pairs of states
5 cosme: Simulation: completed
```

Η παράμετρος `-a` του Cosme χρησιμοποιείται για να προσδιορίσει το πρώτο operating guideline μοντέλο το οποίο αντιστοιχεί στο υποτιθέμενο *μικρότερο* OG, ενώ το υποτιθέμενο *μεγαλύτερο* OG προσδιορίζεται μέσω της παραμέτρου `-b`. Η παράμετρος `--simulation` προσδιορίζει ότι η εργασία που θα εκτελεστεί είναι αυτή της *προσομοίωσης* (*simulation*) μεταξύ των OG μοντέλων, δηλαδή, αν ισχύει η σχέση εξειδίκευσης του Ορισμού 13. Τέλος, με την ένδειξη `-v` εμφανίζονται αναλυτικά δεδομένα εκτέλεσης. Το αποτέλεσμα της εκτέλεσης του Cosme υποδηλώνει μια θετική απάντηση στο ερώτημα της υποκαταστασιμότητας, καθώς η εργασία προσομοίωσης είναι επιτυχής (**`completed`**) (γραμμή 5) (μια αρνητική απάντηση θα αποτυπώνονταν ως `failed` προσομοίωση). Συνεπώς η υπηρεσία *RO-DemoShop* μπορεί να υποκαταστήσει την υπηρεσία *PO-DemoShop*. Επιπλέον, για να αποφασιστεί η ευθυγραμμίας θα πρέπει να εξεταστεί η υποκαταστασιμότητα και προς την αντίθετη κατεύθυνση, δηλαδή κατά πόσο το S_P μπορεί να υποκαταστήσει το S_R, με παρόμοιο τρόπο. Επίσης, το εργαλείο Cosme παρέχει την επιλογή `--equivalence` η οποία μπορεί να χρησιμοποιηθεί για να εξεταστεί απευθείας η ευθυγραμμία.

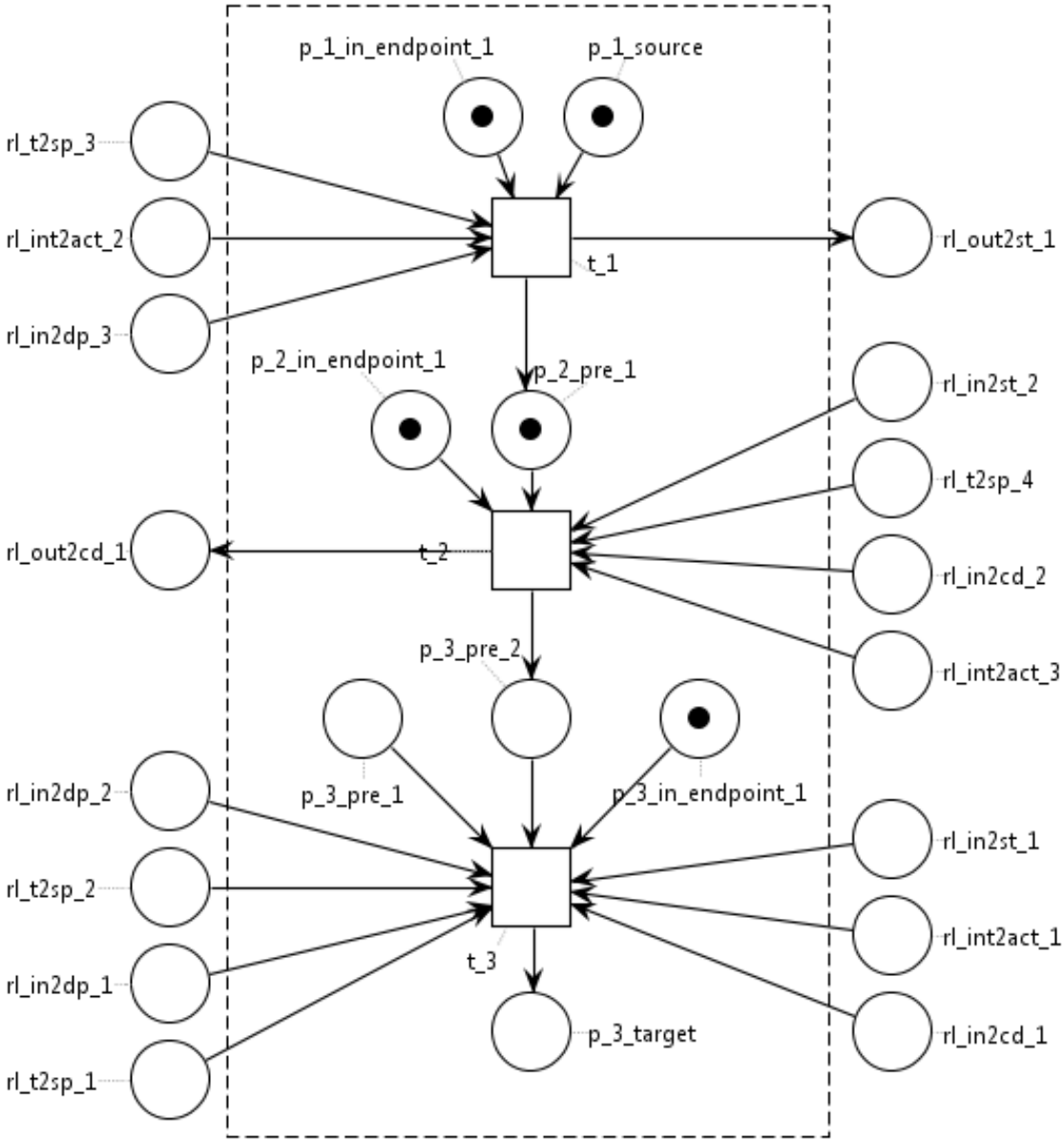
Απουσία ευθυγραμμίας

Στην υποενότητα αυτή, συζητάμε και παρουσιάζουμε πως το προτεινόμενο περιβάλλον-πλαίσιο μπορεί να χρησιμοποιηθεί για να διαπιστωθεί η απουσία ευθυγραμμίας μεταξύ μιας ΠΣΔ και μιας ΠΣΠ διεπαφής εξαιτίας δύο τύπων αναντιστοιχιών: αναντιστοιχία στοιχείων διεπαφής (ή διαφοροποίηση διεπαφών open net) και συμπεριφορική αναντιστοιχία.

Αναντιστοιχία στοιχείων διεπαφής: αυτός ο τύπος αναντιστοιχίας αναφέρεται στην ύπαρξη στοιχείων δεδομένων στη μια διεπαφή του δικτύου, τα οποία δεν μπορούν να συσχετιστούν με στοιχεία του άλλου δικτύου. Μια τέτοια περίπτωση μπορεί να εξεταστεί εύκολα χρησιμοποιώντας την έννοια της διαφοροποίησης διεπαφών στα open net. Όπως συζητήθηκε ήδη, η ισοδυναμία διεπαφών ελέγχεται συγκρίνοντας το I_{N_P} με το I_{N_R} και το O_{N_P} με το O_{N_R}. Στο πρωτότυπό μας, πριν τον έλεγχο του accordance μέσω των εξωτερικών εργαλείων, ελέγχουν την ισοδυναμία των διεπαφών των δικτύων $IEQ_{S_P,S_R} = I_{N_P} \equiv I_{N_R} \wedge O_{N_P} \equiv O_{N_R}$.

Συμπεριφορική αναντιστοιχία: αυτός ο τύπος αναντιστοιχίας αναφέρεται στις διαφορές το αναμενόμενο πρωτόκολλο ή αλληλουχία αλληλεπιδράσεων με την υπηρεσία. Παρουσιάζουμε μια τέτοια περίπτωση κάνοντας μια τροποποίηση στο μοντέλο της υπηρεσίας *PO-DemoShop* που άρει την ευθυγραμμία. Συγκεκριμένα, το εσωτερικό πρωτόκολλο για το S_P προσδιορίζεται από το ακολουθιακό διάγραμμα του Σχήματος 6.5 τροποποιείται έτσι ώστε οι λειτουργίες της υπηρεσίας θα πρέπει να εκτελούνται ακολουθιακά υποχρεωτικά: `searchCatalog` → `createOrder` → `addItem`. Το τροποποιημένο εσωτερικό πρωτόκολλο της υπηρεσίας αποτυπώνεται στο M_P μέσω του τροποποιημένου συνόλου `ProceduralTransition` στοιχείων. Συγκεκριμένα, το τροποποιημένο M_P περιέχει δύο `ProceduralTransition` στοιχεία, $pt_1 = \{pi_1, pi_2\}$ ανδ $pt_2 = \{pi_2, pi_3\}$. Εφαρμόζοντας τη διαδικασία μετασχηματισμού που απεικονίζεται στο Σχήμα 5.2 δημιουργούνται δύο open net, N'_P και N'_R, ένα για την υπηρεσία *PO-DemoShop* και ένα για την υπηρεσία *RO-DemoShop*, αντίστοιχα. Το N'_R ταυτίζεται με το N_R το οποίο απεικονίζεται στο Σχήμα 6.11, καθώς δεν τροποποιούνται τα M_R και M_A μοντέλα. Αντίθετα, το N'_P, το οποίο απεικονίζεται στο Σχήμα 6.12 διαφέρει από το N_P (Σχήμα 6.10) εξαιτίας του ενημερωμένου εσωτερικού πρωτοκόλλου υπηρεσίας στο M_P.

Όμοια με παραπάνω, εκτελούνται τα εργαλεία PETRI και WENDY για να υπολο-



Σχήμα 6.12: N'_P: Το open net που δημιουργήθηκε για το τροποποιημένο M_P μοντέλο της υπηρεσίας *PO-DemoShop*

γιστεί το $OG(N'_P)$. Στη συνέχεια, χρησιμοποιώντας το Cosme, εξετάζεται το $OG(N'_P)$ ως προς το αν υπάρχει σχέση εξειδίκευσης με το $OG(N_R)$.

