DECISION MAKING IN NON-DETERMINISTIC ENVIRONMENTS

GEORGE MARKOU

PH.D. DISSERTATION

SUPERVISED BY

ASSOC. PROFESSOR IOANNIS REFANIDIS

SUBMITTED TO

DEPARTMENT OF APPLIED INFORMATICS

UNIVERSITY OF MACEDONIA

THESSALONIKI, GREECE

OCTOBER 2015

European Union
European Social Fund

OPERATIONAL PROGRAMME
EDUCATION AND LIFELONG LEARNING
investing in knowledge society
MINISTRY OF EDUCATION & RELIGIOUS AFFAIRS
MANAGING AUTHORITY
Co-financed by Greece and the European Union

NSRF 2007-2013
programme for development
EUROPEAN SOCIAL FUND

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Fund Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

ADVISORY COMMITTEE

Ioannis Refanidis, Associate Professor
Department of Applied Informatics, University of Macedonia, Greece

Hector Geffner, Professor
Departament de Tecnologies de la Informació i les Comunicacions (DTIC), Universitat Pompeu Fabra, Spain

Dimitrios Hristu-Varsakelis, Associate Professor
Department of Applied Informatics, University of Macedonia, Greece

EXAMINATION COMMITTEE

George Stephanides, Professor

Department of Applied Informatics, University of Macedonia, Greece

Ilias Sakellariou, Lecturer

Department of Applied Informatics, University of Macedonia, Greece

Georgios Evangelidis, Professor

Department of Applied Informatics, University of Macedonia, Greece

Dimitris Vrakas, Assistant Professor

School of Informatics, Aristotle University of Thessaloniki, Greece

ABSTRACT

This dissertation tackles the problem of automated web service composition, taking into account the non-deterministic nature of web services. Web services, a major ingredient of the Semantic Web, aim to solve interoperability problems between heterogeneous systems, with transparency over the underlying technologies used to implement them and the platforms they are based on. Importantly, due to the semantic markup of a web service, it is possible to automate the tasks of web service discovery, selection and composition.

Since no single web service may provide the desired functionality, it is necessary to perform the task of web service composition in order to achieve it. However, web services exist and operate in an ever-changing and expanding environment; for that reason, searching for the appropriate web services for each goal is hard. What makes web service composition difficult and time-consuming is the additional burden of monitoring whether a web service taking part in an existing solution is still active and has the same usage and interface. Moreover, the possible uncertainty in the web service's execution results must also be taken into account in the web service composition process.

In the last decades, a plethora of methods for planning in classical domains have been proposed providing promising results. In parallel, and frequently inspired by such deterministic methods, algorithms and tools for planning in non-deterministic settings, either in fully, partially, or non-observable domains have been developed. Such methods can be efficiently utilized for the purposes of automated web service composition, using existing tools, following a translation of the original semantic web service composition domain to an AI planning one.

The dissertation has three main parts. First, it critically reviews existing non-deterministic AI planning approaches, as well as deterministic and non-deterministic web service composition ones. Second, it reviews the evaluation process in web service composition approaches and proposes a new set of reproducible use case scenarios. Third, and most importantly, it proposes a non-deterministic AI planning algorithm aimed for automated web service composition. The aforementioned have been implemented into a web-based application, **MADSWAN**, that comprises a semantic web service registry, an editor of semantic description files, as well as manual and automated web service composition modules. The system adheres to the reusable nature of web services by utilizing existing open source projects as sub-element and makes use of an anytime probabilistic planner, **MAPPPA2**, tailor-made for web service composition problems so as to tackle the inherent non-determinism in the web service composition domain.

Keywords: web services; composition; non-determinism; contingent plans; critical review; planning;

ΠΕΡΙΛΗΨΗ

Η παρούσα διατριβή αντιμετωπίζει το πρόβλημα της αυτοματοποιημένης σύνθεση υπηρεσιών ιστού λαμβάνοντας υπόψη τη στοχαστική φύση των υπηρεσιών ιστού. Οι υπηρεσίες ιστού, ένα σημαντικό συστατικό του Σημασιολογικού Ιστού, έχουν ως στόχο να λύσουν τα προβλήματα διαλειτουργικότητας μεταξύ ετερογενών συστημάτων, παρέχοντας προτεραιότητα στη διαφάνεια σε σχέση με τις υποκείμενες τεχνολογίες που χρησιμοποιούνται για την υλοποίηση τους και τις πλατφόρμες στις οποίες βασίζονται. Είναι σημαντικό δε ότι λόγω της σημασιολογικής σήμανσης των υπηρεσιών ιστού είναι δυνατόν να αυτοματοποιηθούν οι εργασίες της εύρεσης των κατάλληλων υπηρεσιών ιστού, καθώς και της επιλογής και σύνθεσης τους.

Δεδομένου του ότι μια υπηρεσία ιστού συχνά δεν προσφέρει μια επιθυμητή λειτουργία από μόνη της, είναι απαραίτητο να εκτελεστεί μια διαδικασία σύνθεσης υπηρεσιών ιστού για την επίτευξή της. Ωστόσο, οι υπηρεσίες ιστού λειτουργούν σε ένα διαρκώς μεταβαλλόμενο περιβάλλον. Για το λόγο αυτό, η αναζήτηση για τις κατάλληλες υπηρεσίες ιστού που επιτυγχάνουν κάποιον συγκεκριμένο στόχο είναι μια δύσκολη διαδικασία. Αυτό που καθιστά την διαδικασία σύνθεσης υπηρεσιών ιστού δύσκολη και χρονοβόρα είναι η πρόσθετη επιβάρυνση του να ελέγχεται το κατά πόσο είναι ακόμη ενεργή μια υπηρεσία ιστού που λαμβάνει μέρος σε μια υπάρχουσα λύση και αν έχει ακόμα την ίδια χρήση και διεπαφή. Επιπλέον, πρέπει επίσης να ληφθεί υπόψη η πιθανή αβεβαιότητα στα αποτελέσματα της εκτέλεσης των υπηρεσιών ιστού.

Κατά τις τελευταίες δεκαετίες, μια πληθώρα μεθόδων για τον αιτιοκρατικό σχεδιασμό ενεργειών έχουν προταθεί επιτυγχάνοντας ιδιαίτερα ελπιδοφόρα αποτελέσματα. Παράλληλα, και συχνά ορμώμενες από τέτοιες αιτιοκρατικές μεθόδους, εργαλεία για το σχεδιασμό ενεργειών σε περιβάλλοντα με αβεβαιότητα, είτε σε πλήρως, είτε σε εν μέρει είτε σε μη παρατηρήσιμα περιβάλλοντα έχουν αναπτυχθεί. Τέτοιες μέθοδοι μπορούν να χρησιμοποιηθούν αποτελεσματικά για τους σκοπούς της αυτοματοποιημένης σύνθεσης υπηρεσιών ιστού, χρησιμοποιώντας ήδη υπάρχοντα εργαλεία, μετά από έναν μετασχηματισμό του αρχικού σημασιολογικού προβλήματος σύνθεσης υπηρεσιών ιστού σε ένα πρόβλημα σχεδιασμού ενεργειών.

Η διατριβή αποτελείται από τρία μέρη: Πρώτον, παρέχει μια κριτική ανασκόπηση της βιβλιογραφίας του πεδίου του σχεδιασμού ενεργειών σε περιβάλλοντα με αβεβαιότητα, καθώς και των πεδίων της αιτιοκρατικής και μη αιτιοκρατικής σύνθεσης υπηρεσιών ιστού. Δεύτερον, εξετάζει τη διαδικασία αξιολόγησης των προσεγγίσεων σύνθεσης υπηρεσιών ιστού και προτείνει ένα νέο σύνολο αναπαράξιμων σεναρίων χρήσης. Τρίτον, και κυριότερο, προτείνει έναν μη-αιτιοκρατικό αλγόριθμο σχεδιασμού ενεργειών που εφαρμόζεται στο πεδίο της αυτοματοποιημένης σύνθεσης υπηρεσιών ιστού. Τα προαναφερθέντα μέρη έχουν υλοποιηθεί σε μια διαδικτυακή εφαρμογή, ονομαζόμενη **MADSWAN**, η οποία αποτελείται από ένα σημασιολογικό μητρώο υπηρεσιών ιστού, έναν επεξεργαστή αρχείων περιγραφής σημασιολογικών υπηρεσιών ιστού, καθώς και διαδικασίες χειροκίνητης και αυτοματοποιημένης σύνθεση υπηρεσιών ιστού. Το σύστημα ακολουθεί τους κανόνες που επιβάλλει η φύση των υπηρεσιών ιστού μέσω της χρήσης υφιστάμενων προγραμμάτων ανοικτού κώδικα ως στοιχεία του, ενώ

χρησιμοποιεί έναν πιθανοτικό αλγόριθμο σχεδιασμού ενεργειών οποιουδήποτε χρόνου, τον **MAPPPA2**, ο οποίος στοχεύει συγκεκριμένα στην επίλυση προβλημάτων σύνθεσης υπηρεσιών ιστού, προκειμένου να αντιμετωπίσει την εγγενή αβεβαιότητα που υπάρχει στο εν λόγω πρόβλημα.

Λέξεις κλειδιά: υπηρεσίες ιστού, σύνθεση, αβεβαιότητα, πλάνα πολλαπλών ενδεχομένων, κριτική ανασκόπηση, σχεδιασμός ενεργειών.

ACKNOWLEDGEMENTS

Having completed both my bachelor and my MSc studies at the Department of Applied Informatics of the University of Macedonia, I only saw it as a sensible step to continue exploring my research interests here; I was very fortunate to be supported throughout my entire journey by Assistant Professor Ioannis Refanidis, the supervisor of my PhD dissertation and academic mentor since my MSc studies. I thank him not only for his scientific guidance, but also for dedicating unlimited hours to the supervision of my thesis, and more importantly, for letting me having the creative freedom that I required in certain aspects of my research. Finally, I would like to thank him for his encouragement that I pursue both academic and industrial scholarships, which helped me immensely to successfully complete my PhD.

It was an honour that two prominent researchers worldwide participated in my Advisory Committee. I would like to express my deep gratitude to Professor Hector Geffner, from the Universitat Pompeu Fabra, and Associate Professor Dimitris Hristu-Varsakelis, from the University of Macedonia, for the trust and discreet support that they provided during the entire course of my research.

This dissertation was completed in the context of its co-financing by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. I fully acknowledge and appreciate that without this scholarship this PhD dissertation would have probably not been possible.

The process of completing a PhD is exceptionally hard, mostly in regard to the psychological elements of it. For this reason, the acknowledgement that your research work is actually valuable can vastly encourage you to push forward; I would like to thank the organizing committees of the following conferences for choosing me to be the recipient of their respective awards: 2nd ESWC Summer School 2012 for the best poster award; the 7th International Conference on Internet and Web Applications and Services (ICIW '12) for the best paper award; and ICTAI 2014 for both their best student paper award and the best paper award.

I would like to thank my long-time girlfriend Mirela, for putting up with me whenever I was stressed, angry or disappointed; all three things not entirely uncommon during these years. Finally, as any important task in life, a PhD dissertation cannot be completed without the support of your family; I would like to express my gratitude towards my parents, Kyros and Eleni, for their incontrovertible love and support. Even when I believed I could not do it, they never doubted me.

ΕΥΧΑΡΙΣΤΙΕΣ

Έχοντας ολοκληρώσει τόσο το βασικό πτυχίο μου όσο και το μεταπτυχιακό μου στο τμήμα Εφαρμοσμένης Πληροφορικής του Πανεπιστημίου Μακεδονίας. θεώρησα την εκπόνηση διδακτορικής διατριβής εδώ ως τη λογική συνέχεια της εξερεύνησης των ερευνητικών μου ενδιαφερόντων. Στάθηκα ιδιαιτέρως τυχερός που καθ' όλη τη διάρκεια αυτού του ταξιδίου μου είχα την υποστήριξη του Αναπληρωτή Καθηγητή Ιωάννη Ρεφανίδη, ως επιβλέποντα της διδακτορικής μου διατριβής και ακαδημαϊκού μέντορά μου από την αρχή των μεταπτυχιακών σπουδών μου. Τον ευχαριστώ όχι μόνο για την επιστημονική καθοδήγηση του, αλλά και για την αφιέρωση απεριόριστων ωρών για την επίβλεψη της διατριβής μου, και κυρίως, για το ότι μου επέτρεψε να έχω τη δημιουργική ελευθερία που χρειαζόμουν για ορισμένες πτυχές της έρευνας μου. Τέλος, θα ήθελα να τον ευχαριστήσω για την ενθάρρυνσή του στο να επιδιώκω τόσο ακαδημαϊκές όσο και επιχειρηματικές υποτροφίες, γεγονός που βοήθησε ιδιαίτερα να ολοκληρώσω επιτυχώς το διδακτορικό μου.

Ήταν τιμή μου που δυο εξέχοντες ερευνητές παγκοσμίως συμμετείχαν στην συμβουλευτική επιτροπή μου. Θα ήθελα να εκφράσω την βαθιά ευγνωμοσύνη μου προς τον Καθηγητή Hector Geffner του Universitat Pompeu Fabra , και προς τον Αναπληρωτή Καθηγητή Δημήτριο Χρήστου-Βαρσακέλη του Πανεπιστημίου Μακεδονίας για την εμπιστοσύνη και την διακριτική υποστήριξη που μου παρείχαν καθ' όλη τη διάρκεια της έρευνάς μου.

Η παρούσα έρευνα ολοκληρώθηκε στο πλαίσιο του Επιχειρησιακού Προγράμματος "Εκπαίδευση και Δια Βίου Μάθηση" που συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους, μέσω της πράξης «Ηράκλειτος ΙΙ». Αναγνωρίζω πλήρως και εκτιμώ ιδιαιτέρως ότι χωρίς αυτή την υποτροφία αυτή η ολοκλήρωση αυτής της διδακτορικής διατριβής κατά πάσα πιθανότητα δεν ήταν δυνατή.

Η διαδικασία ολοκλήρωσης ενός διδακτορικού είναι εξαιρετικά δύσκολη, κυρίως σε σχέση με τις ψυχολογικές συνιστώσες της. Για το λόγο αυτό, η αναγνώριση ότι το ερευνητικό έργο του υποψηφίου διδάκτορα είναι όντως πολύτιμο παρέχει ιδιαίτερα σημαντική ώθηση. Θα ήθελα να ευχαριστήσω τις οργανωτικές επιτροπές των ακόλουθων συνεδρίων τα οποία μου απένειμαν βραβεία για τις ακαδημαϊκές εργασίες μου: το 2nd ESWC Summer School 2012 για το βραβείο καλύτερου poster, το 7th International Conference on Internet and Web Applications and Services (ICIW '12) για το βραβείο καλύτερης εργασίας, και το ICTAI 2014, τόσο για το βραβείο καλύτερης φοιτητικής εργασίας τους όσο και για το βραβείο καλύτερης εργασίας.

Θα ήθελα να ευχαριστήσω θερμά την κοπέλα μου, Μιρέλα, που έκανε υπομονή όποτε ήμουν αγχωμένος, ή θυμωμένος ή απογοητευμένος, και τα τρία όχι ιδιαιτέρως ασυνήθιστα φαινόμενα κατά τη διάρκεια αυτών των ετών. Τέλος, όπως καθετί σημαντικό στη ζωή, μια διδακτορική διατριβή δεν μπορεί να ολοκληρωθεί χωρίς την οικογενειακή υποστήριξη. Θα ήθελα να εκφράσω την ευγνωμοσύνη μου προς τους γονείς μου, Κύρο και Ελένη, για την αδιαμφισβήτητη αγάπη και υποστήριξή τους. Ακόμα και όταν εγώ πίστεψα ότι δεν μπορούσα να το κάνω, οι ίδιοι δεν αμφέβαλαν ποτέ.

Στους γονείς μου.

Και για σας τους δυο που δεν προλάβατε να δείτε το τέλος,
αλλά ήσασταν εκεί σε όλο το ταξίδι. Αυτό έχει σημασία.

CONTENTS:

List of Figures:

List of Tables:

List of Algorithms:

Chapter 1.

Introduction

In the last two decades, Service Oriented Computing (SOC) has redefined the way software applications are implemented and consumed; Service Oriented Architecture (SOA) implements loosely coupled and platform-independent distributed computing based on specific standards (Papazoglou & Heuvel, 2007), with web services being its basic building blocks. Their purpose is to aid the modular, low cost and non-time consuming development of software applications, despite the heterogeneity in the environment. A web service is a "*self-describing, self-contained software module available via a network*" (Papazoglou, 2008); it is described and discovered through the use of standard languages, such as the Web Services Description Language (WSDL) (Chinnici et al., 2007); and is usually accessed through the use of internet-based protocols.

Semantic web services are web services with a formal description of their capabilities in an ontology language, such as OWL (McGuinness & van Harmelen, 2004). This semantic markup is important as in that way these capabilities can be unambiguously described, identified and interpreted in terms of their meaning and not solely based on their syntax. Ontologies are used so as to provide a common understanding of a particular domain; as such, semantic web services can be used to solve interoperability problems between heterogeneous systems, with transparency over the underlying technologies used to implement them and the platforms they are based on. Furthermore, the semantic markup of a web service in a language such as OWL-S (Martin et al., 2004), that is, of what a service does, how it is used and the effects it has, also enables the automation of tasks such as web service discovery, selection and composition.

Web services exist and operate in an ever expanding environment; the number of available web services is growing continuously making the web services' discovery phase more difficult. In the past few years, companies worldwide have begun to offer web services that allow the use of a subset of their functionality, such as Twitter[1], LinkedIn[2] or OPAP[3]. Moreover, web services publication websites witnessed a significant increase in the number of web services registered and used; ProgrammableWeb[4] in specific claims that the web service Application Programming Interfaces (APIs) registered in it have increased from only a few hundred in 2005 to more than 10,000 in 2013 (ProgrammableWeb, 2014) and currently to almost 14,000. Finally, the use of web services does not only facilitate the low cost and non-time consuming development of applications, but is also a significant source of revenue for business; for example, Amazon Web Services (AWS) alone generated $1.82 billion in revenue in just the second quarter of 2015, an increase of over 80% in comparison to 2014 (Novet, 2015). As such, the widespread adoption and use of technologies such as social networks, the Web of Things (Guinard, 2011) and cloud computing has become a positive factor in the continuing growth of web service related standards and research.

[1] https://dev.twitter.com/rest/public
[2] https://developer.linkedin.com/docs/rest-api
[3] http://www.opap.gr/en/web/guest/web-services
[4] http://www.programmableweb.com/

1.1 Web Service Composition and Non-Deterministic Planning

Although there is a vast amount of available web services, a single web service with a desired functionality may not always be available; in such cases, a combination of several atomic or composite web services may be able to offer the requested functionality. In this sense, the true value of SOA lies in being able to compose multiple web services in order to create a composite one with the new functionality, in a manner that dramatically reduces the required cost and time and is based on the reusability of its components.

The plethora of web services as well as the fact that web services may change their interface or part of their usage multiple times throughout their lifespan, hinders even a human expert from manually completing the task of web service composition. More importantly, even if web services remain static, it is always probable that their execution may not be successful or have the desired result. As such, non-determinism should be taken into account during web service composition. A web service composition process should be able to handle such issues effectively, by detecting and responding to these changes automatically, in a way that a human would probably not be able to.

AI planning (Russell & Norvig, 2003) (Ghallab et al., 2004) is the task of coming up with a partially ordered set of actions that achieve a goal. By considering web service composition as the task of finding a partially ordered set of web services, such that their aggregate behavior achieves the intended goal, the connection between web service composition and AI planning is apparent. For this reason, the relative literature focuses on automated non-deterministic web service composition methods that view the problem as one in which web services are atomic planning operators and utilize AI planning techniques to tackle it. AI planning problems may concern either fully or partially observable, or not observable at all environments, with the actions comprising them having fully predictable (deterministic) effects or unpredictable (non-deterministic) ones, with or without a probabilistic model over their occurrence. Since web services are used in the real world, their success cannot be guaranteed. This thesis assumes that web service composition is inherently a non-deterministic problem and, as such, non-deterministic planning techniques have to be used to tackle it. Automated web service composition comprising non-deterministic web services with complete information has been shown to be EXP-hard (Nam et al., 2011).

1.2 Contribution

This thesis starts with a critical review of non-deterministic planning approaches and web service composition methods; it is an extensive and in depth review of such approaches, which, to the best of our knowledge, is currently missing from the recent literature. The literature review reveals that non-deterministic web service composition approaches are scarce and that no standard evaluation set in relation to web service composition exists. Moreover, there is a plethora of relevant methods that either

evaluate their methodology on case studies without referring to quantitative criteria or not at all. This trend has begun to be reversed in the recent years, however, the problem still remains.

For this reason, the thesis proposes a set of evaluation use case scenarios that is based on an existing, open, test collection of semantic web services. The set can be used to evaluate a wide variety of web service composition methods, ranging from deterministic to non-deterministic ones containing preferences and iteration. Importantly, the use case scenarios are described in detail, both in terms of their goals and of the web service descriptions and the inputs/outputs that comprise them and, as such, can easily be reproduced in other systems.

The main contribution of the thesis, however, lies in the implementation of a web-based application related to semantic web services that supports multiple phases of web service composition. The application, named **MADSWAN**, is based on open source software components that utilize the current web service standards, in the spirit of web services themselves, that is, having a modular architecture and making efficient re-use of existing modules. A typical user of **MADSWAN** can advertise a new web service in its online registry, as well as retrieve and edit the web service descriptions stored in it. Moreover, it is possible to manually create workflows by employing OWL-S control constructs and bind them to web service descriptions in the online registry; finally, users can automatically create composite web services that meet complex goal specifications, both assuming a deterministic environment and a non-deterministic one.

The thesis also introduces a new non-deterministic web service composition algorithm, called **MAPPPA2** (an anagram of **A**nytime **P**robabilistic **P**lanning via **A**lternative **P**lan **M**erging), which assumes a fully observable probabilistic domain, with web services being defined in regards to preconditions and effects over ontological concepts, that is, at a functionality level. The approach adopts two basic assumptions. First, it is assumed that it is always possible from any state in the problem to return to the initial one. This is considered possible through the assumption that the actions in the planning domain do not have delete effects. This is a plausible assumption for the domain of web services, since web services usually concern information gathering. However, even web services that perform transactions, e.g., purchasing goods, cannot prevent the execution of another web service of the same type, provided that there is no limit in the available resources (e.g., money). Second, it is assumed that re-executing the same web service within a short period of time, i.e., within the current plan execution episode, will result in the same output. This is a realistic assumption in a web service composition domain, as, e.g., a web service that produces undesirable effects due to network failure or the unavailability of a product in a web store, is not probable to be available again in a very short amount of time.

MAPPPA2 is based on an anytime contingent planning framework; a determinized version of the original probabilistic problem is produced, while retaining the information of the original problem in regard to the probabilities and cost of the web services/actions. This problem is then solved repeatedly by a deterministic planner, generating multiple plans which are then merged in a decision tree, using the initial non-deterministic actions as its decision nodes. Merging prioritizes the alternative plans

according to their expected utility, which is updated each time a plan is selected for merging, by taking into account potential common web services between the newly merged plan and the remaining ones.

This work has been inspired by planning methods that determinize the original non-deterministic problem and then employ classical deterministic planners to solve it; it is, however, specifically targeted for web service composition problems and differs from the existing related work in that it takes the probabilities and the cost of the original non-deterministic actions into consideration, while generating a contingent plan. This algorithm was proven to be highly efficient in the evaluation performed based on the proposed use case scenarios.

1.3 Structure of Thesis

The rest of the thesis is structured as follows:

Chapter 2 provides a critical review of non-deterministic AI planning; the related background theory is presented, along with a categorization of planning approaches as probabilistic planning, determinization methods, planning in the belief state space or translation methods.

Chapter 3 describes the web services and the standards on which they are based. Then, it expands on the definition of web services to include semantic web services and presents a plethora of semantic web content standard and semantic web services' approaches. Finally, it presents a survey in regard to web service composition approaches, whether manual, semi-automated, or fully automated ones and focus on non-deterministic AI planning ones.

Chapter 4 introduces in detail the requirements that govern the implementation of the web service composition system. It presents a set of requirements related both to web service composition process, e.g., in regard to the elements of the composition or the data model that it should abide to and to the system itself, that is, general requirements and the system's goals. Moreover, a set of use case scenarios are presented, mostly based on an existing, publicly available test collection of semantic web services. These scenarios are used in the evaluation of the manual and the automated web service composition modules, both in regard to the deterministic and the non-deterministic planning algorithms.

Chapter 5 presents the web-based application and examines the implementation details of its modules; in specific, it presents the basis of the application, the semantic web service registry along with its sub-modules and reviews its conformance to the requirements set in Chapter 4. Afterwards, it presents the manual web service composition module and reviews its conformance to the requirements set in Chapter 4, as well as its ability to implement the use case scenarios presented in the same section.

Chapter 6 presents the non-deterministic web service composition algorithm and the quantitative evaluation of the automatic module of the proposed system, based on the previously presented use case scenarios. Moreover, it describes a case study referring to the incorporation of web service composition and automated planning into intelligent calendars.

Finally, chapter 7 concludes the dissertation and poses future research directions.

The appendix contains additional information concerning the presented use case scenarios.

1.4 Associated Publications

1.4.1 Conference Papers

- Markou, G. & Refanidis, I., 2012. Towards Automatic Non-Deterministic Web Service Composition. In *7th International Conference on Internet and Web Applications and Services (ICIW'12)*. Stuttgart, Germany, 2012.
    - This paper describes the ongoing work of the web based system presented in Section 5.1; specifically, it presented the basis of the application, i.e., the registry, and a brief description of the use case scenarios in Sections 4.2.3-4.2.5.
- Markou, G. & Refanidis, I., 2012. Towards an Automatic Non-Deterministic Web Service Composition Platform. In *8th International Conference on Next Generation Web Services Practices (NWeSP '12)*. Sao Carlos, Brazil, 2012.
    - This paper describes our ongoing implementation of the system presented in Chapter 5; specifically, it presented the manual web service composition module and the implementation of the use case scenarios through it.
- Markou, G. & Refanidis, I., 2013. Mad Swan: A Semantic Web Service Composition System. In *10th Extended Semantic Web Conference (ESWC '13)*. Montpellier, France, 2013. Lecture Notes in Computer Science.
    - This paper gives a high-level description of the implemented web based system presented in Chapter 5 along with its modules, excluding the automated web service composition ones.
- Markou, G., 2013. Decision Making in Non-Deterministic Environments. In *Doctoral Consortium of the 23rd International Conference on Automated Planning and Scheduling (ICAPS '13)*. Rome, Italy, 2013.
    - This is the participation of the PhD student at the ICAPS-2013 doctoral consortium, presenting there the progress of his work and his future goals.
- Markou, G. & Refanidis, I., 2014. Anytime Planning for Web Service Composition via Alternative Plan Merging. In *26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '14)*. Limasson, Cyprus, 2014.
    - This paper presents the contingent web service composition algorithm described in Section 6.1 along with the simple contingent use case scenario presented in Section 4.2.6 and a preliminary version of a subset of the evaluation results in Section 6.2.
- Markou, G., Alexiadis, A. & Refanidis, I., 2015. Web Services and Automated Planning for Intelligent Calendars. In *6th Italian Workshop on Planning and Scheduling (IPS '15)*. Ferrara, Italy, 2015.
    - This paper describes the integration of web service composition and AI planning into intelligent calendars that is presented in Section 6.3

1.4.2 Journal Articles

- Markou, G. & Refanidis, I., 2013. Composing semantic web services online and an evaluation framework. *International Journal on Advances in Internet Technology*, 6(3-4), pp. 114-31.
    - This article includes a preliminary version of the algorithm that is presented in Section 6.1, along with the experimental evaluation of the deterministic automated web service composition module that is presented in Section 6.2.1.
- Markou, G. & Refanidis, I., 2015. Cost-Sensitive Probabilistic Contingent Planning for Web Service Composition. *International Journal of Artificial Intelligence Tools (IJAIT)*, (Special issue on ICTAI 2014). *To appear*.
    - This article significantly extends and improves over the conference paper (Markou & Refanidis, 2014); it describes the contingent web service composition algorithm presented in Section 6.1 along with the evaluation results presented in Section 6.2.
- Markou, G. & Refanidis, I., 2016. Non-Deterministic Planning Methods for Automated Web Service Composition. *Artificial Intelligence Research (AIR)*, 5(1), pp. 14-35.
    - This article presents a subset of the critical review of non-deterministic planning in Chapter 2 and the automated web service composition survey in Section 3.3.4.

1.4.3 Awards

- Best paper award at the 26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '14).
- Best student paper award at the 26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '14).
- Best paper award at the 7th International Conference on Internet and Web Applications and Services (ICIW '12).
- Best poster award at the 2nd ESWC Summer School 2012.

Chapter 2.

Non-Deterministic Planning

Planning is the problem of deciding which actions have to be executed next in order to achieve a goal (Geffner, 1998). These actions can have fully predictable (deterministic) or unpredictable (non-deterministic) effects, with or without a probabilistic model over their occurrence. Moreover, the world can be either fully observable, or partially observable, or not observable at all, the latter being the case for conformant planning. In a deterministic setting, the outcome of an action applied in a specific state is fully predictable and results in a single state; if represents the state before the execution of action a and s' the state after its execution, then the transition function is $s' = f(s, \alpha)$. In a non-deterministic setting though, the next state depends on which effect of the action actually occurred. If the model is probabilistic, then an action can have n possible mutually exclusive effects, $e_1, e_2, \dots, e_n$, each e_i with probability p_i of occurrence, $0 < p_1, p_2, \dots, p_n \leq 1$, so that $\sum_{i=1}^{n} p_i = 1$. As such, the possible resulting states are those that have $P_a(s' | s) > 0$, with $\sum_{i=1}^{n} P_a(s' | s) = 1$. A domain may allow for observations that provide feedback about the action results and form the beliefs in regard to the possible states that occur.

This chapter provides an up-to-date survey of non-deterministic planning methods, focusing on contingent planning and the support of non-determinism in their formalisms. First, the theoretical background in relation to non-deterministic planning is reviewed; then the first – naïve – non-deterministic methods is briefly presented, focusing on a line of research that simplifies the original domain by removing its non-deterministic elements and then solves it using a classical planner. Afterwards, non-deterministic methods that search through the belief space for a solution to the problem, using a compact representation of their belief states are presented. Finally, the state-of-the-art planners that translate the original non-deterministic domain into an equivalent one in a sound and - in most cases - complete manner are reviewed. Part of the following work has been described in (Markou & Refanidis, 2016).

2.1 Problem formulation

A non-deterministic planning domain is a triple $D = (S, A, \gamma)$, where S is a finite set of states, A is a finite set of actions and $\gamma : S \times A \rightarrow 2^S$ is the state-transition function. A fully observable non-deterministic planning problem is a triple $P = (s_0, s_g, D)$, where $s_0 \in S$ is the initial state and $s_g \subseteq S$ is the goal (Fu et al., 2011). Given such a planning problem, the set of states to which any action has been assigned under policy π is represented by S_π, where $S_\pi \subseteq S$, and the set of reachable states from s using π by $S_\pi(s)$; therefore, a policy is a function $\pi : S_\pi \rightarrow A$. That is, $\forall s \in S_\pi : \exists\, a \in A$, such that $(s, a) \in \pi$, and given a state $s \in S_\pi$, the solution dictates that the action to be executed is $\pi(s)$. Finally, $S_\pi(s)$ denotes the set of states reachable from s using π.

In probabilistic environments, a planning domain is defined by $D = (S, A, \gamma, Pr)$. S, A and γ are the same as in the general definition of non-deterministic domains and $Pr : S \times A \times S \rightarrow [0,1]$ is the probability-transition function. If the domain also incorporates actions costs, then $D = (S, A, \gamma, Pr, Co)$, with $Co : S \times A \times S \rightarrow \mathbb{N}$ being a bounded cost-function. The set of all actions that

can be applied to a state s is $A_D(s) = \{\alpha \in A : \gamma(s,\alpha) \neq 0\}$. A planning problem is also defined in the same way as a general non-deterministic one.

For Fully Observable Probabilistic (FOP) domains, the states of the world can be completely observed at runtime. Following the previous notations, solutions to FOP planning problems can be (total) policies $\pi : S \rightarrow A$, or partial functions from a set S_π *to* A, with $s_0 \in S_\pi$ and S_π closed under policy π; then, for $s \in S_\pi$, the solution π dictates to apply action $\pi(s)$ in state s (Kuter, 2004). If $a \in A_D(s)$ then $Pr(s,a,s')$ is the probability of reaching state s' if action a is applied to s.

Σ_π denotes the execution structure for π, whereas V_π denotes the set of the states in Σ_π. For any two states s and $s' \in V_\pi$, if there is a path in Σ_π from s to s', then s' is a π-descendant of s in Σ_π. Then, if for a state s it holds that there is no π-descendant of it that satisfies the goal, then s is a dead-end (Kuter, 2004). A policy π is closed with respect to a state s iff $S_\pi(s) \subseteq S_\pi$. If the goal state can be reached using π from all (π-reachable) states $s' \in S_\pi(s)$, then π is considered proper w.r.t. a state s. Iff π is both closed and proper w.r.t. the initial state s_0, then it is deemed a valid solution (Bryce & Buffet, 2008). Valid solutions can be either strong, or strong cyclic; a policy π is acyclic if for all possible executions $\tau = s_0 s_1 s_2 \ldots$ of π from s_0, it holds that $s_i \neq s_j$, for any $i \neq j$ (Ramírez et al., 2013). In general, though, solutions to non-deterministic problems can be either weak (with a chance of success), strong (guaranteed to achieve the goal despite non-determinism), or strong cyclic (guaranteed to achieve the goal with iterative trial-and-error strategies) (Cimatti et al., 2003).

An alternative definition of different type of solutions is in terms of probability of success; Assuming that each outcome of an action α from state s has a non-zero probability, weak plans reach the goal with a probability p, $0 < p < 1$. Strong plans reach the goal with probability $p = 1$, in at most n steps; assuming that each outcome of an action a from state s has a non-zero probability, then a strong plan reaches the goal in at most $n \leq |S|$ steps, since the execution path of a strong (acyclic) plan cannot pass through the same state twice. Strong cyclic plans reach the goal with probability $p = 1$, without a bound in the number of required steps, that is, infinite paths are allowed, but with probability $p = 0$ of reaching the goal. It should be noted that since strong solutions to a planning problem are a subset of the strong cyclic solutions, there are problems in which strong solutions do not exist, but strong cyclic ones do (Cimatti et al., 2003).

(Muise et al., 2014) define a partially observable environment (Bertoli et al., 2006) with sensing actions, i.e., a PPOS domain, in a fashion similar to (Bonet & Geffner, 2011), as follows: a PPOS domain is a tuple $D = \langle A, O, I, G \rangle$, A representing the set of actions and O the set of observations; I is a set of clauses that specifies the initial state and G is a conjunction of atoms specifying the goal. Literal l holds in a state s iff s assigns l to be true. As l is a set of clauses, it may hold in a number of states; the set of all states where l holds comprise the belief state b.

For $a \in A$, its preconditions $Pre(a)$ are a conjunction of atoms and its conditional effects $Eff(a)$ are a set of pairs $\langle c, l \rangle$ where c is a condition and l is a literal, implying that l occurs in the next state if c holds in the previous state. An action a is applicable in a state s iff $Pre(a)$ holds in s; it is applicable in a belief state b iff a is applicable in every $s \in b$. In a similar manner, when an action a is performed in b, a successor belief state b' is constructed by performing a in each $s \in b$. Observations $o \in O$ are

also represented by pairs $\langle c, l \rangle$; when c is true, o denotes the truth value of l. This result is achieved by treating observations as "special" actions that have c as the precondition and l as the effect; Thus, when such an observation action o is executed in b, the successor belief state b' is the maximal set of states in b agreeing on l. Moreover, a belief state b' is reachable from b if there is a sequence of actions and/or observations that when executed in b result in b'.

Finally, contingent planning, that is, planning that outputs plans that are conditional, containing different branches for different observation action results, can be constructed either in an offline or online fashion. The former allows for the generation of solutions with decision points based on the outcomes of observation actions and guarantees as the achievement of the goal. The resulting plans are larger but more general and have the ability to avoid dead-ends (Muise et al., 2014). However, the size of full contingent plans that are represented by trees is exponential in the maximum number of observations that are comprised in a single branch of the tree (Palacios et al., 2014). Offline methods can output both belief policies and tree ones. A belief policy is a function mapping belief states into actions, whereas a tree policy is a function mapping executions into actions (Palacios et al., 2014), with the former being represented by graphs and the latter by trees (Geffner & Bonet, 2013). Online methods, on the other hand, focus on choosing the appropriate action for execution for the current belief state.

2.2 Probabilistic Planning

One of the first methods that used classical planning techniques to tackle probabilistic problems was Mahinur (Onder & Pollack, 1999), a probabilistic partial-order planner that generates a (weak) base plan and then improves it with three types of repairs:

- **Corrective repair of branching points**, used in order to decide what should be done if the desired outcome of a branching action does not occur.
- **Preventive repair**, used to ensure that the desired outcome of a branching action actually occurs.
- **Replacement**, used to remove a branching action in order to replace it with an alternative one.

Mahinur searches for the contingencies, which, if they were to fail, would affect the plan the most and uses the aforementioned repairs to deal with them. Moreover, it incorporates a method similar to plan merging, by creating plans that contain joined branches. However, the first attempts to utilize classical planning techniques such as partial order planning, e.g., Mahinur or Cassandra (Pryor & Collins, 1996), are not competitive with subsequent, more modern planners.

APROPOS2 (Majercik, 2002) is an anytime probabilistic contingent planner that chooses the most likely outcomes of all actions at any time; it then generates a base plan under this assumption and, if time permits, considers less likely outcomes of actions to modify it. APROPOS2 is based on a

conversion of the original probabilistic problem to a satisfiability (SAT) one (Garey & Johnson, 1979) and generates solutions for partially observable probabilistic propositional problems.

(Meuleau & Smith, 2003) present an explicit and general method for automated bounded branching planning (Bonet, 2010), with their main focus being fully-observable problems. They define three variants of k-contingency planning: a general one, which outputs the best plan with at most k branches in total; linear k-contingency planning, which outputs contingent plans that consist of a main sequence of actions with k branches attached to it, with these branches not having any other branch attached to them; and, finally, balanced k-contingency planning, with plans having at most k branch points in each trajectory that leads to the goal from the initial state. The authors opt for an anytime, optimal, balanced k-contingency planning method that is based on a formalization of Partially Observable Markov Decision Processes (POMDPs) (Kaelbling et al., 1998) to compute plans for non-deterministic problems with full observability (Bonet, 2006). Their method allows for the use of Dynamic Programming (Sniedovich, 2010) to speed up the state space search. The problem is viewed as a POMDP, where the number of contingencies is incorporated as a state feature and k copies of the state space are generated. Then, each time a sensing action is executed, the current state transitions to a logically equivalent one in which less contingencies exist. The algorithm outputs a tree-shaped plan containing up to k branches. For $k = 0$, the output is an action sequence solving a conformant problem; for $k = \infty$, the algorithm outputs the optimal policy for the problem at hand. The method was evaluated in two domains, taken from (Kaelbling et al., 1998) and (Hyafil & Bacchus, 2003).

2.3 Determinization Methods

One of the prevalent methods to tackle non-deterministic problems is the determinization of the original non-deterministic domain and the generation of a solution to the deterministic problem that is also a weak solution for the original problem. Table 2.1 summarizes the methods that are presented in this section, along with some of their characteristics.

The first efficient method that utilized this approach was FF-Replan (Yoon et al., 2007), the winner of the 2004 International Probabilistic Planning Competition (IPPC-04) (Younes & Littman, 2004) and the top performer in IPPC-06[5]. FF-Replan utilizes the FF (FastForward) planner (Hoffmann, 2001) (Hoffmann & Nebel, 2001) to generate a single plan for a deterministic version of the original Markov Decision Process (MDP) (Puterman, 1994) problem, whereas it produces a new plan each time one of its simulated executions of the current plan results in an unexpected state (from that state to the goal state). (Yoon et al., 2007) introduced two methods of creating a determinized version of the original non-deterministic problem. The first was single-outcome determinization, which selects only one probabilistic effect as the outcome of each probabilistic action, without taking into

[5] Available at http://ldc.usb.ve/~bonet/ipc5/, Accessed on February 11, 2015

account the rest of its effects or its probabilities. The second one, all-outcomes determinization, creates a new action for each of the outcomes of the probabilistic effects in the original non-deterministic action.

Table 2.1. Synopsis of determinization methods

Method	***Online/ Offline***	***Complete***	**Replanning**	***Determinization***	***Plan strength***
FF-Replan	Online	-	•	All-outcomes Single-outcome	Weak
NDP	Offline	•[1]	-	All-outcomes	Strong cyclic
RFF	Offline	-	•	Single-outcome	Weak + robust[4]
FF-Hindsight	Online	-		Hindsight	Weak + robust[4]
(Jiménez et al., 2006)	Online	-	•	All-outcomes + costs	Weak + robust[4]
(Foss et al., 2007)	Online	-	•	All-outcomes + costs	Weak + robust[4]
(Dearden et al., 2003)	Offline	-		Single-outcome	Weak + robust[4]
GPT	Offline	•[2]		Single-outcome dynamic[3]	Weak + cost optimal

1 Under conditions.

2 For certain classes of problems.

3 Multiple single outcome determinizations/ trials are used.

4 The output plans are weak but with some guarantees against failing / have been made more robust against contingencies.

Assuming a non-deterministic planning domain D with a set of actions $A = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$, then for an action $\alpha_i \in A$ with k different probabilistic outcomes, the all-outcomes determinization is the set of deterministic actions $Det_{\alpha_i} = \{\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{ik}\}$, along with a state-transition function $\bar{\gamma}$ such that for every state s, $\gamma(s, \alpha_i) = \bar{\gamma}(s, \alpha_{i1}) \cup \dots \cup \bar{\gamma}(s, \alpha_{ik})$ (Kuter et al., 2008); the single-outcome determinization for action α_i simply chooses one of the actions in Det_{α_i} to substitute α_i.