```
1 $ cosme -a ro-demoshop.owfn.normalized.bits.og -b po-demoshop.owfn.normalized.bits.og --simulation -v
2 cosme: Operating guideline A contains 19585 markings
3 cosme: Operating guideline B contains 18689 markings
4 cosme: bit compare failed in pair A(2068), B(2276)
5 cosme: compared 127 pairs of states
6 cosme: Simulation: failed
```

Το παραπάνω αποτέλεσμα υποδηλώνει ότι το N_P δεν είναι σε accordance με το N_R καθώς υπάρχουν partner του N_R τα οποία δεν είναι partner του N_P. Συνεπώς, η τροποποιημένη ΠΣΔ διεπαφή του S_P δε βρίσκεται σε ευθυγραμμίας με την ΠΣΠ διεπαφή του S_R.

ΚΕΦΑΛΑΙΟ 7

ΣΥΜΠΕΡΑΣΜΑΤΑ

Οι αρχιτεκτονικές ΠΣΠ και οι αρχιτεκτονικές ΠΣΔ αντιμετωπίζονται παραδοσιακά από την ερευνητική κοινότητα ως δύο ανεξάρτητα αρχιτεκτονικά στυλ στην υπηρεσιοστρεφή υπολογιστική. Όσο οι υπηρεσίες προσανατολισμένες διαδικασίες κυριάρχησαν το πεδίο της υπηρεσιοστρεφούς υπολογιστικής, οι αρχιτεκτονικές προσανατολισμένες σε πόρους εστίαζαν ορχικά κυρίως σε μικρού εύρους εφαρμογές Παγκόσμιου Ιστού. Ωστόσο, καθώς οι ερευνητές και οι επαγγελματίες της περιοχής αναγνώρισαν την ανάγκη να χρησιμοποιηθεί το ίδιο το αρχιτεκτονικό στυλ του Παγκόσμιου Ιστού στην υπηρεσιοστρεφή υπολογιστική οι αρχιτεκτονικές ΠΣΠ και ιδιαιτέρως το REST αρχιτεκτονικό στυλ έχει κερδίσει σημαντική προσοχή και αποδοχή μεταξύ τόσο των παρόχων υπηρεσιών όσο και των καταναλωτών υπηρεσιών. Δεν αποτελεί έκπληξη ότι πολλοί πάροχοι υπηρεσιών προσφέρουν και συντηρούν πλέον δυικά API υπηρεσιών για τους πελάτες τους. Στην παρούσα διατριβή εστιάζουμε σε δύο ερευνητικές προκλήσεις οι οποίες ανέκυψαν εξαιτίας της απαίτησης για και της συνύπαρξη των ΠΣΔ και ΠΣΠ σχεδιαστικών υποδειγμάτων διεπαφών σε SOA οικοσυστήματα.

Η πρώτη πρόκληση πηγάζει από την απαίτηση για εναλλακτικές εκθέσεις της λειτουργικότητας των υπηρεσιών και των δεδομένων τους (π.χ. ένα REST API αντί μιας WSA διεπαφής υπηρεσίας), οι οποίες τείνουν να αυξάνουν το κόστος και τα ρίσκα που συνδέονται με την ανάγκη για την εν νέου υλοποίηση και συντήρηση δυικών υλοποιήσεων των δυνατοτήτων των υπηρεσιών. Υπό αυτή την άποψη, οι προσαρμογή υπηρεσιών και συγκεκριμένα η προσαρμογή υποδειγμάτων διεπαφών, ανέκυψε ως μια μεθοδολογία που επιτρέπει διαφορετικά σχεδιαστικά υποδείγματα διεπαφών υπηρεσιών να προσφέρονται χρησιμοποιώντας μια μοναδική υποκείμενη υλοποίηση υπηρεσίας. Η έκθεση υλοποιήσεων υπηρεσιών ΠΣΔ μέσω διεπαφών ΠΣΠ απαιτεί την κατεύθυνση προσαρμογής υποδείγματος διεπαφών ΠΣΔ-προς-ΠΣΠ (procedure-oriented-to-resource-oriented (P2R)). Η προσαρμογή αυτή περιλαμβάνει την αναγνώριση ενός μοντέλου πόρων και τον προσδιορισμό δια-υποδειγματικών συσχετίσεων των διεπαφών. Στις περισσότερες περιπτώσεις, οι εργασίες αυτές ενέχουν σημαντική ιδιόχειρη προσπάθεια και απαιτούν εξειδικευμένα έγγραφα, προδιαγραφές και μοντέλα. Υπό αυτή την άποψη, μια πρόκληση η οποία διερευνάται στην παρούσα διατριβή είναι η αυτοματοποιημένη αναγνώριση P2R λογικής προσαρμογής, θεωρώντας ως αποκλειστικά δεδομένα εισόδου,

προδιαγραφές διεπαφών, επεξεργάσιμες από μηχανές (π.χ. στη μορφή WSDL εγγράφων).

Η δεύτερη πρόκληση πηγάζει από την πρακτική πολλών παρόχων υπηρεσιών οι οποίοι τείνουν να προσφέρουν διαφορετικές εκθέσεις (τόσο ως RESTful υπηρεσίες όσο και ως παραδοσιακά Web services) των ίδιων, ή παρόμοιων, δυνατοτήτων υπηρεσιών. Το γεγονός αυτό εμφανίζεται είτε εξαιτίας διαφορετικών υλοποιήσεων, είτε εξαιτίας δραστηριοτήτων προσαρμογής που προηγούνται χρονικά. Στο πλαίσιο αυτό, υπάρχει μια ανάγκη για μια μέθοδο που να επιτρέπει την αποτίμηση του αν οι διαφορετικές εκθέσεις δυνατοτήτων υπηρεσιών που προσφέρονται από τους παρόχους υπηρεσιών, εκτίθενται με συνεπή και λειτουργικά ισοδύναμο τρόπο έτσι ώστε να μπορούν να θεωρούνται εναλλακτικές ως προς την οπτική του πελάτη. Όταν δύο τέτοιες υπηρεσίες μπορούν χρησιμοποιηθούν εναλλακτικά λέμε ότι βρίσκονται σε *ευθυγραμμία*. Υπό αυτή την έννοια, μια ακόμη πρόκληση που διερευνάται στην παρούσα διατριβή είναι η δυνατότητα να αποτιμάται η υποκαταστασιμότητα μεταξύ υπηρεσιών που εκτίθενται μέσω διαφορετικών υποδειγμάτων υπηρεσιών.

Στο τρέχον κεφάλαιο, συνοψίζουμε τις συνεισφορές της παρούσας διατριβής οι οποίες εισήχθησαν στο Κεφάλαιο 1 και συσχετίζουμε τη συζήτηση με τα ερευνητικά ερωτήματα (E1-E6) τα οποία παρουσιάστηκαν στην Ενότητα 1.2. Επιπλέον, εξετάζουμε ένα σύνολο ανοιχτών θεμάτων και ερευνητικών κατευθύνσεων για το μέλλον, ως προς με κάθε πρόκληση.

7.1 Περίληψη συνεισφορών

Ζητήματα προσαρμογής υποδειγμάτων διεπαφών και συμμόρφωση των API στο REST

Σχετικά με τα ερευνητικά ερωτήματα E1 και E2 που παρουσιάστηκαν στο Κεφάλαιο 1, στην παρούσα διατριβή διερευνούμε τις διαστάσεις σχετικές με τις απαιτήσεις, τη σχεδίαση, τη μοντελοποίηση και την αποτίμηση του προβλήματος της P2R προσαρμογής. Συγκεκριμένα, στο Κεφάλαιο 3 εξετάζονται οι λειτουργικές και μη λειτουργικές απαιτήσεις της P2R προσαρμογής και προτείνεται ένα υψηλού αφαιρετικού επιπέδου μοντέλο διαδικασίας προσαρμογής το οποίο αναλύει και αποσυνθέτει την P2R πρόκληση προσαρμογής σε μικρότερες, περισσότερο διαχει-

ρίσιμες εργασίας. Τελικά, στο Κεφάλαιο 3 παρουσιάζονται περιβάλλοντα-πλαίσια για την αξιολόγηση και αποτίμηση του αποτελέσματος των προσεγγίσεων P2R προσαρμογής υποδείγματος. Στο πλαίσιο αυτό, παρουσιάστηκε το Uniform Interface Constraint Framework το οποίο παρέχει μια συστηματική προσέγγιση ως προς την αποτίμηση της συμμόρφωσης μιας ΠΣΠ διεπαφές στον περιορισμό Ομοιόμορφης διεπαφής του REST.

Εξαγωγή API προσανατολισμένου σε πόρους

Σχετικά με τα ερευνητικά ερωτήματα Ε3 και Ε4 που παρουσιάστηκαν στο Κεφάλαιο 1, η πρόκληση της P2R προσαρμογής διερευνήθηκε περαιτέρω με τον ορισμό και την ανάπτυξη ενός περιβάλλοντος πλαισίου P2R προσαρμογής το οποίο αυτοματοποίησε πυρηνικές δραστηριότητες αναγνώρισης λογικής προσαρμογής και παρείχε μια υλοποίηση η οποία μπορεί να χρησιμοποιηθεί για την προσαρμογή κατά τον χρόνο εκτέλεσης. Συγκεκριμένα, στο Κεφάλαιο 4 προτείνουμε μια προσέγγιση η οποία χρησιμοποιεί τεχνικές εξαγωγής πληροφορίας και μετασχηματισμού μοντέλων, για να αυτοματοποιήσει τη διαδικασία *εξόρυξης μοντέλου πόρων*. Επιπλέον, προτείναμε τεχνικές για την *εξαγωγή αντιστοιχιών διεπαφών*, οι οποίες συσχετίζουν στοιχεία των ΠΣΔ διεπαφών οι οποίες δίνονται ως δεδομένα εισόδου, με στοιχεία της ΠΣΠ διεπαφής που αναγνωρίζεται.

Η εξαγωγή των μοντέλων πόρων REST υλοποιήθηκε χρησιμοποιώντας δύο φάσης εξαγωγής. Η πρώτη φάση είναι η διαδικασία *εξαγωγής πόρων*, η οποία επιστρέφει μια ιεραρχία τύπων πόρων. Η δεύτερη φάση είναι η διαδικασία *εξαγωγής αναπαραστάσεων* η οποία αναγνωρίζει ορισμούς δομών αναπαραστάσεων βασισμένη στις λειτουργίες που σχετίζονται με κάθε πόρο.

Πιο συγκεκριμένα, η φάση εξαγωγής πόρων εφαρμόζεται σε πέντε βήματα. Αρχικά, η ΠΣΠ διεπαφή παρέχεται ως ένα WSDL έγγραφο το οποίο μοντελοποιείται ως ένα σύνολο Μοντέλων Υπογραφών που περιλαμβάνουν πληροφορίες σχετικές με τις υπογραφές των λειτουργιών και αποτελούν μια μέθοδο αφαίρεσης από συγκεκριμένους μηχανισμούς προδιαγραφής διεπαφών. Υπό αυτή την έννοια, η προσέγγιση μπορεί να υποστηρίξει επιπλέον γλώσσες περιγραφής διεπαφών, υπό την προϋπόθεση ότι χρησιμοποιούνται κατάλληλοι μεταφραστές της κάθε IDL γλώσσας. Δεύτερον, τα ονόματα των λειτουργιών αναλύονται χρησιμοποιώντας τεχνικές NLP έτσι ώστε να αναγνωριστούν λεκτικές μονάδες (token) και να επισημειωθούν

με Part-Of-Speech (POS) ετικέτες, αντιμετωπίζοντας τα ως τμήματα μιας φράσης. Χρησιμοποιώντας στα POS tags και στην αλληλουχία τους, αναγνωρίζεται ένα μοντέλο διαφορετικών τύπων όρων και σχέσεων όρων για κάθε λειτουργία, το οποίο αναφέρεται ως Operation Terms Model (OTM). Έπειτα, τα OTM ενοποιούνται σε ένα μοντέλο στο επίπεδο της υπηρεσίας, στο οποίο αναφερόμαστε ως Service Terms Model (STM). Τρίτον, οι λειτουργίες της υπηρεσίας συσχετίζονται με ένα σύνολο προκαθορισμένων κατηγοριών σκοπού οι οποίες υποδηλώνουν τη σημασιολογία της πρόθεσης για αλληλεπίδραση (π.χ. δημιουργία, διαγραφή, κλπ.), έτσι ώστε να δοθεί η δυνατότητα να συσχετιστούν πράξεις χειρισμού σε τύπους πόρων και να δημιουργηθούν ευρετικοί κανόνες μετασχηματισμού μοντέλων οι οποίοι χρησιμοποιούνται σε επόμενα βήματα. Τέταρτον, το STM μοντέλο αναλύεται έτσι ώστε να αναγνωριστούν οι πυρηνικές εννοιολογικές οντότητες (Core Conceptual Entitites) της υπηρεσίας. Τελικά, χρησιμοποιώντας τις οντότητες αυτές, εφαρμόζεται ένα σύνολο κανόνων αναγνώρισης τύπων πόρων έτσι ώστε να παραχθεί μια ιεραρχία τύπων πόρων στην οποία αναφερόμαστε ως Resource Types Model (RTM). Το RTM θεωρείται ως σκελετός του ΠΣΠ API καθώς παρέχει τους διάφορους τύπους πόρων, όπως και υπαρξιακές εξαρτήσεις μεταξύ τους.

Στη συνέχεια, για τον υπολογισμό αναπαραστάσεων πόρων και αντιστοιχίσεων διεπαφών, η παρούσα διατριβή παρέχει μια μέθοδο αναγνώρισης μιας συλλογής σχέσεων θεμελιώδους ανταπόκρισης (base correspondence) μεταξύ τύπων πόρων που περιλαμβάνονται σε ένα Resource Types Model και λειτουργιών που περιλαμβάνονται σε μια ΠΣΔ διεπαφή (π.χ. ένα έγγραφο WSDL το οποίο μοντελοποιείται από ένα σύνολο Μοντέλων Υπογραφών). Χρησιμοποιώντας το υποσύνολο των σχέσεων θεμελιώδους ανταπόκρισης στις οποίες συμμετέχει κάθε τύπος πόρου, υπολογίζονται αναπαραστάσεις χρησιμοποιώντας τις παραμέτρους των αντίστοιχων υπογραφών λειτουργιών.

Η εξαγωγή αντιστοιχιών διεπαφών υλοποιείται στο πλαίσιο ενός συνόλου σχέσεων θεμελιώδους ανταπόκρισης οι οποίες έχουν αναγνωριστεί. Συγκεκριμένα, η τεχνική που προτείνεται αναγνωρίζει *α*′) αντιστοιχίες σκοπού, οι οποίες συσχετίζουν λειτουργίες με κανονικοποιημένες πράξεις χειρισμού, *β*′) αντιστοιχίες τελικού σημείου, οι οποίες συσχετίζουν ΠΣΔ σημεία αλληλεπίδρασης με ΠΣΠ σημεία αλληλεπίδρασης, *γ*′) αντιστοιχίες αναγνωριστικού, οι οποίες συσχετίζουν στοιχεία αναγνώρισης που περιέχονται στα δεδομένα εισόδου λειτουργιών με δυναμικά τμήματα προτύπων αναγνωριστικών πόρων και *δ*′) αντιστοιχίες μηνυμάτων, οι οποίες συσχε-

τίζουν στοιχεία εισόδου και εξόδου υπηρεσιών με αναπαραστάσεις πόρων. Επίσης, στο πλαίσιο του περιβάλλοντος που προτείνεται, υλοποιήσαμε ένα δομοστοιχείο-προσαρμογέα χρόνου εκτέλεσης και το ενσωματώσαμε στο Apache Tuscany SCA runtime. Υπό αυτή την άποψη, ΠΣΔ δομοστοιχεία μπορούν να παρέχουν ένα REST binding, δεδομένης μιας οδηγίας προσαρμογής, η οποία μπορεί να δημιουργηθεί μέσω της προσέγγισης προσαρμογής που προτείνεται.

Η εφαρμογή της προσέγγισης προσαρμογής δείχνεται μέσω ενός συνοδευτικού παραδείγματος μιας υπηρεσίας διαχείρισης παραγγελιών (SimpleOMS) και εξετάζεται περαιτέρω μέσω μιας απ' άκρο εις άκρο περιπτωσιολογικής μελέτης μιας δημοφιλούς υπηρεσίας στο διαδίκτυο (Amazon S3) στο Κεφάλαιο 6. Επιπλέον, στο Κεφάλαιο 6 αποτιμήσαμε το αποτέλεσμα της εξαγωγής πόρων, εφαρμόζοντας την προσέγγιση σε ένα ευρύ σύνολο υπηρεσιών που αντλήθηκε από ένα ανοιχτό αποθετήριο υπηρεσιών. Πιο συγκεκριμένα, αξιολογήσαμε την ακρίβεια της προσέγγισης και λάβαμε θετικά αποτελέσματα, χρησιμοποιώντας τόσο τυπικές μετρικές Ανάκτησης Πληροφορίας, όπως και μια μετρική ομοιότητας η οποία υπολογίζεται χρησιμοποιώντας την minimum weighted maximum graph matching (MWMGM) οντολογική απόσταση. Επιπρόσθετα, εξετάσαμε επίσης την επίδραση που έχει η διαδικασία προσαρμογής στην παραγωγικότητα ενός προγραμματιστή, όταν η προτεινόμενη διαδικασία εξαγωγής πόρων ενσωματώνεται σε ένα περιβάλλον ανάπτυξης. Τα αποτελέσματα δείξανε ότι μειώνει σημαντικά τόσο την προσπάθεια όσο και τον χρόνο που απαιτείται.

Υποκαταστασιμότητα υπηρεσιών διαφορετικών υποδειγμάτων και αποτίμηση ευθυγραμμίας

Σχετικά με τα ερευνητικά ερωτήματα Ε5 και Ε6 τα οποία παρουσιάστηκαν στο Κεφάλαιο 1, στην παρούσα διατριβή στοχεύσαμε στην αντιμετώπιση δύο βασικών ζητημάτων που αφορούν στην πρόκληση αποτίμησης της δια-υποδειγματικής υποκαταστασιμότητας και ευθυγραμμίας υπηρεσιών.

Το πρώτο ζήτημα αφορά στη διερεύνηση των εννοιολογικών αντιστοιχιών μεταξύ των προσανατολισμένων σε πόρων και των προσανατολισμένων σε διαδικασίες αρχιτεκτονικών στυλ. Αυτού του είδους οι εννοιολογικές αντιστοιχίες αναφέρονται στις θεωρητικές αρχές της Τεχνολογίας Λογισμικού, αλλά παρ' όλα αυτά έχουν πρακτικές συνέπειες αφενός στη σχεδίαση ΠΣΠ API από ΠΣΔ διεπαφές και αντίστροφα,

και αφετέρου στην ευθυγραμμία, συντήρηση και παράλληλη εξέλιξη υπηρεσιών που παρέχονται από δυικά API. Υπό αυτή την άποψη, προτείνουμε εννοιολογικά μετα-μοντέλα για την μοντελοποίηση ΠΣΔ και ΠΣΠ API και ορίσαμε συσχετιστικές αντιστοιχίες μεταξύ των δύο αυτών εννοιολογικών μετα-μοντέλων. Το πεδίο της εργασίας μας είναι αυτό των Web Services και των RESTful HTTP services, ως ενδεικτικά παραδείγματα συμπαγών αρχιτεκτονικών υπηρεσιοστρεφούς υπολογιστικής, η ανάλυση των οποίων μας βοήθησε στη σχεδίαση και την ανάπτυξη των αντίστοιχων αφηρημένων μεταμοντέλων υπηρεσιών.

Το δεύτερο ζήτημα αφορά στη σχεδίαση και την ανάπτυξη ενός πρακτικού περιβάλλοντος-πλαισίου το οποίο επιτρέπει την αποτίμηση συμπεριφορικής ισοδυναμίας μεταξύ ενός ΠΣΠ API (π.χ. SOAP Web service) και ένα ΠΣΠ REST API, δεδομένου ενός συνόλου συσχετίσεων μεταξύ των διεπαφών των υπηρεσιών. Στο πλαίσιο αυτό, προτείναμε μια προσέγγιση μετασχηματισμού μοντέλων η οποία επιτρέπει το σταδιακό μετασχηματισμό μοντέλων API υπηρεσιών σε open nets και την επακόλουθη χρήση της θεωρίας της *συμφωνίας* (*accordance*) για την αποτίμηση του αν δύο open nets που αντιστοιχούν στις δύο υπηρεσίες βρίσκονται σε συμφωνία, δηλαδή η μια υπηρεσία μπορεί να εξυπηρετήσει τουλάχιστον όλους τους δυνατούς πελάτες της άλλης.

Η προσέγγιση που προτείνεται επιδείχθηκε μέσω ενός συνοδευτικού παραδείγματος στο Κεφάλαιο 5 και αξιολογήθηκε μέσω μιας περιπτωσιολογικής μελέτης στο Κεφάλαιο 6 στην οποία μια υπηρεσία ΠΣΔ και μια υπηρεσία ΠΣΠ με σχετικές δυνατότητες υπηρεσίας ενός online ηλεκτρονικού καταστήματος δειγμάτων, χρησιμοποιήθηκαν για να εξεταστεί η μεταξύ τους υποκαταστασιμότητα και ως εκ τούτου και η ευθυγραμμία. Από τη μελέτη της περιοχής διαπιστώσαμε το πρόβλημα της δια-υποδειγματικής υποκαταστασιμότητας υπηρεσιών δεν έχει μελετηθεί σε βάθος από την ερευνητική κοινότητα και ότι η παρούσα διατριβή επεκτείνει τα όρια της περιοχής αυτής στοχεύοντας στο να ρίξει φως στην ευθυγραμμία και την αντιστοιχία των αρχιτεκτονικών προσανατολισμένων σε διαδικασίες και των αρχιτεκτονικών προσανατολισμένων σε πόρους.

7.2 Μελλοντικές ερευνητικές κατευθύνσεις

Η παρούσα διατριβή οδηγεί σε ένα σύνολο δυνατών μελλοντικών ερευνητικών κατευθύνσεων. Πρώτον, οι ερευνητικές κατευθύνσεις σχετικές με το περιβάλλον-πλαίσιο P2R προσαρμογής που προτάθηκε περιλαμβάνουν την ενσωμάτωση τεχνικών μηχανικής μάθησης για την εξόρυξη ειδικών ως προς το πεδίο μελέτης μοντέλων λειτουργιών, αντί της χρησιμοποίησης των υπαρχόντων τεχνικών επισημείωσης POS και τους κανόνες μετάφρασης. Επιπλέον, μπορεί να αναβαθμιστεί ο ρόλος των παραμέτρων εισόδου και εξόδου κατά τη διάρκεια της φάσης της προεπεξεργαστικής ανάλυσης, έτσι ώστε να παραχθούν πιο λεπτομερή μοντέλα για την αναγνώριση οντοτήτων και, τελικά, πόρων. Μια άλλη κατεύθυνση στην περιοχή της προσαρμογής υποδειγμάτων διεπαφών υπηρεσιών είναι η υλοποίηση τεχνικών προσαρμογής μιας-προς-πολλές αλληλεπιδράσεις, στις οποίες τεχνικές ο προσαρμογέας μεσολαβεί στην επικοινωνία πελάτη-εξυπηρετητή με τρόπο ο οποίος χαρακτηρίζεται από καταστάσεις, έτσι ώστε να αντιστοιχίζει πολλαπλές αλληλεπιδράσεις προσανατολισμένες σε πόρους σε μοναδικές αλληλεπιδράσεις προσανατολισμένες σε διαδικασίες και αντίστροφα. Μέχρι σήμερα υπάρχουν περιορισμένες συνεισφορές στην κατεύθυνση μιας προσέγγισης P2R προσαρμογής η οποία θα μπορούσε να αντιμετωπίσει μια τέτοια απαίτηση προσαρμογής (μια-προς-πολλές αλληλεπιδράσεις).

Δεύτερον, σχετικά με το περιβάλλον-πλαίσιο αποτίμησης δια-υποδειγματικής υποκαταστασιμότητας, οι μελλοντικές ερευνητικές κατευθύνσεις περιλαμβάνουν το δυναμικό χειρισμό των συνθηκών στις μεταβάσεις μεταξύ πελάτη-εξυπηρετητή και στους ελέγχους υπερμέσων. Συνεπώς, η διαδικασία παραγωγής των open nets θα μπορούσε να επεκταθεί ώστε να περιλαμβάνει κανόνες οι οποίοι αντιστοιχούν τις συνθήκες σε στοιχεία Petri net, τα οποία αποτυπώνουν τόσο τη ροή όσο και τα στοιχεία δεδομένων τα οποία σχετίζονται με τις συνθήκες αυτές. Υπό αυτή την έννοια, οι συνθήκες μπορούν να αποτιμηθούν ως τμήμα του ίδιου του Petri net μοντέλου, εξαλείφοντας την ανάγκη για στην αποτίμησή τους προκαταβολικά και απογραμμικά (offline). Επιπλέον, μια άλλη ενδιαφέρουσα ερευνητική κατεύθυνση είναι η επέκταση της αποτίμησης υποκαταστασιμότητας και ευθυγραμμίας ώστε να καλύπτει μη λειτουργικές διαστάσεις των υπηρεσιών. Σε ένα τέτοιο πλαίσιο, τα εννοιολογικά μοντέλα πρέπει να επεκταθούν ώστε να αποτυπώνουν μελήματα σχετικά με την ποιότητα ενώ ενδεχομένως να απαιτηθούν περαιτέρω τεχνικές συλλογιστικής

για την εξέταση της ευθυγραμμίας μεταξύ των υπηρεσιών. Τελικά, το μεταμοντέλο αντιστοιχίσεων υπηρεσιών θα μπορούσε να χρησιμοποιηθεί για την αναγνώριση κοινών δομών υψηλού αφαιρετικού επιπέδου για τη μοντελοποίηση υπηρεσιών, οι οποίες θα μπορούσαν τελικά να χρησιμοποιηθούν για τον ορισμό ενός ενοποιημένου *μουτέλου δυνατοτήτων υπηρεσιών* (*service capability model*). Ένα τέτοιο μοντέλο δυνατοτήτων υπηρεσιών θα μπορούσε να παρέχει τις απαραίτητες αφαιρέσεις που καταστούν τη συσχέτιση των υποδειγμάτων διεπαφών υπηρεσιών ως μια περισσότερο ρητή και σαφής διαδικασία. Επιπρόσθετα, ένα τέτοιο μοντέλο δυνατοτήτων θα μπορούσε να παρέχει τη βάση για τη σύλληψη μιας ενοποιημένης προσέγγισης στη σχεδίαση και στην πραγμάτωση συστημάτων υπηρεσιών, διαχωρίζοντας τους σχετικούς με τη σχεδίαση διεπαφών περιορισμούς, από τη μοντελοποίηση δυνατοτήτων, επιτρέποντας έτσι πολλαπλά υποδείγματα διεπαφών να υποστηρίζονται με ένα ευέλικτο και διαφανή ως προς την υλοποίηση, τρόπο.

ΠΑΡΑΡΤΗΜΑ Α'

SIMPLEOMS WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="SimpleOMSImplService" targetNamespace="http://simpleoms.sample.softeng.ece.ntua.gr/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://simpleoms.sample.softeng.ece.ntua.gr/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <wsdl:types>
<xs:schema xmlns:tns="http://simpleoms.sample.softeng.ece.ntua.gr/" xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://simpleoms.sample.softeng.ece.ntua.gr/" version="1.0">
  <xs:element name="addOrderItem" type="tns:addOrderItem"/>
  <xs:element name="addOrderItemResponse" type="tns:addOrderItemResponse"/>
  <xs:element name="checkout" type="tns:checkout"/>
  <xs:element name="checkoutResponse" type="tns:checkoutResponse"/>
  <xs:element name="createOrder" type="tns:createOrder"/>
  <xs:element name="createOrderResponse" type="tns:createOrderResponse"/>
  <xs:element name="getOrder" type="tns:getOrder"/>
  <xs:element name="getOrderItemShippingStatus" type="tns:getOrderItemShippingStatus"/>
  <xs:element name="getOrderItemShippingStatusResponse" type="tns:getOrderItemShippingStatusResponse"/>
  <xs:element name="getOrderResponse" type="tns:getOrderResponse"/>
  <xs:element name="getOrderShippingStatus" type="tns:getOrderShippingStatus"/>
  <xs:element name="getOrderShippingStatusResponse" type="tns:getOrderShippingStatusResponse"/>
  <xs:element name="getSubmittedOrders" type="tns:getSubmittedOrders"/>
  <xs:element name="getSubmittedOrdersResponse" type="tns:getSubmittedOrdersResponse"/>
  <xs:element name="removeOrder" type="tns:removeOrder"/>
  <xs:element name="removeOrderItem" type="tns:removeOrderItem"/>
  <xs:element name="removeOrderItemResponse" type="tns:removeOrderItemResponse"/>
  <xs:element name="removeOrderResponse" type="tns:removeOrderResponse"/>
  <xs:element name="searchCatalog" type="tns:searchCatalog"/>
  <xs:element name="searchCatalogResponse" type="tns:searchCatalogResponse"/>
  <xs:element name="submitOrder" type="tns:submitOrder"/>
  <xs:element name="submitOrderResponse" type="tns:submitOrderResponse"/>
  <xs:complexType name="createOrder">
    <xs:sequence>
      <xs:element minOccurs="1" name="auth" type="tns:Auth" maxOccurs="1"/>
      <xs:element minOccurs="1" name="order" type="tns:Order" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Auth">
    <xs:sequence/>
  </xs:complexType>
  <xs:complexType name="Order">
    <xs:sequence>
      <xs:element minOccurs="0" name="orderId" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="createOrderResponse">
    <xs:sequence>
      <xs:element minOccurs="1" name="mitsos" type="xs:string" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="getOrderItemShippingStatus">
    <xs:sequence>
      <xs:element minOccurs="1" name="auth" type="tns:Auth" maxOccurs="1"/>
      <xs:element minOccurs="1" name="orderId" type="xs:string" maxOccurs="1"/>
      <xs:element minOccurs="1" name="orderItemId" type="xs:string" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="getOrderItemShippingStatusResponse">
    <xs:sequence>
      <xs:element minOccurs="1" name="shippingStatus" type="tns:ShippingStatus" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ShippingStatus">
    <xs:sequence>
      <xs:element minOccurs="0" name="shippingStatus" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="getOrderShippingStatus">
    <xs:sequence>
      <xs:element minOccurs="1" name="auth" type="tns:Auth" maxOccurs="1"/>
      <xs:element minOccurs="1" name="orderId" type="xs:string" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="getOrderShippingStatusResponse">
    <xs:sequence>
      <xs:element minOccurs="1" name="shippingStatus" type="tns:ShippingStatus" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="removeOrder">
    <xs:sequence>
      <xs:element minOccurs="1" name="auth" type="tns:Auth" maxOccurs="1"/>
      <xs:element minOccurs="1" name="orderId" type="xs:string" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="removeOrderResponse">
    <xs:sequence/>
  </xs:complexType>
  <xs:complexType name="getSubmittedOrders">
    <xs:sequence>
      <xs:element minOccurs="1" name="auth" type="tns:Auth" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="getSubmittedOrdersResponse">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="orders" type="tns:Order"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="searchCatalog">
    <xs:sequence>
      <xs:element minOccurs="1" name="auth" type="tns:Auth" maxOccurs="1"/>
      <xs:element minOccurs="0" name="query" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="searchCatalogResponse">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="catalogItems" type="tns:CatalogItem"/>
```

```
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="CatalogItem">
    <xs:sequence/>
  </xs:complexType>
  <xs:complexType name="getOrder">
    <xs:sequence>
      <xs:element minOccurs="1" name="auth" type="tns:Auth" maxOccurs="1"/>
      <xs:element minOccurs="1" name="orderId" type="xs:string" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="getOrderResponse">
    <xs:sequence>
      <xs:element minOccurs="1" name="order" type="tns:Order" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="submitOrder">
    <xs:sequence>
      <xs:element minOccurs="1" name="auth" type="tns:Auth" maxOccurs="1"/>
      <xs:element minOccurs="1" name="orderId" type="xs:string" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="submitOrderResponse">
    <xs:sequence/>
  </xs:complexType>
  <xs:complexType name="addOrderItem">
    <xs:sequence>
        <xs:element minOccurs="1" name="auth" type="tns:Auth" maxOccurs="1"/>
        <xs:element minOccurs="1" name="orderId" type="xs:string" maxOccurs="1"/>
        <xs:element name="item" type="tns:OrderItem" maxOccurs="1" minOccurs="1"></xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="OrderItem">
    <xs:sequence>
      <xs:element minOccurs="0" name="orderItemId" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="addOrderItemResponse">
    <xs:sequence>
      <xs:element minOccurs="1" name="itemId" type="xs:string" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="checkout">
    <xs:sequence>
      <xs:element minOccurs="1" name="orderId" type="xs:string" maxOccurs="1"/>
      <xs:element minOccurs="1" name="payment" type="tns:Payment" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Payment">
    <xs:sequence>
      <xs:element minOccurs="0" name="payment" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="checkoutResponse">
    <xs:sequence/>
  </xs:complexType>
  <xs:complexType name="removeOrderItem">
    <xs:sequence>
        <xs:element minOccurs="1" name="auth" type="tns:auth" maxOccurs="1"/>
        <xs:element minOccurs="1" name="orderId" type="xs:string" maxOccurs="1"/>
        <xs:element name="itemId" type="xs:string" maxOccurs="1" minOccurs="1"></xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="removeOrderItemResponse">
    <xs:sequence>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
  </wsdl:types>
  <wsdl:message name="getOrderShippingStatusResponse">
    <wsdl:part name="parameters" element="tns:getOrderShippingStatusResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="removeOrder">
    <wsdl:part name="parameters" element="tns:removeOrder">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getSubmittedOrders">
    <wsdl:part name="parameters" element="tns:getSubmittedOrders">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getOrder">
    <wsdl:part name="parameters" element="tns:getOrder">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="submitOrder">
    <wsdl:part name="parameters" element="tns:submitOrder">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getOrderResponse">
    <wsdl:part name="parameters" element="tns:getOrderResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getOrderItemShippingStatusResponse">
    <wsdl:part name="parameters" element="tns:getOrderItemShippingStatusResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="checkout">
    <wsdl:part name="parameters" element="tns:checkout">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="removeOrderItem">
    <wsdl:part name="parameters" element="tns:removeOrderItem">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="removeOrderItemResponse">
    <wsdl:part name="parameters" element="tns:removeOrderItemResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="createOrder">
    <wsdl:part name="parameters" element="tns:createOrder">
```

```
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="checkoutResponse">
    <wsdl:part name="parameters" element="tns:checkoutResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getOrderItemShippingStatus">
    <wsdl:part name="parameters" element="tns:getOrderItemShippingStatus">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getOrderShippingStatus">
    <wsdl:part name="parameters" element="tns:getOrderShippingStatus">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="addOrderItemResponse">
    <wsdl:part name="parameters" element="tns:addOrderItemResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="removeOrderResponse">
    <wsdl:part name="parameters" element="tns:removeOrderResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="createOrderResponse">
    <wsdl:part name="parameters" element="tns:createOrderResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="searchCatalog">
    <wsdl:part name="parameters" element="tns:searchCatalog">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="searchCatalogResponse">
    <wsdl:part name="parameters" element="tns:searchCatalogResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getSubmittedOrdersResponse">
    <wsdl:part name="parameters" element="tns:getSubmittedOrdersResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="addOrderItem">
    <wsdl:part name="parameters" element="tns:addOrderItem">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="submitOrderResponse">
    <wsdl:part name="parameters" element="tns:submitOrderResponse">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="SimpleOMS">
    <wsdl:operation name="createOrder">
      <wsdl:input name="createOrder" message="tns:createOrder">
    </wsdl:input>
      <wsdl:output name="createOrderResponse" message="tns:createOrderResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getOrderItemShippingStatus">
      <wsdl:input name="getOrderItemShippingStatus" message="tns:getOrderItemShippingStatus">
    </wsdl:input>
      <wsdl:output name="getOrderItemShippingStatusResponse" message="tns:getOrderItemShippingStatusResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getOrderShippingStatus">
      <wsdl:input name="getOrderShippingStatus" message="tns:getOrderShippingStatus">
    </wsdl:input>
      <wsdl:output name="getOrderShippingStatusResponse" message="tns:getOrderShippingStatusResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="removeOrder">
      <wsdl:input name="removeOrder" message="tns:removeOrder">
    </wsdl:input>
      <wsdl:output name="removeOrderResponse" message="tns:removeOrderResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getSubmittedOrders">
      <wsdl:input name="getSubmittedOrders" message="tns:getSubmittedOrders">
    </wsdl:input>
      <wsdl:output name="getSubmittedOrdersResponse" message="tns:getSubmittedOrdersResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="searchCatalog">
      <wsdl:input name="searchCatalog" message="tns:searchCatalog">
    </wsdl:input>
      <wsdl:output name="searchCatalogResponse" message="tns:searchCatalogResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getOrder">
      <wsdl:input name="getOrder" message="tns:getOrder">
    </wsdl:input>
      <wsdl:output name="getOrderResponse" message="tns:getOrderResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="submitOrder">
      <wsdl:input name="submitOrder" message="tns:submitOrder">
    </wsdl:input>
      <wsdl:output name="submitOrderResponse" message="tns:submitOrderResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="addOrderItem">
      <wsdl:input name="addOrderItem" message="tns:addOrderItem">
    </wsdl:input>
      <wsdl:output name="addOrderItemResponse" message="tns:addOrderItemResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="checkout">
      <wsdl:input name="checkout" message="tns:checkout">
    </wsdl:input>
      <wsdl:output name="checkoutResponse" message="tns:checkoutResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="removeOrderItem">
      <wsdl:input name="removeOrderItem" message="tns:removeOrderItem">
    </wsdl:input>
      <wsdl:output name="removeOrderItemResponse" message="tns:removeOrderItemResponse">
```

```
    </wsdl:output>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="SimpleOMSImplServiceSoapBinding" type="tns:SimpleOMS">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="createOrder">
      <soap:operation soapAction="urn:CreateOrder" style="document"/>
      <wsdl:input name="createOrder">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="createOrderResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getOrderItemShippingStatus">
      <soap:operation soapAction="urn:GetOrderItemShippingStatus" style="document"/>
      <wsdl:input name="getOrderItemShippingStatus">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getOrderItemShippingStatusResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getOrderShippingStatus">
      <soap:operation soapAction="urn:GetOrderShippingStatus" style="document"/>
      <wsdl:input name="getOrderShippingStatus">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getOrderShippingStatusResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="removeOrder">
      <soap:operation soapAction="urn:RemoveOrder" style="document"/>
      <wsdl:input name="removeOrder">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="removeOrderResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getSubmittedOrders">
      <soap:operation soapAction="urn:GetSubmittedOrders" style="document"/>
      <wsdl:input name="getSubmittedOrders">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getSubmittedOrdersResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="searchCatalog">
      <soap:operation soapAction="urn:SearchCatalog" style="document"/>
      <wsdl:input name="searchCatalog">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="searchCatalogResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="getOrder">
      <soap:operation soapAction="urn:GetOrder" style="document"/>
      <wsdl:input name="getOrder">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getOrderResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="submitOrder">
      <soap:operation soapAction="urn:SubmitOrder" style="document"/>
      <wsdl:input name="submitOrder">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="submitOrderResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="addOrderItem">
      <soap:operation soapAction="urn:AddOrderItem" style="document"/>
      <wsdl:input name="addOrderItem">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="addOrderItemResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="checkout">
      <soap:operation soapAction="urn:Checkout" style="document"/>
      <wsdl:input name="checkout">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="checkoutResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="removeOrderItem">
      <soap:operation soapAction="urn:RemoveOrderItem" style="document"/>
      <wsdl:input name="removeOrderItem">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="removeOrderItemResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="SimpleOMSImplService">
    <wsdl:port name="SimpleOMSImplPort" binding="tns:SimpleOMSImplServiceSoapBinding">
      <soap:address location="http://localhost:9090/SimpleOMSImplPort"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

ΠΑΡΑΡΤΗΜΑ Β'

ΜΟΝΤΕΛΑ *DEMOSHOP*

B.1 *DemoShop* M_P