The all-outcomes determinization process is also used by the Fast Incremental Planner (FIP) (Fu et al., 2011) and by NDP (Kuter et al., 2008). NDP is very similar to FF-Replan; in contrast to FF-Replan, though, NDP may also be sound and complete provided that a sound and complete planner

is employed. Moreover, NDP, as well as FIP that builds on NDP, generate strong cyclic solutions. FF-Replan on the other hand, generates a single weak plan every time it is executed.

The all-outcomes determinization ignores the probabilities attached to the probabilistic outcomes, thus rendering the planning methods overly optimistic. That is, planners simply choose the most convenient action, one that achieves the desired effect, regardless of its underlying (true) likelihood. As such, the planner does not attempt to avoid actions with trivial probability of succeeding in outputting the desired effect and can select, at any time, the action with the most helpful outcome. FF-Replan does not offer any quality guarantees in general and, in essence, it is not a contingent planning method, although similar to such ones. However, it was very successful in planning competitions and was used as the basis for a plethora of other non-deterministic planning methods, such as RFF (Teichteil-Koenigsbuch et al., 2008).

In contrast to FF-Replan, RFF generates offline policies and provides some guarantees against failing; if a failure still occurs during execution, RFF updates its previously generated policy. It generates multiple plans from the initial state(s) to the goal using single-outcome determinization (in particular, the most probable outcome in each case). Then, it creates a policy with a low probability of causing replanning – and not necessarily of reaching the goal – during execution through Monte-Carlo simulation (Berman & Paul, 2005). Monte-Carlo simulation computes the probability that a partial policy will fail and, if it is sufficiently low, the current policy is returned; otherwise, the policy is further expanded. RFF favors plan modification instead of replanning from scratch, whenever possible. Moreover, if the execution results in a state for which the policy does not specify any action, RFF tries to reach neighbor states for which an action has already been specified, instead of replanning from scratch. However, as a last resort, RFF may require replanning, so it is not optimal and it does not handle dead end states.

(Yoon et al., 2008) present an alternative, dynamic determinization process; FF-Hindsight, is a generalization of FF-Replan, which, instead of solving a single determinized problem, randomly produces multiple determinized problems and combines their solutions. The value of each state is approximated by sampling these (non-stationary) problems originating from it, with the selected outcome for each action varying each time; the problems are then solved in hindsight and the values of the states are combined. In that way, the determinization process is no longer static as in FF-Replan, which can be simply considered an optimistic approximation of hindsight optimization. However, this process is computationally expensive; as (Yoon et al., 2010) note, the amount of computation required at each state is linear in the number of actions applicable in it, as well as in the number of determinized problems generated. For this reason, in (Yoon et al., 2010) several improvements to FF-Hindsight are incorporated in FF-Hindsight+.

First, in order to avoid estimating the value of actions that may not be useful in the final plan, a method for detecting only the potentially useful ones is proposed. Second, instead of generating new plans from scratch, the determinism in the domain is utilized by reusing relevant plans in case a state transition occurs as expected by previously sampled determinizations. In that way, new plans are

generated only when the states evolve in a different manner than the one dictated from their (anticipated) trajectory. Finally, the action evaluation method is modified in order to make it more probable that at least one deterministic plan reaches a goal; this is achieved by using the all-outcomes determinization instead of the sampled ones, thus guiding the search towards paths that reach the goal, regardless of their actual probability. As a result of these changes, the experimental evaluation of FF-Hindsight+ proved it to be competitive with past winners of the International Probabilistic Planning Competition, namely FF-Replan, FPG (Buffet & Aberdeen, 2006) and RFF, respectively for IPPC 2004, 2006 and 2008.

(Jiménez et al., 2006) utilized the all-outcomes determinization in an attempt to take the actual outcomes' probabilities into account. The method combined the determinization with a translation of these probabilities to associated cost values for the new deterministic actions' outcomes, each denoting the risk of failing. In that way, a metric planner compliant with the Planning Domain Description Language (PDDL) (McDermott, 1998) was able to improve the robustness of the plans by trying to minimize the sum of the negative logarithms of the success probabilities, which in turn minimizes the product of the failure probabilities.

(Foss et al., 2007) use a determinization translation process for incremental contingency planning; an initial deterministic seed plan is generated that is then iteratively improved through an analysis and repair cycle. The original plan is analyzed so as to find outcomes that are relatively probable and result in dead ends and then precautionary actions are added in relation to the most problematic outcomes. Recoverable failures, on the other hand, are left as-is in the plan and are repaired through online replanning.

The method presented in (Dearden et al., 2003) formulates the probabilistic domain in a slightly different manner than the aforementioned methods. The initial state may involve uncertainty about the variables that participate in it, with the uncertainty being characterized by probability distributions over their different values. The same is true for the probabilistic effects of actions. A seed plan is initially constructed though the use of a deterministic planner, assuming that each action is executed as dictated by its expected behavior. Then, additional (deterministic) branches are generated and added to the existing plan incrementally, so as to improve the overall utility of the plan. The best place to insert a branch is computed based on the amount of utility gained in the seed plan from adding it at the specific place. Since this computation is expensive, an approximation is used through Monte Carlo simulation and by propagating utility distributions through a graph. This process is repeated either until the plan is sufficiently robust, or the process runs out of time.

(Bonet & Geffner, 2000) formulate the problem of non-deterministic planning in partially observable domains as a search in belief space, that is, with each belief state representing a set of states. The problem is defined as one in which the initial state is unknown but sensing is possible during execution time. Belief states can also represent probability distributions over states, instead of simple sets of states. The proposed method, GPT (General Planning Tool), can tackle both conformant planning problems and problems formulated as MDPs and POMDPs. GPT uses heuristic search to

find plans; particularly, it makes use of a generalization of the LRTA* (Korf, 1990) algorithm named RTDP (Real Time Dynamic Programming) (Barto et al., 1995). RTDP constructs a policy by executing multiple simulations of an execution from an initial state to a goal one (or until a step limit is reached), based on the current approximation of the final policy. After each action is selected/executed, this policy is updated. An admissible heuristic, the max heuristic (Bonet & Geffner, 2001) is used to guide the search. It measures the positive interaction in the belief states and provides an estimate of how difficult it is to achieve the sub-goals in the worst case. A variation of the GPT planner, mGPT (Bonet & Geffner, 2005), extracts different classes of lower bounds from determinizing the original problems and uses them in combination with various heuristic-search algorithms that use these lower bounds to focus their policy updates. mGPT competed in IPPC-04 achieving mixed results.

2.4 Planning in the Belief State Space

This sub-section presents a set of planners that tackle the problem of non-deterministic planning in belief state space, while trying to scale up efficiently. Table 2.2 provides a synopsis of the presented methods and their characteristics.

The Model Based Planner (MBP) (Bertoli et al., 2001) is a general non-deterministic planner that deals with domains that incorporate different degrees of observability: fully, partially observable, or not observable at all. It supports uncertainty in the initial state and the actions, as well as temporally extended goals. Domains are considered to be non-deterministic finite-state machines (FSMs) (Hopcroft & Ullman, 1979) and plans are defined as deterministic FSMs. The plans are generated with the help of Symbolic Model Checking techniques (McMillan, 1992), with belief states being defined as sets of states that contain common observations and represented as Binary Decision Diagrams (BDDs) (Bryant, 1992), so as to efficiently handle the large state space and the FSMs.

In specific, belief states are represented as propositional formulae and the state space search is conducted as a set of logical transformations over them. The use of BDDs allows not to enumerate the state space explicitly and to ignore irrelevant information in relation to the belief states. However, the size of BDDs is sensitive to the order of variables and may be very large, while the computation of the belief states' successors requires the generation of intermediate formulae that may be of exponential size (To et al., 2011). Thus, in certain cases, the methods that make use of them can suffer from an explosion in state space (Cimatti et al., 2003) and, as a result, may not scale well.

Table 2.2. Synopsis of methods for planning in the belief state space

Method	*Full / Partial / No Observability*	*Scale up method*	Plans
MBP	•/•/•	BDDs	Strong/ Strong cyclic[1]
YKA	•/•/-	BDDs	Strong
POND	-/•/•	LUG	Strong
PC-SHOP	-/•/-	Domain-specific knowledge	-
ND-SHOP2	•/-/-	Domain-specific knowledge	Strong cyclic
YoYO	•/-/-	Domain-specific knowledge + BDDs	Strong / Strong cyclic
NDP2	•/-/-	Abstraction + All-outcomes determinization	Strong cyclic
Gamer	•/-/-	Translation to two-player turn taking game + BDDs	Strong / Strong cyclic
FIP	•/-/-	State re-use + goal alternative	Strong cyclic
PRP	•/-/-	State relevance + All-outcomes determinization	Strong cyclic
Contingent-FF	-/•/-	Implicit representation of belief states	Strong
FPG	-/•/-	Local optimization + function approximator for actions	-
DNFct	-/•/•	DNF Representation of belief states	-

1 Depending on the underlying algorithm, under full observability, MBP may generate weak, strong or strong cyclic plans. Under partial or no observability, the plans are strong.

MBP utilizes various planning algorithms to solve the problem, and depending on its type, generates plans of different form and guarantees of reaching the goals. In the case of partial or no observability, the generated plans are strong. (Cimatti et al., 2003) compare MBP to various other planners, namely GPT, UMOP (Jensen & Veloso, 2000), SIMPLAN (Kabanza et al., 1997), QBFPLAN (Rintanen, 1999) and SGP (Weld et al., 1998). MBP proved effective in a wide range of problems, although a blow up in its state space occurred even in problems with a limited number of state variables. MBP clearly outperformed only UMOP, as its comparison with the rest of the planners indicates that each suffers from a different drawback, e.g., SGP's bottleneck stems from the enumeration of the initial states. However, the systems under evaluation solve different problems, formulated in different languages, with each problem encoding having an impact on the planners' performance.

Similar to MBP, BDDs are also used in (Rintanen, 2002) who presents two backward search algorithms and argues that forward search algorithms are required to choose between branching and performing an action; the former leading to larger plans, the latter exchanges useful branch points for

potentially smaller plans. The steps of these algorithms represent both the application of actions and branching in the plan. In the case of partial observability, the first algorithm exhaustively compute sets containing all the maximal belief states with an increasing distance to the goal (belief) state, with the distance representing the maximum number of actions needed to reach the goal from the belief states in the set. The second one heuristically selects a single belief state at a time based on its cardinality. Both algorithms are suboptimal and generate solutions in the form of Directed Acyclic Graphs (DAGs) (Thulasiraman & Swamy, 1992). In the case of full observability, the algorithms perform breadth-first search backwards from the goal states, traversing the entire search tree up to a level.

In this case the plan search is essentially a computation of the distance from every belief state to the goal; based on these distances it is straightforward to extract a plan. This blind enumeration of the belief state space, however, proved to be inefficient, as their high number becomes a bottleneck to the problem's solutions. This led to a new factored representation of the belief space that is able to identify new belief states without blind enumeration, while also allowing algorithms to generate non-deterministic plans backwards (Rintanen, 2003). In this way, all branching points can be automatically handled, and in the case of fully observable planning problems, the representation allows for a single BDD to represent the entire belief space. This new method, YKA, is competitive with MBP, solving all the problems in the evaluation. MBP is, however, considerably faster in a number of problem instances.

Bryce et al. present POND (Partially-Observable Non-Deterministic planner) (Bryce et al., 2004), a planner with the main advantage of using a single planning graph to plan in belief space. LUG labels the propositions/actions with formulas that describe the initial worlds in which they can be applied (Hoffmann & Brafman, 2005); it also guides the search through providing a heuristic for a progression (in the contingent setting) planner. LUG condenses the information in the states comprising a belief state in a single graph representing their optimistic projection and is used to estimate the number of actions required to reach each belief state. Its generation ends when the goal (belief state) can be achieved by the literals appearing in one of its levels, provided these literals have labels that indicate that the goal is supported in all possible worlds. The fact that the goal must be supported in all possible worlds is similar to the max heuristic of GPT; the one used in POND, however, is inadmissible. The actual search in the state space is performed with a top down AO* (Nilsson, 1980) (Bonet & Geffner, 2000) method, in which the nodes are belief states and the hyper-edges are actions.

(Bryce et al., 2006) compare heuristics based on various graph data structures. They conclude that heuristics that do not use graphs, or use a single one, have limited ability to help planners scale, while those that use multiple graphs scale better, but are computationally costly. Multi graph heuristics are deemed better than single graph, with LUG providing the best results in terms of scalability. The evaluation examined the performance of POND against MBP, GPT, SGP and YKA, with the heuristics used in POND being more effective and informative in comparison to cardinality heuristics and the max heuristic in GPT. The evaluation also indicates that POND is up to par with BDD-based planners (MBP and YKA) and Graphplan-based (Blum & Furst, 1997) ones (SGP).

PC-SHOP (Bouguerra & Karlsson, 2005) is a method based on the classical Hierarchical Network Planner (HTN) (Ghallab et al., 2004) planner SHOP (Simple Hierarchical Ordered Planner) (Nau et al., 1999). SHOP was extended with the ability to handle partial observability and actions with probabilistic effects, with uncertainty in the problems modeled using explicit enumeration of belief states. Moreover, since the method is based on SHOP, it makes use of domain-knowledge to solve the problem at hand and being a Depth-First Search (DFS) (Cormen et al., 2001) algorithm it makes use of an iterative deepening strategy to prune deep recursive calls of methods. The generated solutions have a tree-structure and no merging of their branches is attempted.

ND-SHOP2 (Kuter & Nau, 2004) is another non-deterministic method based on SHOP, particularly on its successor, SHOP2 (Nau et al., 2001) (Nau et al., 2003). ND-SHOP extends SHOP2 to address fully observable domains with non-deterministic actions and multiple initial states. SHOP2, in contrast to SHOP that requires the subtasks of each method to be totally ordered, is a well-known sound and complete HTN planner that permits the subtasks of each method to be partially ordered, thus allowing interleaving subtasks of different tasks. It is a domain-independent planner which plans for tasks in the same order in which they will be executed. This element makes it possible for the planning system to be aware of the current state of the world at each step and in that way it gains significant reasoning power in regard to its precondition-evaluation mechanism and reducing the complexity of the planning process by avoiding some task-interaction issues. As in SHOP2, the search in ND-SHOP2 is based on a forward-chaining HTN-planning algorithm that exploits domain-specific search-control heuristics to be efficient. In case these techniques allow for a significant pruning of the search space, ND-SHOP2 outperforms MBP. In the general case, however, since the planner does not use symbolic representations of belief states, it scales up worse. For this reason, YoYO (Kuter et al., 2005) combines the HTN-based search of ND-SHOP2 with BDD-based symbolic model checking. In that way, it does not enumerate belief states explicitly as in PC-SHOP or ND-SHOP2, outperforming both methods in all cases, while also solving larger problems.

(Alford et al., 2014) presented a method that improves over the NDP algorithm, called NDP2, which mainly concerns the behavior of the algorithm in relation to unsolvable states. NDP2 uses the all-outcomes determinization to produce multiple weak plans from the current state to the goal and combines them in an effort to build a strong cyclic solution. In order to deal with unsolvable states, NDP2 modifies the generated classical planning problem by rendering some of the actions inapplicable at the first step of any solution for the problem so as to find acyclic plans that avoid visiting known unsolvable states. In this way, instead of increasing the domain description by an exponential amount as in NDP, the process only incurs a quadratic increase per constrained action.

NDP2 was compared experimentally to MBP in six domains, half of which contain unsolvable states. These experiments, which used FF as the classical planner for NDP2, do not indicate that NDP clearly outperforms MBP; three factors determine which algorithm is better suited for each problem. The amount of non-determinism in the domain; whether or not it contains unsolvable states; and which method could better abstract the non-determinism in it, i.e., MBP through the use of BDDs or

NDP through its abstraction mechanisms. In most cases, the abstraction mechanisms of NDP are significantly less efficient than the use of BDDs and as such MBP manages to plan for whole sets of states at once while NDP2 plans for each one separately. On the other hand, the use of an external classical planner allows NDP2 to make use of efficient heuristics and visit fewer states than MBP.

Gamer (Kissmann & Edelkamp, 2008) (Kissmann & Edelkamp, 2009), the winner of the Fully Observable Non-Deterministic (FOND) track at the International Planning Competition of 2008[6], presents a novel method to solve non-deterministic problems, by translating the original problem into a two-player turn-taking game. The translation, which is linear in the parameterized domain, formulates the problem as one in which one player represents the desired moves dictated by the planner and the other player represents the non-deterministic effects in the environment, by compiling each non-deterministic action into two. The formalism used is a PDDL-like language that was previously used to parse and instantiate general playing games; this allows the use of static analysis tools for deterministic planning and a concise description of the domain. The planner solves the problems optimally and produces strong or strong cyclic plans, generating a state-action table in the form of a BDD.

As mentioned in Section 2.3, another method that attempts to improve on NDP is FIP; FIP solves the non-deterministic problem by iteratively expanding a graph consisting of the reachable states until a path to the goal exists for every (non-goal) leaf state (Fu et al., 2011). That is, a weak plan is generated for an – arbitrarily selected – non-goal leaf state and an external classical planner, in this case FF, is used to find a plan. If the classical planner used is sound and complete, then FIP is also sound and complete. Furthermore, FIP was extended, first, by keeping track of the search results of each iteration, allowing it to avoid exploring the same (solved) states more than once and, second, by the addition of an alternative goal heuristic. For each action with non-deterministic effects, the effect included in the current weak plan represents the intended one; the other one represents a failed effect. Then, instead of setting the goal for a weak plan to be the original goal, the heuristic searches for a weak plan that leads to the intended effect. If one is not found, search is restarted to find a weak plan for the original goal. FIP was evaluated against two planners capable of solving strong cyclic FOND problems, namely MBP and Gamer; moreover, FIP was evaluated with and without the two aforementioned extensions. Both versions of FIP, with and without the extensions, have the same problem coverage, outperforming MBP and Gamer significantly. The extensions are particularly beneficial to FIP, as the planner is considerably faster and produces plan sizes that are significantly smaller.

(Muise et al., 2012) present PRP, a planner that comprises both FOND and online probabilistic planning to solve non-deterministic planning problems. PRP first converts a standard non-deterministic PDDL domain to a non-deterministic SAS+ formalism (Backstrom & Nebel, 1995) and returns a strong cyclic plan, or the best policy, if the former does not exist. It uses the all-outcomes determinization to search for a weak plan each time a state that is not handled by the current policy is

[6] Available at http://ippc-2008.loria.fr/wiki/index.html. Accessed on February 15, 2015

encountered; only the relevant parts of the states are used during search, and the determinized problem is solved for various initial states. A weak plan is generated, and when a state that does not have a strong cyclic plan is encountered, replanning is employed. PRP was extended by using local planning, i.e., in case an unhandled state is encountered, PRP generates a local plan to get back to the intended state instead of replanning. Furthermore, it identifies states in which the policy essentially acts as a strong cyclic one, instead of exhaustively enumerating them. The first extension, however, is efficient only when the local plan is extremely short.

PRP was implemented by modifying the Fast Downward planner (Helmert, 2006); as an offline FOND planner, it significantly outperforms FIP, in relation to the output plan size and the planner's run time. When the planner behaves in this way, it is able to find a policy maximizing its probability of success. During this process, it manages to avoid potential dead-ends through simulation, in contrast to other methods that tackle the problem through replanning. This is the reason that PRP achieves the goal with up to several orders of magnitude fewer actions than FF-Replan, and, when the problems considered are not trivial, i.e., they are "probabilistically interesting" (Little & Thiebaux, 2007) with possible dead-ends, PRP also scales better and is significantly faster than FF-Hindsight+. Counter intuitively, although in domains with probabilistic action outcomes PRP simply ignores the probabilities attached to the actions and behaves as a FOND algorithm, it manages to solve most benchmark problems.

(Hoffmann & Brafman, 2005) present Contingent-FF, which treats contingent planning as search through an AND-OR tree (Luger & Stubblefield, 1993) in the space of belief states. The planner, Contingent-FF, represents belief states implicitly through the action-observation sequences that lead to them from the initial state. During search, the propositions that are known in each belief state are computed, with a proposition being considered known in a belief state if it holds in the intersection of the worlds in that belief state. This allows the planner to scale up better in larger problems and also does not require large amounts of memory (To et al., 2011). On the other hand, instead of explicitly representing belief states, the planner has to reason about the entire action sequence that lead to them, which, in essence, trades space for time.

The actual search is performed as a weighted AO* forward search, with OR nodes representing belief states and AND nodes representing actions. It is assumed that actions with unsatisfied preconditions do not cause the plan to fail, but simply do not produce any result. A relaxed problem is produced by ignoring the delete lists and the length of the conformant solution for it is used as the heuristic. The method was experimentally compared against POND and MBP, that is, with methods that tackle the way belief states are handled in different manners. The experiments show that the plans generated by Contingent-FF were, in most cases, similar in terms of quality to those generated by POND and better than those generated by MBP (Hoffmann & Brafman, 2005).

(Buffet & Aberdeen, 2006) present the Factored Policy-Gradient planner (FPG), a probabilistic temporal planner aimed to efficiently tackle large partially observable problems using Reinforcement Learning (Sutton & Barto, 1998) and a local optimization procedure, i.e., online gradient ascent, to

search for plans. Gradient ascent is used for direct policy search, by estimating the gradient of the long term value of the process. The gradients represent the output policy, which is then factored into simple approximate policies for starting each action. These policies map a partial observation to the actual probability of executing the action, representing the amount of usefulness of each one. FPG also aggregates similar states in an implicit manner, by having each policy contain critical observation and not entire states; moreover, Monte-Carlo-like algorithms are used to keep the memory consumption independent of the state space size.

FPG is able to optimize both the makespan and the probability of reaching the goal. On the other hand, the techniques used in the method, that is, local optimization and the simplified parameters, can lead to suboptimal policies. FPG can also be inefficient in large domains where the algorithm does not reach the goal in a short time after it starts, as well as have long learning times since it improves upon an initially random policy. For these reasons, (Buffet & Aberdeen, 2007) presented an improvement of FPG, FF+FPG, which guides its search of the state space by using FF's heuristic, specifically by having FF return the action it would execute in a given state and caching them for efficiency reasons. In that way, FF+FPG follows a policy based on FF while trying to learn its own, using importance sampling. In general, FF+FPG manages to learn better policies than FPG, solving problems that FPG could not solve.

Finally, (To et al., 2011) utilize an earlier idea from (To et al., 2009) so as to encode the belief space in a compact form of disjunctive formulae, called minimal DNF; this method, under certain restrictions, allows for the computation of belief states' successors in polynomial time. The transition function is modified so that it can also handle non-deterministic action effects and observation actions, making the method suitable for PPOS problems. The underlying algorithm used to generate solutions for these problems is an AND/OR forward search one, called PrAO (Pruning AND/OR search). PrAO is based on standard AND/OR graph search algorithms; it is extended so as to prune useless nodes and scale up more efficiently, and keeps track of potential solutions, allowing the remaining AND/OR search graph to comprise the solution tree for the problem, when the search terminates.

2.5 Translation Methods

This section presents a set of methods that are based on the translation of the original non-deterministic problem into another one; this can be another form of non-deterministic domain that can be tackled more easily, e.g., a FOND one, or a classical deterministic problem. These methods are similar in principle to the ones presented in Section 2.3; however, the main difference is that they utilize translations that are, in most cases, complete, so the resulting problems are equivalent to the original ones. Table 2.3 summarizes the methods presented in the following sub-sections and their characteristics.

Table 2.3. Synopsis of translation methods

Method	*Target Contingent Problem*	*Translation to*	*Complete*	Translation Complexity
CLG	Simple	FOND problem	•	Polynomial
CLG+	With dead-ends and $width \geq 1$	FOND problem	•[1]	-
K-Planner	$Width \geq 1$, with restrictions[2]	FOND problem	•[3]	Linear
PO-PRP	PPOS, with restrictions[4]	FOND problem	•	Quadratic
SDR	PPOS with deterministic actions	Classical problem	•[5]	-
MPSR	PPOS with deterministic actions & observations	Classical problem	•	Double exponential
HCP	PPOS with deterministic actions & observations	Classical problem	-	Linear
(Palacios et al., 2014)	PPOS with deterministic actions & observations	Classical problem	•	Polynomial

1 For suitable choices of tags and merges.

2 Problems with invariant non-unary clauses representing the initial situation's uncertainty and hidden fluents in the initial state that do not appear in the body of conditional effects

3 On the condition that the problem's state space is connected.

4 Problems whose initial state comprises only state invariants and in which uncertainty decreases monotonically.

5 If the underlying classical planner is sound and complete.

2.5.1 Translation to FOND Problems

(Albore et al., 2009) argue that a contingent problem P involving uncertainty only in the initial situation and comprising only deterministic actions, cannot be translated to an equivalent classical problem P', since the two problems have different solution forms. Contingent solutions have a tree form that maps the tree's internal nodes to actions, with a solution being a policy tree that solves P when the execution associated with its branches ends up in belief states that comprise the goal and are feasible. However, contingent problems can be translated to equivalent FOND problems $X(P)$ in state space, which, for strong solutions, have a similar solution form. Such a translation is presented in (Albore et al., 2009), in which the non-deterministic actions in $X(P)$ represent the sensing actions of the original problem. In that way the computations can be made in regard to states that are represented

by sets of literals, instead of beliefs states that are represented by sets of states, which is computationally expensive.

This translation can be sound and complete, as well as efficient. However, it can be polynomial in time only if the problem belongs to a specific category of contingent problems, i.e., has bounded contingent width. Problems that have a bounded contingent width of 1 are named simple, and it is argued that almost all of the existing benchmark planning problems are such. This class of problems is characterized by the fact that the uncertainty is related to the initial values of a set of multi-valued variables that are used in goals or action preconditions, but it is not present in the body of conditional effects (Bonet & Geffner, 2011). Moreover, in simple problems, the value of any proposition that appears within an effect condition is initially known and constraints on the value of initially unknown propositions are invariant (Maliah et al., 2014). These constraints are usually used to characterize the device or physical environment in which an agent is acting, and are clauses that hold in every world state (Helmert, 2009). An example of such a problem is the well-known Wumpus (Russell & Norvig, 2002), in which an agent can move in a grid orthogonally with the goal of reaching a destination containing a treasure. The agent has to avoid a monster named Wumpus that lies in an unknown location inside the grid and can do so by detecting the Wumpus if he is in an adjacent location from its smell. In this case, the locations that contain a Wumpus are the invariant hidden variables, as they are initially unknown, never change and do not affect other variables.

The original contingent problem can be translated to an equivalent FOND problem $X(P)$; however, even this problem cannot be solved trivially. The number of literals in the translation can be exponential in the number of literals in the original problem, possibly resulting in extremely large state space, thus prohibiting the scaling up of the method (To et al., 2011). (Albore et al., 2009) suggest that the problem can be solved efficiently though through the use of a classical planner on a relaxation of (P), namely $X^+(P)$, where all the deletes, preconditions and actions with non-deterministic effects of $X(P)$ are dropped. The solution of $X^+(P)$, which can be obtained in polynomial time, is then used as an estimate for the size of the plans of $X(P)$. As the authors note, however, this result does not hold for arbitrary non-deterministic problems, as such a translation can render the problem unsolvable if the non-deterministic actions are removed from it. A variant of $X^+(P)$ is used in (Albore et al., 2009), based on the relaxation of a modification of the original contingent problem. The resulting (classical) problem is called the heuristic model $H(P)$. A special case of the general translation scheme is presented, $X_i(P)$, which is sound, complete and polynomial in the fixed factor i, if the contingent width of the original problem P is less than i. A planner that makes use of such a translation scheme to tackle contingent problems is presented, namely the Closed-Loop Greedy (CLG) planner. CLG uses the $X_1(P)$ translation to efficiently keep track of beliefs, i.e., is sound, complete and polynomial only for simple contingent problems. $H(P)$ is used to select the next actions in a closed-loop fashion. The solutions are generated by starting from the initial state s being equal to $X_1(P)$, generating an action sequence to be applied in s that results in s' through the use of a modified version of FF and continuing in this fashion until s' is a goal state.

CLG can be used in an online or offline fashion; the result of its online use is a single successful execution, while in offline mode it generates full contingent plans. CLG was empirically evaluated against Contingent-FF and POND version 2.2. In online mode it can solve larger problems for which it cannot generate a full contingent plan; in offline mode it outperformed both planners, which ran out of memory or time for the benchmark problems. Moreover, a comparison between CLG, Contingent-FF, and POND 2.2 exists in (To et al., 2011) where the aforementioned planners were evaluated against DNFct in two problem sets. Contingent-FF was executed with and without helpful actions and POND with AO* as the search algorithm. In the first set DNFct was the best planner overall, with the quality of its plans being comparable to the ones outputted by other planners, while also being able to scale up better on most domains. CLG was the second best planner, whereas the other planners demonstrated poor performance. In the second set of problems, the results were similar, with DNFct having again the best performance in all but one domain, with CLG being second best. Despite the success of CLG, the planner does not perform well in domains with dead-ends; in contingent planning, a belief state in which a dead-end state is reachable is itself a dead-end, as the planner cannot reach the goal with certainty from it. That is, in domains with partial information, dead-end belief states do not only arise from dead-end states (Albore & Geffner, 2009). Moreover, problems in which the initial belief state is a dead-end have no solution. In case all the possible belief states at a point are dead-ends, then a policy that tries to maximize the coverage of the solution is not appropriate. As such, CLG does not guarantee that it will work in such domains.

In (Albore & Geffner, 2009) CLG is extended so as to work in contingent problems without solutions. To do so, three modifications are made to CLG. First, assumptions are incorporated into the planning process. Policies are generated from a relaxation of the original problem, in which assumptions are added to the current belief state. These assumptions are represented by new actions that are related to both $X(P)$ and $H(P)$ and can be confirmed or refuted based on the gathered observations. Secondly, costs are assigned to the actions that are related to these assumptions, so as to allow a preference for solutions that do not include them. Finally, the classical planner used is a modified version of FF, named $FF(h_a^s)$, in relation to its definition and computation of relaxed plans and the use of the set-additive heuristic h_a^s (Keyder & Geffner, 2008). The resulting planner, CLG+, cannot only solve simple contingent problems, but also problems with higher contingent width, in contrast to CLG and Contingent-FF and doing so in a more efficient manner than POND. Moreover, CLG+ also scales up better than CLG in offline mode.

A recent planner that extends CLG is K-Planner (Bonet & Geffner, 2011), which introduces a more compact – linear instead of quadratic - translation scheme. Similarly to CLG+, the method is not restricted to simple contingent problems. K-Planner requires that the value of the non-unary clauses representing the uncertainty about the initial situation be invariant throughout the execution of the plan and that the fluents in the initial state that are hidden do not appear in the body of conditional effects, which are assumed to be deterministic. An example of these restrictions in the Wumpus domain is a variant of it in which the wumpus remains in the same position throughout the planner's execution.

The presented translation is sound and complete on the basis of the aforementioned assumptions, as well as on the assumption that the state space of the problem is connected, that is, if for every pair of states that are reachable from the initial state one can be reached from the other. Moreover, the solutions can be produced using a classical planner on a modified version of the translation scheme.

Sensing actions in K-Planner are translated into multiple deterministic actions that provide the knowledge of different values of the sensed variable (Brafman & Shani, 2011), with the planner being allowed to choose the value it will sense; with these assumptions, the method falls in the category of planning under optimism. However, this may be problematic in cases where backing up is not possible, or when multiple sensing actions were chosen to achieve effects that cannot be realized by any initial state. The latter, though, does not affect the soundness of the method, as being an online one, the plan is always executed until the first sensing action and then the current belief state is updated. K-planner was compared against other approaches, e.g., CLG in (Shani et al., 2013). However, as (Brafman & Shani, 2011) note, K-Planner can only be applied in a specific class of contingent problems; this allows it to maintain belief states more easily and generate more efficient translation patterns, thus providing it with a clear advantage against other more general planners.

A method that combines the completeness of CLG for simple, i.e., of width 1, contingent problems with the linear belief tracking translation of K-Planner is presented in (Bonet & Geffner, 2014). The presented planner, LW1 (Linear translations for Width-1 problems) is an on-line partially observable planner that generates a classical plan from the heuristic model $H(P)$ and executes it until the first sensing action. If the sensing action's true value is the same as expected, then the execution proceeds; otherwise, a new classical plan is computed from the current state. Width-1 completeness is achieved through explicitly defining originally implicit conditional effects, instead of using tagged literals as in CLG. This results in the method utilizing two linear translations; the first, for belief tracking, the second for action selection using classical planners. LW1, using FF as the underlying classical planner, is compared experimentally against K-Planner and the Heuristic Contingent Planner (HCP) (Shani et al., 2013); LW1 is the fastest planner in the comparison in most domains, and generally produces shorter execution plans; finally, it also scales up better than both planners, with HCP being the worst of the three in terms of scalability.

PRP, presented in Section 2.4, was subsequently extended to tackle PPOS domains (Muise et al., 2014); in specific, PO-PRP, considers domains with incomplete information regarding the initial state and all the available actions are treated as deterministic, with the exception of the sensing ones. Similar to CLG+, PO-PRP assumes that the initial state comprises a set of state invariants constraints. A further assumption is that the uncertainty in the domain cannot increase; once a previously unknown property becomes known it cannot become unknown again. The compilation presented in (Bonet & Geffner, 2011) was subsequently used in (Muise et al., 2014) to convert the PPOS problem at hand to a FOND one and then use PRP to compute strong cyclic plans that are subsequently converted into DAGs. PO-PRP was experimentally evaluated in comparison to CLG, which it outperformed in relation to both the size of the generated solutions and the required time to compute them.

2.5.2 Translation to Classical Problems

In (Shani & Brafman, 2011) (Brafman & Shani, 2012) the authors tackle probabilistic domains with partial observability, sensing and deterministic actions by using a similar translation method. In short, the online planner based on this method, SDR (Sample, Determinize, Resample), generates a solution for a determinized version of the original domain based on the T0 translation (Palacios et al., 2009), which allows for the incorporation of the current knowledge of the agent in each state. Then, SDR follows the plan as long as the preconditions of the next action hold in all possible states, until the goal is met. If at any step the preconditions of the next action do not hold in all possible states, a new plan is generated. The method does not try to output a complete plan, but only returns the next action to be executed; in contrast to this, when replanning is employed, the result maybe a sequence of actions.

As in (Albore et al., 2009) sensing actions are translated to non-deterministic actions; furthermore, the translation may lead to domains with an exponential (in the input size) number of propositions. (Shani & Brafman, 2011) propose that both of these problems should be tackled through state sampling, by choosing a subset of the original initial states to reduce the size of the domain and then only one of those as the true initial state, so as to remove non-determinism. An arbitrary possible initial state is chosen and the planner assumes that the observations will be sensed as if this is the actual initial state, ignoring all other execution paths that contain contradicting observations. If, while the plan is being executed and the belief states are updated according to the new observations, the chosen initial state is not consistent with the world, replanning is employed.

In regard to the representation of the belief states, SDR adopts a method similar to the one in Contingent-FF, i.e., the implicit representation of the states through propositional formulas that correspond to the history of action execution. However, SDR further simplifies the method, by only maintaining the actions and the observations made, along with the chosen initial state. In this way, only the history of execution is required to determine whether a condition holds, by regressing through the executed plan. This information is cached, by constructing a partially specified belief state that only contains a list of literals that are known to hold in this state, for each step of the execution. In relation to Contingent-FF, this is easier to compute and produces smaller formulas, despite being prone to reconstructing a formula – or parts of it – multiple times.

SDR was evaluated experimentally against the online version of CLG, with SDR using FF to solve the determinized problems. SDR is about one order of magnitude faster than CLG and also scales up better. In domains that both planners solve, they generate plans of similar size, with one outperforming the other depending on the particular domain. Moreover, the addition of a bias for observation to the planner that forces it to sense the value of all unknown propositions at each step, if that is possible and does not affect the current state, resulted in faster executions in most domains. Finally, SDR typically does not require more than a sampled initial belief state of size 2 to perform efficiently, as the plans' lengths do not change significantly with the number of states.

As aforementioned, CLG and K-Planner fall into the category of planning under optimism by allowing the planner to select the values that it senses. SDR, on the other hand, samples a few initial states and chooses an arbitrary one from them, assuming that the observations will be sensed as if it is the true one and disregarding all other different execution paths. These methods are not particularly realistic and can also be ineffective, as until replanning occurs to encompass the new information, a few actions may have already been executed based on the old assumptions of the world, possibly leading to dead-ends.

(Brafman & Shani, 2012) presented a method designed to tackle such issues called MPSR (Multi-Path Sampling Replanner); MPSR is a sound and complete compilation method that translates the original problem to a determinized classical problem, the solutions of which comprise a contingent plan that takes into account all the possible execution paths. MPSR is targeted towards partially observable contingent planning problems that are characterized by uncertainty regarding the initial state and comprise sensing actions, assuming only deterministic actions and observations. The two main ideas behind it are the issues of distinguishability between states and the enhancement of the original set of actions. MPSR stores the information the agent has at runtime in relation to its initial state and keeps track of which states are distinguishable at each time, in essence capturing the belief states. To achieve this, propositions are added to the domain that define whether two states are distinguishable, i.e., whether a proposition exists having a value that is different in the two states and the agent has observed it. The latter idea is used so that a contingent plan can be generated within a classical - linear - one. For this reason, the original set of actions is modified with new actions, each of which has the following behavior: An action a_b is generated for an original action a that produces the same effects as a in states that are in belief state b, but does not produce any effect for any state that is not in b. As such, for different belief states, different versions of the original action are generated; this, however, leads to a significant increase of the number of actions and the size of the generated problem.

Two different versions of MPSR are presented in (Brafman & Shani, 2012); one used offline, generating a complete contingent plan when such a plan exists. This version of MPSR achieves the goal despite the problem's non-determinism but, as aforementioned, produces large solutions, since the classical problem that is generated is linear in the number of initial states and exponential in the number of propositions. For this reason, a modified version of MPSR is presented, which uses an incomplete translation process. However, this version relies on replanning, similarly to CLG and K-Planner, and also makes use of sampling of a subset of the initial states, ignoring all the rest, similarly to SDR. Moreover, the belief maintenance and update is also similar to SDR and Contingent-FF, by only maintaining the actions and the observations made, along with the chosen initial state. At each iteration of MPSR, a multi path translation generates a plan based on a subset of the initial states that is not a valid solution plan; however, it is more informed than the plans generated by SDR, as it takes into consideration more than a single initial state. This plan is executed until either a new observation

is made, altering the belief state and requiring the modification of the agent's knowledge, or until the next action cannot be executed due to its preconditions not being met.

MPSR was evaluated against CLG and SDR (with all planners using FF as the underlying classical planner) and MPSR proved faster than both planners; in domains without dead-ends, however, while MPSR is faster, it also generates longer plans. The use of resampling, on the other hand, allows for the generation of smaller plans at the expense of increasing the planning time. If the evaluated domain contains dead-ends, MPSR is faster, generates smaller plans and scales better than CLG, with SDR not being able to solve such problem instances.

HCP is an online method for contingent planning that utilizes landmark-based heuristics (Hoffmann et al., 2004) to select the next reachable sensing action. Landmarks are essentially used to identify the appropriate sensing actions, which are viewed as sub-goals of the problem; in this way, the search space can be considerably reduced. As in CLG, K-Planner and SDR, the planner tackles partially observable problems with uncertainty about the initial state of the world, containing sensing actions. All actions are deterministic, as well as the observations that result from the sensing ones. HCP is greedy, focusing only on selecting a single sensing action; after the selection, classical planning is used on a modified version of the original planning problem and the plan is then executed. This process is repeated until the goal can be reached without requiring any more information. Thus, HCP is also approximate, as it does not plan directly over the original conformant problem but using its projection. Finally, in contrast to other planners, e.g. CLG, it does not require explicit modeling of belief states or translation techniques that may require exponential space (Maliah et al., 2014). HCP modifies the method in K-Planner, as it extends the translation scheme presented in (Bonet & Geffner, 2011) in order to handle non-simple contingent domains. The modified translation is sound but not complete and, in contrast to K-Planner that plans directly over it, HCP utilizes it in the following way: First to obtain landmarks for the original problem so as to heuristically select the next sensing action and, second, as a projection to transform contingent/conformant problems into classical ones.

CLG and SDR provide sound and complete approximations for contingent planning domains, at the cost of generating larger classical projections. HCP, on the other hand, generates a simpler projection to capture all the possible paths to the goal, to allow for the heuristic to take into account all the possible outcomes of a sensing action so as to measure its usefulness. In the case that the contingent domain is simple, this projection is identical to the one used by K-Planner. In that sense, in simple problems, HCP behaves exactly as K-Planner with the additional incorporation of landmark heuristics. In the general case, however, the projection used by HCP is incomplete, that is, valid branches of contingent plans can exist that do not correspond to a valid plan of the projected problem, but can be used in any contingent planning domain, regardless of its contingent width. For this reason, in the case that the contingent width of the problem is equal to 1, the generated landmarks are the true landmarks of the problem; in any other case, the landmarks can be viewed as an approximate set for the original problem and can only be used as a heuristic (Maliah et al., 2014).

HCP was experimentally compared to CLG, MPSR and K-Planner, using FF as the underlying classical planner. Based on the evaluation, HCP is significantly faster than all other planners in most of the problems; it also solved more instances than the rest of the planners and, in general, generated shorter solutions. It is important that even in domains that do not involve useful landmarks, HCP worked well due to the rest of its heuristic elements; in total, HCP incorporates the minimal cost, number of literals heuristic, number of sensing actions and number of landmarks heuristics. The one that – on each own – contributes the most to the faster generation of plans is the number of landmarks heuristic. However, it is evident from the evaluation that the combination of the various heuristics is the most beneficial to the overall runtime.

Finally (Palacios et al., 2014) present an offline contingent planning method based on (Brafman & Shani, 2012); specifically, two polynomial – in the number of possible initial states – translations of contingent problems into classical ones are presented. One translation converts the sequence of actions in any topological traversal of a policy tree into a classical plan, while the other takes into account only the actions in a single traversal of the policy tree. The latter utilizes a stack element of size k, in which the states that predict an atom p to be false are pushed so as to be dealt with at a later time. The translation is polynomial in k and a stack of size k can be used to generate contingent plans with branches that comprise up to k observations. However, the resulting classical plans may be exponential in k. The method is evaluated on its own using various classical planners, i.e., Fast Downward, LAMA 2011 (Richter & Westphal, 2010), FF and the SIW planner (Lipovetzky & Geffner, 2012) on both translations. FF performs the worst in general, as it frequently runs out of memory in the evaluation benchmarks. Neither SIW nor LAMA dominate the other, as the former generates nodes slower than the latter, but requires fewer nodes expansion to output plans of higher quality. However, LAMA is the classical planner of choice in the evaluation of this method against Contingent-FF, MBP, POND, CLG and DNFct. CLG and DNFct clearly outperform the method in (Palacios et al., 2014), which performs similarly only in relation to earlier contingent planners, such as Contingent-FF, POND and MBP as to the coverage of problems and the generated solutions.

2.6 Discussion

The review of the recent literature on non-deterministic planning methods denotes that there is a plethora of different approaches to tackle the problems of non-determinism, considering various degrees of observability. Methods such as FF-Replan, that are fast, online and make use of – easy to compute – determinizations have proven to be successful, especially in solving planning competitions' problem sets. This type of methods, though, tends to disregard, or not take fully into account, information inherent to the problem, e.g., the probabilities attached to the outcomes of actions. Several methods improved upon the methods of FF-Replan, such as FF-Hindsight that demonstrated significantly better performance, or RFF that was able to generate offline policies as well as provide guarantees against failing.

A somewhat similar line of research, however, appears to be more promising; methods that utilize translations and result into problems that are equivalent to the original one, can tackle problems that are more realistic in their consideration of uncertainty, while also being able to scale up efficiently. Regarding earlier non-deterministic methods, POND is up to par with Contingent–FF as well as, if not better than, BDD-based planners, e.g., MBP and YKA, with MBP being less competitive than FIP and PRP. Contingent-FF, POND, as well as CLG showed worse behavior than DNFct, with the former two methods significantly lagging behind the newer methods. Translation methods in general, however, are able to outperform such approaches. Even CLG, which performs worse than state-of-the-art contingent planners, appears to be significantly more efficient than methods such as POND and MBP. Specifically, the current literature and evaluation results suggest that LW1 should be considered the state-of-the-art contingent planning method. It is both faster than all state-of-the-art planners, produces shorter execution plans and is more scalable than all current approaches. HCP appears to be the second best contingent planer overall, with K-Planner being faster, but less general.

Chapter 3.

Web Services

The World Wide Web Consortium (W3C) defines a Web Service (WS) as "*a software application identified by a Uniform Resource Identifiers (URI)* (URI Planning Interest Group, W3C/IETF, 2001)*, whose interfaces and bindings are capable of being defined, described, and discovered as XML (Bray et al., 2008) artifacts. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols".* (Schlimmer, 2002). (Papazoglou, 2008) provides a similar definition, defining WSs as "*a self-describing, self-contained software module available via a network, such as the Internet, which completes tasks, solves problems, or conducts transactions on behalf of a user or application. WSs constitute a distributed computer infrastructure made up of many different interacting application modules trying to communicate over private or public networks (including the Internet and Web) to virtually form a single logical system*". A WS can take various forms, such as a self-contained business task, e.g., providing the current temperature at a given city; a business process, e.g., automating the purchase of groceries whenever their stock drops behind a certain level; a complex application, such as a trip planner; or a service-enabled resource, such as providing access to a database containing the previous numbers that won a lottery.

In order for WSs to interact with each other so that different systems can cooperate dynamically, there has to a coupling, that is, a certain degree of dependency between them. This coupling should be loose, so that the communicating systems can cooperate with fewer restrictions, without having the same kind of underlying implementation, or even having to know how their partners are in fact implemented. Furthermore, having characteristics such as an asynchronous interaction scheme, or dynamic binding protocols, provides the opportunity to build more freely structured systems, able to sustain various changes in their structure or external events.

3.1 Standard Web Services

In order for WSs to interact with each other so that different systems can cooperate dynamically, there has to a coupling, that is, a certain degree of dependency between them. This coupling should be loose, so that the communicating systems can cooperate with fewer restrictions, without having the same kind of underlying implementation, or even having to know how their partners are in fact implemented. Furthermore, having characteristics such as an asynchronous interaction scheme, or dynamic binding protocols, provides the opportunity to build more freely structured systems, able to sustain various changes in their structure or external events.

Service description is necessary so as to reduce the amount of understanding, custom programming and integration between the service provider and its clients. It is also a requirement for the use of automated composition of WSs, as it provides a machine-understandable specification describing the structure, operational characteristics and non-functional properties of a WS, as well as the transport protocol that the WSs use. However, it is crucial that the description of WSs is done in a consistent manner, so that they can be published in directories where they can be discovered by clients and developers and composed into new ones in an efficient manner. A mandatory step of this process is to be aware of the precise interface of a WS along with the ways it can be accessed through messages.

An example of such a format is the Web Services Description Language (WSDL) (Christensen et al., 2001; Chinnici et al., 2007).

WSDL is an XML format for describing WSs as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. It is used to describe the details of the public interface exposed by a WS, which includes the abstract descriptions of the operations and messages it uses and their bindings to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL allows for the description of the functional characteristics of WSs so that they can be used by others through a network. Essentially, it is used to describe precisely what a service does, i.e., the operations the service provides; where it resides, i.e., details of the protocol- specific address, e.g., a URL; and how to invoke it, i.e., details of the data formats and protocols necessary to access the service's operations. This implies that WSDL represents a "contract" between the service requester and the service provider, in much the same way that an interface in an object-oriented programming language represents a contract between client code and the actual object itself. The basic difference is that WSDL is platform and language independent. The WS description is therefore concerned only with information that both parties must agree upon - not information that is relevant only to one party or the other, such as internal implementation details. It is important to note that WSDL intentionally only describes, in syntactic terms, the available operations, the relevant information regarding the messages that are used by a WS (their data types and protocols supported) and where a WS is located, without delving into the semantics of this information. This part is left to semantic description languages, described in detail in Section 3.2.2. WSDL is designed to be extensible and adaptable so that various message formats or network protocols can be used to interact with WSs.

The most widely used combination of WSDL with message and transfer protocols is in relation to SOAP messages (Gudgin et al., 2007) transmitted over HTTP. SOAP is an XML-based protocol describing the exchange of messages between nodes, without specifying any details about their operating systems, programming environment, or object model framework. It is basically a set of conventions that specify the format of simple request and response messages written in XML along with a set of rules describing how messages are processed as they move along a path. In essence, a SOAP method is simply an HTTP request and response that complies with its encoding rules. It is a common standard messaging protocol used by WSs as it provides a simple, scalable, flexible and lightweight way of exchanging structured and typed information between services, aimed to reduce the cost and complexity of the interoperation of heterogeneous platforms. SOAP defines a simple and extensible XML messaging framework for decentralized, distributed environments that can be used over multiple protocols with a variety of different programming models. In its basis, SOAP messages are requests transferred to an endpoint, running on any platform and implemented in any number of ways (such as Java (Gosling et al., 1996) servlets), on a different part of a network. As such, it specifies a wire protocol, that is, its functions can only send and receive several kinds of transport protocol packets and process XML. Its role should not be confused with a transport protocol, as it only specifies the

form of the data to be exchanged and it is not the actual the method by which that data is transferred from system to system.

Besides the description of WSs and the communication between them, WS have to be discoverable and searchable; furthermore, a repository should be used in relation to this fact and domain-specific taxonomies in relation to them should also be created. WS discovery consists of finding a WS provider along with its services and their definitions and in that process discover what capabilities they offer, how to interact with them and, most importantly, their exact location, that is, a URI at which they can be found. There are basically two types of service discovery:

- **Static discovery**: Service implementation details are bound at design time, service retrieval is performed on a service registry and a human designer is typically needed to incorporate the results, i.e., the WSs descriptions, into the application logic.
- **Dynamic discovery:** Service implementations are not bound so that they can be determined at run-time. The application issues a retrieval operation at run-time against the service registry to find service implementation definitions that match the service interface definition used by the application and the WS requester's preferences. The application chooses the most appropriate service, binds to it and invokes it.

There are two operations that have to be implemented in order to publish a WS in a registry: First a WS has to be described in relation to its business, service and technical information and, second, it has to be registered, that is, have its description stored in the registry. According to (Papazoglou 2008), a registry should have the following characteristics:

- Maximize WSs reuse.
- Create a management structure capable of sustaining a SOA implementation.
- Contain all the metadata regarding WSs and their associated objects.
- Contain information about service providers, requesters and their relationships.
- Provide general- and special-purpose interfaces that address the needs of providers, consumers, administrators and operators.
- Ensure that if the system architecture evolves in a growing fashion, the registry can handle the increasing number of services and service consumers.

A framework aiming to aid the implementation of a WS registry is the Universal Description and Discovery Interface (UDDI) (Clement et al., 2004); its motivation is to provide both the specifications and a set of actual implementations of a WS registry, automating the procedure of registering and querying to locate an appropriate WS. It also enables service requesters to discover information about organizations/ WSs providers, the WSs they offer and technical description of their interfaces. UDDI

is not bound to any particular protocol; while it uses XML to represent the data it stores, its entries can contain any type of resource, XML based or not. It can be characterized as a metadata aggregation service, defining protocols for querying and updating a repository comprising WSs information. Its most common use can be briefly described as such: A developer needs to locate a service providing a specific function; to find it he has to query a registry containing metadata regarding the WSs stored in it to determine which service providers are suited to his preferences and requirements. Having the results, that is, a list of the compatible WS providers, the developer can then send more queries, to get further information about the actual WSs that the providers offer. After a provider and his specific WSs have been chosen, the registry is able to provide bindings/ links to the technical description of the services (WSDL files). However, the use of UDDI has steadily declined through the years and it is now mainly used within the restrictions of a single company, having even been described as "dead" (Tilkov, 2010).

An example of how WSs can be described, communicate with each other, discovered and registered is shown in Figure 3.1, using the aforementioned standards as the means of implementation; a WS provider can create a technical description of a WS in a WSDL document and register it to a UDDI repository. Then, a WS requester can search the registry for a suitable service through queries for compatible WSs providers and their specific services. If a suitable WS is found, a requester can then bind to it. A SOAP request will be created according to the WSDL document that will be sent to the service provider to process it.

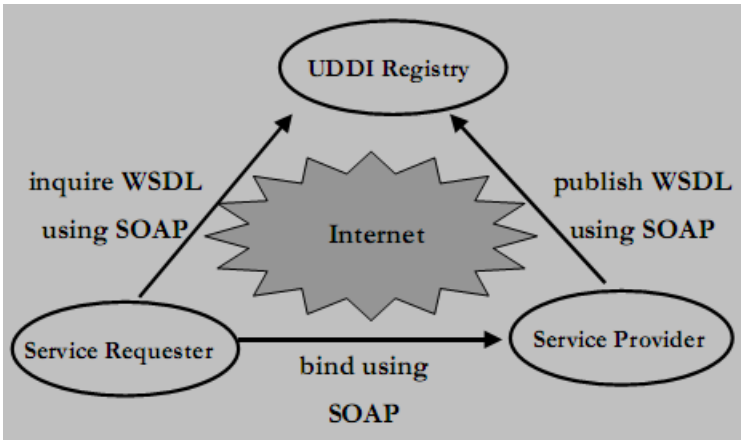

Figure 3.1: Relationship of the functions of UDDI, WSDL and SOAP (Gunzer, 2002)

This section described WSs under the assumption of use of a combination of web protocols and standards, such as WSDL and SOAP; this view was adopted by the (vast) majority of the academic research on automated Web Service Composition (WSC) which has been focused on the automation of WSDL/SOAP WSs (Zhao & Doshi, 2009). An alternative, altogether significantly different view of WSs is that of RESTful WSs (Richardson & Ruby, 2007). While SOAP is a protocol, REST (REST (REpresentational State Transfer) is an architectural style and a design for network-based software applications, e.g., WSs. REST was first described in (Fielding, 2000), with it motivation stemming from

the desire to simplify the process of developing WSs. Such a goal is attempted to be achieved by using HTTP for transferring data, instead of a standardized protocol such as SOAP that works on top of the HTTP layer to provide message transferring capabilities. Moreover, in contrast to the operation-centric view of the combination of SOAP and WSDL, RESTful WSs adopt a resource-centric perspective. In REST, every resource is identified by a unique URI that can be operated on by a subset of the core set of HTTP commands; that is, HTTP is used to retrieve the representations of these resources in varying states (Mulligan & Gracanin, 2009).

SOAP has the advantages of being language, platform and transport independent and being a standardized protocol, in contrast to REST that is based on HTTP and constitutes an architectural style without being specifically defined. Moreover, SOAP is well suited for distributed environments, whereas REST assumes a point-to-point communication and also excels in terms of security and reliability, as it has built-in error handling. However, as (Potti, 2011) notes, REST appears to be more scalable, fast and interoperable. It has also been extensively used in the industry, with Google Maps[7] and Google search[8] being built on the REST principle and no longer supporting SOAP since 2009, and Yahoo, EBay and Amazon also utilizing RESTful WSs (Amazon also offers WSDL/SOAP WSs, as well). However, due to the characteristics of REST and its differences to WSDL/SOAP WSs, the problem of RESTful WSC is considerably different than the one will present in the following sections; as such, we will not discuss RESTful WSC further.

3.2 Semantic Web Services

The previous sub-section described the basis of "traditional Web Services", serving the purpose of separating interface, implementation and deployment. However, this initial basis does not efficiently deal with other important, functional and non-functional, characteristics, such as security or reliability of the WSs framework, or aid the automation of the composition of WSs. Despite XML being flexible, extensible and widely used, its use hinders the framework due to the fact that it does not carry any information relating to the meaning, or semantics, that is associated with the data transferred. It is simply used to define the structure and syntax of data and, thus, a human is usually needed to interpret both the WSDL documents and the XML descriptions of the exchanged data as their meaning can be ambiguous. An analogous problem arises with the UDDI registries, as the descriptions contained there are not formalized and are mostly interpreted by a human reader. As a basic motive for using WSs is to achieve their automated discovery, composition and invocation, a need for semantic annotations arises.

This need is met by the enhancement of the traditional WSs with formal semantic descriptions, expressed with ontologies and description logics. An ontology is a conceptualization of a domain used to formally describe this WSs. The concepts contained in an ontology, along with their properties and

[7] https://developers.google.com/maps/documentation/webservices/
[8] https://developers.google.com/web-search/docs/?hl=en

inter-relationships, are used to create WS descriptions. The semantic metadata attached to Semantic Web Services (SWSs) are used to achieve the automation of WS related tasks. Among the benefits that are gained through the use of SWSs are (Akkiraju, 2007):

- **Reduced cost and time:** While the usage of WSs is growing, so does the time needed for to integrate and process automation projects and their relative cost. SWSs facilitate the automation of the integration of tasks, furthermore resulting in more flexible and adaptable solutions, which can lead to reduced development costs and less time spent on integration.
- **Facilitates the development of flexible and robust systems:** Since runtime environments in business contexts are dynamic, that is, services can become unavailable at any moment, or process changes may have to take place at runtime without disrupting the existing environment, SWS techniques can be used to dynamically discover appropriate replacement WSs or new ones meeting the specified criteria, leading to adaptable, robust and maintainable systems.
- **Formal Models Promote better software system Maintenance:** The use of SWSs dictates the use of ontologies and the general formal documentation of the concepts and the domain used in the implementation process, a fact aiding the future maintenance of the final implemented solutions.

We are mainly interested in WSC, the goal of which is to select and connect various WSs, provided by different partners, into a new one fulfilling a more complex goal. We will elaborate on WSC in Section 3.3. First, we will briefly present the existing Semantic Web standards and Semantic WSs approaches that have occurred in the Semantic Web area (Berners-Lee et al., 2001), a subset of which will be used in later sections.

3.2.1 Semantic Web Content Standards

3.2.1.1 eXtensible Markup Language (XML)

The Extensible Markup Language (XML) (Bray et al., 2008) is a general-purpose markup language, used to share structured data across different information systems. It allows users to define their own tags and it can be used either to encode documents or to serialize data. XML is a W3C recommendation, specifying both the lexical grammar and the requirements for its parsing.

3.2.1.2 Resource Description Framework (RDF)

The Resource Description Framework (RDF) (W3C RDF Working Group, 2004) is a language intended for representing metadata concerning web resources, such as the title or author of a web page. It can also be used to represent information about things that can be identified on the web through

using URIs even when they cannot be directly retrieved on the web, if the meaning of the concept of a "web resource" is generalized.

3.2.1.3 SPARQL

The SPARQL specification (Prud'hommeaux & Seaborne, 2008) defines the syntax and semantics of a query language for RDF and can be used to express queries across diverse data sources, regardless of whether the data is stored natively as RDF or viewed as RDF via middleware.

3.2.1.4 Web Ontology Language (OWL)

The Web Ontology Language (OWL) (McGuinness & van Harmelen, 2004) is used for defining and instantiating web ontologies, each one able to describe classes, along with their related properties and instances. OWL2 (W3C OWL Working Group, 2009) is a new version of the language, adding features to the older one, while remaining compatible with it. Its main use in the WSs framework is to facilitate greater machine interpretability of web content than that supported by non-semantic standards, such as XML and RDF, by providing additional vocabulary along with a formal semantics.

3.2.1.5 RuleML

The Rule Markup Language (RuleML) (RuleML Initiative, 2011) is a markup language developed to express both bottom-up and top-down rules in XML for deduction, rewriting and further inferential-transformational tasks. It can serve as a specification for immediate rule interchange.

3.2.1.6 Semantic Web Rules Language (SWRL)

The Semantic Web Rule Language (SWRL) (Horrocks et al., 2004) is an expressive language based on OWL, allowing the creation of rules that can be expressed in terms of OWL concepts to provide more powerful deductive reasoning capabilities than OWL alone. It combines aspects of sublanguages of OWL, namely OWL-DL and OWL Lite (McGuinness & van Harmelen, 2004), with the Unary/Binary Datalog sublanguages of Rule ML, while also including a high-level abstract syntax for Horn-like rules in both those sublanguages of OWL. A model-theoretic semantics is given to provide the formal meaning for OWL ontologies including rules written in this abstract syntax.

3.2.2 Semantic Web Services Approaches

In this sub-section, we provide a short description of some of the prevalent approaches for describing SWSs. Although they are all ontology-based and none is considered the de facto standard, we will focus on the two approaches that have a widespread use, that is, OWL-S and WSMO.

3.2.2.1 OWL-S

OWL-S (Martin et al., 2004) is an ontology-based approach aimed to provide a semantic description of WSs, which allows their automated discovery, execution and composition. It is the evolution of the DARPA agent markup language for services (DAML-S) (Martin, 2002), which in turn was based on DAML+OIL (van Harmelen et al., 2001). The axioms that are contained in the OWL-S ontology are stated in OWL and, as such, involve class descriptions, subclass hierarchies, and definitions of the kinds of relationships that may hold between different classes. This ontology is divided in three smaller ones/profiles, illustrated in Figure 3.2 and presented in detail below:

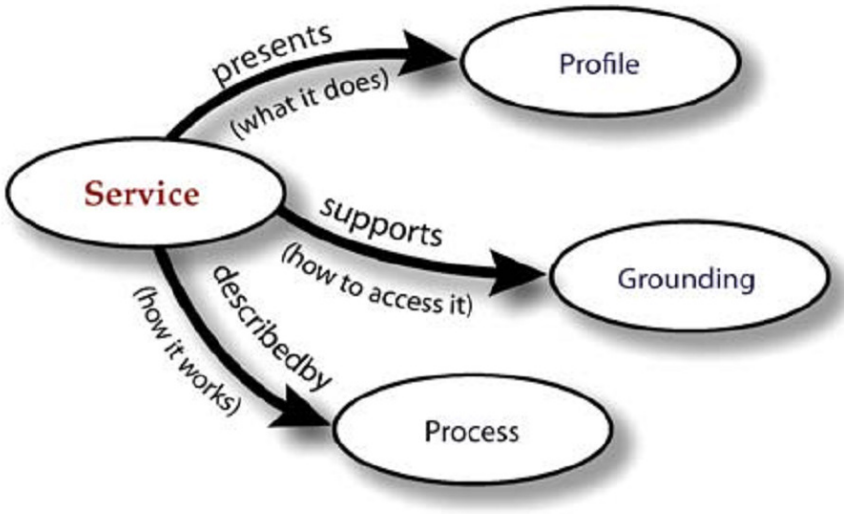

Figure 3.2: Top level of OWL-S Service ontology **(Martin et al., 2004)**

- **Service Profile:** It is used for advertising and discovering WSs, by describing their offered functionalities, along with any constraints and non-functional properties that affect them. The model of OWL-S does not make a distinction between advertising WSs by their providers, or seeking WSs by requesters, so the Service Profile element is used for both. The Service Profile defines the WSs' functionality by using state transitions, that is, specifying the necessary inputs/outputs that are expected to be used/generated, as well as the preconditions/postconditions that must hold before/after a service is executed. In this way, a service requester can determine whether a service's Profile matches the one in the set of candidates that he desires. Furthermore, more than one language can be used to accomplish the task of describing logical expressions for the preconditions and results of Service Profiles (or Service Models). These include SWRL, RDF and PDDL that can be used for treating expressions as string literals and bring together the Semantic Web and the AI research community.

- **Service Model:** It is used to describe the behavioral model of a WS, thus providing a detailed description of its operations to the requester, who can, in this way, understand how to interact with

the service in order to accomplish the desired purpose. Different types of services can be defined in this element, that is, atomic, composite and simple. A simple service is a service that can be implemented either as an atomic or as a composite, that is, it is an abstraction that is not actually invocable. An atomic service is one that corresponds to just one interaction with it. In contrast to that, a composite service comprises multiple steps – each one corresponding to another process, composite or non-composite – that are connected by control constructs and data flow. Each control construct is associated with a property called *components* that expresses the nested control constructs that comprise it and, in some cases, their ordering. A composite process can be viewed as a tree with non-terminal nodes being control constructs, each of which has children specified using components, and the leaves of the tree being invocations of other (composite or non-composite) processes. OWL-S' control constructs are the following: Sequence, Split, Split + Join, Choice, Any-Order, Condition, If-Then-Else, Iterate, Repeat-While and Repeat-Until. Despite their names resembling those of control structures in programming languages, this fact does not mean that a composite process will behave this way, but instead that this behavior can be accomplished by the service client through the sending and receiving of a series of messages. If the composite process has an overall effect, then the client must perform the entire process in order to achieve that effect. Figure 3.3 presents some selected classes and properties of OWL-S' service model.

- **Service Grounding:** It provides information on how a service requester can interoperate with a WS through the exchange of SOAP messages, by mappings elements of this sub-ontology to elements in a WSDL file. Service Grounding is a middle ground between OWL-S' Service Model and WSDL, bringing the two together by mapping atomic processes to WSDL operations, and process' inputs and outputs to operations' inputs and outputs, the former being described by OWL and the latter by XML Schema. Similarly, composite processes use the same mapping - grounding, as composite processes are just a set of other processes, with the additional information of the control and data flow between them being interpreted by an OWL-S process engine. In order for the automated discovery and execution of an OWL-S description to be possible, it must consist of at least one service profile, a process model and one or more groundings. The aforementioned sub-ontologies are parts of the top-level concept Service, which is a point of reference for declaring a WS, as to declare one, an instance of Service must be created. As shown in Figure 3.3, the properties *presents*, *describedBy* and *supports* are properties of Service, with ServiceProfile, ServiceModel and ServiceGrounding being the ranges of those properties respectively. Each instance of Service will *present* a ServiceProfile description, *describedBy* a ServiceModel description, and *support* a ServiceGrounding description (Davies, 2006).

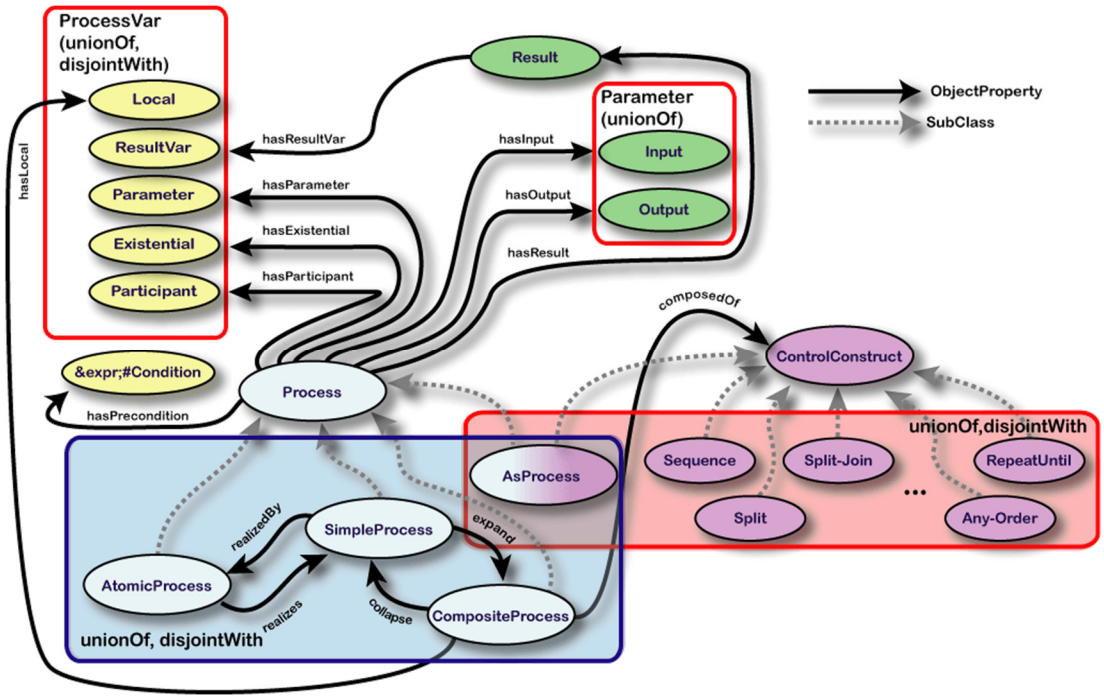

Figure 3.3: Selected classes and properties of OWL-S' service model (Martin et al., 2004)

3.2.2.2 WSMO

The Web Service Modeling Ontology (WSMO) (Lausen et al., 2005) provides a conceptual framework and a formal language for semantically describing all relevant aspects of WSs in order to facilitate the automation of discovering, combining and invoking electronic services over the web. WSMO and OWL-S are very closely related; both aim at the provision of a comprehensive conceptual model for SWSs. An important foundation point of the work on WSMO was the model provided by OWL-S, however the authors of WSMO maintain that OWL-S has a number of serious fundamental flaws that give rise to problems when attempting to use it in practice. The WSMO working group's goal was to create a more complete conceptual model for describing WSs than that of OWL-S, despite both frameworks addressing the same problem.

The conceptual model of WSMO comprises four main elements:

- **Ontologies**: They are used as the underlying data model for SWS, providing a formal terminology of the information used by all the other WSMO elements. They are the basis of all the exchanges of information during the execution of WSs and allow the formal definition of concepts and attributes, as well as of restrictions and rules in regard to them. Their main use, as in other models, is that they provide machine-understandable semantics and ensure semantic interoperability between SWS.

- **Web service descriptions**: They describe the functional (capability) and behavioral (usage) aspects of a WS. WSMO defines a description model that encompasses the information needed for

automatically determining the usability of a WS. Each WS can only offer one functional capability. However, it can offer more than one interfaces needed to access it. A WSMO WS is characterized by four elements; its non-functional properties, its functional capability along with any of its interfaces, a choreography that describes how to interact with it and, finally, an orchestration defining how the functionality of the service is achieved by aggregating other WSs. Similarly to the OWL-S model, a WSMO WS can be considered as a number of state transitions that move from a state to another based and constrained by its defined preconditions, postconditions, assumptions and effects (all being optionally defined). Before a WS is invoked, preconditions define the required state of the information space available to the WS and assumptions define the state of the world outside that information space. In an analogous manner, after a WS is executed, postconditions define the state of the information space and effects describe the state of the world outside that information space. All WSMO does is to define the functionality and behavior of a WS, without having to also define how it is internally implemented.

- **Goals**: They represent the WSs requester's objectives, i.e., describe what objectives a service client wishes to accomplish by interacting with a WS. Goal descriptions are defined from the perspective of a service requester and, thus, they are differentiated from the WSs' descriptions, although both contain the same structure. Both are defined with attributes for non-functional properties, specified functionality/capability, related ontologies, expected behavior of a WS in interfaces and mediators. However, as these descriptions can be semantically similar, but syntactically different, since one is defined by the service requester and the other by the service provider, logical reasoning may be needed in order to achieve the matching of goal and WS descriptions, that is service discovery. These mismatches can be resolved with the use of mediators. This element constitutes one of the main distinctions between OWL-S and WSMO, as in OWL-S only one element is used to describe both the perspective of the service client and the service provider.

- **Mediators**: They are entities that are used to establish interoperability between resources. They handle heterogeneity, that is, resolve possible mismatches between any two WSMO elements that should be interoperable. Similarly to WSMO WS descriptions that do not describe the WSs' implementation, mediators can be optionally grounded in a goal, a WS or another mediator. Each of the four top-level elements that comprise WSMO's conceptual model is represented as a class with several attributes and all of them together are considered as necessary for a complete semantic description of WSs. All WSMO elements have an hasNonFunctionalProperty attribute, that is used to describe any of their non-functional properties, such as QoS (Quality of Service) characteristics.

3.2.2.3 Semantic Web Services Framework (SWSF)

The Semantic Web Services Framework (SWSF) (Battle et al., 2005c) was created with the goal to extend the work of OWL-S and address its disadvantages, namely the fact that – according to the

authors of SWSF – OWL-S is not well suited to describe processes; SWSF provides a full conceptual model and a sufficiently expressive language capable of describing WSs' process models. The framework consists of two elements: The first is the Semantic Web Services Language (SWSL) (Battle et al., 2005a), a general-purpose language that provides formal characterizations of WSs' concepts and descriptions of individual services. It is domain-independent, with any constructs specific to SWS being provided by the Semantic Web Service Ontology (SWSO) (Battle et al., 2005b). SWSL consists of two sublanguages: SWSL-FOL, based on first-order logic (Hodges, 2001), and SWSL-Rules, based on the logic-programming paradigm. The second is SWSO that presents a conceptual model by which WSs can be described, and an axiomatization, or formal characterization, of that model. The complete axiomatization is given in first-order logic, using SWSL-FOL, with a model-theoretic semantics specifying the precise meaning of the concepts named First-Order Logic Ontology for Web Services (FLOWS). Rules Ontology for Web Services (ROWS) is the ontology resulting from the systematic translation of the axioms from FLOWS into SWSL-Rules.

3.2.2.4 WSDL-S

WSDL-S (Rama Akkiraju et al., 2005) is another approach for enhancing the expressivity of WSDL descriptions with semantic annotations, by employing concepts analogous to those in OWL-S, being however agnostic to the semantic representation language used. In contrast to OWL-S and WSMO, it does not specify an ontology for the definition of SWSs; however, (Paolucci & Wagner, 2006) analyzed the relation between OWL-S and WSDL-S and concluded that there is a certain level of overlap between the two approaches, despite there also being differences between them.

3.2.2.5 Semantic Annotations for WSDL (SAWSDL)

The Semantic Annotations for WSDL (SAWSDL) standard (Kopecky et al., 2007) is used to apply semantic annotations to WSDL documents, being heavily influenced by WSDL-S, as among other similarities, they share the same model reference annotation and have similar schema mapping annotations. SAWSDL aims to allow interoperation between different semantic WSs frameworks, such as WSMO and OWL-S, through extending WSDL with pointers to semantics needed in order to achieve automation. SAWSDL extends WSDL by creating an additional layer on top of it that allows WSDL components to specify their semantics. It defines extension attributes that can be applied to WSDL and XML Schema elements so as to annotate WSDL interfaces, operations, and their input/output messages.

3.3 Web Service Composition

3.3.1 Definition

As described in Section 3.1, WSs are self-contained modular applications with a self-describing functionality that can be published, located and invoked automatically. For this reason, they are often used as is, i.e., a single WS provided by a WS provider is invoked by a WS requester. However, more than often the functionality of a single WS is insufficient. Therefore, the ability of connecting the functionality of various heterogeneous, belonging to different organizations, WSs throughout the web at runtime is very appealing. Thus, if no single WS meets the criteria and desired functionality of a service client, there should be the ability to create a new one, based on others, already existing in various repositories, having the desired aspects. This ability will not only reduce the cost, but also the time and complexity needed to create a new WS from scratch.

A well-known and used example in the WSC literature is that of an online travel agent ((McIlraith et al., 2001), (Srivastava & Koehler, 2003), (Curbera et al., 2002), (Menasce, 2002)). The core of this scenario is that if it is known that separate WSs exist with the sole functionality of booking airplane tickets, hotels and rent cars, but a service requester desires the functionality of booking whole trips and not just these separate aspects of one, then the problem of the composition is to create a composite WS that will be able to discover and invoke the necessary WSs to book both a flight and a hotel, rent a car, without committing to a single booking if the other two have not been guaranteed.

However, such a process is difficult and time-consuming for a human, even an expert in the field of WSs. The reasons behind this complexity are mentioned in (Rao & Su, 2003), namely that the number of available WSs has increased considerably in the past few years, it is expected to continue growing and, as such, huge WSs repositories have to be searched in order to successfully accomplish the WSC process. Furthermore, WSs can be created and/or modified dynamically, and the system responsible for the composition process has to be able to detect such changes at runtime in order to act based on the current information. Third, as we have shown in previous sections, there are various frameworks that can be used to create and describe SWSs, thus as a de facto standard for this purpose does not exist, further problems for the manual creation of a composite WS exist. WSs have several interaction patterns and a composition must be able to use WSs as they are defined by their providers. Finally, since the WSs are real world applications, their success is not guaranteed. As such, a specific component invocation can fail or a WS may not produce the desired result. Such a situation will be further discussed in the forthcoming chapters.

3.3.2 Manual Web Service Composition Standards

Manual WSC languages are used in order to be able to compose complex business services that are comprised by more than one WS interoperating with each other. This composition is considered manual as the description of the order of execution of the operations and relations between them, as

well as of the services that provide the functionality, is static and accomplished by human experts. Since our main focus in this thesis is automated WSC, we will restrict to a brief mention of the relevant standards. The following section briefly presents some of the available protocols and languages for manual WSC. It is interesting that, due to the plethora of web service related standards that have been proposed, many having overlapping features and use, an acronym for the phenomenon was ironically proposed - WSAH (Web Services Acronym Hell) (van der Aalst, 2003).

3.3.2.1 BPEL

The Business Process Execution Language for Web Services (BPEL4WS, or more commonly referred to as plain BPEL) (Andrews et al., 2003) is used to define the control logic for WSs' coordination in a business process, as well as the relevant interaction protocols between a business process and its partners. Moreover, it also defines the states and logic of coordination between these interactions. BPEL is an XML-based language that was created with the purpose of combining the best aspects of two different WS flow specification protocols, IBM's WSFL (Web Services Flow Language) (Leymann, 2001), a block-structured language with basic control flow structures, and Microsoft's XLANG (Web Services for Business Process Design) (Thatte, 2001), a graph-oriented composition language, mostly relying on the concept of control links, and not directly supporting iterations. However, the fact that the two languages that influenced BPEL are based on different models and paradigms, lead to a language that had overlapping constructs (van der Aalst et al., 2003).

There are two different types of processes in BPEL, abstract and executable ones. Business interaction protocols are called abstract processes, specifying the public message exchanges between two different parties without revealing the actual implementation of these parties to each other. Executable processes model the behavior of different parties in a business interaction via workflow descriptions that are created using basic and structured activities. Basic activities are the simplest form of interaction with external services, being non-sequenced and individual steps of interaction, and are used for processes such as the handling of exceptions, or to control the passing of data; <*invoke*>, <*reply*>, and <*receive*> are the three basic activities that are used for WSs interactions. Structured activities on the other hand, are related to the internal process' control flow. They define the order in which a set of activities occurs, through the use of constructs such as <*switch*>, <*while*> and <*pick*> that express sequential flow, conditional branching, looping, etc.

3.3.2.2 WSCI

The Web Service Choreography Interface (WSCI) (Arkin et al., 2002) is an XML-based language used for the description of the exchange of messages between stateful WSs that are involved in choreographed interactions with each other. Through the use of it, WSs can interact with each other in an unambiguous manner, as it specifies the temporal and logical dependencies that the message flow has and in that way all the services can conform to the specified collaboration/ contract. However, it

only defines the observable behavior of WSs and it does not deal with the definition of the internal behavior of a WS, that is, it does not provide either the definition or the implementation of the internal processes that are involved in the message flow. It is related to WSDL in the sense that it describes the way that the operations defined in the WSDL documents are choreographed. Thus, the stateless WSDL documents are enriched with choreography attributes and stateful message exchanges, through the dynamic interface provided by WSCI. Similarly to other protocols, WSCI also supports transaction and exception handling.

3.3.2.3 BPML

The Business Process Management Language (BPML) (Arkin, 2002) is an XML-based language developed by the Business Process Management Initiative, which, like BPEL, provides a model and grammar for expressing abstract and executable business processes. BPML incorporates WSCI into its framework (although different coordination protocols can also be used), sharing the same underlying process execution model. WSCI can be used to describe public interactions and BPML for the development of private logic implementations. A process in BPML directs the composition and execution of activities that have a specific functionality, while it can also be a part of a composition. BPML provides process flow constructs and activities similar to that of BPEL, with its activities being either simple or complex, and all activities in a process having a common behavior. BPML defines 17 activity types and three basic process types (Large Scale Distributed Information Systems Lab - University of Georgia, 2005). It also interoperates with WSDL in the sense that it supports importing and referencing WSDL service definitions.

3.3.2.4 ebXML

Electronic Business using eXtensible Markup Language (ebXML) (Cover, 2006) is an XML-based framework having the general purpose of giving businesses the capability to discover one another and conduct electronic commerce exchanges through well-defined messages and standard business processes. ebXML comprises three different standards, the one mostly related to manual WSC being BPSS (Malu et al., 2002):

- **CPP (Collaboration Protocol Profile):** This is similar to a UDDI registry entry
- **CPA (Collaboration Protocol Agreement):** This contains the business agreement among cooperating partners.
- **BPSS (Business Process Specification Schema):** This is used to configure the participating business systems to support their collaboration, by defining the choreography and communications between services.

ebXML determines the actual exchange of business documents and messages between the partners. A set of these exchanges can be defined through BPSS and users can then extract information from the

relevant BPSS documents and configure their systems to support business transactions by agreeing on exchange patterns and a role in the choreographed transactions. ebXML does not directly support the description of data flow in a transaction, however it has explicit support for QoS aspects of exchanges, such as authentication and non-repudiation.

3.3.2.5 WS-CDL

The Web Services Choreography Definition Language (WS-CDL) is defined as "*an XML-based language that describes peer-to-peer collaborations of parties by defining, from a global viewpoint, their common and complementary observable behavior; where ordered message exchanges result in accomplishing a common business goal*" (Kavantzas et al., 2004). It is aimed at defining interoperable, peer-to-peer exchanges regardless of the platform or programming model used. For these exchanges to be able to work, some rules between the participants must be agreed upon. Using WS-CDL, a global definition of the rules and constraints that govern the exchanges is created describing the behavior of all the participants, and from that point any of the parties involved can create solutions that respect this "contract".

3.3.2.6 WSCL

The Web Services Conversation Language (WSCL) (Banerji et al., 2002) allows the abstract interfaces of WSs, that is, the externally visible behavior, e.g., business level conversations or public processes supported by a WS, to be defined. Both the conversations and the WSCL document that defines the sequencing order in which the conversations are exchanged are XML documents. WSCL and WSDL descriptions of a WS can also be combined, either by adding protocol bindings in WSDL to a conversation described in WSCL, by adding choreography to a WSDL port type description, or finally, by providing a full WSDL and WSCL description for the same port type/conversation.

3.3.3 Semi-Automated Web Service Composition

Semi-automated composition refers to the automation of a part of the WSC process, particularly the one related to the search of the appropriate - in terms of meeting the criteria of the composition - WSs in a registry containing a plethora of unsuitable ones, thus still burdening a human expert with defining the interactions between them. As such, a mechanism is needed in order to solve the problem of the difference between the human concepts and the ones computers use. The use of Semantic Web technologies eases the implementation and use of such systems as all information, the functionality of WSs, as well as their properties and relations, have well defined semantics.