```
1  <?xml version="1.0" encoding="ASCII"?>
2  <poService:POServiceDescription xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
       instance" xmlns:poService="http://softeng.ntua.gr/restadapt/poService" xmlns:term="http://softeng.ntua.gr/restadapt/
       termModel" label="PODemoShop">
3    <operations label="searchCatalog">
4      <signature>
5        <input label="SearchCriteria (searchCatalog)"/>
6        <output label="CatalogItems (searchCatalog)"/>
7        <operationName label="searchCatalog">
8          <termModel>
9            <terms xsi:type="term:Intent" value="search" affects="//@operations.0/@signature/@operationName/@termModel/@terms.1"/>
10           <terms xsi:type="term:Concept" value="catalog"/>
11         </termModel>
12       </operationName>
13     </signature>
14   </operations>
15   <operations label="createOrder">
16     <signature>
17       <input label="Authentication (createOrder)" category="METADATA"/>
18       <input label="Order (createOrder)"/>
19       <output label="OrderID (createOrder)"/>
20       <operationName label="createOrder">
21         <termModel>
22           <terms xsi:type="term:Intent" value="create" affects="//@operations.1/@signature/@operationName/@termModel/@terms.1"/>
23           <terms xsi:type="term:Concept" value="order"/>
24         </termModel>
25       </operationName>
26     </signature>
27   </operations>
28   <operations label="addItem">
29     <signature>
30       <input label="Authentication (addItem)" category="METADATA"/>
31       <input label="OrderID (addItem)"/>
32       <input label="Item (addItem)"/>
33       <operationName label="addItem">
34         <termModel>
35           <terms xsi:type="term:Intent" value="add" affects="//@operations.2/@signature/@operationName/@termModel/@terms.1"/>
36           <terms xsi:type="term:Concept" value="item"/>
37         </termModel>
38       </operationName>
39     </signature>
40   </operations>
41   <endpoint identifier="podemoshop.softeng.ntua.gr"/>
42   <scenarios label="standard">
43     <transitions from="//@interactions.0" to="//@interactions.2" condition="//@scenarios.0/@conditions.0" label="searchCatalog-to
         -addItem"/>
44     <transitions from="//@interactions.1" to="//@interactions.2" condition="//@scenarios.0/@conditions.0" label="createOrder-to-
         addItem"/>
45     <conditions expression="true"/>
46   </scenarios>
47   <interactions operation="//@operations.0" source="//@scenarios.0/@transitions.0" label="searchCatalog"/>
48   <interactions operation="//@operations.1" source="//@scenarios.0/@transitions.1" label="createOrder"/>
49   <interactions operation="//@operations.2" target="//@scenarios.0/@transitions.0 //@scenarios.0/@transitions.1" label="addItem"/
       >
50 </poService:POServiceDescription>
```

B.2 *DemoShop* M_R