In semi-automated WSC systems, the search for suitable WSs is dynamic, that is, the selection of atomic processes is automated with the client only specifying a set of constraints and preferences. In contrast to that, the composition process can be static, that is, the client must build an abstract process model before WSC begins, including the set of tasks and the dependencies between them. In this case,

the automation of the process only refers to the selection and binding of WSs, and the goal is similar to the composition of workflows, as in this context a composite WS is very similar to a workflow.

(Sirin et al., 2002) implemented the first system to directly combine the SWS descriptions with actual invocations of their WSDL descriptions, thus being able to actually execute the composed services. The system relies on the user to create a workflow, providing him with the available choices he has at each step. The WSC process begins with a WS from the available registry being selected by the user. A query requesting the information relating to the inputs of the chosen WSs is sent to the knowledge base, and for each of these inputs, a new query is created so as to get the list of the possible WSs that can supply the appropriate data for this input.

EFlow (Casati et al., 2000), a platform defining and managing composite WSs, is another example of semi-automated composition, modeling composite WSs as graphs that define the execution sequence of their nodes. It uses a static workflow generation process, with the graph being manually defined by a domain expert. However, the graph can be updated automatically, as can be the binding of the nodes to concrete services, which is represented by arcs in the graph. The graph itself can contain service, event and decision nodes. Decision nodes define alternative paths and rules that govern the workflow, while service nodes represent calls to WSs, either atomic or composite. Finally, event nodes enable WSs to receive various types of events. EFlow does not support non-determinism, as each time a WS in the workflow is not available at runtime, a new binding of a concrete WSs to the specific node has to be created, since there are no guarantees that the WSs will actually be invocable and running. Furthermore, as (Li et al., 2008) note, the system needs too much manual participation at the execution phase and, although it supports the modification of the process for dynamic WSC, it can't guarantee its correctness.

In (Li et al., 2008) the authors present a method for semi-automated WSC using a process ontology tree, of which each available WS is a node. The proposed framework deals with WSs discovery, composition, verification and execution, and it works in two phases: First BPEL is used to describe the static part of the WSC process and to bind the corresponding WSs. Then, the dynamic part of WSC is described using the process ontology by extracting all the relevant relations and rules from it and then, based on the service client's criteria, the system can search for the target nodes - WSs in the tree. Finally, new and substitute service compositions are automatically created. The proposed framework is semi-automated as a user is needed in the first phase of the system to predefine the fixed part of the process, and then sculpt out the dynamic part of the second phase that is unable to be determined in advance.

The Polymorphic Process Model (PPM) (Schuster et al., 2000) also combines static and dynamic methods for semi-automated WSC. It supports static WSC by referencing processes that consist of other, abstract ones, i.e., sub-processes that lack an actual implementation, but whose functionality is defined completely. These sub-processes are implemented by concrete WSs and are bound at runtime. It also supports dynamic WSC through the modeling of service-based processes, that is, modeling a

service as a state machine, along with the possible states it can be in, as well as the transitions between them. In that way, dynamic WSC can be achieved by reasoning based on the state machine.

In (Albreshne & Pasquier, 2010) OWL-S is used to facilitate the WSs discovery phase, guide the user to limit the available choices for appropriate WSs and define his preferences to finally reach the composition goal. The authors' main objective is to provide generic process templates to non-expert users as a mechanism to ease the WSC process. These templates define a workflow composed of several WSs with specific functions described by their semantic annotation, and are linked with control flow. The user then manually customizes the templates so that they meet his requirements and preferences. In short, the system's phases are the following: First, the system searches for available WSs based on their functions. Then the appropriate ones are selected and a workflow is created with the help of the user who is asked to make decisions regarding it and express his preferences and constraints. Finally, the abstract WSs are bound to concrete ones and the data flow in the process is assigned so that the final composite WS can be executed.

The ASTRO approach (Marconi et al., 2008) is a framework that supports all the phases of WSC, from the specification of control and data flow requirements, to the automated synthesis and execution of the new composite WSs. Its WSC process consists of two phases: During the first one, an expert user specifies control and data flow requirements based on abstract BPEL and WSDL so as to create a preliminary version of the new composite WSs. Based on the user's specifications, an executable composite process, defined as executable BPEL, along with its user interface, described in WSDL and abstract BPEL, are automatically created by the system. Then the result of the first phase can be iteratively enhanced by the expert user in the second phase regarding both the composition requirements and the interface.

(Zahoor et al., 2009) present a rule-based approach for the semi-automated WSC problem, which relies on a user to guide the composition process by selecting WSs instances or constrained WSs types, i.e., create "*nodes*". The user can specify a WS node having a specific type, such as "*Museum*" or "*Cinema*", if its instance is not known a priori, and also add constraints to it that are to be satisfied in the final step of the composition, that is the binding of nodes to concrete WS instances. Before this step, however, the user has to use a set of custom control/data flow connectors to create a flow between different WS nodes or instances. These include connectors such as "*sequence*", "*split*", or "*aggregate*", and are mostly similar to those in OWL-S, with the main difference being that they can handle both control and data flows. The last step in the process, as already mentioned, is that of binding specific nodes to concrete WS instances, using a set of rule-based queries satisfying the associated constraints, which are used both during WSs discovery, and in order to define a specific solution among a set of available ones.

An example of a non-AI planning approach is presented in (Sheng et al., 2009); the output composite WSs are specified as process schemas, and the atomic WSs that comprise the composite ones are selected at runtime, based on non-functional constraints specified by the users. The presented system, CCAP, is based on three core services: coordination, context and event services that schedule

and implement user-configured adaptations of WSs at runtime. CCAP only makes use of technologies such as XML and UDDI, without taking into account the semantic matching capabilities that can be achieved by using ontologies and semantic specifications. Moreover, non-determinism in the execution of WSs is assummed to have already been described in user configured exceptions. (Sheng et al., 2009) evaluate the system's performance based on scalability and adaptability criteria, and also provide a usability study based on 41 users of different educational backgrounds who were asked to use the system and report their experience by answering a questionnaire.

The method in (Birchmeier, 2007) generates alternative plans in a semi-automated way, which are then annotated with quality information regarding them. The planner computes incomplete plans, either due to lack of knowledge or because it was unable to find a plan, and presents them to the user. This is achieved using a combination of A* (Hart et al., 1968) and beam search (Furcy & Koenig, 2005), that is, by keeping a limited list of partial plans. The algorithm only considers a set of k best nodes at each time, k being equal to the number of desired alternative plans. This set is ordered according to a parameter representing the degree of similarity between the path of the first selected node and the path of the node that will be valued. The algorithm stops when k solutions are found, or no more can be found. (Birchmeier, 2007) also suggest a different method to generate these alternative plans, through the use of a mixture of AI planners with variant underlying planning techniques in parallel. It is possible, however, that more than one planner in the mix produce the same plan, resulting in fewer alternative ones. For this type of plan generation, a load balancing method was also developed in order to make the parallel execution of different planners more scalable.

Finally, a semi-automated, interactive and limited, contingent WSC method is presented in (Mediratta & Srivastava, 2006) that allows the intervention of users in order to control the search over the possible plans generated. The problem formulation allows for incomplete initial state descriptions and sensing actions; only part of the total branches, which are exponential in the number of unknown terms, may lead to the goal. For this reason, users are allowed to insert default branches instead of the ones that they are not interested in, as well as define soft constraints. In essence, in order to compensate for the lack of complete modeling, user acceptable plans are allowed, which specify only a subset of the branches that are guaranteed to lead to the goal (Hoffmann et al., 2009).

The search space consists of belief states in an AND-OR graph; sensing actions form the AND part of the graph and the rest of the actions the OR part of it. Users can specify which part of the search space is considered good with the help of a cost measure that estimates heuristic distance, so that the search is focused on those states. In this way, the AND part of the graph can be pruned efficiently, and the Planning Graph heuristic (Ghallab et al., 2004) can be used to prune the rest of the graph. A* is used, with the cost of a belief state being equal to the number of actions needed to be applied from the initial belief state to reach the current one. The cost of each literal is equal to the minimum of the sum of the costs of the actions that have to be executed in order to reach the literal when planning backwards from the goal. Search is conducted in a forward fashion, while the heuristic is computed backwards, so as to approximate the distance from the goal belief state to the current one.

It only takes into account relevant actions, either directly supporting the goals, or indirectly, by supporting actions that support the goals themselves, by using a RIFO process (Nebel et al., 1997). The result of the planning process is complete provided that the user does not define external specifications that prune the search space. The method was compared to a naïve planning method that generated a complete plan under all conditions, filtering specific branches afterwards based on the user's constraints, as well as against MBP and SGP. The evaluation was based on two planning domains from the planning benchmark problem set in SGP. The method performed significantly better than the naïve implementation, but consistently considerably worse than SGP in both domains, while also being worse than MBP in one domain, and better in the other. Results that are consistent with the evaluation results previously presented in (Bryce et al., 2006) and (Cimatti et al., 2003).

3.3.4 Automated Web Service Composition

In its simplest form, automated WSC is achieved by linking existing atomic or composite WSs in sequence in order to create new services. In more complex forms, other control structures can be used, to link services in parallel or use alternative ones, whereas the problem can be formulated to include further constraints, e.g., QoS optimization ones (Rao & Su, 2003). In a deterministic setting, we are forced to assume that we can predict the results of the executed actions precisely. However, such a setting is often restrictive and not realistic in WSC, and the adoption of a non-deterministic assumption, i.e., that the outcome of a WS is not known a priori, allows us to compute more flexible plans. Since WSs operate in an ever-changing environment, they may neither be always active nor have the same interface or even provide the same functionalities; moreover, it is always possible for a WS' execution to be unsuccessful or have undesired effects. As such, this thesis assumes that non-determinism is inherent in the WSC domain and formulates the WSC problem as such. The problem of automated WSC has been shown to be computationally hard, most recently in (Muscholl & Walukiewicz, 2008) and (Nam et al., 2011); in (Muscholl & Walukiewicz, 2008), it is stated that if the WSC problem is formulated as a composition of finite state machines, each one representing a WS, then in its simplest setting, when the composition is fully asynchronous, there is an *EXP-hard* lower bound on the complexity of the task. In (Nam et al., 2011), it is shown that solving the composition problem of non-deterministic WSs with complete information is *EXP-hard*.

A very large part of the current research on automated WSC has been focused on AI planning techniques. Since dynamic WSC methods require the automated generation of plans, i.e., composite WSs, AI planning techniques in various forms can be used. The general assumption that such methods adopt is that WSs can be characterized by preconditions and effects, that is, that WSs are software modules that receive input data and produce outputs according to that. Moreover, it is assumed that the preconditions of a WS describe a state that is required for it to be executable, and its effects describe the world state after its execution. As aforementioned, OWL-S is a SWS approach that has a strong relation to AI planning and PDDL in particular. Its Service Profile specifies a WS' functionality by

using state transitions, that is, specifying the necessary inputs and outputs that are expected to be used, as well as the preconditions and postconditions that must hold before / after a service is executed (see Section 3.2.2). As such, OWL-S to PDDL translations are possible and relatively straightforward when only declarative information is considered. In this way, WSC can be achieved by such a translation and the use of standard AI planning systems to produce valid plans.

The main focus of this section is automated non-deterministic WSC methods that view the problem as one in which WSs are atomic planning operators and use AI planning techniques to tackle it. However, in the interest of providing a complete review of the current state-of-the-art WSC methods, and since non-deterministic WSC methods are scarce in the literature, other methodologies that are deterministic in their problem formulation are also reviewed. All the WSC methods that are reviewed are presented in Table 3.1, along with their differentiating attributes.

In (Bertoli et al., 2010), (Bertoli et al., 2006), (Pistore et al., 2005) WSs are viewed as stateful services, which can be modeled as state transition systems and whose description of behaviors imposes constraints on their interactions. In this line of research, state transitions are equivalent to asynchronous messages that are exchanged as a result of an atomic action's execution (Pistore et al., 2005). The goal of the output WS is described in temporal logic, with the WSC problem being tackled mainly through symbolic model checking techniques. In a similar line of research, e.g., (Berardi et al., 2003), (Berardi et al., 2005), (Berardi et al., 2005b), the "roman model" describes services as state automata that can have conversations with clients that issue requests expressed as virtual services. That is, each WS is modeled as a transition system capturing its possible conversations with a generic client; the result of WSC is the realization of a virtual target service, a concrete composite WS that is also described as a transition system, by combining only actual WSs that are available in a registry (DeGiacomo et al., 2014). In (Berardi et al., 2003) services are modeled as deterministic finite transition systems; in (Berardi et al., 2005) and (Berardi et al., 2005b) non-determinism is taken into account, either by allowing it in terms of the required target service definition, or by allowing only partially controllable WSs with incomplete information on their exact behavior. The roman model was also subsequently utilized in (De Giacomo & Sardina, 2007) and (Calvanese et al., 2008). Methods such as the aforementioned, however, are radically different from the scope of this thesis and, as such, will not be reviewed further in this section.

In this section, first we present several prominent deterministic WSC methods. Next, methods that bridge the gap between non-deterministic and deterministic ones are reviewed; this section presents WSC methods that either generate multiple solutions before selecting a single one, or incorporate alternative WSs to their output plans. Finally, we review the evaluation process of the methods present in the literature.

Table 3.1. Synopsis of web service composition methods

Method	*Determinism*	*Based on*	Heuristic/ Stratified	Evaluation
OWLS-Xplan	Deterministic	FF + HTN decomposition	•/-	IPC3
PORSCE II	Deterministic	LPG-td / JPlan	•/-	Single domain
WSPR	Deterministic	Two-step search algorithm	•/-	EEE05 - ICEBE05
(Oh et al., 2009)	Deterministic	A*	•/-	-
(Huang et al., 2009)	Deterministic	Two-step forward filtering algorithm	-/-	Web Service Challenge 2009
AWSP	Deterministic	Backward / forward search based on A*	•/-	WSBen
ENQUIRER	Deterministic	SHOP2	-/-	Two planning domains
SCUP	Deterministic	SHOP2 + Best first search	•/-	Two planning domains from IPC-5
(Zúñiga et al., 2010)	Deterministic	ND-SHOP2	-/-	Case study implementation
(Zuñiga et al., 2014)	Deterministic	SHOP2	-/-	Case study implementation
(Wagner et al., 2011)	Alternative WSs	Keikaku	-/-	Random problem generator – Against QSynth
(Deng et al., 2014)	Alternative WSs Multiple solutions	BTSC-DFS	-/•	China Web Service Cup -WS Challenge 2009
(Wagner et al., 2012)	Alternative WSs Multiple solutions	Genetic algorithm	-/-	Custom synthetic problems
(Birchmeier, 2007)	Multiple solutions	A* and beam search	•/-	-
(Rodriguez-Mier et al., 2011)	Multiple solutions	Backward search based on A*	•/•	WS Challenge 2008
(Cui et al., 2012)	Multiple solutions	Viterbi algorithm	-/•	Custom problems - Against human expert

(Meyer & Weske, 2006)	Non-deterministic	Enforced Hill-Climbing	•/-	-
(Brafman & Shani, 2012)	Non-deterministic	Conformant-FF	•/-	Two artificial domains - Against the DLVK tool
(Mediratta & Srivastava, 2006)	Non-deterministic	A*	•/-	Two planning domains- Against MBP and SGP
(Kona et al., 2008)	Non-deterministic	Prolog with Constraint Logic Programming	-/•	Single domain - WS Challenge 2006
(Kuzu & Cicekli, 2012)	Non-deterministic	SimPlanner	•/-	Case study implementation
(Dacosta et al., 2004)	Non-deterministic	-	•/•	Implemented prototype
(Zou et al., 2012)	Non-deterministic	FF and SatPlan06	•/-	ICEBE05 – Against WSPR

3.3.4.1 Deterministic methods

Several approaches exist that make use of the connection between PDDL and OWL-S to tackle the WSC problem. OWLS-Xplan (Klusch et al., 2005)(Klusch & Gerber, 2006) was one of the first methods to be based on such a translation; it translates given initial and goal state ontologies in OWL and WSs in OWL-S 1.1 to equivalent domain and problem descriptions specified in PDDL 2.1, and uses a specific AI planner, Xplan (Schmidt, 2005), to generate a plan sequence that solves the planning problem. Xplan is a hybrid planner that combines guided local search with graph planning (specifically, using FF), and a simple form of HTN decomposition. This approach maintains a higher degree of flexibility compared to pure HTN planners like SHOP2, while also providing guarantees of always finding a solution if one exists, without depending on whether the given set of decomposition rules for HTN planning allow the generation of a plan consisting solely of atomic actions.

OWLS-Xplan incorporates a conversion tool that translates OWL-S 1.1 service descriptions to corresponding PDDL 2.1 descriptions, with the domain description containing the definition of all types, predicates and actions, and the problem description all objects, the initial state and the goal state. The translation does not output a standard PDDL file, but a modified version of it in XML, called PDDXML, that simplifies parsing, reading and communicating PDDL descriptions using SOAP. Any non-deterministic changes to the world state during the composition planning process are dealt with by including a replanning component that during the plan execution checks whether all the preconditions of the current action to be applied hold. If not every precondition holds, the system is

informed about which facts are invalid and at which position in the plan the problem occurs. If the original plan cannot be executed without making changes to it, then the replanning component searches for an alternative path in the connectivity graph from the current state to the goal.

PORSCE II (Hatzi et al., 2010), (Hatzi et al., 2011), in combination with VLEPPO (Hatzi et al., 2007), is a similar approach to OWLS-XPlan, in that it transforms the original problem into a new – semantically enhanced one – compatible with PDDL and invokes external planners to solve it. Specifically, the planning domain is derived from the initial WSC problem, and the OWL-S descriptions are transformed into a PDDL planning problem so that independence from specific planners is maintained, while the semantic information of OWL-S is used for the enhancement of the composition process and the possible approximation of the output composite service if exact solutions cannot be found. An important feature of PORSCE II is the enhancement of the planning problems with semantically equivalent or relevant concepts. This feature is implemented through the use of a Description Logic Reasoner, specifically the Pellet DL Reasoner (Sirin et al., 2007). The reasoner computes the inferred relationships between the domain ontologies' concepts, based on different criteria; first, if two ontology concepts have a specific hierarchical relationship, with the possible relationships acknowledged being two concepts *A* and *B* being the following:

- *exact*(*A,B*): The two concepts share the same URI or are equivalent in terms of OWL class equivalence
- *plugin*(*A,B*): Concept *A* is subsumed by concept *B*
- *subsume*(*A,B*): Concept *A* subsumes concept *B*
- *sibling*(*A,B*): The two concepts have a common (not necessarily direct) superclass *T*

Second, if the semantic distance between the two concepts does not exceed a specific threshold, set by the user. This distance is calculated using two different measures, the Edge-Counting Distance and the Upwards Cotopic Distance (Hatzi et al., 2011). If two concepts meet both of the aforementioned criteria, they are considered relevant by the system. In this way, syntactically different but semantically equivalent or even similar concepts can be identified and used. In the latter case, if there are concepts that match exactly, input concepts can still be matched to output ones approximately. This feature, called "semantic relaxation", allows the generation of composite WSs that may be less accurate than desired, but still serve the original preferences of the user as well as possible. The problems can be solved locally, with the use of two different planners, LPG-td (Gerevini et al., 2006), a planner based on local search and planning graphs that handles PDDL 2.2 domains, and JPlan (Manzalawy, 2009), an open-source Java implementation of GraphPlan. However, any PDDL compliant planner can be used in collaboration with the system. The system offers the feature of replacing a specific atomic WS, either by discovering all atomic WSs that could be used alternatively, or if a single atomic WS cannot be found, by perform replanning in order to locate a set of services that have the same effect as the one chosen to be replaced, and can therefore substitute it without causing any problem to the overall solution.

WSPR (Web Service Planner) (Oh et al., 2007) is based on a two-step search polynomial time algorithm. First, the cost of achieving individual parameters is computed, assuming that each WS can be invoked with a non-negative cost; the forward search starts from the initial state generated from the user's request, and then an optimal plan is approximated through the use of regression search, using the results of the first step as guidance. The backward search is heuristic, having the goal of minimizing the length of the solution, i.e., the number of WSs in the plan. Moreover, (Oh et al., 2007) assume that a WS contributing more to match a sub-goal earlier in the search will also be helpful to reach the initial state faster, and as such, tends to favor such services during search. Simultaneously, the heuristic tries to avoid choosing WSs that are only partial matches. (Oh et al., 2009) improve upon WSPR, using A* as the search algorithm. A* is used with the aggregated QoS value up to the current node as the path-cost function, and favors nodes whose contribution to find the remaining parameters is the maximum, using a heuristic estimate that is similar to that of WSPR. The use of A* allows for the generation of optimal compositions if the heuristic is admissible; however as (Rodríguez-Mier et al., 2012) note, the drawback of the heuristic used in (Oh et al., 2009) is that it is not informative in the case when only the last services of a composition produce all the required parameters. Moreover, in this case, no experimental results as to the method's efficiency were provided.

(Huang et al., 2009) also propose a two-step method based on a forward filtering algorithm to prune the search space, followed by a backward – dynamic – search that computes the optimal QoS values for the WSC problem. The filtering algorithm removes two types of services; first, services that cannot be enabled by the available inputs and, second, services that provide the required outputs but not with the optimal value for the response time or throughput.

AWSP (Automatic Web Service Planner) (Wu et al., 2011) is a heuristic search method that utilizes two implementations of the A* algorithm and a custom heuristic state space search. It can function either as a forward state search planner or as a backwards one, using two heuristics that rely on the concept of parameter distance defined by the authors. The distance between two parameters is defined to be equal to one if they are part of the input and the output parameters of a single WS, respectively, and infinite if no invocation between them is possible. In any other case, i.e., when longer paths exist that require the generation of intermediate parameters, parameter distance is equal to the length of the shortest path of invocation between them. This method requires pre-computing the parameter distance between any two pairs, thus if any modification is made to the WS registry, e.g., a WS is added or removed from it, this computation has to be done again.

A variety of methods utilize HTN techniques, particularly SHOP2, as the underlying planning module; The most notable case of HTN planning used for WSC is presented in (Sirin et al., 2004) (Wu et al., 2003), based on SHOP2. Its motive is the fact that OWL-S (and specifically its process sub-ontology) and Hierarchical Task Networks share similar features. Specifically, OWL-S processes can be translated into tasks to be achieved by SHOP2, and the plan that is generated presents a collection of atomic process instances that achieve the desired functionality. First, an OWL-S process model is translated to a SHOP2 domain and an OWL-S WSC problem to a compatible SHOP2 planning

problem. Then the planner generates a plan that can be converted back to an OWL-S process and executed by an OWL-S API. Neither (Sirin et al., 2004) nor (Wu et al., 2003), however, provide any experimental results in regard to time or quality measurements, except for an analysis of a case study.

Other approaches that are based on SHOP2 are ENQUIRER (Kuter et al., 2005) and the SCUP algorithm (Lin et al., 2008). The first extends SHOP2 to be able to cope with domains in which the information about the initial state of the world may not be complete, but it is discoverable through plan-time information-gathering queries, whereas the second tries to incorporate the users' preferences in the WSC process. ENQUIRER keeps the original SHOP2-language mapping unchanged, while extending its usage so that it can deal with the facts that the planner's initial information about the world may be incomplete and as such it should be able to gather information during planning, as well as that some WSs may return the information that the planner needs with a delay or not at all. To do so, it begins by having an incomplete initial world state view and tries to gain relevant information during the planning process. Meanwhile, it explores alternative paths while waiting for information to be returned. In (Lin et al., 2008) the translation techniques used in SHOP2 to encode OWL-S to HTN are adopted, while the authors augment the OWL-S process model with qualitative user preferences. OWL-S service descriptions and preferences described in PDDL 3.0 are translated into an HTN formalism, and a WSC algorithm - SCUP - performs a best-first search over the possible task decompositions, having evaluated them heuristically based on ontological reasoning over the input preferences.

(Zúñiga et al., 2010) make use of AI planning in conjunction with WSMO; a translation between WSML (Web Service Modeling Language), the language providing the syntax and semantics for WSMO, and PDDL occurs before planning; afterwards, an evaluation of the resulting plans occurs so as to choose the best one among them. For the former function, a partial mapping between WSML and PDDL is proposed, whereas for the latter, the non-functional properties of the services comprising the plan are considered. The planner used is ND-SHOP2, which generates all the policies that solve the original problem, with each consisting of an ordered set of actions and the parameters associated with them. Apart from the (selected) best plan, though, the rest are not taken into consideration during plan execution, and if a failure occurs during its execution, a replanning module generates a new plan from scratch. A prototype utilizing this method was implemented modeling a semantic application domain, without any quantitative evaluation of the method.

(Zuñiga et al., 2014) present an improved method; they argue that most of the WSC methods in the recent literature do not take into account all of the following parameters: First, the fact that their architecture should be decoupled, in accordance to the design principles of WSs themselves. Second, the fact that most methods are not designed to allow and facilitate the use of WSs from different sources, as well as their use from non-expert users. Moreover, they argue that the specification of the goal request or the selection and composition of WSs usually requires expert technical knowledge, deterring non-experts users from utilizing them; finally, it is argued that WSC methods should be domain independent, and that AI planning algorithms should be used in WSC methods that deal with

multiple domains that have dissimilar functionalities. However, although (Zuñiga et al., 2014) acknowledge the need for non-deterministic elements in WSC in addition to the use of planning languages that support non-determinism, e.g., PPDDL (Younes & Littman, 2004), their method does not make use of such elements. The translation module receives the user goal as input, generated by using the WSMO goal specification, the ontology and WSs that are related to the chosen domain, and outputs a planning domain and a problem, along with its initial state. The AI planning module, which employs SHOP2 as the underlying planner, receives the translator's output and generates one or more plans. The plans are ordered lists of activities that, if executed successfully, achieve the goal.

3.3.4.2 Middle ground methods

A middle ground between straightforward deterministic methods and those that support non-determinism consists of methods that group similar WSs in their solutions and use them as alternatives. Another type of middle ground methods are those that generate multiple solutions to a problem, or even all the possible solutions to it, rank them according to one or more QoS criteria, and output a single optimal solution.

(Wagner et al., 2011), in contrast to (Zuñiga et al., 2014), argue that the basic problem of most recent WSC planning methods is that they only tackle the problem of flexibility, i.e., automated WSC, while also either trying to maximize the reliability of the produced plans, or achieve QoS goals, but not both. Their method comprises a planning module that generates workflows incorporating alternative backup WSs. The method first generates clusters containing similar WSs, thereby pruning the search space; this (offline) process is based on a direct-acyclic functionality graph to keep track of the alternative services. The functionality of each computed cluster can be represented by its root node, so only the root nodes of the computed clusters are used to produce workflows. Keikaku, a custom regression planner based on iterative deepening DFS, is used in order to generate the workflows, taking the QoS of each cluster in the graph into account. If only the reliability of the workflow is important to WSC, then any service that participates in the workflow can be substituted by another one in the same or in a child node. If additional QoS attributes, such as the WSs' costs, have to be considered, then the algorithm simply selects the services with the best QoS attributes. This choice is based on the assumption that in real world settings the WSs tend to be reliable and, as such, the combination of two WSs should be sufficiently reliable.

(Deng et al., 2014) present a QoS-based WSC method generating the top-k solutions of WSC problems through DFS. The original WSC problem is split into mutually independent smaller problems that can be solved concurrently. Then the best k solutions are returned in relation to some QoS criterion, with the rules in (Zeng et al., 2004) being used to compute the aggregate QoS for WSs either in a sequence execution path or in multiple parallel paths. The sole criterion used is the solution's response time; (Deng et al., 2014) argue, however, that the method can be extended to handle multi-dimensions QoS values, provided that an aggregating function of multiple QoS properties is available.

Finally, the composite solution is retuned, as a DAG. The first step in the method is to transform the original set of WSs into a rule repository, with each WS comprising a rule that can generate both the concepts originally in it and their ancestors. An inverted index of this repository is created, with the rules being indexed by the concepts that can be outputted from them. WSs are identified as either useful or irrelevant, by using an extension of the method in (Hennig & Balke, 2010). Based on this, the concepts that the user provides as being initially available create a set; then the WSs that can be executed by the concepts in the aforementioned set are identified and their outputs are added to this set. This procedure repeats until no more concepts can be added to the set. If the concepts in the output set are a superset of the ones in the user's goal request, then the problem is solvable and all the WSs that have contributed outputs to the set are identified as useful. In any other case, the goal request cannot be met with the specific set of WSs.

Afterwards, the mutually independent tasks created by the original user request are distributed to the various WSC agents who output solution sub-graphs for their particular problem using a backtracking algorithm for WSC based on DFS named BTSC-DFS. BTSC-DFS returns, for each concept in the request, the set of WSs that output it, from the inverted index of the rule repository previously created. Then, a subset of those WSs is chosen to backtrack for, and in case two or more of these WSs share the same inputs, they are considered as one common. For this reason, the output solution may contain sets of WSs instead of single ones. BTSC-DFS first searches for critical paths that output the specific concept; a critical path is defined as an ordered sequence of nodes, in which each node represents one WS or a set of common ones, such as for each pair of neighboring nodes (n_i, n_{i+1}), at least one input of n_{i+1} can be provided by n_i. This set of paths is then merged with non-critical ones, and all the solution sub-graphs are returned in order to constitute a single one. However, (Deng et al., 2014) note that if all WSs are to be considered and all the possible compositions for each merging solution sub-graph generated, then the computational cost may be very high. For this reason, only the WSs with the highest QoS are selected in each case. Moreover, during the merging process, if a solution sub-graph can generate a superset of concepts in comparison to other sub-graphs, then the rest are discarded instead of being merged with it. These optimizations, though, result in the loss of completeness of the method, as some correct compositions are not generated in exchange for improved efficiency.

(Wagner et al., 2012) propose a multi-objective optimization method that outputs multiple alternative solutions to WSC problems through the use of Hierarchical Workflow Graphs (HWG). The problem is first encoded as a genome and then solved through a genetic algorithm, with the QoS elements considered being the cost of WSs, the required time to output a response, and their reliability. A multi-objective optimization is utilized that partially orders the current selections based on the following domination notion: a WS x dominates another x' if all objective functions $f_i(x)$ produce at least the same value as $f_i(x')$ and is strictly better for at least one i. The problem is modeled as a HWG that contains complex service nodes that allow for the representation of alternative workflows schemes, each of which is a DAG. HGW also comprises user requests and atomic tasks, which are

implemented by one or more concrete WSs. The WSs, however, are grouped into sets of functionally equivalent ones that can be distinguished by their QoS properties.

The HGW is used to determine time and input-dependent QoS, whereas an evolutionary algorithm computes a Pareto-optimal set of non-dominated solutions, among which one solution is chosen based on the input QoS preferences. The decision variables in the workflows are encoded as a genome; there are three types of decision variables, namely, selecting a workflow scheme for each complex service; selecting a concrete service for each atomic task; and determining a starting time for each concrete service. Each genome is split into three parts, each corresponding to one decision variable. Then, the genomes are mutated, joined by a crossover operator and selected in each generation, using the non-dominated sorting genetic algorithm II (Deb et al., 2002), simulated binary crossover, uniform distribution mutation operators and binary-tournament selection. The method is evaluated using synthetic problems created by (Wagner et al., 2012), who concluded that a high crossover probability was necessary for the method to be efficient in the first few generations. Moreover, for this test set, optimal solutions could not be computed efficiently even for simple, small problems. Instead, an approximation of optimal solutions is generated. It should also be noted that, as in (Deng et al., 2014), the set of alternative, and in this case also non-dominated, solutions is outputted so that a single one can be chosen, and these are not used as a whole to tackle non-determinism.

(Rodriguez-Mier et al., 2011) first formulate the idea of using A* as a heuristic algorithm to search through a service dependency graph and generate an optimal composition in terms of the number of WSs comprising it. In (Rodríguez-Mier et al., 2012) an optimal and complete backward search method is used to generate multiple optimal solutions to WSC problems, without taking into consideration their non-functional properties. Specifically, based on the input request, an extended service dependency graph, which represents a suboptimal solution, is created dynamically using semantic input-output matching. Each layer of the graph contains all the WSs than can be executed with the outputs of the WSs in the previous one, with each being connected to the previous layer in sequential fashion. This solution is improved by removing services that are not used and combining the ones that have equivalent outputs, in order to produce a non-redundant graph. A backward heuristic search based on A* is then used to generate all optimal solutions that have a different number of services and runpath; the heuristic used is simply the layer in which the specific node is placed, i.e., its distance to the initial state. Finally, the generated solutions are optimized so that parallelism is maximized and the number of services in each one is minimized. The paths explored are reduced dynamically in two ways. First, offline, by replacing WSs that produce the same set of outputs by a representative WS, one that dominates the rest as to the outputs it produces. Second, online, during search, by detecting and combining nodes that can generate equivalent neighbors. In (Rodriguez-Mier et al., 2015) the method is paired with an existing service registry utilizing Linked Data principles and semantic reasoning along with various service discovery and matchmaking configurations. It is shown that unless several indexing optimizations are incorporated the system performs unacceptably; regardless, the time required for the discovery and matchmaking phases dominates that of WSC.

Finally, in (Cui et al., 2012) a deterministic WSC method is presented, which generates all feasible solutions and outputs a single optimal solution based on dynamic programming. All possible execution paths of service compositions are generated by a service graph that represents the WS workflows. Then the Viterbi algorithm (Viterbi, 1967) is used to select among the possible plans, with multiple QoS attributes, such as the WSs' price, response time and reliability. Similarly to the previously presented methods, in (Cui et al., 2012) despite having generated all possible solutions, all other solutions are discarded apart from the optimal one, assuming a deterministic environment. In that way, though, most of the computation time is wasted, when more than one solution could have been outputted, either as alternatives ones to be suggested to the user, or as part of a contingent plan, in case one or more solutions fail. The next section presents various methods that utilize such a methodology, along with other non-deterministic WSC methods.

3.3.4.3 Non-deterministic and contingent planning methods

(Meyer & Weske, 2006) proposed one of the first methods that took uncertainty into account during the WSC, assuming that it was present in the problem's initial state, as well as in the WSs' effects. For this reason, alternative control flow elements were incorporated in the composition process. The method's planning module is based on forward heuristic search in state space, particularly Enforced Hill-Climbing (EHC) (Hoffmann & Nebel, 2001). EHC, however, does not support non-determinism or the generation of alternative control flow, and the result of the planning process is sequential. As such, it was modified by extending its state transition function and the way states are handled. The Relaxed GraphPlan heuristic was used to guide the EHC search, solving a simplified version of the original problem using GraphPlan; (Meyer & Weske, 2006) use the length of this solution as an admissible heuristic that can be computed in polynomial time. Moreover, only the current fact layer is stored in memory in addition to the number of activity layers required to reach it, and not the activity layers themselves, as they are only needed in the case a solution is required.

(Hoffmann et al., 2007), as well as subsequent work in (Sirbu & Hoffmann, 2008) and (Hoffmann et al., 2009), present a method in relation to non-deterministic WSC utilizing AI planning that considers integrity constraints. WS application is considered as a belief update operation and concept subsumption relations are modeled through forward effects, in order to allow for the use of ontologies into AI planners (Kaldeli, 2013). Forward effects in WSC are assumed to be present when all the ramifications of a WS' application concern only propositions involving at least one new constant. When problems fall into the two special cases of WSC under uncertainty that are identified in (Hoffmann et al., 2009), they are tractable and allow for a compilation of the original WSC problem into conformant planning. Subsequently, in these cases, an existing conformant planner is used, namely Conformant-FF (Hoffmann & Brafman, 2006), which is modified to consider on-the-fly output constants.

(Kona et al., 2008) present a WSC algorithm with the ability to generate either a linear chain of services or a directed acyclic graph, which may also be conditional. That is, the composition may be

sequential, non-sequential, or non-sequential conditional; in the latter case, the flow of the resulting composition depends upon the result of the WS' postconditions. The approach is a stratified one, with filtering of WSs occurring at multiple stages of the WSC process. Each stage comprises all the WSs that can be executed using the initial state's input parameters and all the outputs produced in the previous stage, until all the output parameters of the goal state are reached. Then a final filtering process occurs, this time in a backwards fashion, in order to remove redundant services, i.e., ones that are not useful in achieving the goals. The approach was implemented utilizing Prolog (Clocksin & Mellish, 2003) along with Constraint Logic Programming (Apt, 2003).

(Kuzu & Cicekli, 2012) present a conversion schema from OWL-S to PDDL based on (Klusch et al., 2005) and (Kim & Kim, 2006). The presented methodology takes into account the non-determinism in the domain and makes use of a modification of an existing PDDL planner, namely SimPlanner (Onaindia et al., 2001), to tackle it at a high level, through interleaving planning and execution and utilizing SimPlanner's replanning functionality. In case the high level approach does not succeed, web service transaction frameworks, such as WS-Coordination (OASIS, 2009) and WS-BusinessActivity (OASIS, 2007), are also utilized to recover from failure situations and prevent the undesired results of aborted WSC attempts.

(Dacosta et al., 2004) present a method that produces robust plans through the generation of contingency plans, allowing for OWL-S WSs that achieve the same (semantic) tasks. Users are required to input the initial concepts that are available, as well as the goal, in addition to a high level description of the requested workflow comprising the set of activities that have to be executed. In specific, the user inputs activities that are implemented by specific WSs' operations. The notions of useful and redundant operations are defined in (Dacosta et al., 2004), use a different method to distinguish between the two operations than in (Deng et al., 2014); if a WS can generate an output or effect that is part of the user's request, implements an activity in it, or is the input or precondition of another useful operation, then it is considered useful. Then, for each node that represents an activity, only useful WSs' operations can be selected, thus effectively restricting the search space. Redundant operations are identified by creating a graph in which if one operation generates an output that can be used as an input for another one, a transition is created between the two. If all the transitions that leave an operation op_a end at an operation op_b that already has an identical transition incoming from a different operation, e.g., op_c, then op_a is redundant and can safely be discarded.

The method's output is a graph containing control flow constructs and calls to specific WSs' operations, with each path in it being a possible execution path, and each child node comprising an alternative execution possibility. This graph contains all the possible contingency plans; to do so, a stratified method is utilized. First, the user inputs are added to the appropriate set and then several rules are applied continuously, until no rule can be applied anymore, or a plan is feasible. This step restricts the number of possible execution paths that the planner has to search among, and outputs the useful operations' set if a plan is possible. Then, the operations that generate each output, effect and activity populate one vector each, and the sets of vectors are used to output all the possible paths to

the goal; that is, all paths that produce every output of the original request are generated, comprising one element of each vector. Next, redundant operations are removed from the paths, the paths are reordered in relation to their preconditions and input dependencies, and the set of paths is ordered from the best – the one containing the smaller number of services – to the worst. The resulting graph, containing all the possible execution paths, can only fail if an operation fails and there is no other brother or uncle of the node that can be executed. This process is similar to the one in (Rodríguez-Mier et al., 2012). (Dacosta et al., 2004) describe only a prototype as well as an example of the method on a test problem. Thus, the efficiency and time complexity of elements of it, e.g., the combination of all different vectors for each output, effect and activity, or the repeated generation of the redundant operations' graph, are not evaluated.

The method in (Zou et al., 2012) belongs in the category of methods that determinize the original problem and then use a deterministic planner to solve it, in a manner similar to the planning methods presented in Section 2.3. In this case, FF and SatPlan06 (Kautz et al., 2006) are employed. The WSC problem receives as input explicit user defined contingencies, comprises WSDL WSs and does not take into account ontologies. Importantly, its goal is to generate plans for multiple participants that collaborate with each other to achieve a common goal, and supports asynchronous operations of WSs.

Finally, (Wang et al., 2015) present a WSC method that takes uncertainty in the actions' effects into account, through a modification of Graphplan. The method allows for operators with multiple groups of effects as well as branches in the output solutions. Every non-deterministic operator having k groups of different effects, generates k branching actions after its parameters' instantiation, with the branching actions in the same action level that were generated from the same operator being mutually exclusive.

3.3.4.4 Evaluation of web service composition systems

A plethora of reviews of WSC methods exist, with different scope, categorization and focus; the majority of these surveys classify methods based on the AI planning techniques used, (Chan et al., 2007), (Peer, 2005), (Küster et al., 2005), (Sirin, 2004), (Khadka & Sapkota, 2010), (D'Mello et al., 2011), (Oh et al., 2006) and (Digiampietri et al., 2007) being prominent examples of such ones. Older surveys, e.g., (Srivastava & Koehler, 2003) and (Rao & Su, 2003) focused less on AI planning methodologies, whereas most recent focus almost exclusively on such methods. There are, however, a few recent surveys that also review other methods, such as workflow ones, e.g., (Alamri et al., 2006), (Kapitsaki et al., 2007), (Eid et al., 2008) and (Tabatabaei et al., 2008). Moreover, lately, a variety of surveys review WSC methods that support non-determinism as part of a wider categorization, such as in (Küster et al., 2005), (Baryannis & Plexousakis, 2010), (Sheng et al., 2014).

In (Chan et al., 2007), a comparison of planning techniques for WSC is conducted, with the evaluation criteria being based only on qualitative criteria. Some of the criteria that the authors take into consideration are: whether the technique is domain independent, as well as whether it supports

partial observability and non-determinism or not; the standards to which it can be applied to, i.e., its applicability, and its support of concurrency in the execution of WSs. The scalability of the approach is also evaluated, but without any mention to quantitative results; the authors simply evaluate it based on a critique of the algorithm used. In (Feenstra et al., 2007), WSC approaches are criticized for making use of easy to measure criteria, without incorporating more important requirements in their evaluation. Several testing challenges in relation to the effective evaluation of service-centric systems are reported in (Canfora & Di Penta, 2006). Among these challenges are the dynamicity and adaptiveness that is inherent in the output workflows that contain abstract services, due to the fact that they can be automatically bound to various concrete services during the execution of the workflow instances. Other reported challenges are the lack of control that is attributed to a WS being modified during its lifecycle, and the cost of testing that is related to invoking the actual WSs. In (Bozkurt et al., 2010), it is concluded that WS testing is more challenging than testing traditional systems; these findings are consistent with (Feenstra et al., 2007), that is, it is the authors' opinion that the difficulty in testing WSs is partly due to the dynamic nature of WSs and the limited control over them, as well as not having access to their source code.

Several approaches for WSC are evaluated in (Bartalos & Bieliková, 2011); among those not referenced previously, three out of eleven do not provide any evaluation at all. The rest, e.g., (Alrifai et al., 2008), present quantitative experimental results, such as the time needed to achieve a solution; however, none refers to actual scenarios with specific goals that exhibit the system's functionalities. For example, in (Shin & Lee, 2007), another one of the approaches evaluated in (Bartalos & Bieliková, 2011), randomly generated problems are used, with the authors only providing information regarding the number of services and ontology concepts present in them. (Rosenberg et al., 2010) present quantitative experimental results, along with details in relation to the machine that was used to run the experiments and the number of WSs taking part in the tests. A non-standardized language is used to implement the composition model, namely VCL (Vienna Composition Language), without any mention to the structure or the goals of the test problems, so as to allow their replication and comparison to other systems, or to testify that they are non-trivial. Of the approaches evaluated in (Bartalos & Bieliková, 2011), only (Jiang et al., 2010) denotes the test set that was used for evaluation purposes, the 2009 Web Service Challenge dataset (Kona et al., 2009). The actual benchmark domains used in (Li et al., 2008) are specified, consisting of two of the problem files used in (Sirin et al., 2004).

There is a plethora of WSC methods that either evaluate the proposed methodology on case studies without referring to quantitative criteria or not at all. For example, in (Zúñiga et al., 2010) the evaluation of the method is based on a single case study, with no quantitative experimental results being available. (Dacosta et al., 2004) mention an implemented prototype, similarly to (Kuzu & Cicekli, 2012) and (Chen et al., 2009) who implemented a single case study without referring to any quantitative criteria; (Bo & Zheng, 2009) do not provide any kind of evaluation at all.

Fortunately, in recently there has been an increase in WSC methods that provide some kind of quantitative evaluation of their approach; OWLS-XPlan is evaluated using the benchmarks of the 2003

International Planning Competition (IPC3) (Long & Fox, 2003). Specifically, all test cases of the IPC3 test scenarios were used, with a total of 122 problems being tested, separated by different complexity classes, with an increasing number of objects. Its performance is then compared to that of the top four performing participants of that year's competition (FF, SimPlanner, TLPlan (Bacchus & Kabanza, 2000), and SHOP2). The evaluation is focused on the planning capabilities of OWLS-XPlan and not its WSC ones; the results, however, provide a sense of its performance, and especially in contrast to SHOP2, another successful WSC approach. In regard to the average plan quality, OWLS-XPlan had the best performance among all the tested planners. SHOP2 is almost consistently worse, but for the more complex classes it presents a significantly better performance than all the other planners. In regards to the solved problems of the benchmark, as before, SHOP2 does not perform well in problems of low and mid-range complexity, but it has a top performance in the more complex ones. This is also true for the other HTN planner, TLPlan, while OWLS-XPlan solved nearly all the given problem cases. Finally, the experimental results as to the average plan length in relation to the complexity of the problem definition are consistent with the previous ones.

(Hatzi et al., 2011) evaluate their proposed WSC approach by performing experiments using the SemWebCentral OWL-S Test Collection (TC) (SemWebCentral, 2010) version 2.2 revision 1. The experiments performed without semantic relaxation are fairly similar to those performed with semantic relaxation using the edge-counting distance metric, with both approaches having a linear complexity for the transformation process and converging to an average transformation time per WS profile of less than a second. Semantic relaxation using the upwards cotopic metric seems to be more slow. Although the results in regard to the planners are not crucial, as in theory their use is not restrictive, the experiments were definitive as far as to the use of the planners, with LPG-td being several orders of magnitude faster than JPlan in every case.

The evaluation of WSPR is based on two test sets, namely EEE05[9] and ICEBE05[10], the former being manually created by human experts and comprising fewer WSs than the latter. The evaluation criteria in regard to both test sets are the planner's total running time and the number of WSs in the plans. As the method is targeted for WSDL services and UDDI registries, both test sets only use WSDL files; as such, only syntactic matching is assumed, i.e., if two parameters are textually equivalent, they are matched. There are three dimensions that control the algorithm's performance: the number of WSs in a solution, the total number of available WSs and the total number of parameters (from which the initial input parameters and the desired output parameters are generated). The experiments indicated that the performance of WSPR is mainly dependent on the efficiency of the forward search, as it has much worse computation time than the regression one, since in that stage the planner has to visit more states. Although the total planning time did not exceed 30ms in any problem, (Oh et al., 2007) suggest that the planner's search in the parameter space should be more efficient.

[9] Available at http://www.comp.hkbu.edu.hk/~eee05 /contest/ Accessed on March 11, 2015
[10] Available at http://www.comp.hkbu.edu.hk/~ctr/ wschallenge Accessed on March 11, 2015

The method in (Huang et al., 2009) was evaluated experimentally using problems from the Web Service Challenge 2009. The average search time of the method over a set of five test problems is used as the evaluation metric, with the number of WSs in the experiments ranging from 6000 to 12000 and the ontological concepts comprising them from 12000 to 24000 respectively. However, little information is provided by the authors concerning the experiments; the specifics of the test set problems are not presented. The authors mention that the number of services does not affect the algorithm's efficiency heavily, with the results exhibiting that there is linear correlation between the two. Specifically, between test sets comprising 4000 and 12000 services the planning time increases almost three times as much, from 100ms to almost 300ms.

AWSP was evaluated using the WSBen benchmark (Oh et al., 2006), with the running time to find the solution, the number of WSs in it and the states that were expanded during search being the evaluation metrics. Based on this evaluation, the backwards search appears to be substantially better than the forward one, as the state space is much larger in the latter. Additionally, the authors favor their own algorithm instead of A*, as in their experiments, particularly in smaller problems, it generates solutions faster, and with a quality similar to A*; due to the use of an admissible heuristic, though, A* always outputs optimal solutions, whereas their own heuristic state space search does not. However, the heuristic function used cannot take into account the QoS properties of WSs, and the algorithm cannot manage either the parallel execution of WSs or the generation of alternative workflows. (Rodríguez-Mier et al., 2012) argue that since (Wu et al., 2011) do not utilize stratified methods, the search space size cannot quickly reduce. (Wu et al., 2011) on the other hand, argue that although stratified methods are efficient, they can only tackle WSC problems in which the WSs are only information-providing and not world altering.

(Kuter et al., 2005) test the ENQUIRER algorithm in two planning domains, with the first having 100 randomly-generated problem instances, with multiple runs for each problem, each of which incorporates different amount of information available about the initial state. For the second domain, three variations of the original algorithm are evaluated based on how the planner maintains its OPEN list. The criteria used are the way the solutions are affected by the amount of the information available during planning, and the desirability of the generated plans, i.e., to what degree the solution satisfies the user's preferences, in terms of the ordering among the methods applicable to the same task.

Two planning domains from the "Planning with Qualitative Preferences" track of the 2006 International Planning Competition (IPC-5)[11] are used for the evaluation in (Lin et al., 2008), with the method's performance being compared to that of the planner that won the relevant track of the IPC-5, SGPlan, as to the overall violation cost values of the solutions generated by them on both domains. In both domains, SCUP outperformed SGPlan in almost all of the problems.

(Wagner et al., 2011) compare their proposed method against an extension of QSynth (Jiang et al., 2010) that they re-implemented. Their evaluation is based on a random problem generator

[11] http://ldc.usb.ve/~bonet/ipc5/

comprised of OWL-S WSs, containing a number of variations between 2 and 7, depending on the test set, with every set containing a total of 1000 WSs. The method compares favorably against QSynth and its extension in most cases, with a gain of more than 5% in utility. However, in case where there are few variations of the services in the registry, the method is less efficient, generating longer workflows with low reliability.

The comprehensive experimental evaluation of the method in (Deng et al., 2014) considers two problem datasets, from the China Web Service Cup (CWSC2011)[12] and the Web Service Challenge 2009. The evaluation comprises tests with concept numbers ranging from 1,000 to 20,000 WSs, and solution depths up to 10 WSs. The authors conclude that the method's performance does not change considerably when the solution depth increases, due to the fact that only one possible service is selected for backtracking and the method is based on DFS. When compared with methods that generate a single solution, such as the winner of the Web Service Challenge 2009, the method is in general more costly. However, the approach in (Deng et al., 2014) also generates alternative solutions and not a single one. As to the optimizations, the evaluation indicates that they improve the method's efficiency by reducing the solutions' cost by approximately half for each dataset, at the expense of its accuracy.

The method in (Rodriguez-Mier et al., 2011) was experimentally evaluated using eight datasets from the Web Service Challenge 2008 (Bansal et al., 2008), comprising 58 to 8,119 WSDL services. The algorithm was able to find all the optimal solutions in all cases, with the optimizations improving the method's efficiency substantially. On other hand, the method suffers from several disadvantages; first, it only incorporates two control structures in its compositions, sequences and splits, i.e., parallel execution. As such, control structures that support alternative flows cannot be used. Moreover, a pre-computed table mapping inputs to the WSs that require them to be executed is used to generate the dependency graph so as to obtain the necessary WSs for each layer more efficiently. For this reason, each time the description or the functionalities of a service are modified, the dependency graph has to be recomputed, from the service's layer up to the last one, thereby hindering the dynamic adaptability of the method.

The method in (Cui et al., 2012) is evaluated using five test sets, with no information provided about their size, the number of concepts comprising the WSs or any other detail. Its performance is evaluated against an optimal solution selected by a human expert and the average value of the feasible solutions computed earlier. Moreover, the satisfaction generated from the solution in comparison to the problem's complexity, i.e., the number of WSs comprising its solution, is also computed.

The planning method in (Meyer & Weske, 2006) is evaluated against Conformant-FF and Contingent-FF; however, it was significantly slower than either of them, with the authors suggesting that the method's inefficiency is due to the larger search space created by the search for possible parallel invocations and its non-optimized representation of states.

[12] http://debs.ict.ac.cn/cwsc2011/

The method in (Hoffmann et al., 2009) is evaluated using two artificial test scenarios; two versions of the modified Conformant-FF planner, with and without the EHC search algorithm and the helpful actions pruning optimizations, are tested against the DLVK tool (Eiter et al., 2003). Although DVLK is not in itself a WSC tool, it is compatible for evaluation against the method in (Hoffmann et al., 2009), as it can handle ontology axioms directly. Two different encodings of the test problems are tested, with the total runtime, number of search states expanded and the length of the plans being the evaluation metrics. Even with a bound equal to the optimal plan length, DLVK was significantly slower than Conformant-FF, and solved considerably less problem instances.

(Kona et al., 2008) utilize a single use case scenario, with three variations depending on whether the produced workflows are sequential, non-sequential, or conditional non-sequential, along with the Inputs/Outputs/Preconditions/Effects (IOPEs) of the services that take part in the WSC process. The criteria used in the evaluation of the system are quantitative, i.e., the number of WSs participating in the problem, the number of I/O parameters each WS had, and the preprocessing and query execution time needed to obtain a solution. The WSs that take part in the composition comprise a customized version of the 2006 Web Service Challenge (Blake et al., 2006) test collection.

(Zou et al., 2012) transform the original WSC choreography problem into a planning one, solve it using one of the aforementioned planners and build a dependency graph based on it; the graph is then used to create a master plan that is projected to a distributed one. This method is compared to WSPR based on 18 test sets comprising 81,464 WSs from the ICEBE05 WSC challenge. WSPR is shown to be slower than the proposed method, despite solving simpler problems, without contingencies or communication between WSs, mainly due to the efficiency and optimizations of the existing deterministic planners that are used.

Finally, in (Wang et al., 2015) the method was evaluated using the ICEBE05 test set; in the experiments that did not include uncertainty, the algorithm's performance depended on the number of the web services comprising the problem, their number of parameters and the problem's solution length. In the problems that incorporated uncertainty, by modifying the ICEBE05 problems by adding preconditions and multiple effects, the evaluation demonstrated that - as expected - there is a difference in the planning time required to reach a solution with and without uncertainty; the problems with uncertainty usually require approximately double the time than the respective deterministic ones. The method's performance relies on the number and location of the non-deterministic actions in the outputs solution and their number of uncertain effects

3.3.5 Discussion

The review of the current literature in Section 3.3.4 reveals that there is no definitive answer to the problem of automated WSC. The main reasons are that a standard benchmark for WSC does not exist, as well as that the problem of WSC is not defined in a similar way in each approach. However, it is important to note that the (vast) majority of approaches using AI planning to perform WSC,

especially those using one of the already available standards and not an internal representation of their own, use OWL-S to define the WSs that take part in the process (Chan et al., 2007). On the other hand, almost all approaches that make use of a planning language, either as a final problem description language or as an intermediate one, use PDDL as their formalization of choice.

The presented approaches differ in several elements concerning the properties of the domains in which they act, as well as in regard to the characteristics of the WSC algorithms they use. Some of the properties that characterize the various approaches as they have been identified from the review are the following:

- The applicability of the methodology, that is, the extent to which the current de facto standards in relation to WSs and their semantic content are supported, such as WSDL or OWL-S.
- Whether partial observability of the domain is supported.
- Whether the WSs and the domain is considered to be of non-deterministic nature. An action in a planning domain should model the possibility that if it is executed in a state it may lead to several different states, so as to represent more realistically the fact that real world WSs will not always act in the way they are supposed to, e.g., by failing to execute correctly, being unavailable, etc.
- Whether the approach in question takes into consideration additional elements of WSs, that is non-functional properties such as QoS characteristics, e.g., their response time, operational availability, cost of use, message encryption abilities, or their reputation based on user feedback.
- The implementation point of view. Some authors argue that WSC should consider WSs as stateful processes that perform complex multi-phase interaction protocols with each other. Others base their approaches on the assumption that WSs are simply atomic components, which, given some inputs, return some outputs, in a single request–response step. Moreover, those who support the former alternative usually consider WSs as having asynchronous interactions with other services, i.e., as processes that can evolve independently and interact with other WSs only through asynchronous message exchanges.
- Whether an approach supports the concurrent execution of more than one WS, or if it only allows a single WS to be executed at any given time.
- Whether an approach is domain independent, or it relies on domain-specific knowledge and heuristics in order to complete the WSC task successfully.
- The performance of the planners used and their scalability.

The relative immaturity of the WSC research field in combination with the diversity of (semantic) WS standards used and the variety of methods in terms of their scope, restrict the direct comparison between them and the use of a standard test set; a direct comparison between the various approaches

is not straightforward. As (Hoffmann et al., 2009) note "*A comparison to alternative WSC tools is problematic due to the widely disparate nature of what kinds of problems these tools can solve, what kinds of input languages they understand, and what purpose the respective developers had in mind*". This is the reason why the majority of the approaches presented in Section 3.3.4 are either evaluated against planning approaches, e.g., in the cases of OWLS-XPlan, (Meyer & Weske, 2006) or (Hoffmann et al., 2009), or utilize WS evaluation datasets, such as EEE05, ICEBE05, or the Web Service Challenge 2009. The only notable exceptions in the above survey are (Zou et al., 2012) who compare their approach to WSPR and (Wagner et al., 2011), who re-implemented a version of QSynth and evaluated their approach against it.

The approaches that utilize AI planning techniques are among the most utilized in the literature; the reason behind their popularity lies in the fact that already implemented planning systems and algorithms can be used, without having to implement new ones, and with the ability to interchange various planners, the only restriction being their compatibility with the planning formalization used. Since this formalization is almost always PDDL, and most planners conform to some subset of it, a very large number of planning systems is immediately available for use. Furthermore, the algorithms used have already been reviewed extensively in the literature, and as such their weaknesses and advantages have been identified, as well as their theoretical properties. Another advantage of such methods is the fact that most of the aforementioned characteristics, such as the applicability of the methodology, whether or not WSs can be executed concurrently, the domain dependency of the planner, or the scalability of the approach, do not depend on the approach itself but on the choice of planning systems. As this choice can be among a plethora of planning systems, the problems in regard to these elements are lessened to a great extent.

As far as the translation from a WSC problem to a planning one is concerned, the idea is very intuitive, since WS approaches such as WSMO and OWL-S have been influenced heavily from planning languages such as PDDL. For that reason, such a mapping – or at least some parts of it - is relatively natural and straightforward. However, there are elements of this translation for which a satisfying mapping has not been proposed; in particular, while the transcription of types and properties to PDDL predicates is relatively simple, there are problems, such as the one in (Bo and Zheng 2009) who do not consider a translation for OWL-S's Repeat-While, Repeat-Until, Any-Order and Split + Join control constructs, as there is no composite action in PDDL corresponding to them.

Additional problems arise, in regard first to the world state during planning time, and second to the differences between OWL-S and PDDL operators. As far as the first is concerned, most approaches adopt the Closed-World Assumption, without considering changes in the world state during planning time, while OWL adopts an Open-World Assumption. Such a difference can lead to incomplete mappings. Furthermore, while OWL-S supports inputs, outputs, preconditions and effects, PDDL only comprises preconditions and effects. For this reason, it cannot represent non-physical knowledge, and an indirect mapping has to be implemented, e.g., as in (Klusch et al., 2005) which introduces a new predicate into the domain, with a variable that can either be bound to an input or an output parameter.

Finally, the incorporation of non-determinism in WSC still remains an open problem; however, the advances in the non-deterministic planning research field have allowed for the consequent evolution of analogous WSC methods. Although in the past the vast majority of methods concerning the generation of composite WSs were focused on using classical planners in conjunction with a deterministic problem formulation, the literature review indicates that there has been a change in recent years. Some methods utilize the determinization schemes proposed by FF-Replan and its planning successors, while others make direct use of non-deterministic planners, such as Conformant-FF on non-deterministic problems formulations

Chapter 4.

Requirements and Use Case Scenarios

This chapter presents the requirements analysis that preceded the development of the **MADSWAN** system. Furthermore, the chapter introduces a set of use case scenarios for WSC systems, as well as a set of mockups that correspond to the implementation of a subset of this scenarios, in order to have a clear aim of what the GUI of the final system should look like.

4.1 System Requirements

Based on the extensive literature review of the previous chapters we propose six sets of requirements that can be used as a guidance for the design and implementation of a WSC system. Sections 4.1.1-4.1.6 refer to its internal structure, in regard to the composition elements, control and data flow and its data model, and to the goals that we aim to achieve through its use.

4.1.1 General Requirements

1) **The end user provides the goal of the composition (mandatory)**: The WSC process initiates with the user stating the goals that must be achieved. Beforehand, he must have already provided the initial state of the system, that is, the current state before the WSC process had started. Finally, he should be able to provide any necessary data that may be required in the initial state by the relevant WSs.
2) **The WSC process should fulfill the user's request goal (mandatory)**: The WSC should act in accordance to the user's request, that is, achieve the stated goals from the initial state, creating the composite WS using the provided request data, if that is possible.
3) **WSC can be either fully automated or manual (mandatory)**: The end user should have the option to use the web-based registry to navigate through the available SWSs and choose among them those that will participate in the final WSC; then, he should be able to orchestrate the WSs' input and outputs so that the goals are achieved, without having unsupported inputs. The system should notify him of any flaws in this process. Alternatively, he can utilize a fully automated WSC process, which does not require from him to select the necessary WSs that achieve the goals and place them in the correct order.
4) **The final system should utilize existing open source components (optional)**: In accordance with the motivation behind WSC, i.e., reusability of its components, this thesis proposes for WSC systems to use already established, open source elements, in order to promote re-usability and facilitate evaluation.
5) **The final system should be platform independent (optional):** The application's interface should be web-based, this being accessible through a variety of operating systems and web browsers.
6) **The final system should require an intermediate user skill level (optional):** The system's usability orientation will be towards users that have at least a basic understanding of the WS

philosophy and the technologies that govern it. As such, users are expected to understand concepts such as WSs' inputs and outputs, as well as ontologies. However, they are not expected to be experts in the WS research area or understand the specifics behind the WSC process.

4.1.2 Composition Elements Requirements

1) **The functionality of WSs is described semantically (mandatory)**: In order to automate the WSC process, it is needed to describe the functionality of the available WSs semantically.
2) **Services have IOPES (mandatory)**: A system supporting semantic WSs must provide the means for the definition of input, output, precondition and effects of the parameterization of the WSs' functionality. Generally, a WS can have zero or more input and output parameters; before its execution all of its preconditions must hold, and after it all of its effects have occurred, if it is a deterministic one, or a subset of them, if it is a non-deterministic one.
3) **The composition comprises services and their interaction (mandatory)**: The WSC process is based on the transformation of the WSs' semantic IOPEs into a planning domain and problem, and the solution of that problem. The solution should comprise a set of web services, atomic or composite, that achieve the required goal, along with the OWL-S control constructs that control their interactions.
4) **The output of the composition is a SWS itself (optional)**: The result of the WSC process is a PDDL plan that achieves the required goals. However, the system should convert the PDDL plan back to OWL-S, that is create an OWL-S profile and its process description. The creation of grounding instances and WSDL definitions for the new WS will not be mandatory. In short, the profile description of the new composite WS will treat it as an atomic service with IOPEs, while the process model will be based on OWL-S control constructs to analyze the way the services interact with each other. The exact data flow between them, that is which output is provided as input to the next WS, will not be necessarily defined.
5) **Newly created composite SWSs can be part of the composition (optional):** The purpose of automated WSC is to improve the reusability of already existing processes that provide a certain functionality. As such, it is purposeful to store the results of past WSC processes as new semantic descriptions of composite WSs into the web-based registry, so that they can be used later as part of new WSC processes.
6) **SWSs are described using OWL-S (optional)**: As a direct result of 1), 2) and 4), a formalism to semantically describe the SWSs and their IOPEs should be selected. The standard of choice is OWL-S, for reasons that have been analyzed in previous chapters. A semantic WSC system should fully support the upload, download, removal and editing of the 1.1 version of OWL-S
7) **The test collection used to implement the use case scenarios is the OWL-S Test Collection (optional):** The largest collection of OWL-S semantic descriptions of WSs at the time of pursuing

this thesis was OWL-S TC. Its features and the advantages that come from its use will be reviewed in Section 4.2.2.

4.1.3 Control Flow Requirements

1) **Use of OWL-S control constructs (mandatory)**: The new composite WS should be decomposable into other (atomic or composite) WSs based on a subset of the following OWL-S control constructs: Sequence, Split, Split+Join, Any-Order, If-Then-Else, Choice, Iterate Repeat-While and Repeat-Until. It should be noted that these control constructs do not dictate a behavior that will be implemented by the composite WS they refer to, but a behavior that can be achieved by the clients that interact with it by sending and receiving messages. As such, the effect that it has will only be achieved if the entire process is executed successfully.

4.1.4 Data Flow Requirements

1) **WSs have data interactions (mandatory)**: The WSs that form the composite one can exchange data with each other. Specifically, they can receive data as input either from the initial data that the user provides or from the output data from other WSs. Note that data exchange between WSs within a composite one implies ordering constraints between them.

4.1.5 Data Model Requirements

1) **Data elements are typed (mandatory)**: The semantic description of WSs ensures that the data that is exchanged between them is typed. This fact ensures that all the data exchanges are valid, and eases the WSC process as the planner does not consider for composition WSs that do not output or receive as input the right type of parameters.
2) **An ontology is required to define the type of data element types (mandatory)**: To describe the semantics of input/output concepts of WSs various ontologies can be used. These ontologies are imported in the semantic description of each WS and define the type of the data elements, as well as their relation to each other (such as their super-classes and sub-classes).
3) **The language used to define ontologies that indicate the data element types is OWL (mandatory)**: As a direct result of 1) and 2), a formalism to define the relative ontologies should be selected. Since OWL-S is the semantic WS approach selected to describe the WSs available in the registry, and it is an OWL ontology itself, the ontology language should also be OWL.
4) **The planner realizes the hierarchy defined by the ontologies (optional)**: As a result of 1), it would be beneficial for a WSC planner to be aware of the data element structure, that is, to be aware whether an ontology concept is a sub-class of another, and as such can be used to provide an output of the desired type. Note that PDDL supports typed objects.

4.1.6 System Goals

1) **Capability of dealing with the domain's uncertainty (mandatory)**: In order to create a realistic WSC system, one must take into consideration the non-determinism that is inherent in the WSC problems. Firstly, WSs are processes that may fail to execute correctly for various reasons, such as network congestion, or service invocation errors. More importantly, the result of a WS' execution may not be always known a priori. For this reason, we opt for the incorporation of a non-deterministic planning algorithm in the WSC process.
2) **Automated WSC should be supported, besides its manual counterpart (mandatory):** Automated WSC is generally preferred compared to manual WSC, as it can handle larger domains involving more WSs and can be orders of magnitude faster. As such, automated WSC should be able to tackle use case scenarios that comprise a significant number of WSs in the domain.
3) **Quantitative performance measurement (mandatory)**: The quantitative criteria that will be considered for the evaluation of the proposed system should include some of the following:
    - Number of WSs considered for WSC
    - Preprocessing time (parsing of ontologies' concepts, etc)
    - Transformation time (of a WS domain to PDDL one)
    - Planning time (to output a valid plan that achieves the goal)
    - Optimality of the outputted plans (in terms of cost and expected utility)
    - Contingent or deterministic plan generation

This section presented the requirements that drive the implementation of a new prototype WSC system, taking also into account their non-deterministic nature. To test these requirements, several use case scenarios are introduced, with various amounts of non-determinism and complexity each. Table 4.1 presents all the discussed requirements synoptically.

Table 4.1: Summary of the system's requirements

General Composition Requirements		
General Requirements		
1	The end user provides the goal of the composition	Mandatory
2	The WSC process should fulfill the user's request goal	Mandatory
3	WSC can be either fully automated or manual	Mandatory
4	The final system should utilize existing open source components	Optional
5	The final system should be platform independent	Optional
6	The final system should require an intermediate user skill level	Optional
Composition Elements Requirements		
1	The functionality of WSs is described semantically	Mandatory
2	Services have IOPES	Mandatory
3	The composition comprises services and their interaction	Mandatory
4	The output of the composition is a SWS itself	Optional
5	Newly created composite SWSs can be part of the composition	Optional
6	SWSs are described using OWL-S	Optional
7	The test collection used to implement the use case scenarios is OWL-S TC4	Optional
Control Flow Requirements		
1	Use of OWL-S control constructs	Mandatory
Data Flow Requirements		
1	WSs have data interactions	Mandatory
Data Model Requirements		
1	Data elements are typed	Mandatory
2	An ontology is used to define the type of data element types	Mandatory
3	The language used to define ontologies that indicate the data element types is OWL	Mandatory
4	The planner realizes the hierarchy defined by the ontologies	Optional
System Goals		
1	Capability of dealing with the domain's uncertainty	Mandatory
2	Automated WSC should be supported, besides its manual counterpart	Mandatory
3	Quantitative performance measurement	Mandatory

4.2 Use Case Scenarios

As shown in Sections 3.3.3 and 3.3.4, no standard WS testbed or test collection exists; in Section 3.3.5 we discussed the reasons why it is currently very hard to evaluate a WSC approach objectively against another one.

A well-known and used example of WSC use case scenario in the relevant literature is that of an online travel agent ((Zaremba et al., 2006), (McIlraith et al., 2001), (Srivastava & Koehler, 2003), (Curbera et al., 2002), (Menasce, 2002)), which presumes that separate WSs exist, each (on its own) capable of booking airplane tickets, hotels and renting cars, but a service requester desires to book an entire trip that includes booking a flight to travel to a specific destination, booking a hotel for the duration of the visit, and renting a car for the same duration. The problem of the composition is to discover the necessary WSs that will form the composite WS, along with its accompanying control flow. The frequent use of this scenario, as well as the fact that it depicts a real-world situation, could constitute it as a candidate for use as a benchmark in the domain of WSC. However, virtually every reference to the online travel agency scenario in the bibliography describes a modified version of the one in (Haas, 2002).

One of the goals of this thesis is to create a set of use case scenarios that could be used and reproduced by other WSC systems, that is, clearly define detailed ones that are based on an existing open test collection. The thesis adopts the OWL-S TC, since in the past few years it has been used extensively, as a test set in the recent S3 contests (Klusch, 2011), or in several approaches in the recent literature (Klusch et al., 2005) (Canfora & Di Penta, 2006) (Mesmoudi et al., 2011). We have designed five use case problems; with one exception, they are based on the service descriptions contained in a single domain of OWL-S TC, although several modifications were made to the descriptions of some services, and a few new descriptions were added to some domains in order to design more useful scenarios. All the modifications to the original test collection, as well as the full set of WSs that comprise them can be found in Appendix A.

This section first presents some of the use case scenarios that have been used in the recent literature, followed by a brief discussion of OWL-S TC. Finally, it introduces the use case scenarios that exhibit the proposed system's capabilities and describes in detail the necessary SWSs used in the scenarios and their inputs/outputs, with the ontologies' concepts used in each one shown in parentheses (following the format "*ontology#concept*"). This set of scenarios provides use cases that can efficiently evaluate the capabilities of WSC methodologies in a way that is both reproducible and extensible. They allow for a system to showcase that it can indeed cope with non-determinism in the WSC domain, and output both sequential and conditional plans. The correctness of the automated WSC solution plan for a problem is automatically validated. It is worth noting that in the evaluation of the various modules of the proposed system, different subsets of the use case scenarios that are presented is utilized.

4.2.1 Use Case Scenarios in the Literature

This section presents the use case scenarios used in the recent literature; scenarios that involve travelling arrangements, e.g., transport and accommodation, are among the most popular in the literature, as demonstrated by the use of the online travel agent use case. The following approaches all use a version of that scenario, although each describes the motivating example in a different way. (Chen et al., 2009) describe an e-travel scenario that requires that a user should first reach a travel destination and then attend an exhibition there. For this purpose more than one plan may be available to achieve the goal, and they are evaluated based on their expected utility. In this work it is not mentioned whether the WSs used actually exist, e.g., in publicly available registries, are part of a test set, whereas the standard used to describe them is not specified.

A similar scenario is used in (McIlraith & Son, 2002). The scenario involves an author who must plan his trip to a conference. Based on this scenario, if the person were to perform this task manually, he would have to find the location and dates of the particular conference, book the airplane tickets or a rental car to travel to his destination and, then, arrange for his accommodation at a nearby hotel. The implementation of the proposed approach is a generic travel procedure with the relevant Prolog code being provided for a subset of this procedure. The result of the WSC process is able to select and book the transportation to the required location either via air or by car, book a hotel and transportation from/to the airport and email the user the specified itinerary, along with updating an online expense claim.

(Dong et al., 2006) describe a use case in which a user wants to travel from one location to another, with the preference to only stop once during his transport. In case his flight is cancelled and the company does not automatically reschedule him on another flight, he must find a way to travel to his destination through the combination of other existing flights. His requirements are specified in terms of goals, such as departure/arrival city and his airline preference. The authors specifically refer to the use of OWL-S services, mention all the available flight details for a particular day (with each flight being a specific WS) and provide a simple ontology that defines a hierarchy between the services. Moreover, they refer to specific details of their output composite WS, such as the number of WSs that it consists of (just two in each of their examples) and the actual flights that the user should board, i.e., the actual WSs used.

(Hatzi et al., 2008) presents a typical travel scenario, with a user providing the dates for his trip and a sightseeing activity, and the composite WS outputting a reservation for a hotel, its price and a map of the area. For this scenario, the actual OWL-S TC services that are used mainly serve the purpose of keeping the WSs considered for composition to a realistic size, with the authors having created new semantic descriptions that fit their purpose, that is, providing the required inputs/outputs and relevant ontologies' concepts.

A final example in the travel category is the one presented in (Kona et al., 2008), or in specific, the various versions of a basic travel scenario that includes flight, hotel and rental car reservations. The

authors present a different modification of this scenario for each method of WSC that they present; in the case of sequential composition, i.e., when the services comprising the composite WS are part of a linear plan, the scenario used was the aforementioned, with the client desire being to first book a flight, then a hotel to stay at and, finally, rent a car to travel while he remains at his destination. A modified scenario, used for the case when atomic services could be combined in a non-sequential fashion, i.e., more than one services could be involved at any stage of the WSC process, was the following: A client still wants to travel and book a flight, hotel and car, but this time he must travel internationally, and for this reason he must acquire a valid visa. One of the services has as an effect that the client acquires an approved visa, which is the necessary precondition in order for a person to be able to make travel reservations. As such, the former service has to be executed first, and only after it completes successfully (and outputs an approved visa) can the rest of the booking services begin their execution. A third scenario is presented for the case of non-sequential composition that states that before one applies for a visa he should make a tentative flight and hotel reservation. After that, an application for a visa can be made, which, unlike the previous scenario, is not always successful. In case it is, the client can proceed to buy the relative hotel and flight reservations, and book a car as well. If not, these reservations must be cancelled. In this scenario, the results of the WS' execution must be evaluated at runtime, and based on them, a different set of services should be executed.

Another large category of use cases for WSC is the one related to the purchase of specific goods and services. In (Papapanagiotou & Fleuriot, 2011), as well as in (Rao et al., 2006), a use case scenario is presented related to the order of a ski set. In it, a set of value-adding services are used along with a core service providing the main functionality, i.e., return the price of a ski set in US dollars based on the brand, model and length of the requested ski, to compose a composite service. The composite service should output the price of a ski set in a different currency given the user's height, weight and skill level, while having a specific price limit.

In the scenario presented in (Yu & Reiff-Marganiec, 2006) there are six services, each implementing a different functionality in the domain of online TV shopping. There is a computer locator service that can locate the city of the user based on his IP address; a TV information WS that a client can use to get information in regard to TV sets; and a WS representing a TV shop chain which has local shops that fulfill the shopping requests based on the customers' locations. The next two WSs deal with the delivery of the purchased goods and their insurance, while the last one is a TV license service. The WSs are only described in terms of functionality, with no specifics being given on their implementation. The goal of the scenario is to buy a TV from an online shop along with insurance for it, and get it delivered to a specific address, which the user provides as input to the WS (along with the computer's IP which is automatically provided). The authors provide two alternative sequential plans that achieve the aforementioned goal and evaluate their system's performance qualitatively, e.g., based on the degree of automation, the reusability of the system and the level of users' skills that are required.

(Hatzi et al., 2010) refer to an e-bookstore that allows its clients to buy books from it by providing as input a book title and its author, along with their personal information concerning their address and

their credit card. As a result, the store's composite WS should charge the specified credit card for the purchase and output information about the shipping date and the customs cost for the purchased book. The WS always considers just one bookstore and the requested book is always assumed to be in stock, that is, the scenario is fully deterministic.

(McDermott, 2002) refers to a use case scenario, in which a WS that sells in bulk to computer manufacturers requires as input the order itself and the relevant payment information. The output of the WS is either a confirmation number in case the operation is successful or a failure message in case it is not. The implemented example is that of a simple purchasing problem, in which the WS must send a request to a fictional book selling website to ask if they have a certain book in stock and receive its answer.

Finally, (Sirin et al., 2003), (Ponnekanti & Fox, 2002) and (Wu et al., 2003) present some use case scenarios that do not involve either travelling or the purchase of various products; (Ponnekanti & Fox, 2002) provide two use cases and create seven different composite WSs. The first scenario implements a movie and dinner planner composite service. It receives as input the user's name, his location, the movie he desires to view and a specific cuisine and it outputs an itinerary from the user's location first to a restaurant serving the inputted cuisine, then to a theater that the selected movie is showing, and finally back to the original location. The second scenario receives as input two stock market symbols and outputs an email message containing current quotes for the two symbols.

(Sirin et al., 2003) provide two use case scenarios; the former assumes a WS that translates text between several language pairs and one that provides a dictionary for the English language. If a client asks for a WS that functions as a dictionary for the French language, neither can fulfill the goal. However, if the two services are composed and the initial French text is translated to English, the result is provided to the dictionary service and then translated back to French, then the desired goal can be achieved. The latter use case refers to a sensor network environment that is comprised of simple sensor services, each one related to a WS that outputs the relative sensor data, and sensor processing services that combine the data from different sensors to output a new result, such as recognizing an object. Moreover, each sensor is also described by functional and non-functional properties that state whether a task is compatible with it.

SHOP2 (Wu et al., 2003), (Sirin et al., 2004) uses a use case based on (Berners-Lee et al., 2001). In it a mother with two children has medical problems and wants to visit a physician who will prescribes a series of activities, a few of which require making an appointment. The appointments have to come in a specific order, e.g., the follow up check must come after the treatments, and the children have to be accommodated so that they can actually drive their mother to the appointments, e.g., two different appointments can't be placed on the same day. The authors state that the implemented system uses actual WSs when possible, adding semantic markup when necessary (as some of the WSs used only had WSDL descriptions) or creating new SWS in some cases, as some functionalities, for example an online pharmacy store, could not be fulfilled through actual WSs.

4.2.2 OWL-S Test Collection

OWL-S is the most widespread formalism for the description of SWS in the research community, used in a plethora of approaches, e.g., (Sirin et al., 2004) (Srinivasan et al., 2004), (Mrohs et al., 2005), (Birchmeier, 2007), (Bo & Zheng, 2009), (Ziaka et al., 2011), (Guo et al., 2005), (Vaculín & Sycara, 2007), (Aslam et al., 2006) and (Chung et al., 2005); OWL-S TC is a collection of OWL-S WS descriptions created with the primary purpose of supporting and measuring the performance of semantic WS matchmaking algorithms, supporting versions 1.0 and 1.1 of the OWL-S formalism. The majority of descriptions has been retrieved from UDDI registries or converted semi-automatically from the corresponding WSDL descriptions of WSs. While there is no standard test for the evaluation of the performance of WSC systems, OWL-S TC is perhaps the only WSs collection of significant size[13]. It has also been used continuously in the annual international competition Semantic Service Selection S3, with its latest version, TC v.4, being used in the two latest contests (S3 Contest, 2011) (Klusch et al., 2013), and the European project SEALS (Semantic Evaluation at Large Scale) (Cabral et al., 2011). Furthermore, it is one of the recommendations of Semantic Technologies Institute (STI) (STI International, 2011) for the evaluation of semantic tools. Additionally, as mentioned in the previous sections, older versions have been used by similar systems in the literature, e.g., version 2.1 in (Klusch et al., 2009) and version 2.2 in (Hatzi et al., 2011), as well as in one of the biggest freely available online semantic registries, Opossum (Kuster et al., 2008). The widespread use of this collection confirms the suitability of its use for the evaluation purposes of this thesis, while allowing efficient comparison with other WSC applications. As such, this thesis opts to use OWL-S as the language to semantically describe the WSs in our approach; the latest version of OWL-S TC (TC4) was preferred over the older one (TC 2.2), although it has been used less in the literature, because it comprises more OWL-S descriptions (1083 versus 1007), comprises more domains (nine versus seven)[14], and because of the correction of various bugs in the available descriptions and the enhancement of their textual descriptions.

4.2.3 Movie Search Scenario

The first use case scenario proposed by this thesis (MS-UC_1) is fully deterministic, allowing for the output of a fully serialized composite WS; it uses a subset of the SWS descriptions of a single domain of the OWL-S TC v.4, namely the Communication domain. In the following example, the ontologies' concepts used are shown in parentheses (following the format "*ontology#concept*"), all of which are present in the relevant ontologies in the latest version of OWL-S TC. Specifically, MS-UC_1

[13] SAWSDL-TC (SemWebCentral, 2008) is a similar size set of WSs with semantic descriptions in the SAWSDL formalism. But since the creation is based on the semi-automatic conversion of the original OWL-S files from the OWL-S Test collection in SAWSDL, we do not regard it as a distinct collection of SWS. Furthermore, OWL-S is significantly more widely used compared to SAWSDL.

[14] Version 2.2 contained the following domains: education, medical care, food, travel, communication, economy, weapon. Version 4 added the geography and simulation domains.

models a situation where a user knows the title of a film (books.owl#Title or my_ontology.owl#Film) and wants to retrieve all the comedy films (my_ontology.owl#ComedyFilm) that exist with this title, along with their respective prices; however, he desires to know all the relevant pricing information in regard to a comedy film, i.e., its regular price (concept.owl#Price), its maximum price (concept.owl#MaxPrice), the recommended one (concept.owl#RecommendedPrice), and the price with (concept.owl#TaxedPrice) and without taxes (concept.owl#TaxFreePrice). Finally, he wants to store all of these results to a database (ontosem.owl#database) so that he can remember to buy them in the future.

In order for the tests to be more meaningful in regard to the system's capabilities, we have added several new OWL-S descriptions to the Communication domain, each a modification of an existing one. For the test case, however, all the SWS that are contained in the relevant domain can be included in the WSC process, that is, all 58 descriptions of the Communication domain, so that the problem is of realistic size. Variations of the experiments could possibly include a smaller set including only the relevant WSs, or all the available SWSs in the online registry. The reasoning behind the addition of new descriptions in the domain is to allow for the creation of different plans that achieve the same goal in order to evaluate the system's ability to produce optimal plans efficiently. The new SWSs, the relevant ones from the Communication domain that can be used in this scenario and their inputs/outputs can be found in Section Ap.A.1 of the Appendix.