```
1  <?xml version="1.0" encoding="ASCII"?>
2  <roService:ROServiceDescription xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
       instance" xmlns:roService="http://softeng.ntua.gr/restadapt/roService" label="RODemoShop">
3    <interactionPoints resource="//@resources.0" label="READ_store" entry="true">
4      <action key="GET (READ_store)" type="ACTION"/>
5      <request contextResourceIdentifier="//@resources.0/@ridt">
6        <controlData key="Host (READ_store)" value="{ServiceDescription/Context/Host/identifier}" type="HOST"/>
7      </request>
8      <response exposedRepresentations="//@resources.0/@representations.0" contextResourceIdentifier="//@resources.0/@ridt"
           affordances="//@hypermedia.0 //@hypermedia.1">
9        <controlData key="WWW-Authenticate (READ_store)" value="Basic" type="AUTHENTICATION_POLICY"/>
10       <controlData key="Response-Code (READ_store)" value="200" type="RESPONSE_CODE"/>
11     </response>
12   </interactionPoints>
13   <interactionPoints resource="//@resources.1" label="READ_products_query">
14     <action key="GET (READ_products_query)" type="ACTION"/>
15     <request contextResourceIdentifier="//@resources.1/@ridt">
16       <controlData key="Host (READ_products_query)" value="{ServiceDescription/Context/Host/identifier}" type="HOST"/>
17     </request>
18     <response exposedRepresentations="//@resources.1/@representations.0" contextResourceIdentifier="//@resources.1/@ridt">
19       <controlData key="Response-Code (READ_products_query)" value="200"/>
20     </response>
21   </interactionPoints>
22   <interactionPoints resource="//@resources.2" label="CREATE_carts">
23     <action key="POST (CREATE_carts)" value="" type="ACTION" actionType="CREATE"/>
```

```
    <request exposedRepresentations="//@resources.2/@representations.0" contextResourceIdentifier="//@resources.2/@ridt">
      <controlData key="Host (CREATE carts)" value="{ServiceDescription/Context/Host/identifier}" type="HOST"/>
      <controlData key="Authentication (CREATE carts)" value="" type="AUTHENTICATION"/>
    </request>
    <response contextResourceIdentifier="//@resources.2/@ridt" exposedResourceMetadata="//@resources.2/@resourceMetadata.0">
      <controlData key="Response-Code (CREATE carts)" value="201" type="RESPONSE_CODE"/>
      <controlData key="Location (CREATE carts)" value="" type="LOCATION"/>
    </response>
  </interactionPoints>
  <interactionPoints resource="//@resources.3" label="CREATE item">
    <action key="PUT (CREATE item)" type="ACTION" actionType="CREATE"/>
    <request exposedRepresentations="//@resources.3/@representations.0" contextResourceIdentifier="//@resources.3/@ridt">
      <controlData key="Host (CREATE item)" value="{ServiceDescription/Context/Host/identifier}" type="HOST"/>
      <controlData key="Authentication (CREATE item)" value="" type="AUTHENTICATION"/>
    </request>
    <response contextResourceIdentifier="//@resources.3/@ridt">
      <controlData key="Response-Code (CREATE item)" value="201" type="RESPONSE_CODE"/>
    </response>
  </interactionPoints>
  <context application="demoshop">
    <host identifier="rodemoshop.softeng.ntua.gr"/>
  </context>
  <resources label="store">
    <ridt context="//@context" label="/store/">
      <pattern label="/store/"/>
    </ridt>
    <representations label="store" direction="SC">
      <representationData>
        <stateData key="status" value="" multiplicity="1"/>
      </representationData>
      <representationMetadata key="Content-Type (filtered products)" value="application/vnd.ntua.store+json" type="MEDIA_TYPE"/>
      <onActionType>READ</onActionType>
    </representations>
    <resourceMetadata key="Allow" value="GET" type="ALLOWS_ACTION"/>
  </resources>
  <resources label="products query">
    <ridt context="//@context" label="/store/products/{?query}">
      <pattern label="/store/products/{?query}">
        <staticPart label="products (/store/products/{?query})" position="1"/>
        <dynamicPart label="query (/store/products/{?query})" position="2" expressionType="QUERY"/>
      </pattern>
    </ridt>
    <representations label="filtered products" direction="CS">
      <representationData>
        <stateData xsi:type="roService:StateElement" key="products" value="" multiplicity="1"/>
      </representationData>
      <representationMetadata key="Content-Type (filtered products)" value="application/vnd.ntua.products+json" type="MEDIA_TYPE"
          />
      <onActionType>READ</onActionType>
    </representations>
    <resourceMetadata key="Allow" value="GET" type="ALLOWS_ACTION"/>
  </resources>
  <resources label="carts">
    <ridt context="//@context" label="/store/carts/">
      <pattern label="/store/carts/">
        <staticPart label="carts (/store/carts/)" position="1"/>
      </pattern>
    </ridt>
    <representations label="cart item">
      <representationData>
        <stateData xsi:type="roService:StateElement" key="cart" value="" multiplicity="1"/>
      </representationData>
      <representationMetadata key="Content-Type (carts)" value="application/vnd.ntua.cart+json" type="MEDIA_TYPE"/>
      <onActionType>CREATE</onActionType>
    </representations>
    <resourceMetadata key="Link (carts)" value="{}" type="LINKS_TO"/>
  </resources>
  <resources label="item">
    <ridt context="//@context" label="/store/carts/{cart.id}/items/{item.id}">
      <pattern label="/store/carts/{cart.id}/items/{item.id}">
        <staticPart label="carts (/store/carts/{cart.id}/items/{item.id})" position="1"/>
        <staticPart label="items (/store/carts/{cart.id}/items/{item.id})" position="4"/>
        <dynamicPart label="cart (/store/carts/{cart.id}/items/{item.id})" position="3"/>
        <dynamicPart label="item (/store/carts/{cart.id}/items/{item.id})" position="5"/>
      </pattern>
    </ridt>
    <representations label="item">
      <representationData>
        <stateData xsi:type="roService:StateElement" key="item" multiplicity="1"/>
      </representationData>
      <representationMetadata key="Content-Type (item)" value="application/vnd.ntua.item+json" type="MEDIA_TYPE"/>
      <onActionType>UPDATE</onActionType>
      <onActionType>READ</onActionType>
    </representations>
  </resources>
  <hypermedia source="//@interactionPoints.0" target="//@interactionPoints.1" label="READ store - to - READ products query"
      requiredForGoal="true"/>
  <hypermedia source="//@interactionPoints.0" target="//@interactionPoints.2" label="READ store - to - CREATE carts"
      requiredForGoal="true"/>
  <hypermedia source="//@interactionPoints.2" target="//@interactionPoints.3" label="CREATE carts - to - CREATE item"
      requiredForGoal="true"/>
</roService:ROServiceDescription>
```

B.3 *DemoShop* M_A

```
<?xml version="1.0" encoding="ASCII"?>
<mappingRules:Alignment xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
    " xmlns:mappingRules="http://softeng.ntua.gr/restadapt/mappings" xmlns:poService="http://softeng.ntua.gr/restadapt/
    poService">
  <rules xsi:type="mappingRules:InputToDynamicPartMapping" label="rl_in2dp_1">
```