4.2.4 E-bookstore Scenario

The second use case scenario (MS-UC_2) refers to an e-bookstore that allows its clients to buy books from it. Similarly to MS-UC_1, it uses a subset of the SWS descriptions of a single domain of OWL-S TC, i.e., the Education one. However, this scenario contains non-determinism in the users' choices, as well as the results of the actions. In detail, MS-UC_2 states that initially the user provides as input a book title (books.owl#Title) or its ISBN (portal.owl#ISBN), his address (order.owl#Address) and a preferred method of payment for the purchased item. As a result, if the book is available in stock at the specific e-bookstore the final composite WS should use the specified method of payment (finance_th_web.owl#payment, whether cash or credit, that is cheque, debit, or credit card) to purchase the item; that is, pay the price it costs (finance_th_web.owl#Retail_Sale) based on the previous steps of the WSC process, and record the address for the item in the user's shopping cart. The composite WS should result in a purchased item (order.owl#PurchasedItems), and the output of information regarding the purchased book; specifically, the book's author (books.owl#Author), its type (hard-cover or paperback) (books.owl#Book-Type), size (small/large/medium) (books.owl#Size) and the paid price (concept.owl#Price) for it. If, however, the book is not in stock, no payment should be made and no output information should be displayed to the client.

As in MS-UC_1, we have added or modified several OWL-S descriptions; moreover, for MS-UC_2, we have also modified some of the concepts' relationships in the relevant ontologies for our evaluation

purposes. These modifications are presented in detail in Section Ap.A.2 of the Appendix. For the test case all the SWS that are contained in the relevant domain can be included in the WSC process, that is, all 285 OWL-S descriptions in the education domain of OWL-S TC4, so that the problem is of realistic size. In variations of this scenario, and for evaluation purposes, a smaller set, e.g., containing only the SWS that refer to books[15], or larger set, e.g., also containing the SWS of the economy domain (359 in total) or all the available SWSs in the online registry can be used to test the performance of a WSC system in various situations.

As mentioned above, this scenario contains elements of non-determinism and is, thus, more complex than MS-UC_1. Apart from the initial decision of use between a book title or ISBN number, which can result in the use of a different WS, the system has to be able to accommodate (if that is possible, based on the available WSs) all possible forms of payment, that is, cash or credit, and their relative sub-categories (all forms of cards). More importantly, though, the fact that a book is not assumed to always be in stock defines a precondition that dictates the system's flow non-deterministically; depending on the status of the book, the composite WS could alternatively have to charge the client's credit card, or abort its operations.

4.2.5 Camera Purchase Scenario

The third use case scenario (MS-UC_3) refers to a person who desires to buy an analog camera with a specific product code and of a specific type; since he is seriously interested in photography and wants his camera to have more advanced features, he would rather buy this SLR camera than a – simpler - compact one. However, if such a camera is not in stock, then he would rather buy a compact one than none at all. Finally, if neither is available at any store, he will settle for any camera at all.

As all the previous scenarios, MS-UC_3 also uses a subset of the SWS descriptions of a single domain of OWL-S TC, namely, the Economy domain. It is also similar to MS-UC_2, as it regards the purchase of an item and it also features elements of non-determinism. In contrast to the second scenario, however, where only one seller was assumed to exist and the user wanted to purchase his book from that specific store, in this scenario, the user has various options to make his purchase from. Furthermore, in this scenario the user has a clear preference between products, but is also willing accept the purchase of alternative ones if he has to.

In detail, the described scenario states that initially the client provides as input to the composite WS the type of camera he desires to buy, which is an analog, non-APS, standard SLR camera (extendedCamera.owl#SLR). However, if this camera is not available, the scenario presumes that the user would also be satisfied with buying another type of analog camera that has less features, specifically an analog, non-APS, standard compact camera (extendedCamera.owl#Compact). In either case, the user wants a specific analog camera and for that reason he also provides the desired camera's product code (extendedCamera.owl#ProductCode). Finally, though, if no store is found that has an analog,

[15] 67 of the SWS descriptions in the education domain of OWL-S TC4 are relevant to our scenario.

non-APS, standard SLR or a compact camera in stock, then he will settle for any kind of camera, whether analog or digital (extendedCamera.owl#Camera). The user should also state in advance which stores should be considered as alternatives for him to buy the camera from; in specific, he can choose to provide as input a shopping mall (Mid-level-ontology.owl#ShoppingMall), a retail store (Mid-level-ontology.owl#RetailStore) or a specific chain of retail stores, with the available ones being Walmart (Mid-level-ontology.owl#WalmartStore) and Media Markt (Mid-level-ontology.owl#MediaMarktStore). He can also state that all mercantile organizations can be considered as alternatives (SUMO.owl#MercantileOrganization). Again, the user can provide one or more inputs, but in this case all the available choices are considered equally preferable alternatives. We assume that the user does not differentiate between the alternative stores he has provided as input, as long as he finds the camera he desires in stock. For simplification reasons we assume that the price of the camera is identical for all the available stores; however, he might wish to exclude some of the available stores from his alternatives' list, as, for example, he might have had a negative experience with this store in the past.

Having inputted the desired product type (extendedCamera.owl#SLR) along with its product code (extendedCamera.owl#ProductCode) and the alternative stores that can be used to buy it from, the composite WS should find a store that sells this product and check whether it has it in stock or not. If it is, it should add it to the user's shopping cart (ShoppingCart.owl# ShoppingCartRequestItems), and if not, it should continue to search for another store that sells it. If it can't find any store that has the SLR camera in stock, it will repeat the aforementioned process, this time searching for a compact camera (extendedCamera.owl#Compact). If no compact camera is in stock either, then the composite WS will search for any camera available in stock.

The output event of the service can only be an addition to the user's shopping cart or no action at all. We assume that there is a single, "*universal*" WS that can check the availability of every product code in every store, like a price comparison site (e.g., http://www.pricegrabber.com/). If, however, the item is not in stock, no action will be taken (by the atomic service itself; the composite WS should search for alternative cameras/stores). Since the SLR and Compact camera concepts are subclasses of the camera class, and the retail stores (along with specific shops – Media Markt and Walmart) and shopping malls are subclasses of the mercantile organization class, all of these concepts should also be accepted as input from the WS.

As in the previous scenarios, we have added or modified several OWL-S descriptions for evaluation purposes; the OWL-S descriptions comprising the economy domain are 359 in total. More information about the relevant WSs in the economy domain, the changes that were made to them and the relevant ontologies and the relation between the different camera types in the extendedCamera ontology, is presented in Section Ap.A.3 of the Appendix.

4.2.6 Simple Contingent Use Case Scenario

The fourth use case scenario (MS-UC4) comprises a simple dummy problem for the contingent planning algorithm that will be presented in Section 6.1; as such it is not based on OWL-S TC and the problem does not describe a real-life use case. It does test, however, the desired features of the contingent algorithm, that is, whether alternative plans can be computed and merged together. Figure 4.1 presents the problem; it consists of a (single) initial state s_0 and the goal state G (each state is represented by a circle). Actions a_2 and a_7 are deterministic (denoted by a straight line), while the rest of the actions are probabilistic (denoted by a dotted line); for each probabilistic action a_i, $i \in \{1,3,5\}$, its (single) effect is achieved with a probability $prob_{\alpha_i} = 0.8$, $\forall i \in \{1,3,5\}$, and with a probability of $1 - prob_{\alpha_i} = 0.2$, it fails and no effect is achieved (the current state does not change). Actions a_4 and a_6 have two different effects, each having a different probability of being produced when executing the related action, with $prob_{\alpha_{41}} = 0.9$, $\text{prob}_{\alpha_{42}} = 0.1$ and $prob_{\alpha_{61}} = 0.8$, $prob_{\alpha_{62}} = 0.2$. The cost associated with each action is shown opposite it, e.g., $cost_{\alpha_{11}} = 4$ and $cost_{\alpha_7} = 2$.

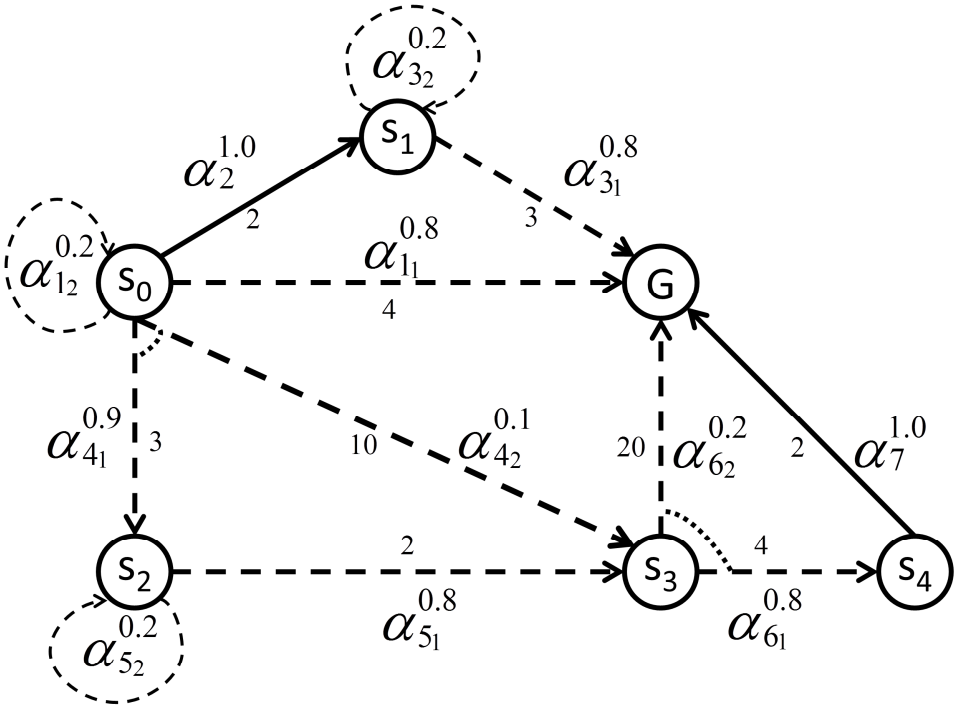

Figure 4.1: Simple contingent use case scenario

4.2.7 Complex Contingent Use Case Scenario

The fifth use case scenario (MS-UC_5) was specifically defined for the contingent planning algorithm that will be presented in Section 6.1; it is more complex than MS-UC4 and, similar to MS-$UC_{1\text{-}3}$, it uses a subset of the SWS descriptions of a single domain of OWL-S TC, namely the Geography domain. In detail, MS-UC_5 describes a situation in which a traveler who is on a business trip is required to visit two locations at difference cities of a European country. He has to travel to both locations within the same day by car, knowing only their respective addresses (protonu.owl#Address); his goal is to obtain driving directions (geographydataset.owl# DrivingDirections) that include both the locations, a map related to the two locations (geographydataset.owl#Map), as well as a visit to an ATM (that is identified by its protont.owl#latitude and protont.owl#longitude) so that he has some cash for the trip. Moreover, he desires to know the local time of sunrise (geographydataset.owl#Sunrise) and sunset (geographydataset.owl#Sunset) for the day he will be traveling, so that he can adjust his trip accordingly for as long as there is sunlight. Finally, he would like to know the total distance (geographydataset.owl# GeographicDistance) he has to cover in miles (geographydataset.owl#InternationalMileLengthUnit).

It is assumed that the traveler has knowledge about and can initially provide as input to the composite WS the respective addresses (protonu.owl#Address) of the cities (protonu.owl#City) that he will visit, as well the states (protonu.owl#State) and country (protonu.owl#Country) they belong to. Moreover, it is assumed that he can provide the dates (protonu.owl#Date) of his travel, and that he has a specific language (protont.owl#Language) in which the composite WS' results will be returned, as well as that he is able to provide valid identifiers (either geographydataset.owl#Code or geographydataset.owl#UniqueIdentifier) that allow him to use specific WSs.

As in the previous scenarios, there have been added several OWL-S descriptions for testing purposes; these services and their input/outputs are presented in detail in Section Ap.A.4 of the Appendix. For the test case, however, all the SWS that are contained in the relevant domain can be included in the WSC process, that is, all 60 OWL-S descriptions in the geography domain of OWL-S TC4, so that the problem is of realistic size.

4.2.8 Manual Web Service Composition Scenario Mockups

Finally, before the implementation of the manual WSC module, and so as to guide that process, several mockups of the application that was going to be created were designed in order to have a clear goal as to how the final system's GUI should be be based on the requirements presented in Table 4.1 and the previously presented use case scenarios in Sections 4.2.3-4.2.5. The mockups that were consequently created for this purpose, using the Pencil Project,[16] are shown in Figures 4.2-4.28.

[16] Available at http://pencil.evolus.vn/

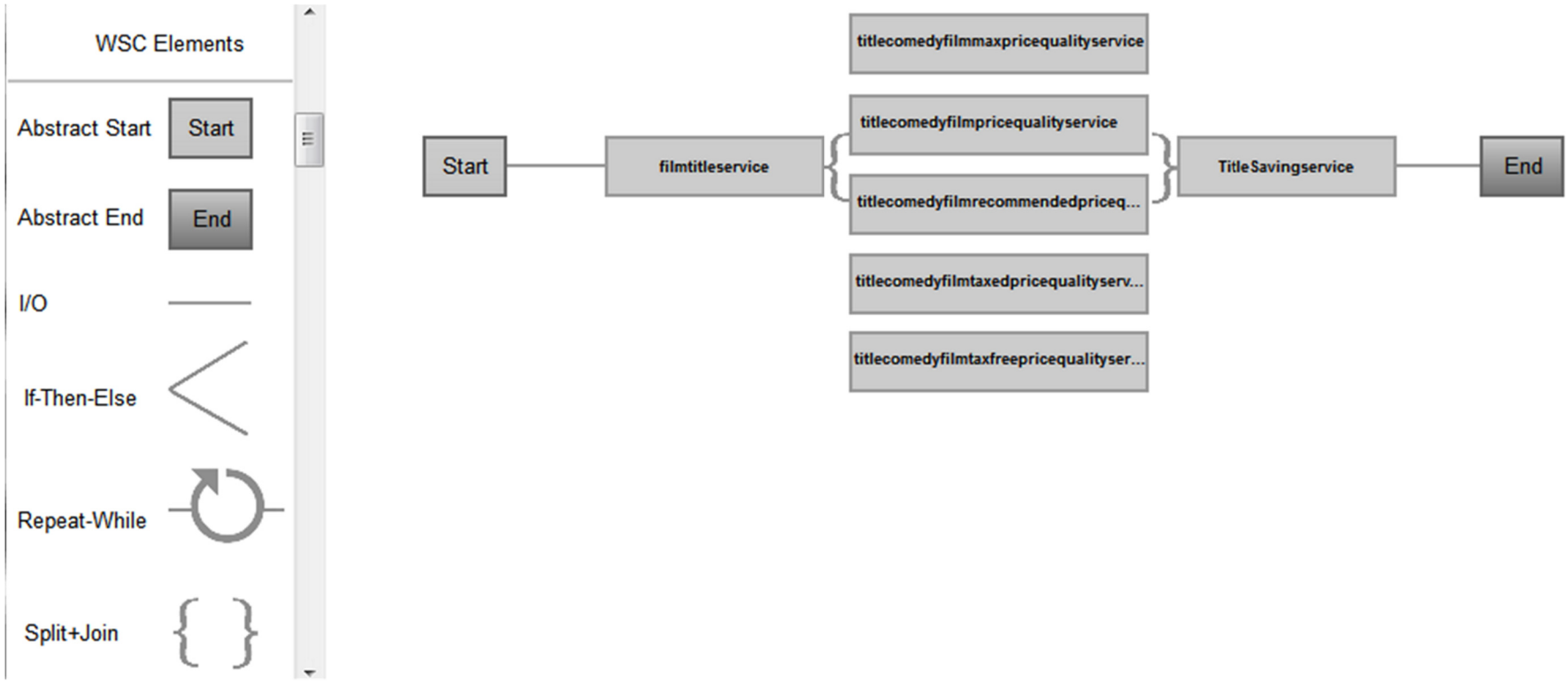

Figure 4.2: Movie search scenario mockup

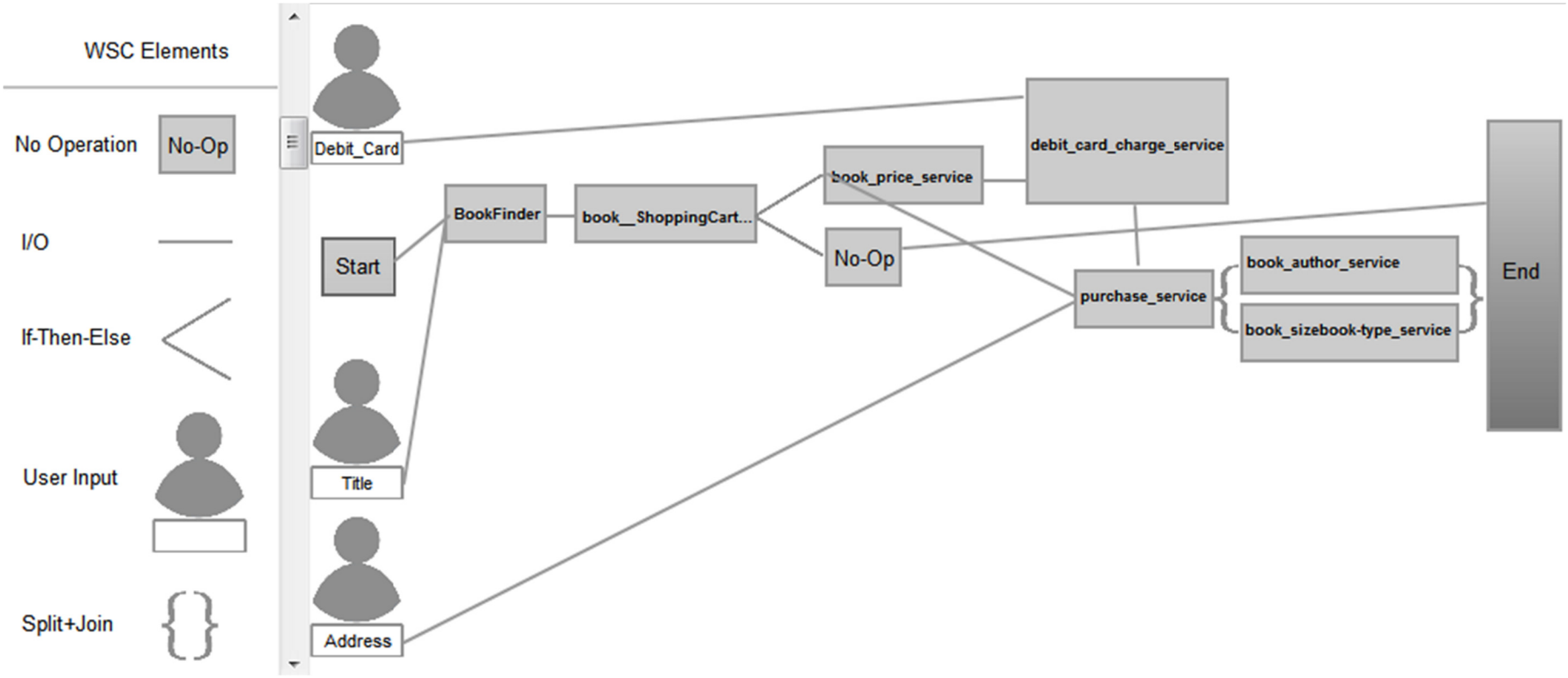

Figure 4.3: E-bookstore scenario mockup

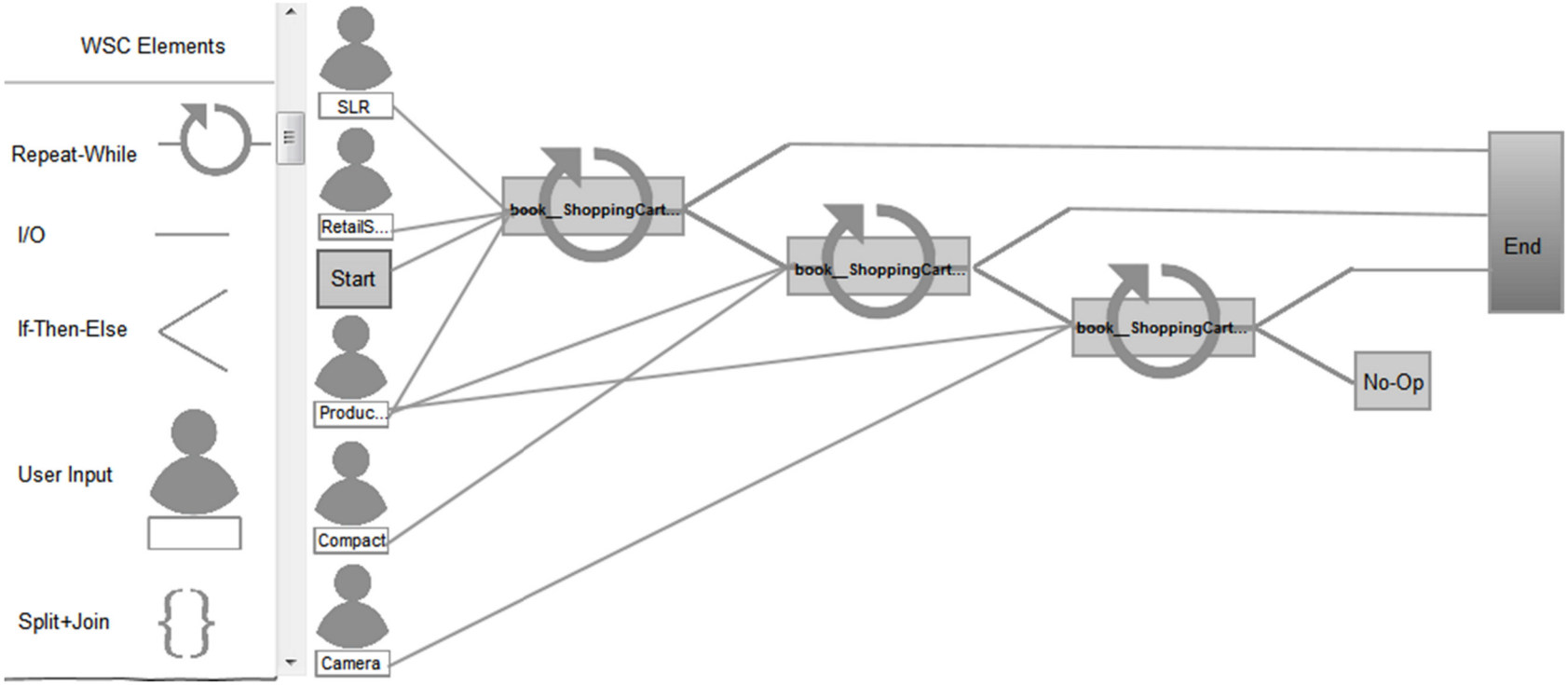

Figure 4.4: Camera purchase scenario mockup

Chapter 5.

Implementation of a Semantic Web Service Composition System

This section presents the WSC research prototype that has been developed within this thesis, named **MADSWAN** (an anagram of "**M**anual **AND A**utomated **S**emantic **W**SC"). **MADSWAN** is a system aiming to automate WSC procedures through the combination of semantic web technologies and AI planning techniques.

In this thesis' view, a WSC system should follow the same principles as WSs themselves. Since WSs rely on the idea of maximizing the reuse of loosely coupled components (McGovern et al., 2003), **MADSWAN** has been implemented by making use of open source, already existing, software components as much as possible, in a bottom-up approach. This led to a reduced required effort in comparison to creating entirely new components, whereas it allowed the use of well-established standards instead of proprietary ones, thus facilitating the comparison of different WSC systems to each other. The overall architecture of **MADSWAN**, including its constituent components, is shown in Figure 5.1.

The automated WSC process includes the following phases (Rao & Su, 2003):

- presentation, or advertisement, of a single service in a registry;
- translation between external and internal service specification languages for the domain, i.e., a WS description language and a planning one;
- generation of a composition process model;
- evaluation of the output composite service.

MADSWAN supports all the aforementioned phases of the WSC process. Indeed, a typical user can advertise a new WS in the online registry, as well as retrieve and edit the WSs stored in it through the system's online interface. Moreover, it is possible to manually create workflows based on OWL-S control constructs and bind them to WS descriptions and concepts that are present in the online registry. Finally, users are currently able to generate composition process models automatically, based on AI planning following a translation of the original semantic WSC domain, described in OWL-S, to an AI planning one, defined in PDDL. The evaluation process of the system itself is based on the use case scenarios presented in Chapter 4.

This chapter provides a rigorous analysis of **MADSWAN**, presenting a working prototype; we describe the modules of the system in detail, that is, the WS registry that constitutes its basis, the XML editor, and the manual WSC module. The algorithms used in the automated WSC module will be presented in a next chapter (Chapter 6). **MADSWAN** is, to the best of our knowledge, the first system of its kind with a publicly available prototype[17] able to support various stages of the WSC process. More information, as well as video tutorials on the basic use of the system are available at (Markou, 2014b).

[17] The current version of the online prototype is accessible at http://madswan.uom.gr/

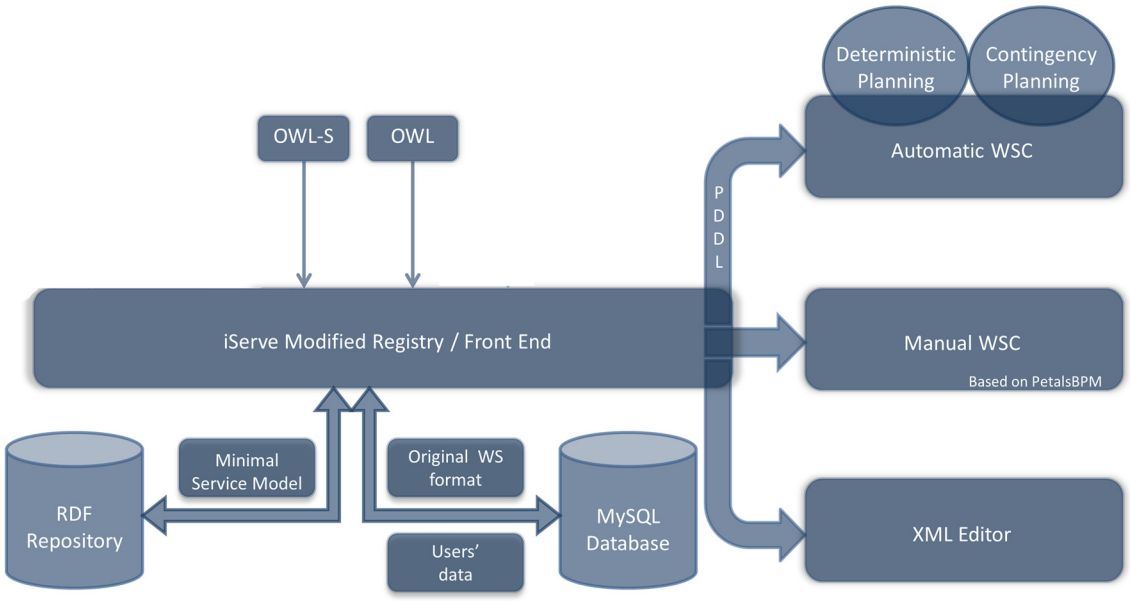

Figure 5.1: Architecture of **MADSWAN**.

Other web-based systems similar to **MADSWAN** exist, such as YaWSA (Macdonald, 2007); YaWSA provides a web-based interface that supports a WSC process, however, it is no longer available for public use and, most importantly, it provided no other capabilities related to different phases of WSC, such as a registry. (Du et al., 2006) present a system supporting multiple phases of WSC, including WS browsing, the creation of composite services and service flow execution. It is based on common process pattern instances, essentially abstract composite services, which are used to bridge the gap between the users' requirements and the technical service descriptions. However, the common process pattern instances are manually created by users, in a process similar to creating a flowchart diagram.

The system that is most closely related to **MADSWAN** in terms of functionalities is the one implemented in the SUPER (Semantics Utilized for Process management within and between EnteRprises) project (Information Society Technologies, 2006). The major objective of SUPER was to bridge the gap between the business needs expressed by business people and the actual Information Technology infrastructures intended to support them, while also supporting in a more efficient way the reuse and automation of business processes. For this reason, it implements a semantic-based and context-aware framework platform that supports the management of business processes in a scalable manner, through the use of semantic WSs' technologies. The final platform includes modules for the automated discovery, substitution, composition and execution of business process implementations. Furthermore, three use case scenarios were developed for the needs of the project, all based on a telecoms domain, covering the fields of fixed telephony, traffic routing and the management of mobile environments.

Despite the different objective of SUPER project in comparison to **MADSWAN**, it shares a lot of similarities with it. The system's interface was alike the one in the manual composition module

presented here, using the BPMN (Object Management Group, Inc, 2011) standard as one of its basic elements. More importantly, the two systems share a similar architecture, e.g., the inclusion of modules for the discovery, translation and composition of semantic WSs, even though the underlying standard for the description of semantic WSs is different; WSMO in SUPER and OWL-S in **MADSWAN**. The evaluation process of SUPER was conducted solely based on interviews with a sample set of the system's users expressing their view on criteria such as the completeness and support, e.g., in terms of tools, of the system, or on its reuse of open source software and standards and its overall correctness. In our case, the evaluation process is based on quantitative criteria. The most important difference, however, lies in the WSC approach; WSC is used in SUPER to refine relevant parts of a business process model by searching for partial replacements in a process model and does take non-determinism into account.

5.1 Semantic Web Service Registry

The basic functionality related to WSC is storing and retrieving web service descriptions. In order to support semantic WS discovery in a more meaningful way, this thesis opted against the use of UDDI; As aforementioned in Section 3.1, UDDI's search mechanism is based on the description of the WSs' capabilities using a classification schema that does not provide for a semantic description of their content. For this reason, instead of using UDDI, or approaches such as (Srinivasan et al., 2004) and (Luo et al., 2006) that bridge the gap between semantic WSs and UDDI (in most cases between OWL-S and UDDI), this thesis opted to adapt an existing web based application, namely iServe (Pedrinaci et al., 2010), as the core of our application. Specifically, the original iServe application was modified and extended by incorporating new functionalities through the implementation and addition of new software modules.

5.1.1 Basis of the Registry - iServe

iServe is an open source platform used for the publishing and discovery of various formalisms of WSs in relation to the Service Oriented Architectures for All (SOA4All, 2011) project; it supports WSDL, SAWSDL, OWL-S, MicroWSMO (Kopecký & Vitvar, 2008) and WSMO-Lite (Fensel et al., 2010) descriptions, by converting each WS description into linked data (Bizer et al., 2009) irrelevant to its original formalism. A common ontology that expresses the "vocabulary" of the WSs' descriptions is used, namely the Minimal Service Model (MSM) (iServe Team, 2010) ontology, depicted in Figure 5.2. MSM is a simple RDF(S) ontology that can be used to model semantic information of WSs and Web APIs so that they can be published and searched in registries. In short, it defines services that have a number of operations that contain Input/Output Messages, as well as error ones.

iServe incorporates various semantic web tools, such as a semantic annotation editor (Maleshkova et al., 2009) or a WS recommender system (Liu et al., 2010). However, such tools are beyond the scope

of this thesis' goals, and as such have not been integrated in **MADSWAN**. **MADSWAN** only utilizes the web-based application that allows users to browse the directory, search or add WSs to it, along with the RDF store and its SPARQL endpoint, with which one can interact in order to retrieve WSs or perform queries on their descriptions. iServe is patented under version 2.0 of the license Apache (The Apache Software Foundation, 2004), which allows the use and modification of its source code. One of the reasons why **MADSWAN** was based on iServe is that its graphic interface was implemented using Google Web Toolkit (GWT) (Google, 2011) in conjunction with the external library Ext-GWT[18]. These tools allow the implementation of web applications in Java that automatically compile to Javascript (Flanagan, 2006), which is compatible with all modern browsers. Moreover, the architecture of GWT follows the rules of Web 2.0 (O'Reilly, 2007), while also being open source software supported by Google.

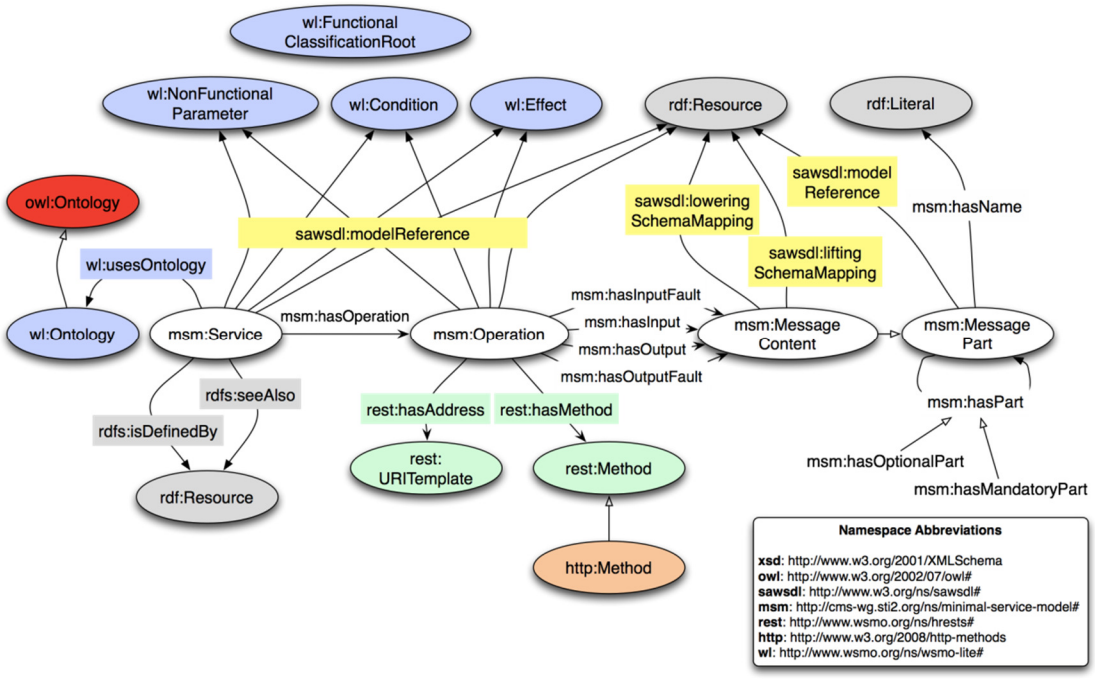

Figure 5.2: The MSM Ontology (Semantic Technology Institute International Working Group, 2009)

5.1.2 Web Based Application

This sections presents the implemented web application, as well as its differences and extensions from/to the original iServe application.

5.1.2.1 User accounts

A **MADSWAN** user can login either as an unregistered user with reduced rights or as a registered one with full rights; unregistered users are not allowed to add new WSs to the registry or edit/delete existing

[18] http://www.sencha.com/products/gxt/

ones (Figure 5.3), in essence, only being allowed to browse the existing WSs. **MADSWAN** implements local accounts based on storing user data in the local MySQL database and sending confirmation emails to the users to in order to support the registration of users; a user must enter a desired username and email account, both of which should not already exist in the application's database; an HTML email is sent to the user, including his username and a random - seven character – string (in fact the first seven digits of a random UUID (Oracle, 2011)) that can be used as temporary password (Figure 5.4) for his first login as a registered user. It should be noted that the users' passwords are not stored as is in the MySQL database; only their hashes are kept. Specifically, an OpenBSD Bcrypt scheme is used to create the hashed key (Provos & Mazières, 1999). Additionally, the application utilizes cookies so as to "remember" recurring users.

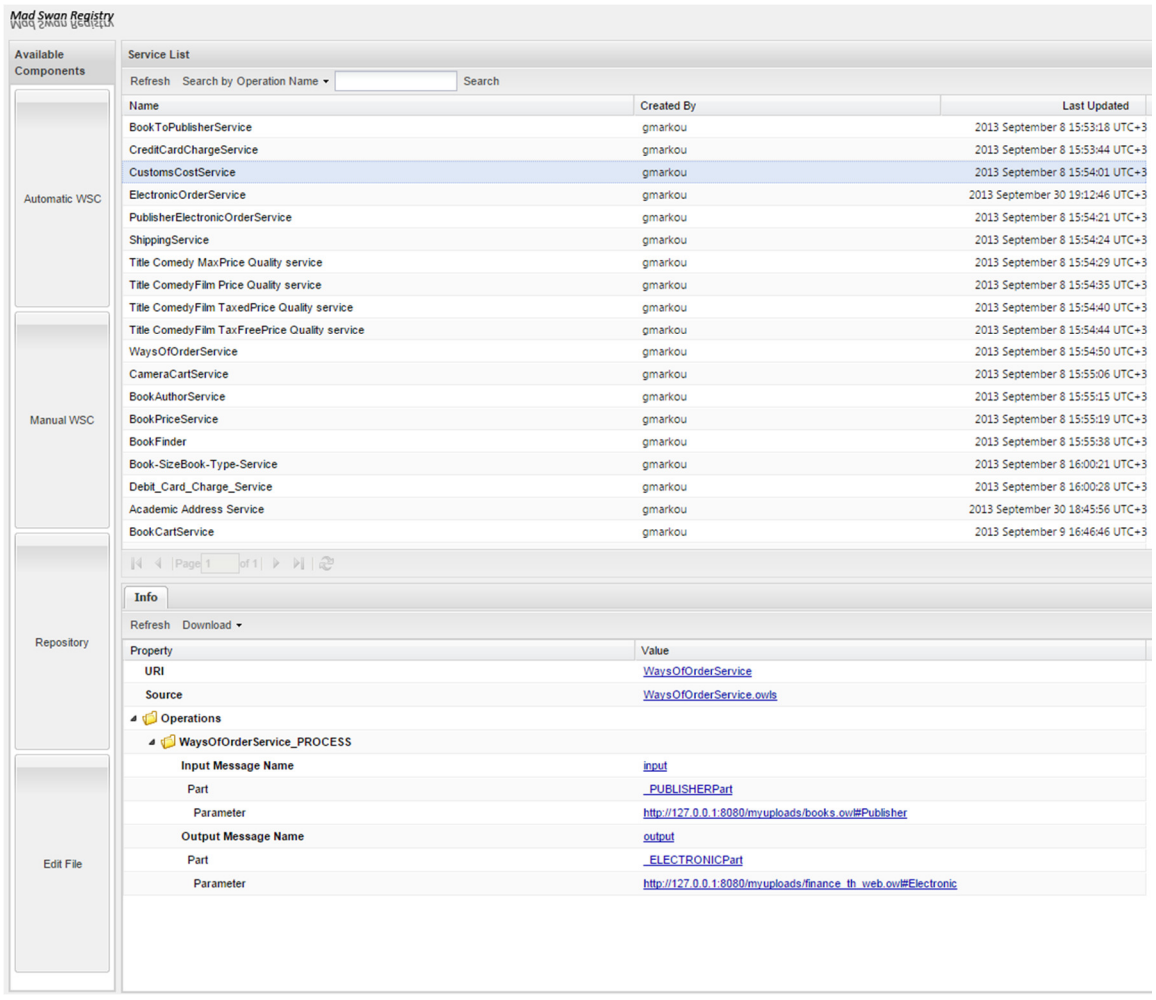

Figure 5.3: Unregistered users' GUI

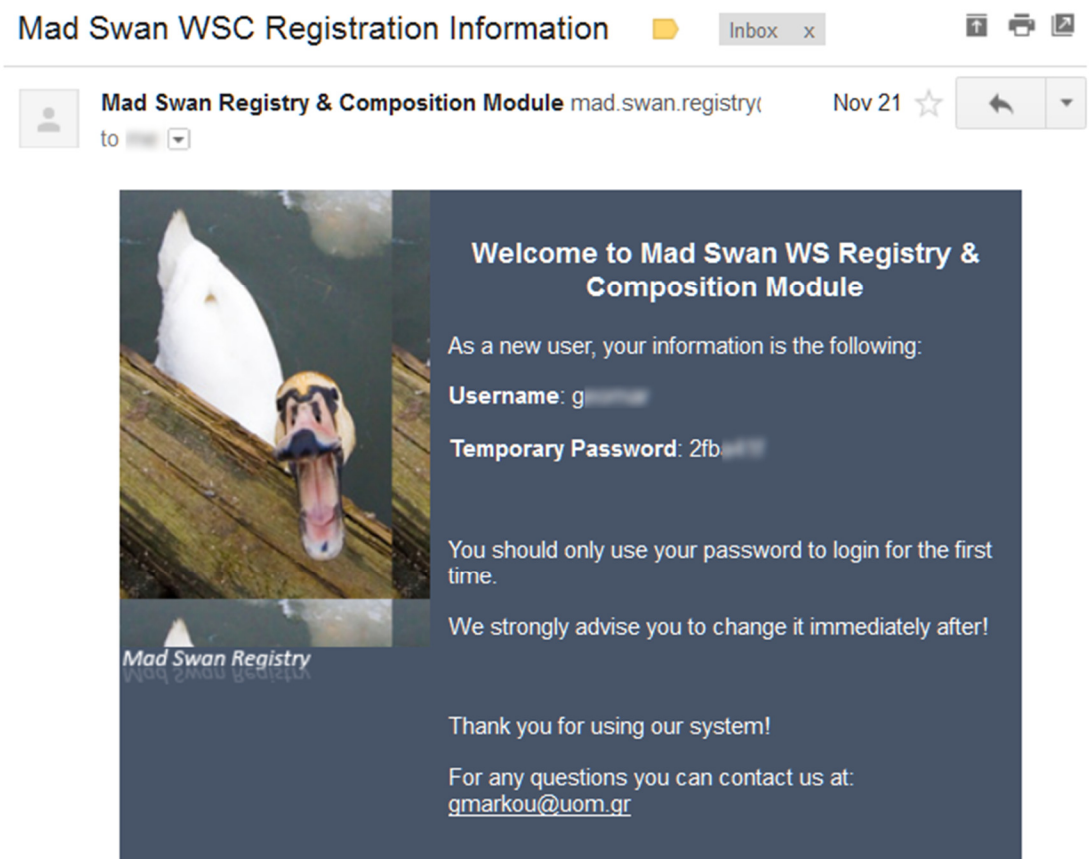

Figure 5.4: HTML formatted email to registered user

5.1.2.2 Add - Remove – Edit OWL-S Web Services

Registered users can search for a WS, upload or delete one to/from the registry and edit the existing ones through the GUI application. The user has the ability to search for WSs in the registry using their operations' names, the category they belong to under the taxonomy used, their input/output parameters, their address at which they can be found/used, or their unique identifier (ID) (Figure 5.5). Registered users can also upload new WSs to the registry; in particular, OWL-S v1.1 descriptions that exist locally on the users' computers can be uploaded to the web application (Figure 5.6). These descriptions are then stored and converted to linked data based on the MSM ontology, that is, in RDF triples. Additionally, the option to also store the description in its original form as an OWL-S file in the MySQL database has been added, i.e., as a binary large object - BLOB); this allows for more efficient editing of the stored WSs in the editing module. When the description of a WS has been successfully stored in both databases, the user can see the basic information of the semantic description in the linked data form, such as its supported functions and the required input and output parameters, as shown in Figure 5.7.

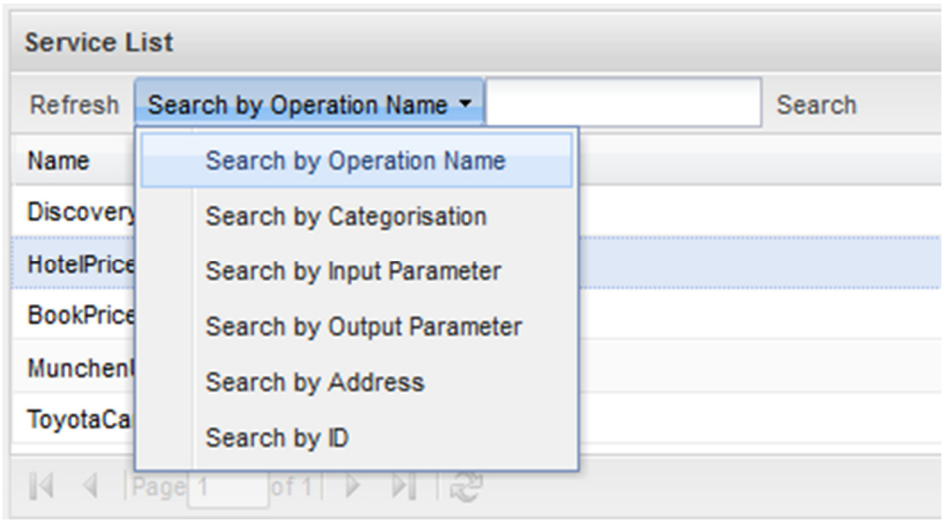

Figure 5.5: Search for a web service in the registry

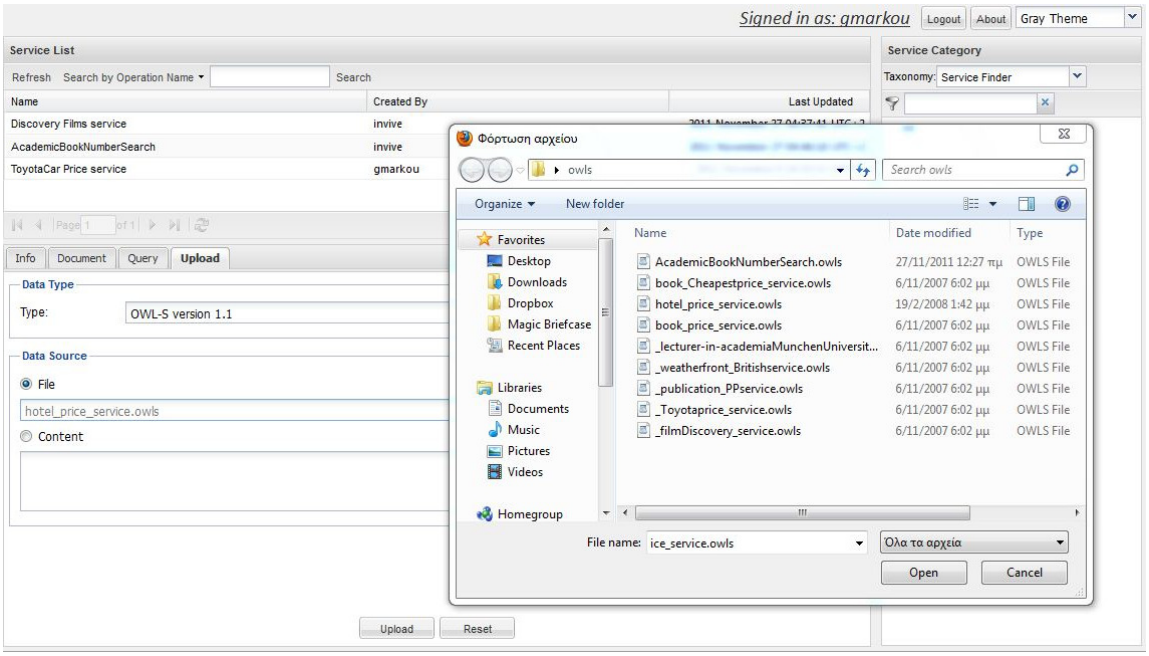

Figure 5.6: Uploading an OWL-S description to the registry

During the addition of a new WS in the registry, additional elements regarding it, such as the user's and the WS' ID are stored. These elements are used in two cases: first, in case a user attempts to upload another WS with the same ID, the application warns the user and asks him whether he wants to replace the existing WS or store it as an alternative one (Figure 5.8). Second, in case a user chooses to delete an existing WS, then he is only allowed to do so if he was the one that had initially uploaded it. This precaution is taken to prevent malicious use of the system.

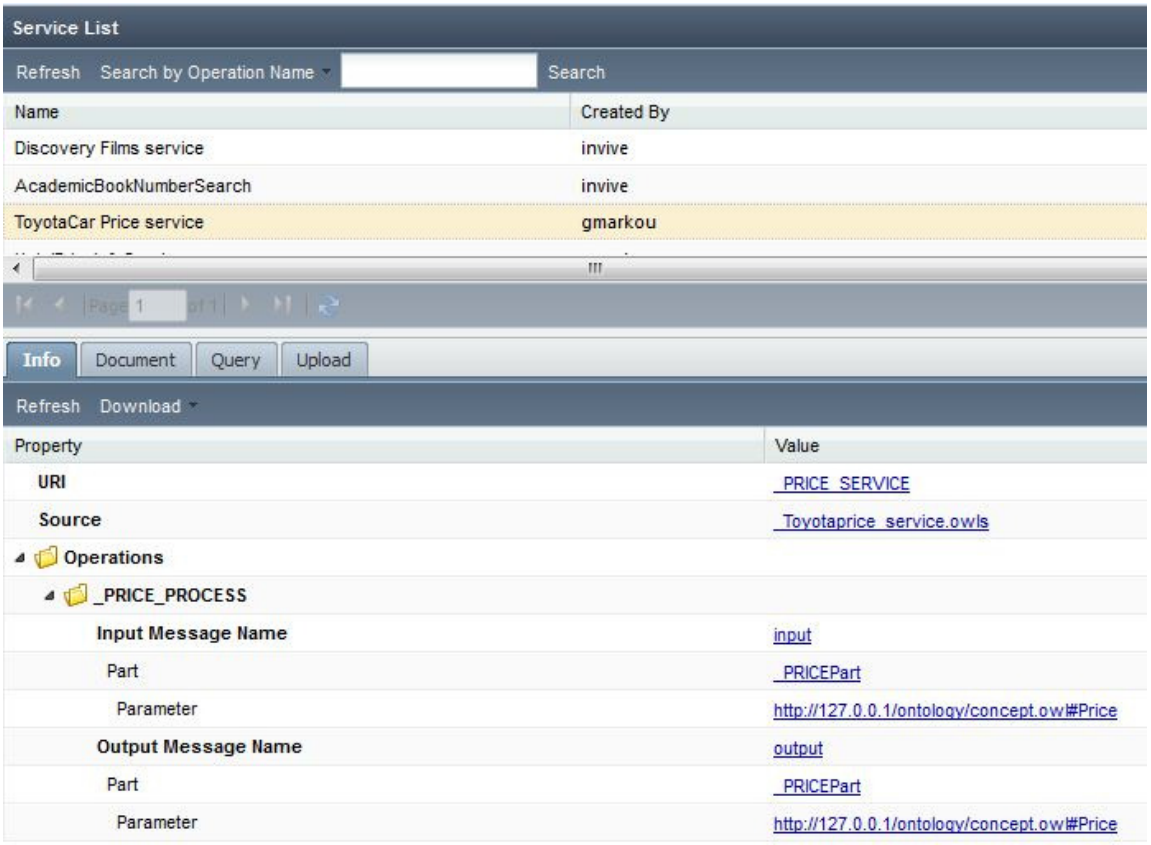

Figure 5.7: Information regarding the uploaded WS

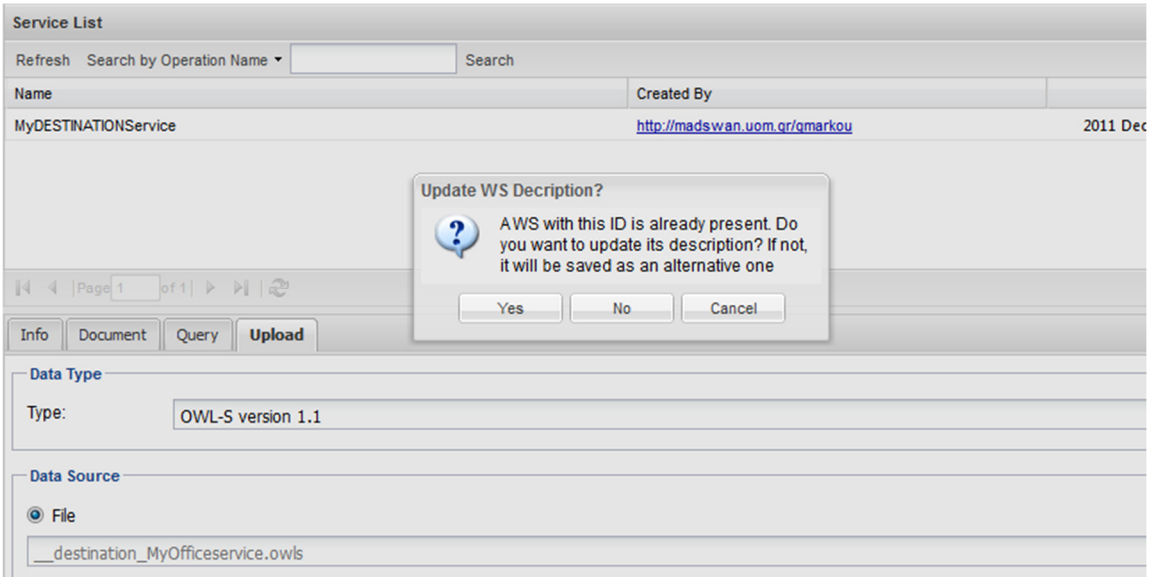

Figure 5.8: Uploading a web service with an existing ID; user's choices

An additional module that was added to the original application was an XML editor with syntax coloring in order to edit the existing OWL-S descriptions. The editor also has pre-existing templates for the most commonly used tags in OWL-S, e.g., *<service:presents/>, <profile:hasInput/>* , as seen in Figures 5.9-5.10.

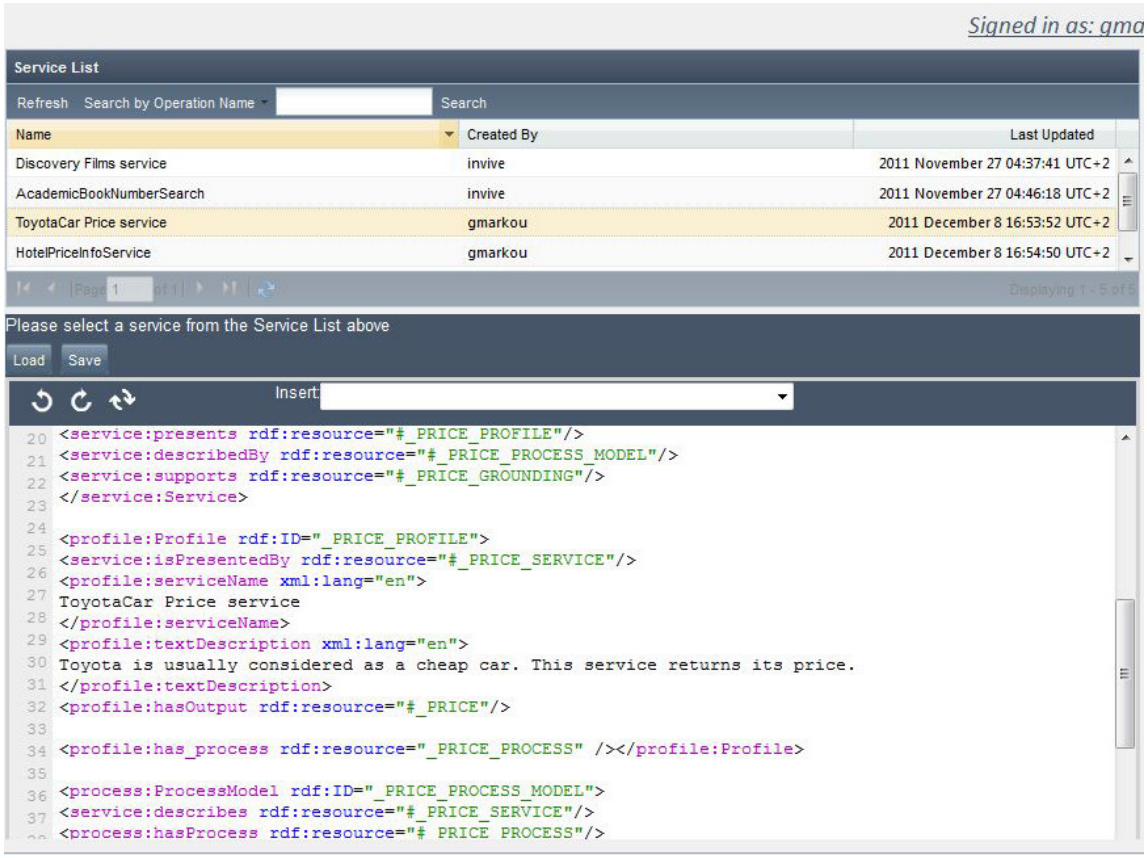

Figure 5.9: Online editor with syntax coloring

Finally, apart from the addition of the aforementioned capabilities, various aesthetic modifications to the original GUI were implemented. For example, the addition of different color combination profiles for the GUI, the use of different notification messages, the initial presentation of all existing WSs instead of an empty list in the homepage, and various bug fixes.

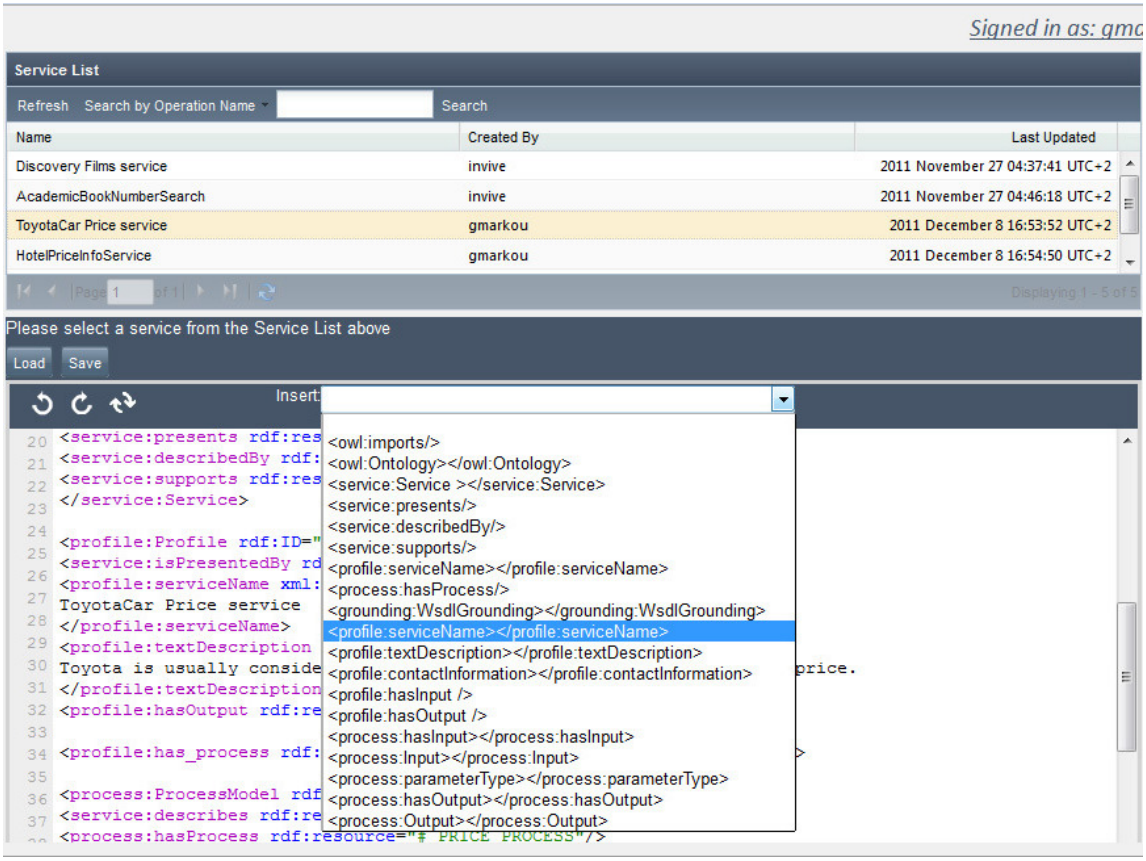

Figure 5.10: Online editor, pre-existing tag templates

5.1.2.3 Automated web service composition module

One of the basic features of the **MADSWAN** system is its ability to solve WSC problems automatically; this feature is restricted for registered users only and is based on the WSs that are available in the registry at any given time. This section discusses the options available to the users concerning the automated WSC module. In specific, users can create WSC problems comprising all the WSs present in the registry or a subset of them that they select, or use a pre-existing problems, as shown in Figure 5.11. For the first two cases, the ontology concepts present in the selected WSs are parsed and divided into inputs and outputs, which the users being able to select and form the initial state and goals from the relevant concepts, as shown in Figure 5.12. In order to parse the necessary ontology concepts a translation from the original WS domain to PDDL first occurs.

Our translation process is similar to the one in (Klusch et al., 2005) and based on a modifications of the one in (Hatzi et al., 2011). In contrast to (Hatzi et al., 2011), the translation can now also generate the planning files that comprise non-deterministic actions, in cases where the domain is non-deterministic. The basic elements of this process are the translation of the original OWL-S Service Profile ID to the action's name in the planning domain file; the Service Profile's *hasInput* and

hasPrecondition properties of each atomic web service to the action's preconditions; and the *hasOutput* and *hasEffect* properties of each atomic web service to the action's effects.

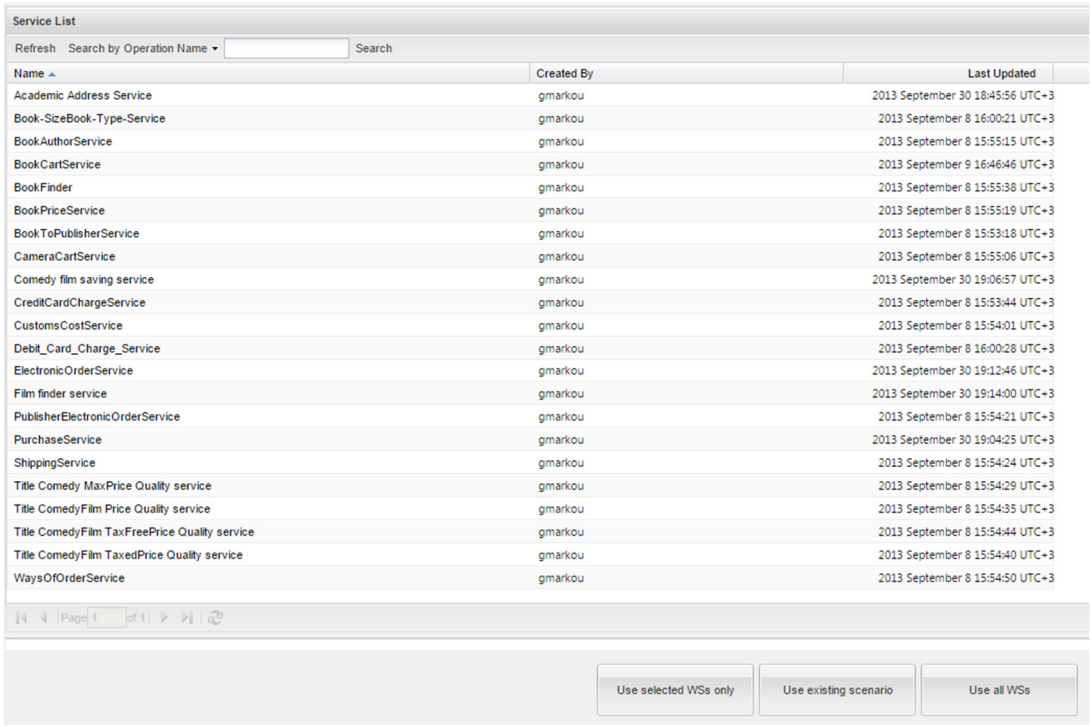

Figure 5.11: Available options for automated web service composition.

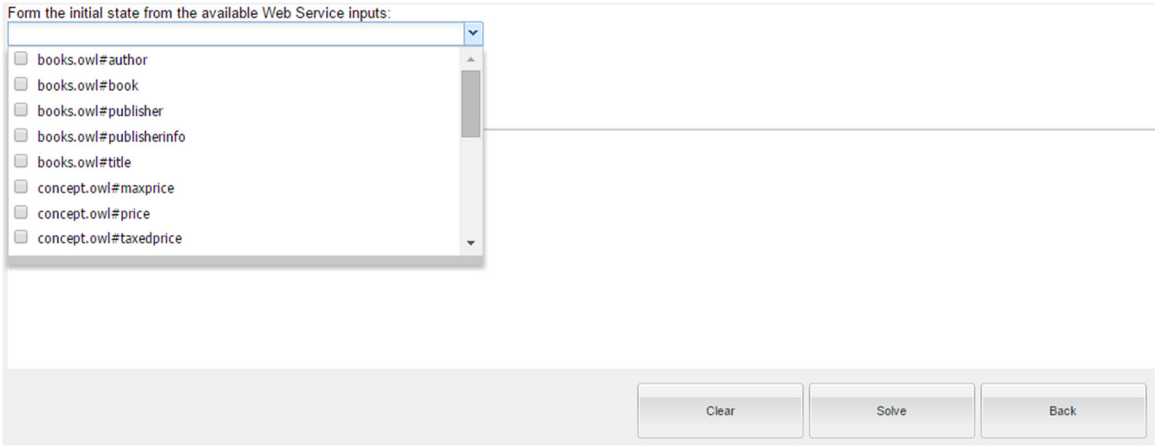

Figure 5.12: Selection of initial state for the web service composition problem.

In case the user opts to solve a pre-existing problem, he can choose between problems that have been previously stored in the system by other users, as shown in Figure 5.13; users are only shown the initial state and the goals of the problem, in terms of the ontology concepts used, and then the WSC algorithm solves the selected problem from scratch. The methodology and algorithms used for the WSC process will be described in Chapter 6. Each user can create and solve a problem in the manner shown above and, after it has been solved, he has the option to store it in the system for future use.

The same holds for the solution of the problem, that is, users can store the output composite WS in OWL-S format in the **MADSWAN** registry. In order to implement the translation from the planning solution back to OWL-S, the approach we followed is based on (Ziaka et al., 2011). A sample solution for a deterministic problem is shown in Figure 5.14, and a part of the composite OWL-S WS that is stored in the registry is shown in Figure 5.15 (specifically one of the atomic processes that comprise it).

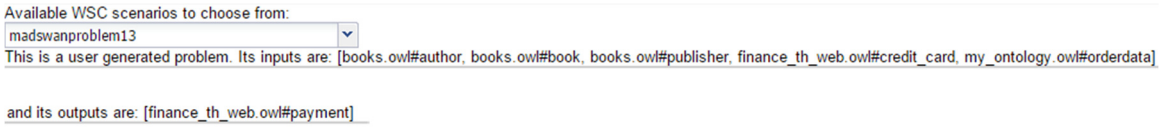

Figure 5.13: Selecting pre-existing web service composition problem to be solved.

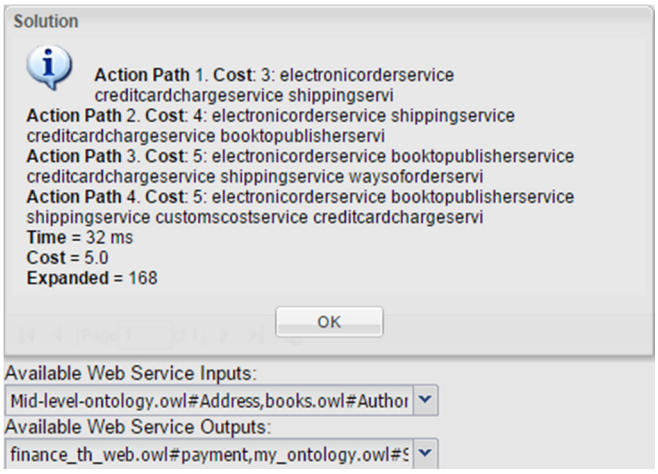

Figure 5.14: Sample solution for deterministic web service composition problem.

```
<j.0:CompositeProcess rdf:ID="CompositeProcess">
    <j.0:composedOf>
        <j.0:Perform>
            <j.0:process>
                <j.0:AtomicProcess rdf:about="#http://195.251.210.239:8080/myuploads/creditcardchargeservice.owls">
                    <j.0:hasInput>
                        <j.0:Input rdf:about="#http://195.251.210.239:8080/myuploads/credit_card">
                            <j.0:parameterType rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#anyURI"
                            >http://195.251.210.239:8080/myuploads/credit_card</j.0:parameterType>
                        </j.0:Input>
                    </j.0:hasInput>
                    <j.0:hasInput>
                        <j.0:Input rdf:about="#http://195.251.210.239:8080/myuploads/orderdata">
                            <j.0:parameterType rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#anyURI"
                            >http://195.251.210.239:8080/myuploads/orderdata</j.0:parameterType>
                        </j.0:Input>
                    </j.0:hasInput>
                    <j.0:hasOutput>
                        <j.0:Output rdf:about="#http://195.251.210.239:8080/myuploads/payment">
                            <j.0:parameterType rdf:datatype=
                            "http://www.w3.org/2001/XMLSchema#anyURI"
                            >http://195.251.210.239:8080/myuploads/payment</j.0:parameterType>
                        </j.0:Output>
                    </j.0:hasOutput>
                </j.0:AtomicProcess>
```

Figure 5.15: Part of a sample composite OWL-S web service output.

5.1.3 Assessment Based on the Pre-Defined Requirements

This sub-section presents an analysis of the **MADSWAN** application and its automated WSC module in regard to the requirements, as they were proposed in Table 4.1; in summary, the implemented application fulfills all of the mandatory requirements, as well as all but one of the optional ones. As to the general requirements, the user chooses both the initial state and the goals through the available ontology concepts that comprise the selected WSs (requirement 1). The system then solves the created problem and returns a solution comprised from subset of the selected WSs, if one exists. The correctness of the solution is automatically validated by checking that all of the ontological concepts that were present in the goals have been generated by the WSs in the plan (requirement 2); moreover, the user can opt to manually design and solve a WSC problem, through the use of the manual WSC module, as it is presented in Section 5.2.2 (requirement 3). The implemented system is open source (requirement 4), any modules it uses or modifies are also open source, and it is not platform dependent (requirement 5) as it is web-based and only requires a compatible browser to be accessed. Moreover, as we require a user of the system to be familiar with basic WSs' concepts as well as with the basics of OWL-S, but not to understand either the specifics of OWL-S or the WSC algorithms used, we only assume an intermediate user skill level to be required (requirement 6);

In regard to the composition elements requirements, all the WSs in the registry are either part of the OWL-S TC collection or a modified version of a WS in it, and as such are described in OWL-S, fulfilling the composition elements requirements 1, 6 and 7. Moreover, users select among their IOPEs to create the WSC problem, which can be solved automatically generating a solution that comprises WSs and their interactions and can be translated to a composite OWL-S WS (composition elements requirements 3 and 4). The composite OWL-S WS implies the use of OWL-S constructs to control the flow of data between the various atomic/composite WSs that comprise it (control flow requirement 1 and data flow requirements 1 and 2), and since the participating WSs are OWL-S their data is typed, based on OWL ontologies (data model requirements 1-3). However, for the time being, the planner does not make use of reasoning to compute the inferred relationships between the domain ontologies' concepts. As such, the optional data model requirement 4 is not yet fulfilled. Finally, since the result of the WSC process can be an OWL-S WS, it can be consequently stored in the **MADSWAN** registry and used in future WS compositions (composition elements requirement 5).

Table 5.1 presents the aforementioned results in regard to whether the implemented system complies with the pre-defined requirements or not. "✓" suggests that the system complies with the requirement and "✗" that the system does not comply.

Table 5.1: System requirements and whether or not the web based registry complies with them

General Composition Requirements			Complies
General Requirements			
1	The end user provides the goal of the composition	Mandatory	✓
2	The WSC process should fulfill the user's request goal	Mandatory	✓
3	WSC can be either fully automated or manual	Mandatory	✓
4	The final system should utilize existing open source components	Optional	✓
5	The final system should be platform independent	Optional	✓
6	The final system should require an intermediate user skill level	Optional	✓
Composition Elements Requirements			
1	The functionality of WSs is described semantically	Mandatory	✓
2	Services have IOPES	Optional	✓
3	The composition comprises services and their interaction	Mandatory	✓
4	The output of the composition is a SWS itself	Mandatory	✓
5	Newly created composite SWSs can be part of the composition	Optional	✓
6	SWSs are described using OWL-S	Optional	✓
7	The test collection used to implement the use case scenarios is OWL-S TC4	Optional	✓
Control Flow Requirements			
	Use of OWL-S control constructs	Mandatory	✓
Data Flow Requirements			
1	WSs have data interactions	Mandatory	✓
Data Model Requirements			
1	Data elements are typed	Mandatory	✓
2	An ontology is used to define the type of data element types	Mandatory	✓
3	The language used to define ontologies that indicate the data element types is OWL	Mandatory	✓
4	The planner realizes the hierarchy defined by the ontologies	Optional	✗

5.2 Online Manual Web Service Composition Editor

This section describes the online manual WSC module that has been incorporated in the implemented system, extending an existing open source application. The module cooperates with the registry described in Section 5.1 so as to retrieve the currently stored WSs in the system, and allows the users to manually create complex composite WSs based on OWL-S constructs.

5.2.1 Basis of the Online Editor - Petals BPM

The second major module of **MADSWAN**, which concerns the manual web service composition functionality, is based on Petals BPM (EBM Websourcing, 2014), an existing BPMN 2.0 modeler. Petals BPM was developed in order to address the business process design issues encountered in a large number of organizations and has been used in several research projects, among which are Synergy (SYNERGY, 2011), CHOReOS (CHOREOS, 2011) and ISTA3 (PetalsLink, 2011). Some of its main features include the following capabilities:

- Creation and editing of BPMN 2.0 diagrams.
- Importing and exporting of the created diagrams to a BPMN 2.0 XML file.
- Importing and exporting of the created diagrams to a XML Process Definition Language (XPDL) format (The Workflow Management Coalition, 2012) 2.1 file.
- Exporting of the created diagrams to a BPEL 2.0 file.
- BPMN Syntax validation

Petals BPM is distributed under the open source license AGPL (Affero General Public License) (Free Software Foundation, Inc, 2007) and as such, it is allowed to use and edit its code freely under its terms and conditions. It has been implemented using GWT, and for that reason it was possible to integrated it with **MADSWAN**. It should be noted that, although many of Petals BPM's available features are stable, it is still in an early development phase.

5.2.2 Implemented Application

5.2.2.1 Support for OWL-S constructs

All the elements of Petals BPM that referred to BPMN 2.0 were removed, particularly the various options to create BPMN 2.0 projects, e.g., Executable Processes or Descriptive Collaboration Projects. Moreover the original BPMN 2.0 palette that comprised of BPMN 2.0 concepts, such as swimlane objects (e.g., collapsed pools, lanes, etc), text annotation artifacts, or BPMN specific events (e.g., Intermediate Throwing Message events or Intermediate Catching Timer events), was replaced by a set of OWL-S constructs and a few "helper" constructs that are used to provide a more intuitive interface. In detail, the OWL-S constructs currently supported by the application are distinguished in five categories: Activities, Events, Control Constructs, Connectors and Data. Each category contains the following elements:

- Activities
  - Generic Task
  - Web Service Task
- Events
  - Start
  - End
- Control Constructs
  - If-Then-Else Gateway
  - Split+Join Gateway
  - End Split+Join
  - Repeat-While Gateway
  - End Repeat-While
- Connectors
  - Regular Sequence Flow
  - If Sequence Flow
  - Else Sequence Flow
- Data
  - Input Data
  - Output Data

The ⟨Sequence⟩ control construct is expressed through the use of regular sequence flows, while each of the ⟨If-Then-Else⟩, ⟨Split+Join⟩ and ⟨Repeat-While⟩ control constructs is represented with a gateway of the control constructs category; the latter two are only used along with an end "helper" construct, i.e., the ⟨End Split+Join⟩ and ⟨End Repeat-While⟩ "helper" constructs are used in conjunction with the regular ⟨Split+Join⟩ and ⟨Repeat-While⟩ constructs in order to enclose elements between them. WSs are represented either through Web Service Task activities that are necessarily bound to specific WSs already existing in the **MADSWAN** registry, or Generic Tasks that can represent abstract, unbound services. Their necessary inputs and outputs are represented with the two respective elements that comprise the Data category. Other helper constructs are the dedicated ⟨If⟩ and ⟨Else⟩ sequence flows that can only be used along with an ⟨If-Then-Else⟩ gateway. Moreover, there are ⟨Start⟩ and ⟨End⟩ constructs that are necessary to signify the beginning and end of a workflow, respectively. That is, with the exception of input data that can precede a ⟨Start⟩ construct to signify that the input is valid for the whole workflow, no other construct can be used before a ⟨Start⟩ construct or after an ⟨End⟩ one. The available OWL-S elements along with their graphical representation can be seen in Figure 5.16.

Figure 5.16: The available OWL-S elements

5.2.2.2 GUI details

Having presented the OWL-S elements that form the basis of the application, this section discusses the application's GUI. Specifically, the existing GUI has been simplified, in order to suit the needs of **MADSWAN**'s target users, who are considered to be familiarized with basic concepts of the WSC domain as well as OWL-S, but are non-expert users, nonetheless. Furthermore, since the application was built as a standalone one , whereas the aim was to integrate it in the existing **MADSWAN** web-based application, a stripped down, more basic version has been created , with certain graphical elements removed or unified with others.

A second important feature that was implemented in order to simplify the use of the system is a type of color coding of the graphical elements that comprise the WS workflows. Thus, every element that represents the beginning of an event/construct has a specific color, i.e., green, and the same holds for the end of such events (with those elements being colored orange). For this reason, If-Then-Else, Split+Join, Repeat-While graphical elements that signify the beginning of the ⟨If-Then-Else⟩, ⟨Split+Join⟩, ⟨Repeat-While⟩ control constructs, respectively, as well as Start, If sequence flow and Data input ones are colored green. On the other hand, End Split+Join and End Repeat-While graphical elements, along with End, Else sequence flow and Data output ones are colored orange. This feature is apparent in Figure 5.17.

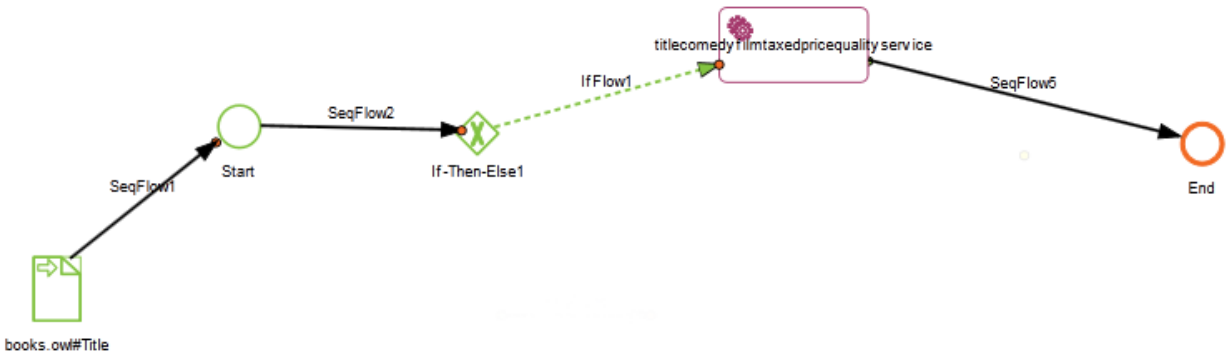

Figure 5.17: Color coding of graphical elements start / end

5.2.2.3 Constructs' properties

Each of the constructs that comprise the OWL-S palette has specific properties and attributes that refer to it, either in regard to its graphical use, e.g., every construct of the gateway type should have at least an incoming and one outgoing sequence flow to be used correctly, or its use as an OWL-S element, e.g., the fact that a Web Service Task must have a binding to an actual WS in order to be used in a valid OWL-S description. As such, it was important to program such properties for each of the constructs based on (Martin et al., 2004), so that rules regarding their use could be established.

Thus, in the application each construct has a unique name by default; these default names cannot be changed by the users for all the elements in the Events, Control Constructs or Connectors categories. The default name for Generic Tasks can either be left as is, or changed to anything the user desires as they represent abstract elements. Web Service and Data elements (Input/Output ones) must be bound to WSs and ontological concepts already in the **MADSWAN** registry. For this reason, the Web Service Tasks' name list box in the properties tab of the application is automatically populated with the names of the WSs that are present at any time in the registry's Service List tab (and is bound to it when the user selects it). The same holds for the name list boxes in the data input and outputs elements, which are populated with the set of ontological concepts (either inputs or outputs) that exist in the registry. These features can be seen in Figures 5.18 and 5.19.

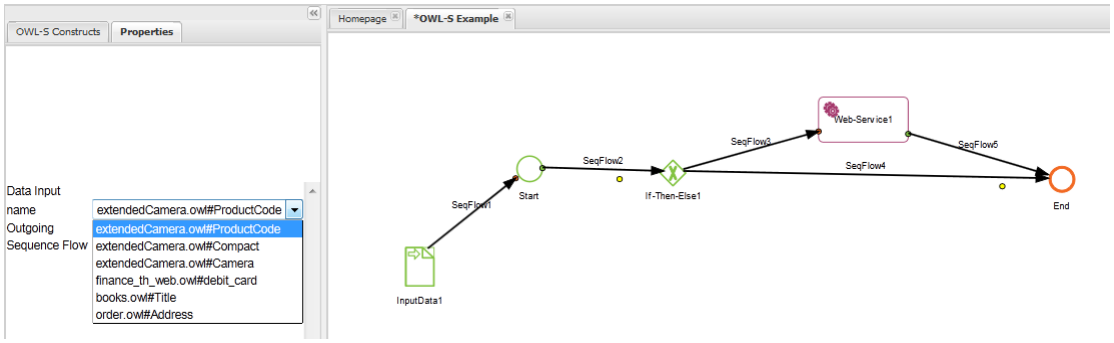

Figure 5.18: Data binding in the manual editor

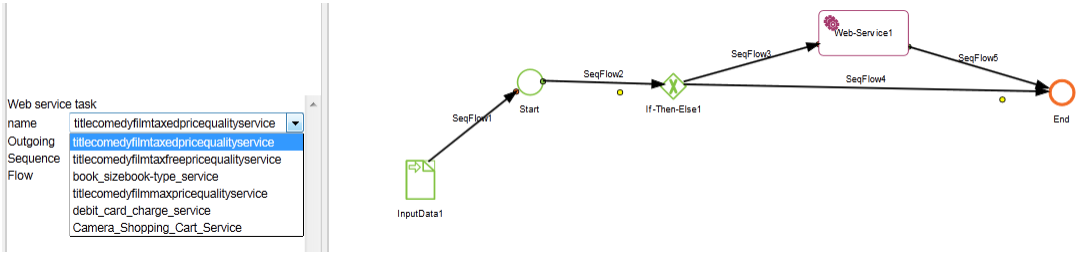

Figure 5.19: Web service binding in the manual editor

5.2.2.4 Preemptive help

As per the requirements of Table 4.1, the users of **MADSWAN** should use it without being experts in WSC. Since each of the constructs that comprise the OWL-S palette has specific properties and attributes, it is not expected from such users to be familiar with all of these properties. For this reason, during the manual creation of the workflows users are guided in regard to the correct use of the available constructs, whenever they input one in a workflow (having the choice, however, to disable such notifications if they don't need them). These messages are closely related to the rules that will be presented in the next sub-section (Section 5.2.2.5). However, Figures 5.20-5.21 present two examples of such messages; the former refers to the insertion of a Start event in the workflow, notifying the user that only one of these events is allowed to be in any workflow, and that all workflows must have both a Start and an End event in order to be valid. The latter refers to the insertion of a data input construct, notifying the user that leaving the default name of the object unchanged is not allowed and that it is required to bind such an object to an ontological concept in the **MADSWAN** registry.