```
    <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.3"/>
    <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.2"/>
    <parameter xsi:type="poService:InputParameter" href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@input.1"
        />
    <identifier href="ipss/demoshopmini/demoshopmini.res#//@resources.3/@ridt"/>
    <input href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@input.1"/>
    <dynamicPart href="ipss/demoshopmini/demoshopmini.res#//@resources.3/@ridt/@pattern/@dynamicPart.0"/>
  </rules>
  <rules xsi:type="mappingRules:InputToStateMapping" label="rl_in2st_1">
    <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.3"/>
    <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.2"/>
    <parameter xsi:type="poService:InputParameter" href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@input.2"
        />
    <representation href="ipss/demoshopmini/demoshopmini.res#//@resources.2/@representations.0"/>
    <input href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@input.2"/>
    <state href="ipss/demoshopmini/demoshopmini.res#//@resources.3/@representations.0/@representationData/@stateData.0"/>
  </rules>
  <rules xsi:type="mappingRules:IntentToActionMapping" label="rl_int2act_1">
    <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.3"/>
    <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.2"/>
    <operationName href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@operationName"/>
    <control href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.3/@action"/>
    <intent href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@operationName/@termModel/@terms.0"/>
    <action href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.3/@action"/>
  </rules>
  <rules xsi:type="mappingRules:InputToControlDataMapping" label="rl_in2cd_1">
    <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.3"/>
    <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.2"/>
    <parameter xsi:type="poService:InputParameter" href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@input.0"
        />
    <control href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.3/@request/@controlData.1"/>
    <input href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@input.0"/>
    <controlData href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.3/@request/@controlData.1"/>
  </rules>
  <rules xsi:type="mappingRules:InputToDynamicPartMapping" label="rl_in2dp_2">
    <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.3"/>
    <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.2"/>
    <parameter xsi:type="poService:InputParameter" href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@input.1"
        />
    <identifier href="ipss/demoshopmini/demoshopmini.res#//@resources.3/@ridt"/>
    <input href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@input.2"/>
    <dynamicPart href="ipss/demoshopmini/demoshopmini.res#//@resources.3/@ridt/@pattern/@dynamicPart.1"/>
  </rules>
  <rules xsi:type="mappingRules:TermToStaticPartMapping" label="rl_t2sp_1">
    <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.3"/>
    <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.2"/>
    <operationName href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@operationName"/>
    <resourceIdentifier href="ipss/demoshopmini/demoshopmini.res#//@resources.3/@ridt"/>
    <term href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@operationName/@termModel/@terms.1"/>
    <staticPart href="ipss/demoshopmini/demoshopmini.res#//@resources.3/@ridt/@pattern/@staticPart.0"/>
  </rules>
  <rules xsi:type="mappingRules:TermToStaticPartMapping" label="rl_t2sp_2">
    <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.3"/>
    <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.2"/>
    <operationName href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@operationName"/>
    <resourceIdentifier href="ipss/demoshopmini/demoshopmini.res#//@resources.3/@ridt"/>
    <term href="ipss/demoshopmini/demoshopmini.proc#//@operations.2/@signature/@operationName/@termModel/@terms.1"/>
    <staticPart href="ipss/demoshopmini/demoshopmini.res#//@resources.3/@ridt/@pattern/@staticPart.1"/>
  </rules>
  <rules xsi:type="mappingRules:TermToStaticPartMapping" label="rl_t2sp_3">
    <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.1"/>
    <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.0"/>
    <operationName href="ipss/demoshopmini/demoshopmini.proc#//@operations.0/@signature/@operationName"/>
    <resourceIdentifier href="ipss/demoshopmini/demoshopmini.res#//@resources.1/@ridt"/>
    <term href="ipss/demoshopmini/demoshopmini.proc#//@operations.0/@signature/@operationName/@termModel/@terms.1"/>
    <staticPart href="ipss/demoshopmini/demoshopmini.res#//@resources.1/@ridt/@pattern/@staticPart.0"/>
  </rules>
  <rules xsi:type="mappingRules:InputToDynamicPartMapping" label="rl_in2dp_3">
    <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.1"/>
    <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.0"/>
    <parameter xsi:type="poService:InputParameter" href="ipss/demoshopmini/demoshopmini.proc#//@operations.0/@signature/@input.0"
        />
    <identifier href="ipss/demoshopmini/demoshopmini.res#//@resources.1/@ridt"/>
    <input href="ipss/demoshopmini/demoshopmini.proc#//@operations.0/@signature/@input.0"/>
    <dynamicPart href="ipss/demoshopmini/demoshopmini.res#//@resources.1/@ridt/@pattern/@dynamicPart.0"/>
  </rules>
  <rules xsi:type="mappingRules:IntentToActionMapping" label="rl_int2act_2">
    <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.1"/>
    <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.0"/>
    <operationName href="ipss/demoshopmini/demoshopmini.proc#//@operations.0/@signature/@operationName"/>
    <control href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.1/@action"/>
    <intent href="ipss/demoshopmini/demoshopmini.proc#//@operations.0/@signature/@operationName/@termModel/@terms.0"/>
    <action href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.1/@action"/>
  </rules>
  <rules xsi:type="mappingRules:OutputToStateMapping" label="rl_out2st_1">
    <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.1"/>
    <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.0"/>
    <parameter xsi:type="poService:OutputParameter" href="ipss/demoshopmini/demoshopmini.proc#//@operations.0/@signature/@output
        .0"/>
    <representation href="ipss/demoshopmini/demoshopmini.res#//@resources.1/@representations.0"/>
    <output href="ipss/demoshopmini/demoshopmini.proc#//@operations.0/@signature/@output.0"/>
    <state href="ipss/demoshopmini/demoshopmini.res#//@resources.1/@representations.0/@representationData/@stateData.0"/>
  </rules>
  <rules xsi:type="mappingRules:InputToControlDataMapping" label="rl_in2cd_2">
    <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.2"/>
    <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.1"/>
    <parameter xsi:type="poService:InputParameter" href="ipss/demoshopmini/demoshopmini.proc#//@operations.1/@signature/@input.0"
        />
    <control href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.2/@request/@controlData.1"/>
    <input href="ipss/demoshopmini/demoshopmini.proc#//@operations.1/@signature/@input.0"/>
    <controlData href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.2/@request/@controlData.1"/>
  </rules>
  <rules xsi:type="mappingRules:TermToStaticPartMapping" label="rl_t2sp_4">
    <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.2"/>
    <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.1"/>
    <operationName href="ipss/demoshopmini/demoshopmini.proc#//@operations.1/@signature/@operationName"/>
    <resourceIdentifier href="ipss/demoshopmini/demoshopmini.res#//@resources.2/@ridt"/>
    <term href="ipss/demoshopmini/demoshopmini.proc#//@operations.1/@signature/@operationName/@termModel/@terms.1"/>
    <staticPart href="ipss/demoshopmini/demoshopmini.res#//@resources.2/@ridt/@pattern/@staticPart.0"/>
  </rules>
  <rules xsi:type="mappingRules:InputToStateMapping" label="rl_in2st_2">
```

```
108       <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.2"/>
109       <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.1"/>
110       <parameter xsi:type="poService:InputParameter" href="ipss/demoshopmini/demoshopmini.proc#//@operations.1/@signature/@input.1"
              />
111       <representation href="ipss/demoshopmini/demoshopmini.res#//@resources.2/@representations.0"/>
112       <input href="ipss/demoshopmini/demoshopmini.proc#//@operations.1/@signature/@input.1"/>
113       <state href="ipss/demoshopmini/demoshopmini.res#//@resources.2/@representations.0/@representationData/@stateData.0"/>
114     </rules>
115     <rules xsi:type="mappingRules:IntentToActionMapping" label="rl_int2act_3">
116       <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.2"/>
117       <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.1"/>
118       <operationName href="ipss/demoshopmini/demoshopmini.proc#//@operations.1/@signature/@operationName"/>
119       <control href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.2/@action"/>
120       <intent href="ipss/demoshopmini/demoshopmini.proc#//@operations.1/@signature/@operationName/@termModel/@terms.0"/>
121       <action href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.2/@action"/>
122     </rules>
123     <rules xsi:type="mappingRules:OutputToControlDataMapping" label="rl_out2cd_1">
124       <rip href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.2"/>
125       <pi href="ipss/demoshopmini/demoshopmini.proc#//@interactions.1"/>
126       <parameter xsi:type="poService:OutputParameter" href="ipss/demoshopmini/demoshopmini.proc#//@operations.1/@signature/@output
              .0"/>
127       <control href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.2/@response/@controlData.1"/>
128       <output href="ipss/demoshopmini/demoshopmini.proc#//@operations.1/@signature/@output.0"/>
129       <controlData href="ipss/demoshopmini/demoshopmini.res#//@interactionPoints.2/@response/@controlData.1"/>
130     </rules>
131     <proc href="ipss/demoshopmini/demoshopmini.proc#/"/>
132     <res href="ipss/demoshopmini/demoshopmini.res#/"/>
133   </mappingRules:Alignment>
```

ΠΑΡΑΡΤΗΜΑ Γ΄

ΕΝΝΟΙΟΛΟΓΙΚΑ ΜΟΝΤΕΛΑ ΥΠΗΡΕΣΙΩΝ

C.1 Μεταμοντέλο MM_P

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="procedureOrientedService" nsURI="http://softeng.ntua.gr/restadapt/
        poService"
    nsPrefix="poService">
  <eClassifiers xsi:type="ecore:EClass" name="POServiceDescription" eSuperTypes="SOC.ecore#//ServiceDescription">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType_http://www.eclipse.org/emf/2002/Ecore#//
        EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="operations" lowerBound="1"
        upperBound="-1" eType="#//Operation" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="endpoint" lowerBound="1"
        eType="#//Endpoint" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="scenarios" lowerBound="1"
        upperBound="-1" eType="#//Scenario" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="interactions" lowerBound="1"
        upperBound="-1" eType="#//ProceduralInteraction" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Signature">
    <eStructuralFeatures xsi:type="ecore:EReference" name="operation" lowerBound="1"
        eType="#//Operation" eOpposite="#//Operation/signature"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="input" upperBound="-1"
        eType="#//InputParameter" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="output" upperBound="-1"
        eType="#//OutputParameter" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="operationName" lowerBound="1"
        eType="#//Name" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Endpoint">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="identifier" eType="ecore:EDataType_http://www.eclipse.org/emf/2002/
        Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="ProceduralInteraction">
    <eStructuralFeatures xsi:type="ecore:EReference" name="operation" lowerBound="1"
        eType="#//Operation"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="outputAssignment" eType="#//InputAssignment"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="inputAssignment" eType="#//OutputAssignment"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="source" lowerBound="1"
        upperBound="-1" eType="#//ProceduralTransition" eOpposite="#//ProceduralTransition/from"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="target" lowerBound="1"
        upperBound="-1" eType="#//ProceduralTransition" eOpposite="#//ProceduralTransition/to"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType_http://www.eclipse.org/emf/2002/Ecore#//
        EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Operation">
    <eStructuralFeatures xsi:type="ecore:EReference" name="signature" lowerBound="1"
        eType="#//Signature" containment="true" eOpposite="#//Signature/operation"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" lowerBound="1" eType="ecore:EDataType_http://www.eclipse.org/
        emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="InputParameter" eSuperTypes="#//Parameter"/>
  <eClassifiers xsi:type="ecore:EClass" name="OutputParameter" eSuperTypes="#//Parameter"/>
  <eClassifiers xsi:type="ecore:EClass" name="InputAssignment">
    <eStructuralFeatures xsi:type="ecore:EReference" name="bindings" upperBound="-1"
        eType="#//AssignmentBinding" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="OutputAssignment">
    <eStructuralFeatures xsi:type="ecore:EReference" name="bindings" upperBound="-1"
        eType="#//AssignmentBinding" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Parameter" abstract="true">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType_http://www.eclipse.org/emf/2002/Ecore#//
        EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="optional" eType="ecore:EDataType_http://www.eclipse.org/emf/2002/Ecore
        #//EBoolean"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="parameterName" lowerBound="1"
        eType="#//Name" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="customType" eType="ecore:EClass_SOC.ecore#//CustomType"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="primitiveType" eType="ecore:EClass_SOC.ecore#//PrimitiveType"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="category" eType="#//ParameterCategory"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Scenario">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType_http://www.eclipse.org/emf/2002/Ecore#//
        EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="transitions" lowerBound="1"
        upperBound="-1" eType="#//ProceduralTransition" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="conditions" upperBound="-1"
        eType="#//Condition" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Name">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType_http://www.eclipse.org/emf/2002/Ecore#//
        EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="termModel" eType="ecore:EClass_TermModel.ecore#//TermModel"
        containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="ProceduralTransition">
    <eStructuralFeatures xsi:type="ecore:EReference" name="from" lowerBound="1" eType="#//ProceduralInteraction"
        eOpposite="#//ProceduralInteraction/source"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="to" lowerBound="1" eType="#//ProceduralInteraction"
        eOpposite="#//ProceduralInteraction/target"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="condition" eType="#//Condition"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType_http://www.eclipse.org/emf/2002/Ecore#//
        EString"/>
```

```
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Condition">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="expression" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
        defaultValueLiteral=""/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="ParameterCategory">
    <eLiterals name="APPLICATION_DATA"/>
    <eLiterals name="METADATA"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="AssignmentBinding">
    <eStructuralFeatures xsi:type="ecore:EReference" name="parameter" lowerBound="1"
        eType="#//Parameter"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
</ecore:EPackage>
```

C.2 Μεταμοντέλο MM_R

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="resourceOrientedService" nsURI="http://softeng.ntua.gr/restadapt/roService"
    nsPrefix="roService">
  <eClassifiers xsi:type="ecore:EClass" name="ResourceIdentifierTemplate">
    <eStructuralFeatures xsi:type="ecore:EReference" name="context" lowerBound="1"
        eType="#//Context"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="pattern" eType="#//Pattern"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" lowerBound="1" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Context">
    <eStructuralFeatures xsi:type="ecore:EReference" name="host" lowerBound="1" eType="#//Host"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="application" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="parameters" eType="ecore:EClass SOC.ecore#//ContextParameter"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Pattern">
    <eStructuralFeatures xsi:type="ecore:EReference" name="staticPart" upperBound="-1"
        eType="#//StaticPart" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="dynamicPart" upperBound="-1"
        eType="#//DynamicPart" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="StaticPart" eSuperTypes="#//Part"/>
  <eClassifiers xsi:type="ecore:EClass" name="DynamicPart" eSuperTypes="#//Part">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="expressionType" eType="#//DynamicPartExpressionTypes"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Host">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="identifier" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Part">
    <eOperations name="getOwningResourceType" eType="#//ResourceType"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="position" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Action" eSuperTypes="#//ControlDatum">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="actionType" eType="#//ActionTypes"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Representation">
    <eStructuralFeatures xsi:type="ecore:EReference" name="representationData" eType="#//RepresentationData"
        containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="direction" eType="#//Direction"
        defaultValueLiteral="ANY"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="representationMetadata"
        lowerBound="1" upperBound="-1" eType="#//RepresentationMetadatum" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="onActionType" upperBound="5"
        eType="#//ActionTypes"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="RepresentationData">
    <eStructuralFeatures xsi:type="ecore:EReference" name="stateData" upperBound="-1"
        eType="#//StateDatum" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="RepresentationMetadatum" eSuperTypes="#//KeyValuePair">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" eType="#//RepresentationMetadataTypes"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="StateDatum" eSuperTypes="#//KeyValuePair">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="multiplicity" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="optional" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="type" eType="ecore:EClass SOC.ecore#//DataType"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="KeyValuePair" abstract="true">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="key" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="Direction">
    <eLiterals name="CS" value="1"/>
```

```
69      <eLiterals name="SC" value="2"/>
70      <eLiterals name="ANY"/>
71    </eClassifiers>
72    <eClassifiers xsi:type="ecore:EClass" name="ControlDatum" eSuperTypes="#//KeyValuePair">
73      <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" eType="#//ControlDatumType"/>
74    </eClassifiers>
75    <eClassifiers xsi:type="ecore:EClass" name="ResourceInteractionPoint">
76      <eStructuralFeatures xsi:type="ecore:EReference" name="action" lowerBound="1"
77          eType="#//Action" containment="true"/>
78      <eStructuralFeatures xsi:type="ecore:EReference" name="resource" lowerBound="1"
79          eType="#//ResourceType"/>
80      <eStructuralFeatures xsi:type="ecore:EReference" name="request" lowerBound="1"
81          eType="#//RequestSpecification" containment="true"/>
82      <eStructuralFeatures xsi:type="ecore:EReference" name="response" lowerBound="1"
83          eType="#//ResponseSpecification" containment="true"/>
84      <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//
              EString"/>
85      <eStructuralFeatures xsi:type="ecore:EAttribute" name="entry" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//
              EBoolean"
86          defaultValueLiteral="false"/>
87    </eClassifiers>
88    <eClassifiers xsi:type="ecore:EClass" name="ResourceMetadatum" eSuperTypes="#//KeyValuePair">
89      <eStructuralFeatures xsi:type="ecore:EAttribute" name="type" eType="#//ResourceMetadataTypes"/>
90    </eClassifiers>
91    <eClassifiers xsi:type="ecore:EClass" name="ROServiceDescription" eSuperTypes="SOC.ecore#//ServiceDescription">
92      <eStructuralFeatures xsi:type="ecore:EReference" name="interactionPoints" lowerBound="1"
93          upperBound="-1" eType="#//ResourceInteractionPoint" containment="true"/>
94      <eStructuralFeatures xsi:type="ecore:EReference" name="context" lowerBound="1"
95          eType="#//Context" containment="true"/>
96      <eStructuralFeatures xsi:type="ecore:EReference" name="resources" upperBound="-1"
97          eType="#//ResourceType" containment="true"/>
98      <eStructuralFeatures xsi:type="ecore:EReference" name="hypermedia" upperBound="-1"
99          eType="#//HypermediaRelation" containment="true"/>
100     <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//
              EString"/>
101   </eClassifiers>
102   <eClassifiers xsi:type="ecore:EClass" name="ResourceType">
103     <eStructuralFeatures xsi:type="ecore:EReference" name="ridt" lowerBound="1" eType="#//ResourceIdentifierTemplate"
104         containment="true"/>
105     <eStructuralFeatures xsi:type="ecore:EReference" name="representations" upperBound="-1"
106         eType="#//Representation" containment="true"/>
107     <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//
              EString"/>
108     <eStructuralFeatures xsi:type="ecore:EReference" name="resourceMetadata" upperBound="-1"
109         eType="#//ResourceMetadatum" containment="true"/>
110   </eClassifiers>
111   <eClassifiers xsi:type="ecore:EClass" name="RequestSpecification" eSuperTypes="#//InteractionSpecification"/>
112   <eClassifiers xsi:type="ecore:EClass" name="ResponseSpecification" eSuperTypes="#//InteractionSpecification">
113     <eStructuralFeatures xsi:type="ecore:EReference" name="affordances" upperBound="-1"
114         eType="#//HypermediaRelation"/>
115     <eStructuralFeatures xsi:type="ecore:EReference" name="exposedResourceMetadata"
116         upperBound="-1" eType="#//ResourceMetadatum"/>
117   </eClassifiers>
118   <eClassifiers xsi:type="ecore:EClass" name="HypermediaRelation">
119     <eStructuralFeatures xsi:type="ecore:EReference" name="source" eType="#//ResourceInteractionPoint"/>
120     <eStructuralFeatures xsi:type="ecore:EReference" name="control" lowerBound="1"
121         eType="#//HypermediaControl" containment="true"/>
122     <eStructuralFeatures xsi:type="ecore:EReference" name="target" eType="#//ResourceInteractionPoint"/>
123     <eStructuralFeatures xsi:type="ecore:EReference" name="condition" eType="#//HypermediaCondition"
124         containment="true"/>
125     <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" lowerBound="1" eType="ecore:EDataType␣http://www.eclipse.org/
              emf/2002/Ecore#//EString"/>
126     <eStructuralFeatures xsi:type="ecore:EAttribute" name="requiredForGoal" lowerBound="1"
127         eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//EBoolean"/>
128   </eClassifiers>
129   <eClassifiers xsi:type="ecore:EClass" name="HypermediaControl">
130     <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//
              EString"/>
131   </eClassifiers>
132   <eClassifiers xsi:type="ecore:EClass" name="HypermediaCondition">
133     <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//
              EString"/>
134     <eStructuralFeatures xsi:type="ecore:EAttribute" name="expression" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/
              Ecore#//EString"/>
135   </eClassifiers>
136   <eClassifiers xsi:type="ecore:EClass" name="StateElement" eSuperTypes="#//StateDatum">
137     <eStructuralFeatures xsi:type="ecore:EReference" name="stateElementContent" upperBound="-1"
138         eType="#//StateDatum" containment="true"/>
139   </eClassifiers>
140   <eClassifiers xsi:type="ecore:EEnum" name="ActionTypes">
141     <eLiterals name="READ"/>
142     <eLiterals name="CREATE" value="1" literal="CREATE"/>
143     <eLiterals name="UPDATE" value="2"/>
144     <eLiterals name="DELETE" value="3"/>
145     <eLiterals name="PROCESS" value="4"/>
146   </eClassifiers>
147   <eClassifiers xsi:type="ecore:EClass" name="InteractionSpecification" abstract="true">
148     <eStructuralFeatures xsi:type="ecore:EReference" name="exposedRepresentations"
149         upperBound="-1" eType="#//Representation"/>
150     <eStructuralFeatures xsi:type="ecore:EReference" name="contextResourceIdentifier"
151         lowerBound="1" eType="#//ResourceIdentifierTemplate"/>
152     <eStructuralFeatures xsi:type="ecore:EReference" name="controlData" upperBound="-1"
153         eType="#//ControlDatum" containment="true"/>
154   </eClassifiers>
155   <eClassifiers xsi:type="ecore:EEnum" name="RepresentationMetadataTypes">
156     <eLiterals name="CUSTOM_TYPE" literal="CUSTOM_TYPE"/>
157     <eLiterals name="LANGUAGE" value="1"/>
158     <eLiterals name="CONTENT_LOCATION" value="3" literal="CONTENT_LOCATION"/>
159     <eLiterals name="ENCODING" value="4"/>
160     <eLiterals name="MODIFICATION_ID" value="5"/>
161     <eLiterals name="MODIFICATION_TIME" value="6"/>
162     <eLiterals name="MEDIA_TYPE" value="7"/>
163   </eClassifiers>
164   <eClassifiers xsi:type="ecore:EEnum" name="ResourceMetadataTypes">
165     <eLiterals name="CUSTOM_TYPE" value="1"/>
166     <eLiterals name="ALLOWS_ACTION" value="1"/>
167     <eLiterals name="LINKS_TO" value="2"/>
168     <eLiterals name="SERVER" value="3"/>
169     <eLiterals name="IS_A" value="4"/>
170   </eClassifiers>
171   <eClassifiers xsi:type="ecore:EEnum" name="DynamicPartExpressionTypes">
```

```
    <eLiterals name="CUSTOM_TYPE" value="1"/>
    <eLiterals name="ELEMENT" value="2"/>
    <eLiterals name="QUERY" value="3"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EEnum" name="ControlDatumType">
    <eLiterals name="CUSTOM_TYPE" value="1"/>
    <eLiterals name="HOST" value="2"/>
    <eLiterals name="CONDITIONAL_BY_ID" value="3"/>
    <eLiterals name="CONDITIONAL_BY_TIME" value="4"/>
    <eLiterals name="ACCEPT_MEDIA_TYPE" value="5" literal="ACCEPT_MEDIA_TYPE"/>
    <eLiterals name="ACCEPT_LANGUAGE" value="6"/>
    <eLiterals name="CACHING" value="7"/>
    <eLiterals name="LOCATION" value="8"/>
    <eLiterals name="RESPONSE_CODE" value="9"/>
    <eLiterals name="TIME" value="10"/>
    <eLiterals name="USER_AGENT" value="11"/>
    <eLiterals name="AUTHENTICATION_POLICY" value="12"/>
    <eLiterals name="AUTHENTICATION" value="13" literal="AUTHENTICATION"/>
    <eLiterals name="ACTION" value="14"/>
  </eClassifiers>
</ecore:EPackage>
```

C.3 Μεταμοντέλο MM_A

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="mappingRules" nsURI="http://softeng.ntua.gr/restadapt/mappings"
    nsPrefix="mappingRules">
  <eClassifiers xsi:type="ecore:EClass" name="MappingRule" abstract="true">
    <eStructuralFeatures xsi:type="ecore:EReference" name="rip" lowerBound="1" eType="ecore:EClass␣ResourceOrientedService.ecore
        #//ResourceInteractionPoint"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="pi" lowerBound="1" eType="ecore:EClass␣ProcedureOrientedService.ecore
        #//ProceduralInteraction"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//
        EString"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="IntentToActionMapping" eSuperTypes="#//OperationNameToControlDataMapping">
    <eStructuralFeatures xsi:type="ecore:EReference" name="intent" lowerBound="1"
        eType="ecore:EClass␣TermModel.ecore#//Intent"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="action" lowerBound="1"
        eType="ecore:EClass␣ResourceOrientedService.ecore#//Action"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="InputToStateMapping" eSuperTypes="#//ParameterToRepresentationMapping">
    <eStructuralFeatures xsi:type="ecore:EReference" name="input" lowerBound="1" eType="ecore:EClass␣ProcedureOrientedService.
        ecore#//InputParameter"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="state" lowerBound="1" eType="ecore:EClass␣ResourceOrientedService.
        ecore#//StateDatum"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="InputToDynamicPartMapping" eSuperTypes="#//ParameterToResourceIdentifierMapping">
    <eStructuralFeatures xsi:type="ecore:EReference" name="input" lowerBound="1" eType="ecore:EClass␣ProcedureOrientedService.
        ecore#//InputParameter"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="dynamicPart" lowerBound="1"
        eType="ecore:EClass␣ResourceOrientedService.ecore#//DynamicPart"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="OutputToDynamicPartMapping" eSuperTypes="#//ParameterToResourceIdentifierMapping">
    <eStructuralFeatures xsi:type="ecore:EReference" name="output" lowerBound="1"
        eType="ecore:EClass␣ProcedureOrientedService.ecore#//OutputParameter"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="dynamicPart" lowerBound="1"
        eType="ecore:EClass␣ResourceOrientedService.ecore#//DynamicPart"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="TermToStaticPartMapping" eSuperTypes="#//OperationNameToResourceIdentifierMapping">
    <eStructuralFeatures xsi:type="ecore:EReference" name="term" lowerBound="1" eType="ecore:EClass␣TermModel.ecore#//Term"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="staticPart" lowerBound="1"
        eType="ecore:EClass␣ResourceOrientedService.ecore#//StaticPart"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="OutputToStateMapping" eSuperTypes="#//ParameterToRepresentationMapping">
    <eStructuralFeatures xsi:type="ecore:EReference" name="output" lowerBound="1"
        eType="ecore:EClass␣ProcedureOrientedService.ecore#//OutputParameter"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="state" lowerBound="1" eType="ecore:EClass␣ResourceOrientedService.
        ecore#//StateDatum"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="OutputToResourceMetadataMapping" eSuperTypes="#//MappingRule">
    <eStructuralFeatures xsi:type="ecore:EReference" name="output" lowerBound="1"
        eType="ecore:EClass␣ProcedureOrientedService.ecore#//OutputParameter"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="resourceMetadata" lowerBound="1"
        eType="ecore:EClass␣ResourceOrientedService.ecore#//ResourceMetadatum"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="InputToRepresentationMetadataMapping"
      eSuperTypes="#//ParameterToRepresentationMapping">
    <eStructuralFeatures xsi:type="ecore:EReference" name="input" lowerBound="1" eType="ecore:EClass␣ProcedureOrientedService.
        ecore#//InputParameter"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="representationMetadata"
        lowerBound="1" eType="ecore:EClass␣ResourceOrientedService.ecore#//RepresentationMetadatum"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Alignment">
    <eStructuralFeatures xsi:type="ecore:EReference" name="rules" upperBound="-1"
        eType="#//MappingRule" containment="true"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="proc" lowerBound="1" eType="ecore:EClass␣ProcedureOrientedService.
        ecore#//POServiceDescription"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="res" lowerBound="1" eType="ecore:EClass␣ResourceOrientedService.ecore
        #//ROServiceDescription"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="IntentToStaticPartMapping" eSuperTypes="#//OperationNameToResourceIdentifierMapping
      ">
    <eStructuralFeatures xsi:type="ecore:EReference" name="intent" lowerBound="1"
        eType="ecore:EClass␣TermModel.ecore#//Intent"/>
    <eStructuralFeatures xsi:type="ecore:EReference" name="staticPart" lowerBound="1"
        eType="ecore:EClass␣ResourceOrientedService.ecore#//StaticPart"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="InputToStaticPartMapping" eSuperTypes="#//ParameterToResourceIdentifierMapping">
```

```
66      <eStructuralFeatures xsi:type="ecore:EReference" name="input" lowerBound="1" eType="ecore:EClass_ProcedureOrientedService.
            ecore#//InputParameter"/>
67      <eStructuralFeatures xsi:type="ecore:EReference" name="staticPart" lowerBound="1"
68          eType="ecore:EClass_ResourceOrientedService.ecore#//StaticPart"/>
69    </eClassifiers>
70    <eClassifiers xsi:type="ecore:EClass" name="EndpointToControlDataMapping" eSuperTypes="#//MappingRule">
71      <eStructuralFeatures xsi:type="ecore:EReference" name="controlData" lowerBound="1"
72          eType="ecore:EClass_ResourceOrientedService.ecore#//ControlDatum"/>
73      <eStructuralFeatures xsi:type="ecore:EReference" name="endpoint" lowerBound="1"
74          eType="ecore:EClass_ProcedureOrientedService.ecore#//Endpoint"/>
75    </eClassifiers>
76    <eClassifiers xsi:type="ecore:EClass" name="InputToControlDataMapping" eSuperTypes="#//ParameterToControlDataMapping">
77      <eStructuralFeatures xsi:type="ecore:EReference" name="input" lowerBound="1" eType="ecore:EClass_ProcedureOrientedService.
            ecore#//InputParameter"/>
78      <eStructuralFeatures xsi:type="ecore:EReference" name="controlData" lowerBound="1"
79          eType="ecore:EClass_ResourceOrientedService.ecore#//ControlDatum"/>
80    </eClassifiers>
81    <eClassifiers xsi:type="ecore:EClass" name="OutputToControlDataMapping" eSuperTypes="#//ParameterToControlDataMapping">
82      <eStructuralFeatures xsi:type="ecore:EReference" name="output" lowerBound="1"
83          eType="ecore:EClass_ProcedureOrientedService.ecore#//OutputParameter"/>
84      <eStructuralFeatures xsi:type="ecore:EReference" name="controlData" lowerBound="1"
85          eType="ecore:EClass_ResourceOrientedService.ecore#//ControlDatum"/>
86    </eClassifiers>
87    <eClassifiers xsi:type="ecore:EClass" name="OutputToRepresentationMetadataMapping"
88        eSuperTypes="#//ParameterToRepresentationMapping">
89      <eStructuralFeatures xsi:type="ecore:EReference" name="output" lowerBound="1"
90          eType="ecore:EClass_ProcedureOrientedService.ecore#//OutputParameter"/>
91      <eStructuralFeatures xsi:type="ecore:EReference" name="representationMetadata"
92          lowerBound="1" eType="ecore:EClass_ResourceOrientedService.ecore#//RepresentationMetadatum"/>
93    </eClassifiers>
94    <eClassifiers xsi:type="ecore:EClass" name="ParameterToResourceIdentifierMapping"
95        abstract="true" eSuperTypes="#//MappingRule">
96      <eStructuralFeatures xsi:type="ecore:EReference" name="parameter" lowerBound="1"
97          eType="ecore:EClass_ProcedureOrientedService.ecore#//Parameter"/>
98      <eStructuralFeatures xsi:type="ecore:EReference" name="identifier" lowerBound="1"
99          eType="ecore:EClass_ResourceOrientedService.ecore#//ResourceIdentifierTemplate"/>
100   </eClassifiers>
101   <eClassifiers xsi:type="ecore:EClass" name="OperationNameToResourceIdentifierMapping"
102       abstract="true" eSuperTypes="#//MappingRule">
103     <eStructuralFeatures xsi:type="ecore:EReference" name="operationName" lowerBound="1"
104         eType="ecore:EClass_ProcedureOrientedService.ecore#//Name"/>
105     <eStructuralFeatures xsi:type="ecore:EReference" name="resourceIdentifier" lowerBound="1"
106         eType="ecore:EClass_ResourceOrientedService.ecore#//ResourceIdentifierTemplate"/>
107   </eClassifiers>
108   <eClassifiers xsi:type="ecore:EClass" name="ParameterToRepresentationMapping" abstract="true"
109       eSuperTypes="#//MappingRule">
110     <eStructuralFeatures xsi:type="ecore:EReference" name="parameter" lowerBound="1"
111         eType="ecore:EClass_ProcedureOrientedService.ecore#//Parameter"/>
112     <eStructuralFeatures xsi:type="ecore:EReference" name="representation" lowerBound="1"
113         eType="ecore:EClass_ResourceOrientedService.ecore#//Representation"/>
114   </eClassifiers>
115   <eClassifiers xsi:type="ecore:EClass" name="ParameterToControlDataMapping" abstract="true"
116       eSuperTypes="#//MappingRule">
117     <eStructuralFeatures xsi:type="ecore:EReference" name="parameter" lowerBound="1"
118         eType="ecore:EClass_ProcedureOrientedService.ecore#//Parameter"/>
119     <eStructuralFeatures xsi:type="ecore:EReference" name="control" lowerBound="1"
120         eType="ecore:EClass_ResourceOrientedService.ecore#//ControlDatum"/>
121   </eClassifiers>
122   <eClassifiers xsi:type="ecore:EClass" name="OperationNameToControlDataMapping" abstract="true"
123       eSuperTypes="#//MappingRule">
124     <eStructuralFeatures xsi:type="ecore:EReference" name="operationName" lowerBound="1"
125         eType="ecore:EClass_ProcedureOrientedService.ecore#//Name"/>
126     <eStructuralFeatures xsi:type="ecore:EReference" name="control" lowerBound="1"
127         eType="ecore:EClass_ResourceOrientedService.ecore#//ControlDatum"/>
128   </eClassifiers>
129 </ecore:EPackage>
```

C.4 Μεταμοντέλο κοινόχρηστων στοιχείων

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <ecore:EPackage xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3       xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="serviceOrientedComputing" nsURI="http://softeng.ntua.gr/restadapt/
            soc"
4       nsPrefix="soc">
5     <eClassifiers xsi:type="ecore:EClass" name="ServiceDescription">
6       <eStructuralFeatures xsi:type="ecore:EReference" name="dataTypes" upperBound="-1"
7           eType="#//CustomType" containment="true"/>
8     </eClassifiers>
9     <eClassifiers xsi:type="ecore:EClass" name="SimpleType" eSuperTypes="#//CustomType">
10      <eStructuralFeatures xsi:type="ecore:EReference" name="primitiveType" lowerBound="1"
11          eType="#//PrimitiveType" containment="true"/>
12    </eClassifiers>
13    <eClassifiers xsi:type="ecore:EClass" name="PrimitiveType" abstract="true" eSuperTypes="#//DataType"/>
14    <eClassifiers xsi:type="ecore:EClass" name="DataType" abstract="true">
15      <eStructuralFeatures xsi:type="ecore:EAttribute" name="label" eType="ecore:EDataType_http://www.eclipse.org/emf/2002/Ecore#//
            EString"/>
16      <eStructuralFeatures xsi:type="ecore:EAttribute" name="multiplicity" eType="ecore:EDataType_http://www.eclipse.org/emf/2002/
            Ecore#//EString"/>
17    </eClassifiers>
18    <eClassifiers xsi:type="ecore:EClass" name="CompositeType" eSuperTypes="#//CustomType">
19      <eStructuralFeatures xsi:type="ecore:EReference" name="subTypes" upperBound="-1"
20          eType="#//CustomType" containment="true"/>
21    </eClassifiers>
22    <eClassifiers xsi:type="ecore:EClass" name="String" eSuperTypes="#//PrimitiveType"/>
23    <eClassifiers xsi:type="ecore:EClass" name="Boolean" eSuperTypes="#//PrimitiveType"/>
24    <eClassifiers xsi:type="ecore:EClass" name="Decimal" eSuperTypes="#//PrimitiveType"/>
25    <eClassifiers xsi:type="ecore:EClass" name="Float" eSuperTypes="#//PrimitiveType"/>
26    <eClassifiers xsi:type="ecore:EClass" name="Double" eSuperTypes="#//PrimitiveType"/>
27    <eClassifiers xsi:type="ecore:EClass" name="Duration" eSuperTypes="#//PrimitiveType"/>
28    <eClassifiers xsi:type="ecore:EClass" name="DateTime" eSuperTypes="#//PrimitiveType"/>
```

```
29    <eClassifiers xsi:type="ecore:EClass" name="Date" eSuperTypes="#//PrimitiveType"/>
30    <eClassifiers xsi:type="ecore:EClass" name="Time" eSuperTypes="#//PrimitiveType"/>
31    <eClassifiers xsi:type="ecore:EClass" name="gYearMonth" eSuperTypes="#//PrimitiveType"/>
32    <eClassifiers xsi:type="ecore:EClass" name="gYear" eSuperTypes="#//PrimitiveType"/>
33    <eClassifiers xsi:type="ecore:EClass" name="gMonthDay" eSuperTypes="#//PrimitiveType"/>
34    <eClassifiers xsi:type="ecore:EClass" name="gDay" eSuperTypes="#//PrimitiveType"/>
35    <eClassifiers xsi:type="ecore:EClass" name="gMonth" eSuperTypes="#//PrimitiveType"/>
36    <eClassifiers xsi:type="ecore:EClass" name="hexBinary" eSuperTypes="#//PrimitiveType"/>
37    <eClassifiers xsi:type="ecore:EClass" name="base64Binary" eSuperTypes="#//PrimitiveType"/>
38    <eClassifiers xsi:type="ecore:EClass" name="anyURI" eSuperTypes="#//PrimitiveType"/>
39    <eClassifiers xsi:type="ecore:EClass" name="CustomType" abstract="true" eSuperTypes="#//DataType"/>
40    <eClassifiers xsi:type="ecore:EClass" name="ContextParameter">
41      <eStructuralFeatures xsi:type="ecore:EAttribute" name="key" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//
              EString"/>
42      <eStructuralFeatures xsi:type="ecore:EAttribute" name="value" eType="ecore:EDataType␣http://www.eclipse.org/emf/2002/Ecore#//
            EString"/>
43    </eClassifiers>
44  </ecore:EPackage>
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Software architecture definitions. http://www.sei.cmu.edu/architecture/start/glossary/definition-form.cfm. Accessed: 2014-06-30.

[2] Software architecture glossary. http://www.sei.cmu.edu/architecture/start/glossary/. Accessed: 2014-06-30.

[3] Rosa Alarcon, Cesare Pautasso, and Erik Wilde, editors. *WS-REST '12: Proceedings of the Third International Workshop on RESTful Design*, New York, NY, USA, 2012. ACM.

[4] Rosa Alarcon, Cesare Pautasso, and Erik Wilde, editors. *WS-REST'13 Workshop*, New York, NY, USA, 2013. ACM.

[5] J Algermissen. Classification of http-based apis, 2010.

[6] Michael Athanasopoulos and Kostas Kontogiannis. Identification of rest-like resources from legacy service descriptions. In *Reverse Engineering (WCRE), 2010 17th Working Conference on*, pages 215–219. IEEE, 2010.

[7] Michael Athanasopoulos and Kostas Kontogiannis. Extracting rest resource models from procedure-oriented service interfaces. *Journal of Systems and Software*, 100:149–166, 2015.

[8] Michael Athanasopoulos, Kostas Kontogiannis, and Chris Brealey. Towards an interpretation framework for assessing interface uniformity in rest. In *Proceedings of the Second International Workshop on RESTful Design*, pages 47–50. ACM, 2011.

[9] Michael Athanasopoulos, Kostas Kontogiannis, and Chris Brealey. Considerations of adapting service-offering components to restful architectures. *book 'Migrating to SOA and Cloud Environments: Challenges in Service Oriented Architecture and Cloud Computing Environments', eds. AD Ionita, G. Lewis and M. Litoiu, IGI Global,(to appear 2012)*, 2012.

[10] Xiaoying Bai, Wenli Dong, W-T Tsai, and Yinong Chen. Wsdl-based automatic test case generation for web services testing. In *Service-Oriented System Engineering, 2005. SOSE 2005. IEEE International Workshop*, pages 207–212. IEEE, 2005.

[11] A Banerji, C Bartolini, D Beringer, V Chopella, K Govindarajan, A Karp, H Kuno, M Lemon, G Pogossiants, S Sharma, et al. Web services conversation language (wscl) 1.0 w3c note, march 2002.

[12] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2 edition, 2003.

[13] Twan Basten and Wil MP van der Aalst. Inheritance of behavior. *The Journal of Logic and Algebraic Programming*, 47(2):47–145, 2001.

[14] Robert Battle and Edward Benson. Bridging the semantic web and web 2.0 with representational state transfer (rest). *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):61–69, 2008.

[15] James Bean. *SOA and Web Services Interface Design: Principles, Techniques, and Standards*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009.

[16] James Bean. *SOA and web services interface design: principles, techniques, and standards*. Morgan Kaufmann, 2009.

[17] Jörg Becker, Martin Matzner, and Oliver Müller. Comparing architectural styles for service-oriented architectures–a rest vs. soap case study. In *Information Systems Development*, pages 207–215. Springer, 2010.

[18] Steffen Becker, Antonio Brogi, Ian Gorton, Sven Overhage, Alexander Romanovsky, and Massimo Tivoli. *Towards an engineering approach to component adaptation*. Springer, 2006.

[19] Zohra Bellahsene, Angela Bonifati, Erhard Rahm, et al. *Schema matching and mapping*, volume 20. Springer, 2011.

[20] Boualem Benatallah, Fabio Casati, Daniela Grigori, Hamid R Motahari Nezhad, and Farouk Toumani. Developing adapters for web services integration. In *Advanced Information Systems Engineering*, pages 415–429. Springer, 2005.

[21] Boualem Benatallah, Fabio Casati, and Farouk Toumani. Representing, analysing and managing web service protocols. *Data & Knowledge Engineering*, 58(3):327–357, 2006.

[22] Tim Berners-Lee. A short history of 'resource' in web architecture. `http://www.w3.org/DesignIssues/TermResource.html`, August 2009.

[23] Antonia Bertolino, Paola Inverardi, Patrizio Pelliccione, and Massimo Tivoli. Automatic synthesis of behavior protocols for composable web-services. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 141–150. ACM, 2009.

[24] Dirk Beyer, Arindam Chakrabarti, and Thomas A Henzinger. Web service interfaces. In *Proceedings of the 14th international conference on World Wide Web*, pages 148–159. ACM, 2005.

[25] Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. Bracketing guidelines for treebank ii style penn treebank project. *University of Pennsylvania*, 97, 1995.

[26] Filippo Bonchi, Antonio Brogi, Sara Corfini, and Fabio Gadducci. A behavioural congruence for web services. In *International Symposium on Fundamentals of Software Engineering*, pages 240–256. Springer, 2007.

[27] Lucas Bordeaux, Gwen Salaun, Daniela Berardi, and Massimo Mecella. When are two web services compatible? In *Technologies for E-Services*, pages 15–28. Springer, 2005.

[28] Mario Bravetti and Gianluigi Zavattaro. Contract based multi-party service composition. In *International Symposium on Fundamentals of Software Engineering*, pages 207–222. Springer, 2007.

[29] Antonio Brogi, Carlos Canal, and Ernesto Pimentel. On the semantics of software adaptation. *Science of Computer Programming*, 61(2):136–151, 2006.

[30] Antonio Brogi and Razvan Popescu. Automated generation of bpel adapters. In *Service-Oriented Computing-ICSOC 2006*, pages 27–39. Springer, 2006.

[31] Paul C Brown. *Architecting composite applications and services with TIBCO*. Addison-Wesley Professional, 2012.

[32] Carlos Canal, Juan Manuel Murillo, and Pascal Poizat. Practical ap-

proaches for software adaptation. In *Object-Oriented Technology. ECOOP 2007 Workshop Reader*, pages 154–165. Springer, 2008.

[33] Pedro A. Castillo, Jose Luis Bernier, Maribel Garcia Arenas, Juan Julian Merelo Guervos, and Pablo Garcia-Sanchez. Soap vs rest: Comparing a master-slave ga implementation. *CoRR*, abs/1105.4978, 2011.

[34] Anna Corazza, Sergio Di Martino, and Valerio Maggio. Linsen: An efficient approach to split identifiers and expand abbreviations. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pages 233–242. IEEE, 2012.

[35] Bruno Costa, Paulo F Pires, Flávia C Delicato, and Paulo Merson. Evaluating rest architecturesb•"approach, tooling and guidelines. *Journal of Systems and Software*, 2015.

[36] Luiz Alexandre Hiane da Silva Maciel and Celso Massaki Hirata. An optimistic technique for transactions control using rest architectural style. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 664–669. ACM, 2009.

[37] Luiz Alexandre Hiane da Silva Maciel and Celso Massaki Hirata. Extending timestamp-based two phase commit protocol for restful services to meet business rules. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 778–785. ACM, 2011.

[38] Jérôme David and Jérôme Euzenat. Comparison between ontology distances (preliminary results). In *The Semantic Web-ISWC 2008*, pages 245–260. Springer, 2008.

[39] Ricardo Ramos de Oliveira, Vieira Sanchez, Robson Vinicius, Julio Cezar Estrella, Renata Pontin de Mattos Fortes, and Valerio Brusamolin. Comparative evaluation of the maintainability of restful and soap-wsdl web services. In *Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), 2013 IEEE 7th International Symposium on the*, pages 40–49. IEEE, 2013.

[40] Akon Dey Dey, Alan Fekete, and Uwe RG'Ahm. Rest+t: Scalable transactions over http. In *International Conference on Cloud Engineering, IC2E*, pages 36–41. IEEE, 2015.

[41] Yanhua Du, Xitong Li, and PengCheng Xiong. A petri net approach to

mediation-aided composition of web services. *Automation Science and Engineering, IEEE Transactions on*, 9(2):429–435, 2012.

[42] Marlon Dumas, Murray Spork, and Kenneth Wang. Adapt or perish: Algebra and visual notation for service interface adaptation. *Business Process Management*, pages 65–80, 2006.

[43] Michael Edwards and Martin Chapman. OASIS Service Component Architecture / Assembly (SCA-Assembly) TC. `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=sca-assembly`, 2007. [Online; accessed 04-Jan-2015].

[44] Greg Meredith Sanjiva Weerawarana Erik Christensen, Francisco Curbera. Web services description language (wsdl) 1.1. `http://www.w3.org/TR/wsdl/`, 2001. [Online; accessed 17-June-2014].

[45] Thomas Erl. *SOA: principles of service design*, volume 1. Prentice Hall Upper Saddle River, 2008.

[46] Thomas Erl, Benjamin Carlyle, Cesare Pautasso, and Raj Balasubramanian. *SOA with REST: Principles, Patterns &Constraints for Building Enterprise Solutions with REST*. Prentice Hall Press, 2012.

[47] Jeff A Estefan, K Laskey, F McCabe, and P Thornton. Reference architecture foundation for service oriented architecture version 1.0. *OASIS Committee Draft*, 2(14):2009, 2009.

[48] Jérôme Euzenat, Pavel Shvaiko, et al. *Ontology matching*, volume 18. Springer, 2007.

[49] Xinyang Feng, Jianjing Shen, and Ying Fan. Rest: An alternative to rpc for web services architecture. In *Future Information Networks, 2009. ICFIN 2009. First International Conference on*, pages 7–10. IEEE, 2009.

[50] R Fielding, J Gettys, J Mogul, H Frystyk, L Masinter, P Leach, and T Berners-Lee. Rfc 2616 - hypertext transfer protocol–http/1.1. 1999.

[51] Roy Fielding. Rest apis must be hypermedia-driven. *Online at http://roy.gbivn.com/untangled/2008/rest-apis-must-be-hypertext-driven*, 2008.

[52] Roy T Fielding and Richard N Taylor. Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2):115–150, 2002.

[53] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, 2000.

[54] Martin Fowler. Richardson maturity model: steps toward the glory of rest. *Online at http://martinfowler. com/articles/richardsonMaturityModel. html*, 2010.

[55] Dimitrios Georgakopoulos and Michael P Papazoglou. *Service-oriented computing*. The MIT Press, 2008.

[56] Christian Gierds, Arjan J Mooij, and Karsten Wolf. *Specifying and generating behavioral service adapters based on transformation rules*. Univ., Inst. fur Informatik, 2008.

[57] Paolo Giorgini, John Mylopoulos, Eleonora Nicchiarelli, and Roberto Sebastiani. Reasoning with goal models. In *Conceptual Modeling' ER 2002*, pages 167–181. Springer, 2003.

[58] Object Management Group. Service oriented architecture modeling language (soaml), version 1.0.1. `http://www.omg.org/spec/SoaML/1.0.1/`, 2012. [Online; accessed 17-June-2014].

[59] The Open Group. Soa reference architecture technical standard. `http://www.opengroup.org/soa/source-book/soa_refarch/index.htm`, 2011. [Online; accessed 17-June-2014].

[60] The Open Group. Soa ontology version 2.0. `http://www.opengroup.org/soa/source-book/ontologyv2/index.htm`, 2014. [Online; accessed 17-June-2014].

[61] Zhifeng Gu, Juanzi Li, and Bin Xu. Automatic service composition based on enhanced service dependency graph. In *Web Services, 2008. ICWS'08. IEEE International Conference on*, pages 246–253. IEEE, 2008.

[62] Dominique Guinard, Iulia Ion, and Simon Mayer. In search of an internet of things service architecture: Rest or ws-*? a developers' perspective. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 326–337. Springer, 2012.

[63] Vivek Gupta. Usage of document/literal wrapped pattern in wsdl design. Usageofdocument/literalwrappedpatterninWSDLdesign, 2011.

[64] Marc J Hadley. Web application description language (wadl), 2006.

[65] ISO/IEC/IEEE. Iso/iec/ieee systems and software engineering – architecture description. *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)*, pages 1–46, Dec 2011.

[66] Chandra Krintz Jayathilaka, Hiranya and Rich Wolski. Service-driven computing with apis: Concepts, frameworks, and emerging trends. In *Handbook of Research on Architectural Trends in Service-Driven Computing*, pages 355–379. IGI Global, 2014.

[67] Wei Jiang, Dongwon Lee, and Songlin Hu. Large-scale longitudinal analysis of soap-based and restful web services. In *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pages 218–225. IEEE, 2012.

[68] Stephen Kell. A survey of practical software adaptation techniques. *J. UCS*, 14(13):2110–2157, 2008.

[69] Sean Kennedy, Robert Stewart, Paul Jacob, and Owen Molloy. Storhm: a protocol adapter for mapping soap based web services to restful http format. *Electronic Commerce Research*, 11(3):245–269, 2011.

[70] Kostas Kontogiannis, Grace A Lewis, and Dennis B Smith. A research agenda for service-oriented architecture. In *Proceedings of the 2nd international workshop on Systems development in SOA environments*, pages 1–6. ACM, 2008.

[71] Jacek Kopecky, Karthik Gomadam, and Tomas Vitvar. hrests: An html microformat for describing restful web services. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, volume 1, pages 619–625. IEEE, 2008.

[72] Heather Kreger. Navigating the soa open standards landscape around architecture. *Joint Paper, The Open Group, OASIS, and OMG*, 2009.

[73] Philippe Kruchten, Henk Obbink, and Judith Stafford. The past, present, and future for software architecture. *Software, IEEE*, 23(2):22–30, 2006.

[74] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[75] Markku Laitkorpi, Johannes Koskinen, and Tarja Systa. A uml-based approach for abstracting application interfaces to rest-like services. In *Reverse Engineering, 2006. WCRE'06. 13th Working Conference on*, pages 134–146. IEEE, 2006.

[76] Markku Laitkorpi, Petri Selonen, and Tarja Systa. Towards a model-driven process for designing restful web services. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 173–180. IEEE, 2009.

[77] James Lewis and Martin Fowler. Microservices. `http://martinfowler.com/articles/microservices.html`, 2014. [Online; accessed 04-Jan-2015].

[78] Zheng Li, Liam O'Brien, and He Zhang. Circumstantial–evidence–based effort judgement for web service composition–based soa implementations. *International Journal of Space-Based and Situated Computing*, 2(1):31–44, 2012.

[79] Mark Little. Rest and transactions? *Online at http://www.infoq.com/news/2009/06/rest-ts*, 2009.

[80] Yan Liu, Qingling Wang, Mingguang Zhuang, and Yunyun Zhu. Reengineering legacy systems with restful web service. In *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*, pages 785–790. IEEE, 2008.

[81] Niels Lohmann. A feature-complete petri net semantics for ws-bpel 2.0. In *Web Services and Formal Methods*, pages 77–91. Springer, 2008.

[82] Niels Lohmann, Peter Massuthe, and Karsten Wolf. *Operating guidelines for finite-state services*. Springer, 2007.

[83] Niels Lohmann and Karsten Wolf. How to implement a theory of correctness in the area of business processes and services. In *Business Process Management*, pages 61–77. Springer, 2010.

[84] C Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F Brown, Rebekah Metz, and Booz Allen Hamilton. Reference model for service oriented architecture 1.0. *OASIS Standard*, 12, 2006.

[85] Nioosha Madani, Latifa Guerrouj, Massimiliano Di Penta, Y Gueheneuc, and Giuliano Antoniol. Recognizing words from source code identifiers using speech recognition techniques. In *Software Maintenance and Reengineering (CSMR), 2010 14th European Conference on*, pages 68–77. IEEE, 2010.

[86] Maria Maleshkova, Carlos Pedrinaci, and John Domingue. Investigating web apis on the world wide web. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 107–114. IEEE, 2010.

[87] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

[88] Alexandros Marinos, Amir Razavi, Sotiris Moschoyiannis, and Paul Krause. Retro: A consistent and recoverable restful transaction model. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 181–188. IEEE, 2009.

[89] Clynch Gary Markey, Philip. A performance analysis of ws-* (soap) & restful web services for implementing service and resource orientated architectures. In *The 12th Information Technology and Telecommunications (IT&T) Conference*. Athlone IT, 2013.

[90] Axel Martens. Analyzing web service based business processes. In *Fundamental Approaches to Software Engineering*, pages 19–33. Springer, 2005.

[91] José Antonio Martín and Ernesto Pimentel. Automatic generation of adaptation contracts. *Electronic Notes in Theoretical Computer Science*, 229(2):115–131, 2009.

[92] Larry Masinter, Tim Berners-Lee, and Roy T Fielding. Rfc uniform resource identifier (uri): Generic syntax - uniform resource identifier (uri): Generic syntax. 2005.

[93] Sabine Massmann, Salvatore Raunich, David Aumüller, Patrick Arnold, and Erhard Rahm. Evolution of the coma match system. *Ontology Matching*, 49, 2011.

[94] Peter Massuthe, Wolfgang Reisig, and Karsten Schmidt. An operating guideline approach to the soa. *ANNALS OF MATHEMATICS, COMPUTING AND TELEINFORMATICS*, 1:35–43, 2005.

[95] Peter Massuthe and Karsten Schmidt. Operating guidelines-an automata-theoretic foundation for the service-oriented architecture. In *Quality Software, 2005.(QSIC 2005). Fifth International Conference on*, pages 452–457. IEEE, 2005.

[96] Peter Massuthe, Alexander Serebrenik, Natalia Sidorova, and Karsten Wolf. Can i find a partner? undecidability of partner existence for open nets. *Information Processing Letters*, 108(6):374–378, 2008.

[97] Peter Massuthe and Daniela Weinberg. Fiona a tool to analyze interacting open nets, 2008.

[98] Peter Massuthe and Karsten Wolf. An algorithm for matching non-deterministic services with operating guidelines. *International Journal of Business Process Integration and Management*, 2(2):81–90, 2007.

[99] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[100] Hamid Reza Motahari Nezhad, Boualem Benatallah, Axel Martens, Francisco Curbera, and Fabio Casati. Semi-automated adaptation of service interactions. In *Proceedings of the 16th international conference on World Wide Web*, pages 993–1002. ACM, 2007.

[101] Hamid Reza Motahari Nezhad, Guang Yuan Xu, and Boualem Benatallah. Protocol-aware matching of web service interfaces for adapter development. In *Proceedings of the 19th international conference on World wide web*, pages 731–740. ACM, 2010.

[102] Gavin Mulligan and Denis Gračanin. A comparison of soap and rest implementations of a service based interaction independence middleware framework. In *Simulation Conference (WSC), Proceedings of the 2009 Winter*, pages 1423–1432. IEEE, 2009.

[103] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[104] Meenakshi Nagarajan, Kunal Verma, Amit P Sheth, John Miller, and Jon Lathem. Semantic interoperability of web services-challenges and experiences. In *Web Services, 2006. ICWS'06. International Conference on*, pages 373–382. IEEE, 2006.

[105] US NHI National Cancer Institute. Service and capability naming standards document. https://wiki.nci.nih.gov/display/SAIF/CBIIT+SAIF+Wiki, 2009.

[106] Jaime Navon and Federico Fernandez. The essence of rest architectural style. In *REST: from research to practice*, pages 21–33. Springer, 2011.

[107] Laskey Ken Newcomer, Eric and Le Hegaret Philippe. Web of Services for Enterprise Computing Workshop Report. http://www.w3.org/2007/04/wsec_report, 2007. [Online; accessed 04-Jan-2015].

[108] Sam Newman. *Building Microservices*. " O'Reilly Media, Inc.", 2015.

[109] OASIS. The oasis ws-i. http://www.oasis-ws-i.org/, 2010.

[110] Michael P Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-oriented computing: a research roadmap. *International Journal of Cooperative Information Systems*, 17(02):223–255, 2008.

[111] Mike P Papazoglou and Willem-Jan van den Heuvel. Service-oriented computing: State-of-the-art and open research issues. *IEEE Computer. v40 i11*, 2003.

[112] Guy Pardon and Cesare Pautasso. Towards distributed atomic transactions over restful services. In *REST: From Research to Practice*, pages 507–524. Springer, 2011.

[113] Guy Pardon and Cesare Pautasso. Atomic distributed transactions: a restful design. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 943–948. International World Wide Web Conferences Steering Committee, 2014.

[114] Cesare Pautasso and Erik Wilde. Why is the web loosely coupled?: a multi-faceted metric for service design. In *Proceedings of the 18th international conference on World wide web*, pages 911–920. ACM, 2009.

[115] Cesare Pautasso, Erik Wilde, and Rosa Alarcon, editors. *Second international workshop on RESTful design (WS-REST 2011)*. ACM, 2011.

[116] Cesare Pautasso, Erik Wilde, and Alexandros Marinos, editors. *WS-REST '10: Proceedings of the First International Workshop on RESTful Design*, New York, NY, USA, 2010. ACM.

[117] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. big'web services: making the right architectural decision. In *Proceedings of the 17th international conference on World Wide Web*, pages 805–814. ACM, 2008.

[118] Santiago Pericas-Geertsen and Marek Potociar. Jax-rs: Java api for restful web services. *Oracle Corporation*, pages 1–84, 2013.

[119] Michael P Perrone and Leon N Cooper. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, DTIC Document, 1992.

[120] Dewayne E Perry and Alexander L Wolf. Foundations for the study of software architecture. *ACM SIGSOFT Software Engineering Notes*, 17(4):40–52, 1992.

[121] Shankar R Ponnekanti and Armando Fox. Interoperability among independently evolving web services. In *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 331–351. Springer-Verlag New York, Inc., 2004.

[122] Paul Prescod. Roots of the rest/soap debate. In *Extreme Markup Languages®*. Citeseer, 2002.

[123] Erhard Rahm. Towards large-scale schema and ontology matching. In *Schema matching and mapping*, pages 3–27. Springer, 2011.

[124] Erhard Rahm and Philip A Bernstein. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350, 2001.

[125] Amir Razavi, Alexandros Marinos, Sotiris Moschoyiannis, and Paul Krause. Restful transactions supported by the isolation theorems. In *Web Engineering*, pages 394–409. Springer, 2009.

[126] Arend Rensink. The groove simulator: A tool for state space generation. In *Applications of Graph Transformations with Industrial Relevance*, pages 479–485. Springer, 2004.

[127] Dominik Renzel, Patrick Schlebusch, and Ralf Klamma. Today's top "restful" services and why they are not restful. In *Web Information Systems Engineering-WISE 2012*, pages 354–367. Springer, 2012.

[128] Leonard Richardson and Sam Ruby. *RESTful web services*. O'Reilly Media, Inc., 2008.

[129] Arthur Ryman Sanjiva Weerawarana Roberto Chinnici, Jean-Jacques Moreau. Web services description language (wsdl) version 2.0 part 1: Core language. `http://www.w3.org/TR/wsdl20/`, 2007. [Online; accessed 17-June-2014].

[130] Juan Manuel Rodriguez, Marco Crasso, Cristian Mateos, Alejandro Zunino, and Marcelo Campo. The easysoc project: a rich catalog of best practices for developing web service applications. *CLEI Electronic Journal*, 14(3):2–2, 2011.

[131] Lior Rokach. *Pattern classification using ensemble methods*, volume 75. World Scientific, 2010.

[132] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[133] Silvia Schreier. Modeling restful applications. In *Proceedings of the Second International Workshop on RESTful Design*, pages 15–21. ACM, 2011.

[134] Len Seligman, Peter Mork, Alon Halevy, Ken Smith, Michael J Carey, Kuang Chen, Chris Wolf, Jayant Madhavan, Akshay Kannan, and Doug Burdick. Openii: an open source information integration toolkit. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 1057–1060. ACM, 2010.

[135] Petri Selonen. From requirements to a restful web service: Engineering content oriented web services with rest. In *REST: From Research to Practice*, pages 259–278. Springer, 2011.

[136] Mary Shaw and David Garlan. *Software architecture: perspectives on an emerging discipline*, volume 1. Prentice Hall Englewood Cliffs, 1996.

[137] Amit P Sheth, Karthik Gomadam, and Jon Lathem. Sa-rest: semantically interoperable and easier-to-use services and mashups. *IEEE Internet Computing*, 11(6):91–94, 2007.

[138] Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching

approaches. In *Journal on Data Semantics IV*, pages 146–171. Springer, 2005.

[139] Pavel Shvaiko and Jérôme Euzenat. Ontology matching: state of the art and future challenges. *Knowledge and Data Engineering, IEEE Transactions on*, 25(1):158–176, 2013.

[140] Munindar P Singh and Michael N Huhns. *Service-oriented computing: semantics, processes, agents*. John Wiley & Sons, 2006.

[141] Christian Stahl, Peter Massuthe, and Jan Bretschneider. Deciding substitutability of services with operating guidelines. In *Transactions on Petri Nets and Other Models of Concurrency II*, pages 172–191. Springer-Verlag, 2009.

[142] Jakob Strauch and Silvia Schreier. Restify: from rpcs to restful http design. In *Proceedings of the Third International Workshop on RESTful Design*, pages 11–18. ACM, 2012.

[143] Wei Tan, Yushun Fan, and MengChu Zhou. A petri net-based method for compatibility analysis and composition of web services in business process execution language. *Automation Science and Engineering, IEEE Transactions on*, 6(1):94–106, 2009.

[144] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[145] Richard N Taylor, Nenad Medvidovic, and Eric M Dashofy. *Software architecture: foundations, theory, and practice*. Wiley Publishing, 2009.

[146] Bipin Upadhyaya, Ying Zou, Hua Xiao, Joanna Ng, and Alex Lau. Migration of soap-based services to restful services. In *Web Systems Evolution (WSE), 2011 13th IEEE International Symposium on*, pages 105–114. IEEE, 2011.

[147] Wil MP van der Aalst. The application of petri nets to workflow management. *Journal of circuits, systems, and computers*, 8(01):21–66, 1998.

[148] Wil MP Van Der Aalst, Niels Lohmann, Peter Massuthe, Christian Stahl, and Karsten Wolf. From public views to private views–correctness-by-design for services. In *Web Services and Formal Methods*, pages 139–153. Springer, 2008.

[149] Wil MP van der Aalst, Niels Lohmann, Peter Massuthe, Christian Stahl, and Karsten Wolf. Multiparty contracts: Agreeing and implementing interorganizational processes. *The Computer Journal*, 53(1):90–106, 2010.

[150] Wil MP van der Aalst, Arjan J Mooij, Christian Stahl, and Karsten Wolf. Service interaction: Patterns, formalization, and analysis. In *Formal Methods for Web Services*, pages 42–88. Springer, 2009.

[151] Steve Vinoski. Putting the" web" into web services. web services interaction models. 2. *Internet Computing, IEEE*, 6(4):90–92, 2002.

[152] Steve Vinoski. Rpc and rest: Dilemma, disruption, and displacement. *Internet Computing, IEEE*, 12(5):92–95, 2008.

[153] Steve Vinoski. Serendipitous reuse. *Internet Computing, IEEE*, 12(1):84–87, 2008.

[154] Atro Voutilainen. Part-of-speech tagging. *The Oxford handbook of computational linguistics*, pages 219–232, 2003.

[155] W3C. Web services activity. `http://www.w3.org/2002/ws/`, 2002.

[156] W3C. Web services architecture. `http://www.w3.org/TR/ws-arch/`, 2004.

[157] W3C. Web services architecture working group. `http://www.w3.org/2002/ws/arch/`, 2004.

[158] Jim Webber, Savas Parastatidis, and Ian Robinson. *REST in practice: Hypermedia and systems architecture*. " O'Reilly Media, Inc.", 2010.

[159] Erik Wilde and Cesare Pautasso. *REST: from research to practice*. Springer, 2011.

[160] Web Service Interoperability Organization (WS-I). Basic profile version 1.2. `http://ws-i.org/Profiles/BasicProfile-1.2-2010-11-09.html`, 2010. [Online; accessed 17-June-2014].

[161] Web Service Interoperability Organization (WS-I). Basic profile version 2.0. `http://ws-i.org/Profiles/BasicProfile-2.0-2010-11-09.html`, 2010. [Online; accessed 17-June-2014].