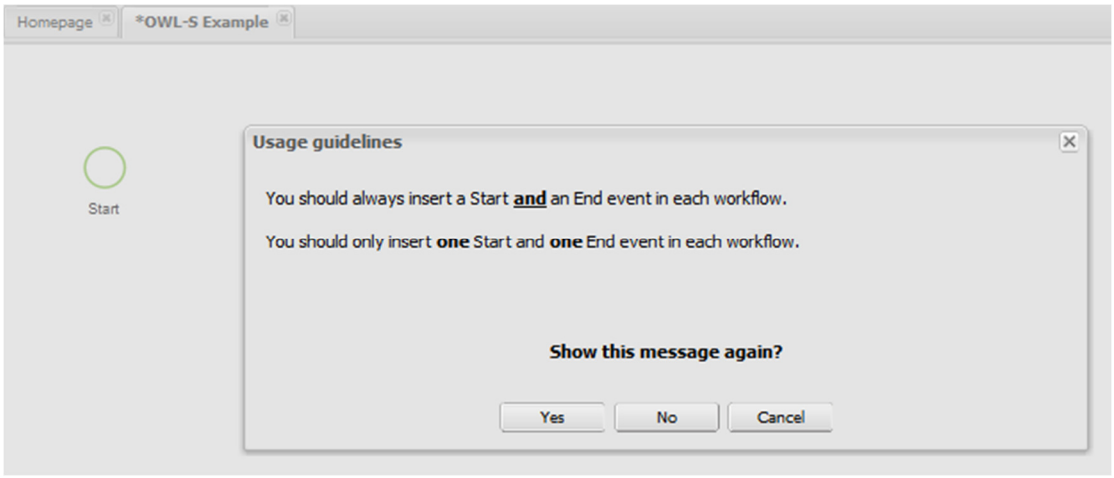

Figure 5.20: Preemptive notification in regard to the insertion of a Start event

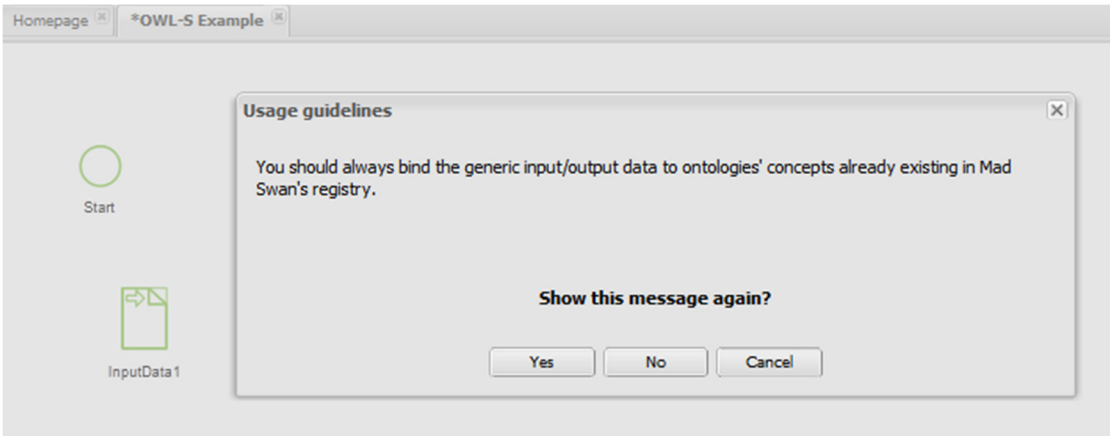

Figure 5.21: Preemptive notification in regard to the insertion of a Data Input construct

5.2.2.5 Validation rules

In the previous section we referred to the preemptive notifications that are presented to the users when they insert an element in the workflow. However, it is not expected that the users will always be able to create valid workflows based solely on that help. As such, the validation module has been extended in order to take into account the properties of the OWL-S constructs that have been implemented for the manual WSC editor. Rules that only applied to BPMN have been removed and new rules have been created to suit our purposes. These new rules are shown in Table 5.2, whereas some of the rules of the original application that were kept from the original application are presented in Table 5.3.

Table 5.2: New validation rules

Rule:	Description:
1	A workflow must have exactly one Start Event
2	A workflow must have exactly one End Event
3	A workflow with a ⟨Split+Join⟩ control construct must also contain an ⟨End Split+Join⟩ one
5	A workflow with an ⟨End Split+Join⟩ control construct must also contain a ⟨Split+Join⟩ one
6	A workflow with a ⟨Repeat-While⟩ control construct must also contain an ⟨End Repeat-While⟩ one
7	A workflow with an ⟨End Repeat-While⟩ control construct must also contain a ⟨Repeat-While⟩ one
8	Any Data construct that is inserted must have its binding set
9	Any Web Service Task construct that is inserted must have its binding set
10	A Data Input cannot have any input connectors
11	A Data Input must have an outgoing connector
12	A Data Output can only be present following an Activity or an ⟨If-Then-Else⟩ gateway if the gateway has an Activity as its source
13	A Data Output must have an incoming connector
14	An ⟨If⟩ Sequence Flow can only have an ⟨If-Then-Else⟩ gateway as its source
15	An ⟨Else⟩ Sequence Flow can only have an ⟨If-Then-Else⟩ gateway as its source
16	A Start Event cannot have incoming connectors except if they originate from a Data Input
17	Only Web Services' Tasks and their Data Outputs can be contained between a ⟨Split+Join⟩ control construct and an ⟨End Split+Join⟩ one
18	An ⟨If-Then-Else⟩ gateway must have exactly one outgoing ⟨If⟩ Sequence Flow and an optional ⟨Else⟩ Sequence Flow

The new rules that were added regard both the correct use of the OWL-S constructs and the general correct creation of the workflow itself. Rules such as that "*a Data Input cannot have any input connectors*" (rule 10 in Table 5.2), or that "*a Start Event cannot have incoming connectors except if they originate from a Data Input*" (rule 16 in Table 5.2), regard the former. Rule 10 requires that a data input is not produced by any other element but only serves as an input to others. Rule 16 requires that a Start Event is always the first element in a workflow except if a data input precedes it, implying that the input is valid for any Activity in the workflow (and not a specific one).

Table 5.3: Rules from the original application that were not modified

Rule number:	Description:
1	A workflow cannot be empty
2	A connector (Regular/ If/ Else Sequence Flow) must have a source and a target
3	A workflow must start with a Start Event
5	A workflow must end with an End Event
6	A connector cannot have the same source and target
7	An Activity (Generic/ Web Service Task) must have at least one incoming or one outgoing connectors
8	A Control Construct must have at least one incoming or outgoing connectors
9	A Start Event must have at least one outgoing Sequence Flow
10	An End Event must have at least one incoming connector
11	An End Event cannot have outgoing connectors

Rules such as that "*only Web Services' Tasks and their Data Outputs can be contained between a ⟨Split+Join⟩ control construct and an ⟨End Split+Join⟩ one*" (rule 17 in Table 5.2) or that "*A Data Output can only be present following an Activity or an ⟨If-Then-Else⟩ gateway if the gateway has an Activity as its source*" comprise the latter category (rule 12 in Table 5.2). Rule 17 stems from the definition of the ⟨Split+Join⟩ control construct in OWL-S, that a "*Split+Join completes when all of its components processes have completed*" (Martin et al., 2004), requiring that only processes, i.e., Activities, can be part of it. Rule 12 aims to impose the fact that a data output is always produced by Activities, which can only be the case if they are direct outputs of one, or if they are the result of an ⟨If⟩/⟨Else⟩ Sequence Flow that originates from one.

An example of the use of the validation rules is shown in Figure 5.22. This simple workflow illustrates an ⟨If-Then-Else⟩ gateway that has two alternatives, one leading to the execution of a Web Service Task and one to not take any action at all. However, based on the rules presented in Tables 5.2 and 5.3, the workflow suffers from the three errors shown in Figure 5.23.

Specifically, the errors reported are related to rules 8, 9 and 18 in Table 5.2. Neither the Web Service Task nor the Data Input inserted in the workflow are bound to a specific WS/ ontological concept present in the **MADSWAN** registry. Those constructs still hold their default names, that is, "InputDataX" and "Web-ServiceX" (X being a unique number assigned to each element of its category). Moreover, the connectors used in conjunction with the ⟨If-Then-Else⟩ gateway are not the dedicated ⟨If⟩ /⟨Else⟩ Sequence Flows, but the regular sequence flow ones.

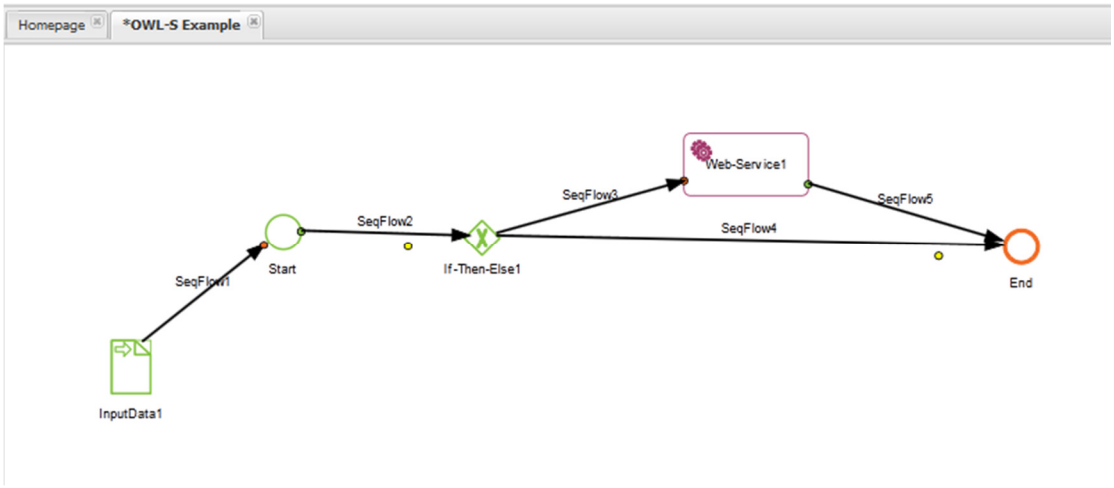

Figure 5.22: Incorrect OWL-S workflow

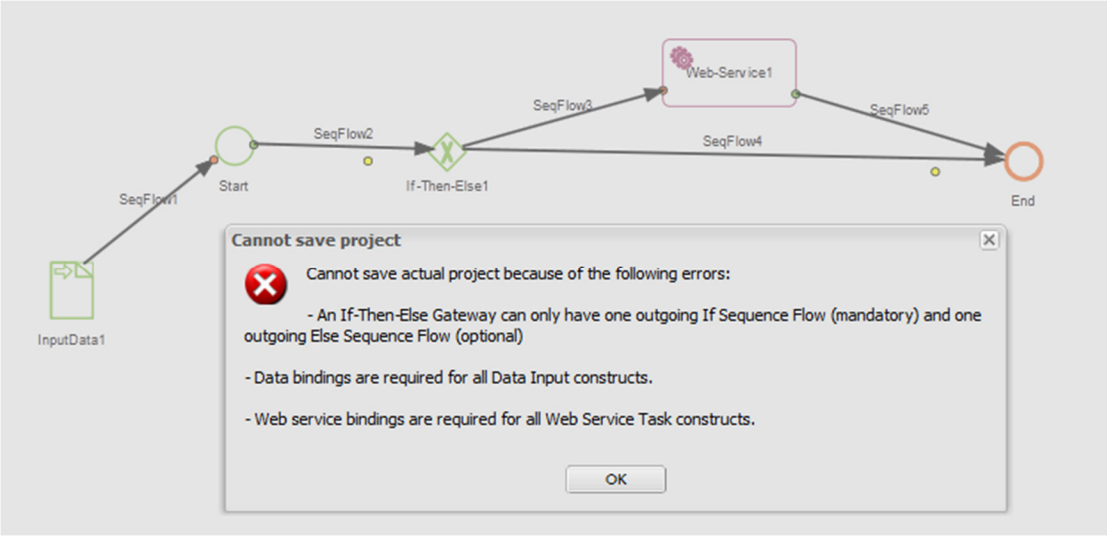

Figure 5.23: Validation errors for the OWL-S workflow shown in Figure 5.22.

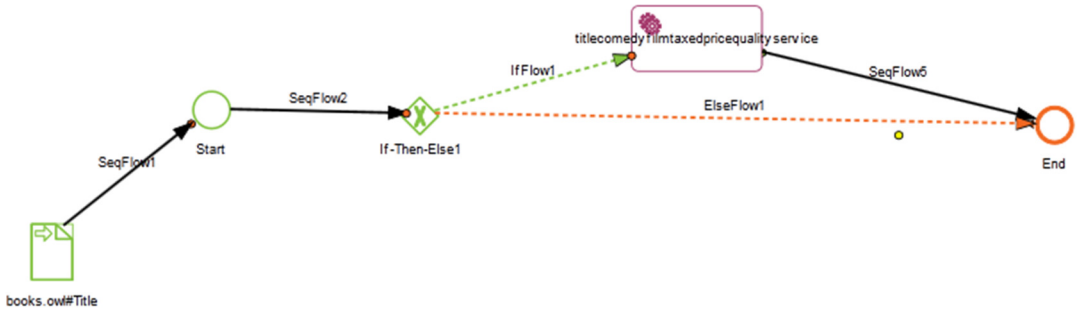

Figure 5.24: Correct OWL-S workflow for the example shown in Figure 5.22.

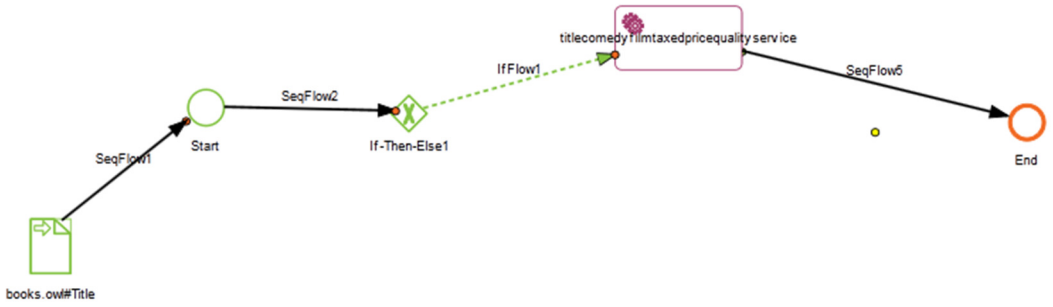

Figure 5.25: Alternative correct OWL-S workflow for the example shown in Figure 5.22.

Figures 5.24 and 5.25 illustrate two alternative workflows that comply with the pre-defined rules. Both have set bindings for the Data Input (specifically, the *books.owl#Title* ontological concept) and the Web Service Task (the *titlecomedyfilmtaxedpricequalityservice* WS) and use the dedicated conditional sequence flows. However, since the else clause of the ⟨If-Then-Else⟩ gateway simply leads to no action taken, the ⟨Else⟩ Sequence Flow can also be removed from the workflow (the specification does not require for both clauses to be defined).

Finally, since Petals BPM was in an early development phase and is not yet completely ready to be used in a production environment, we made efforts to fix several bugs found in the application. In collaboration with the developers of the application, several bugs were reported/corrected such as ones that refer to the internal storage of the workflows, with the application not being able to find some of the elements' sources or edges unless the user first exported the workflow; a bug in relation to the names of the saved projects that prevented them from displaying correctly if the same project was closed and loaded again; a problem in the loading of saved projects, with some of the connectors being displayed as distorted and zigzagged when they should have appeared as straight lines; an error in the display of the correct creation/ modification date of the saved projects, and finally a bug that appeared when a project was saved and loaded that did not allow for the successful save of a workflow if an element was deleted from the original one.

5.2.3 Analysis Based on the Pre-Defined Requirements and Use Case Scenarios

This section presents an analysis of the manual WSC module in regard to the requirements presented in Table 4.1. As to the general requirements, the implemented module fulfills all of them. The user can manually (requirement 3) design a WSC process that fulfills his request goal (requirement 2); the implemented system is open source (requirement 4) and the modules it uses, namely Petals BPM, are also open source; and it is not platform dependent (requirement 5) as it is web-based and only requires a compatible browser to be accessed. Moreover, it requires an intermediate user skill level

(requirement 6); a user of the system should be familiarized with basic WSs' concepts as well as with the basics of OWL-S. Towards that goal, we have implemented preemptive notifications, help tips and color coding of the graphical elements, as well as validation rules that prevent the users from creating invalid workflows.

In regard to the composition elements requirements, as well as the ones related to control and data flow: The implemented application is used to create workflows that comprise of OWL-S constructs, WSs and several helper graphical elements. As such, it complies with requirement 1 of the composition elements requirements. Moreover, the data used in the workflows must be bound by the users during the creation of the workflows with semantically annotated ontological concepts present in the **MADSWAN** registry. The bound data can then be used as IOPEs for the services (composition elements requirement 2), which can themselves be bound to existing OWL-S WSs in the registry; the registry itself has been populated with the entire OWL-S TC v.4.0 (composition elements requirements 3 and 7). The application uses OWL-S constructs to create the workflows (composition elements requirements 1 and 6, control flow requirement 1). However, currently the created workflows are not being translated to OWL-S description files; as a consequence, the result of the manual WSC cannot be imported back into the registry as a SWS. On the other hand, newly created composite WSs from the automated WSC module can be used at a later time in the manual module, having been stored locally in the registry. As such, composition elements requirement 4 is currently not supported, while composition elements requirement 5 is only partially supported. The workflows use connectors and the OWL-S constructs in order to specify the order in which the services are executed and the order and way that the data contained in the workflow is exchanged (data flow requirements 1 and 2).

In relation to the data model requirements, the module abides to the requirements 2 and 3, as all the data inputs/outputs in the workflows are bound to specific ontological concepts (data model requirement 2) that are based on OWL (data model requirement 3). However, the implemented application does not comply with requirements 1 and 4. Specifically, since this is a manual composition editor, the module is based solely on the user to identify which WSs and data inputs/ outputs are relevant and valid for the specific task /composition and cannot identify if two concepts are subclasses of another or if they are equivalent or not.

Table 5.4 presents the aforementioned results in regard to whether the implemented system complies with the pre-defined requirements or not. "✓" suggests that the system complies with the requirement, "✗ - *Relies on the user*" that the system does not comply, but this is due to its nature as a manual module and this requirement is to be achieved by the user's actions.

In summary, the implemented manually web service composition module complies with the majority of the composition requirements, whether optional or mandatory, with the exception of two data model requirements and two composition elements requirements; the former lack of support is due to the system being meant as a manual module, that is, relying on the user to compose the services without providing any intelligence as to the actual WS composition. The latter remains for future work.

Table 5.4: System requirements / manual WSC module compliance with them.

General Composition Requirements			Complies
General Requirements			
1	The end user provides the goal of the composition	Mandatory	✓
2	The WSC process should fulfill the user's request goal	Mandatory	✓
3	WSC can be either fully automated or manual	Mandatory	✓
4	The final system should utilize existing open source components	Optional	✓
5	The final system should be platform independent	Optional	✓
6	The final system should require an intermediate user skill level	Optional	✓
Composition Elements Requirements			
1	The functionality of WSs is described semantically	Mandatory	✓
2	Services have IOPES	Optional	✓
3	The composition comprises services and their interaction	Mandatory	✓
4	The output of the composition is a SWS itself	Mandatory	×
5	Newly created composite SWSs can be part of the composition	Optional	× - Partially supported
6	SWSs are described using OWL-S	Optional	
7	The test collection used to implement the use case scenarios is OWL-S TC4	Optional	✓
Control Flow Requirements			
	Use of OWL-S control constructs	Mandatory	✓
Data Flow Requirements			
1	WSs have data interactions	Mandatory	✓
Data Model Requirements			
1	Data elements are typed	Mandatory	× - Relies on the user
2	An ontology is used to define the type of data element types	Mandatory	✓
3	The language used to define ontologies that indicate the data element types is OWL	Mandatory	✓
4	The planner realizes the hierarchy defined by the ontologies	Optional	× - Relies on the user

The existing implementation of the manual composition module is capable of producing graphical workflows that correspond to the three use case scenarios in Sections 4.2.3-4.2.5. Figures 5.29-5.31 present the manually created workflows representing the movie search, e-bookstore and camera purchase scenarios, respectively. In all figures, the labels of the connecting sequence flows have been removed so as to ease their legibility. The files used by the system's internal load and save functionalities for the three use case scenarios can be found at (Markou, 2014c).

Comparing the intended GUI in Figures 4.2-4.4 with the implemented one in Figures 5.29-5.31 it is evident that they are very similar. However, it is our opinion that the implemented one is more

intuitive and user friendly; the addition of dedicated ⟨If⟩ and ⟨Else⟩ Sequence Flows, as well as of helper elements such as ⟨End Split+Join⟩ and ⟨End Repeat-While⟩ ease the use of the system by providing specific graphical elements that imply the start and end of the OWL-S constructs. This fact is particularly apparent in the use of the ⟨Repeat-While⟩ element in the mockups. Moreover, the color coding used in the final application is helpful in order to prevent errors in the insertion of elements in the workflow, as are the preemptive rules that notify the user as to their correct use. Finally, the use of connectors that can be twisted and take any form help to increase the comprehension of the workflows in relation to the use of simple, straight lines for the connection of the various elements.

As it will be shown in Section 6.2, using the automated WSC module, the solution plan for the tests cases, requires – in all cases – up to only a few seconds; it is apparent that even an experienced user would require at least a few minutes in order to accomplish a similar result visually in the manual WSC module. For that reason, we consider the manual WSC module to mainly be useful for the visualization of WSC workflows, and the creation of a basic "sketch" of a composite WS description, which can then be manually completed by experts.

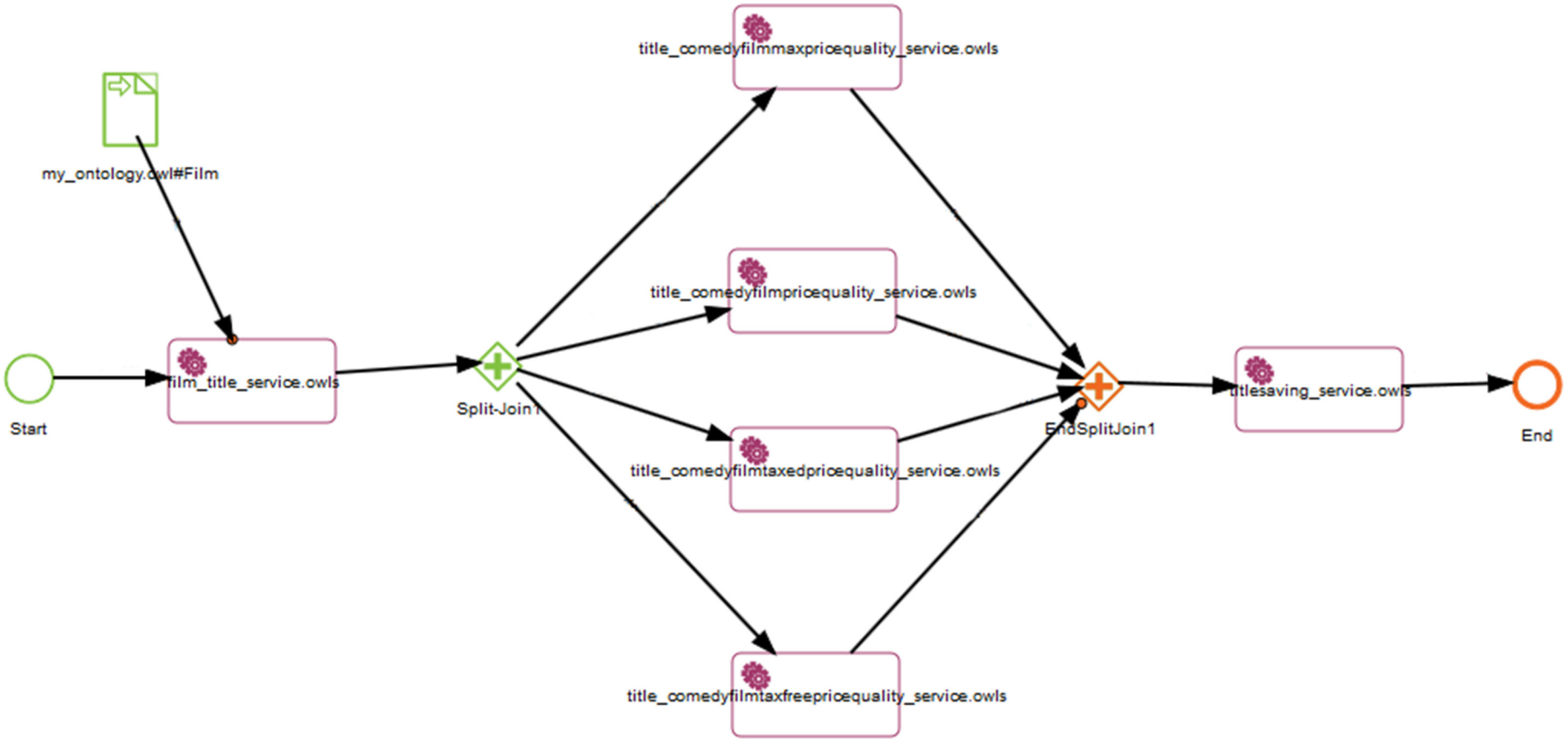

Figure 5.26: Movie database scenario workflow.

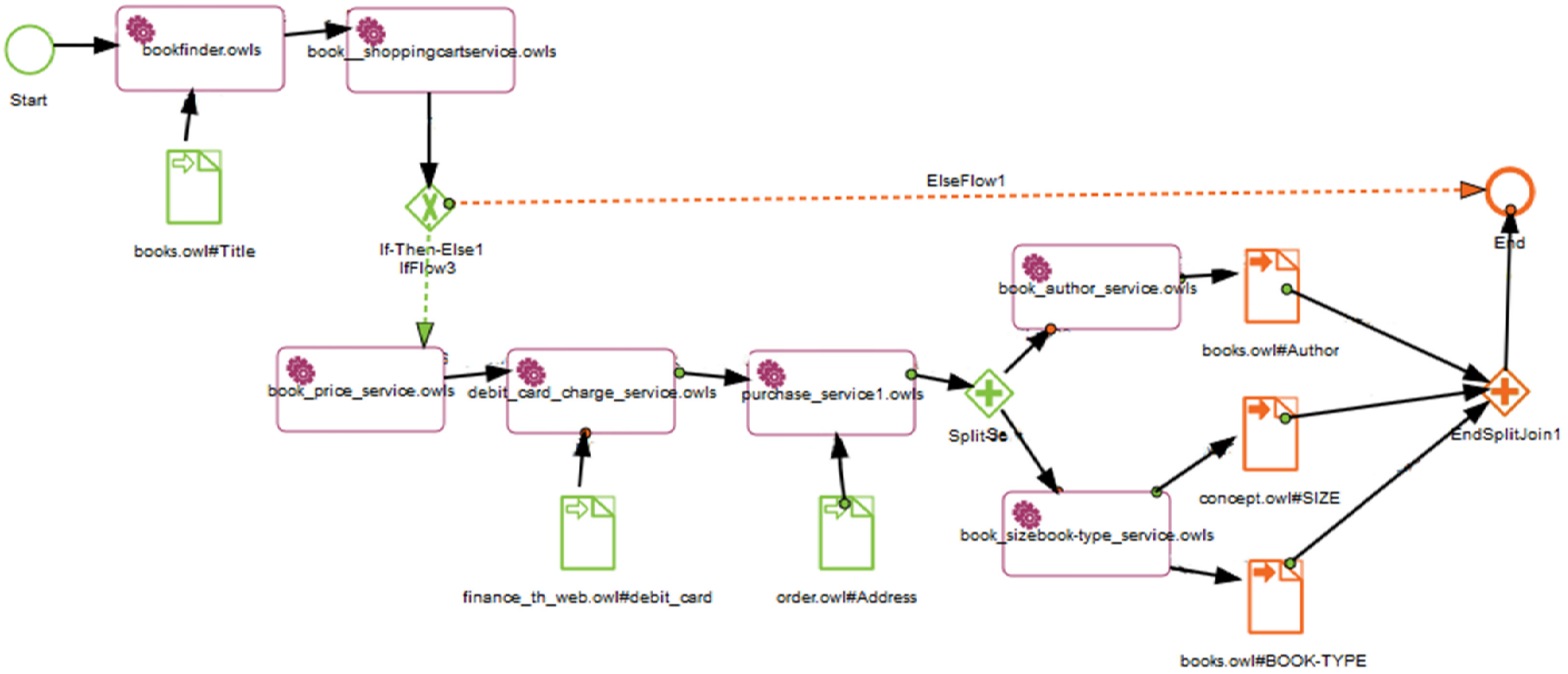

Figure 5.27: Online bookstore scenario workflow.

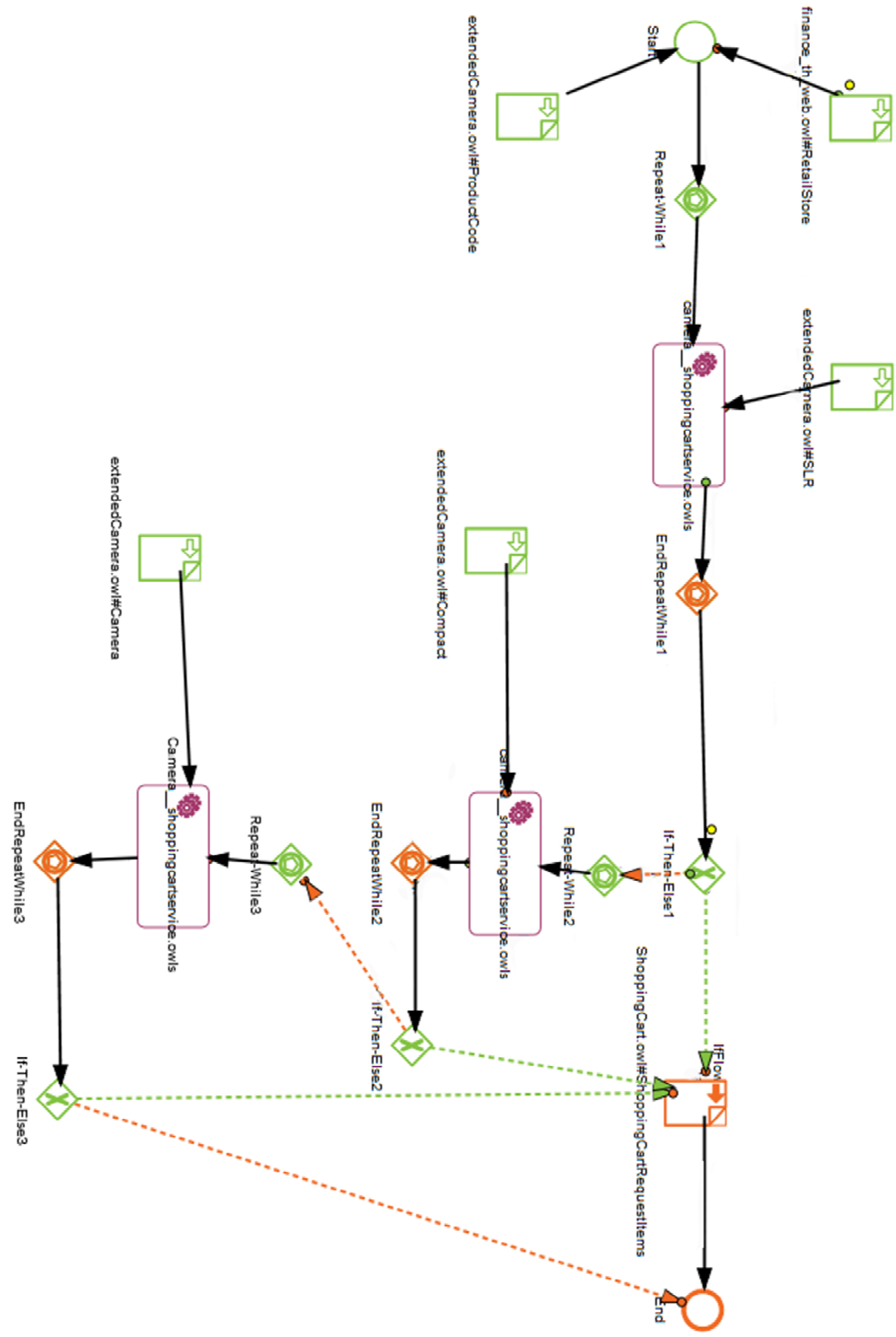

Figure 5.28: Camera search scenario workflow

Chapter 6.

Non-Deterministic Web Service Composition: A Contingent Anytime Approach

Automating the process of composing semantic WSs can lead to great reductions in the time and cost required to develop web-based applications. In chapter 5 it was assumed that web services are deterministic, that is, their outcomes are predictable. Thus, automated web service composition was based on a simple A* algorithm, that resulted in optimal plans in terms of their cost. This chapter assumes that WSs are non-deterministic, that is, they are allowed to have multiple alternative outcomes, each with a probability of occurring attached to it, thus the problem is formulated as a FOP one. Taking into account the nature of the non-determinism inherently in the web service composition problem, the chapter introduces a new algorithm for probabilistic web service composition that results in a good tradeoff between probability of achieving the goals and cost. Additionally, it presents quantitative evaluation results related to both the deterministic and non-deterministic WSC algorithms that are proposed, as well as a case study referring to the application of WSs to the field of intelligent calendars.

6.1 Contingent Anytime Web Service Composition Algorithm

The proposed non-deterministic WSC algorithm, **MAPPPA2** (an anagram of **A**nytime **P**robabilistic **P**lanning via **A**lternative **P**lan **M**erging), is based on an anytime contingent planning framework. Initially, the web service domain is modeled as a probabilistic planning one. Then, a determinized version of it is produced, while retaining the information the original one contains in regard to the probabilities and cost of the web services/actions. The planning problem is heuristically solved repeatedly, until either a time limit is reached, or all non-subsumed plans have been found. Finally, the deterministic plans are merged in a contingent plan, using the initial non-deterministic actions as its decision nodes, under the additional assumption that re-executing the same web service within the current plan execution episode will result to the same output. **MAPPPA2** prioritizes the generation of the alternative plans, as well as their merging into a contingent plan, taking into account their expected utility; however, the overall process does not guarantee that the final contingent plan is optimal in terms of expected utility.

MAPPPA2 is specifically targeted for web service composition problems and differs from related work in that it takes the probabilities and the cost of the original non-deterministic actions into consideration while generating a contingent plan. The expected utility of such a plan monotonically increases as more time is allowed for planning and more plans are generated and added to it.

This work has been inspired by determinization approaches, such as the ones presented in Section 2.3; it differs from related work in that it proposes a contingent algorithm specifically targeted for WSC problems and in that it takes the probabilities and the cost of the original non-deterministic actions into consideration, while generating a contingent plan. We follow the problem formulation that was presented in Section 2.1. Specifically, we define a probabilistic planning domain in the form $D = (S, A, \gamma, Pr, Co)$, where S is a finite set of states, A is a finite set of actions, $\gamma : S \times A \rightarrow 2^S$ is the state-transition function, $Pr : S \times A \times S \rightarrow [0,1]$ is the probability-transition function and $Co : S \times A \times$

$S \to \mathbb{N}$ is a bounded cost-function. Importantly, the definition of the initial state is with respect to closed world semantics.

6.1.1 Alternative Plan Generation and Merging

MAPPPA2 is a cost-sensitive probabilistic anytime contingent planning framework for automated web service composition, consisting of three steps: creating a determinized version of the domain based on the original probabilistic one; generating multiple solutions to the new deterministic problem giving priority to solutions with high expected utility; and, finally, merging these solutions in a single decision tree, which constitutes the contingent plan.

MAPPPA2 is not a generic contingency planning one; it is tailor-made for web service composition problems and specifically for ones where the web services involved do not produce irreversible results. That is, it is assumed that it is always possible from any state in the problem to return to the initial one. This is considered possible through the assumption that the actions in the planning domain do not have delete effects or negated preconditions.

Moreover, it is assumed that once an action has been executed with a specific effect as an outcome, then for the rest of the current plan execution episode it cannot be executed again with a different outcome. This assumption also holds if the action's execution produces undesired effects, i.e., once an action fails, it is assumed that it will always fail in the current plan execution episode. This is a realistic assumption in a WSC domain, as a web service that produces undesirable effects for some reason, e.g., network failure, is not probable to be available again in a very short amount of time. A consequence of this assumption is that the contingent plan is allowed to execute each of the actions at most once in each of its branches; thus, the solutions do not contain infinite branches or cycles.

Consequently, the output (deterministic) solutions are weak ones; each one can only succeed in the case that all the actions in it have the desired effect as their actual outcome. If there are n actions in branch $Plan_i = [a_{1j_1}, a_{2j_2}, \dots, a_{nj_n}]$, the desired outcome j_k of each having a probability of $prob_{\alpha_{kj_k}}$ to actually occur, then each branch has a probability $Prob_{branch} = prob_{\alpha_{1j_1}} \times prob_{\alpha_{2j_2}} \times \dots \times prob_{\alpha_{nj_n}}$ of being executed successfully.

A motivational real world example of such a problem containing web services is the following: Assume a person wants to purchase an item (the problem's goal) from the web using his credit card. Initially, he has available a credit card that has not yet been charged and the product has not been purchased. Using an eBay web service, the user searches for the item among various eBay sellers and finds several ones that have the item in stock. In the case that none of these sellers offers the item in the desired price or ships it to the user's address, the user can return to the initial state, having an available credit card with the goal of purchasing the particular item. He can then choose an alternative action, e.g., using an Amazon web service to search for the item in the Amazon web store.

6.1.1.1 Computing alternative weak plans

MAPPPA2 adopts the all-outcomes determinization approach. The planning process can be based on any search algorithm that returns multiple deterministic plans to the goal. **MAPPPA2** adopts A* with either the max heuristic (h_{max}) or the additive heuristic (h_{add}) (Bonet & Geffner, 2001). Of course, in case the resources (time and memory) are enough to generate all deterministic plans, the selection of the search algorithm or the heuristic does not matter, provided that it is a complete algorithm. A* has been forced to continue finding solutions after the first one is found, whereas it terminates only when either a time limit, or a limit on the number of possible solutions has been reached.

At this point, it is necessary to provide some details about the application of the A*, particularly the computation of the current path cost and the estimate of the cost to the goal, that is, the heuristic value. The overall goal is to create a contingent plan that optimizes the expected utility, which occurs by merging deterministic weak plans. In case the available resources (time and memory) are not sufficient to generate all the weak plans, it is important to generate first the best determinized weak plans, in order to create a contingent plan with the highest possible, given the limited resources, expected utility.

In order to compute the plan's expected utility, it is important to define the utility of achieving the goal of the planning problem. Let U denote this utility. On the other hand, a determinized action's execution incurs some cost. In order to define the expected utility of a weak plan that achieves the goal, denoted by $[a_1, a_2, \dots, a_k]$, with costs $[c_1, c_2, \dots, c_k]$ and probabilities of successful execution $[p_1, p_2, \dots, p_k]$, we adopt the following assumptions: Each determinized action a_i may fail, with probability $(1 - a_1)$. In case an action fails, its cost is not incurred. On the other hand, we do not consider the alternative outcomes of the action, whatever they are, including the case that they may achieve the goal directly. In other words, the expected utility of a weak plan concerns the case where the goal is achieved by the successful execution of all of its actions and not incidentally by any alternative outcome of any plan's action.

A weak solution plan with k actions has $k + 1$ possible outcomes: The k outcomes concern the successful execution of the first $k - 1$ actions and the failure of the k^{th} action, whereas the $(k + 1)^{th}$ outcome concerns the successful execution of the k actions, in which case the goal is achieved. The first k outcomes do not achieve the goal, so they only incur the cost of the actions that have already been executed; the $(k + 1)^{th}$ outcome achieves the goal. So, the expected utility of such a plan is the weighted average of all its alternative outcomes:

$$\mathrm{E}U = \left(\prod_{i=1}^{k} p_i\right)\left(U - \sum_{i=1}^{k} c_i\right) - \sum_{i=1}^{k}\left((1 - p_i)\left(\prod_{j=1}^{i-1} p_j\right)\sum_{j=1}^{i-1} c_j\right) \quad (1)$$

Eq. (1) is a sum of two complex terms. The first term concerns the case where all actions were successfully executed. The second term concerns the case where the first $i - 1$ actions were successfully executed, but action a_i failed. As an example, assume a plan with two actions, $[a_1, a_2]$,

with $c_1 = 2$, $c_2 = 1$, $p_1 = 0.8$ and $p_2 = 0.5$. Assume also that $U = 10$. The expected utility of this plan is:

$$\mathrm{E}U = p_1 p_2 (U - c_1 - c_2) - p_1 (1 - p_2) c_1 = 0.8 \cdot 0.5 (10 - 2 - 1) - 0.8 \cdot 2 = 1.2$$

Note that alternative week solution plans with the same actions, but with a different order on them (if possible), may have different expected utilities. Particularly, for actions that do not have an ordering between them, the expected utility is increased as far as the actions with the highest failure or/and the lowest cost are positioned first. In the aforementioned example, the plan $[a_2, a_1]$ would have the following expected utility:

$$\mathrm{E}U = p_2 p_1 (U - c_2 - c_1) - p_2 (1 - p_1) c_2 = 0.5 \cdot 0.8 (10 - 1 - 2) - 0.5 \cdot 1 = 2.3$$

A* guides its search by the sum of the cost from the initial state to the current state, in addition to the estimated cost from the current state to the goal. In order to compute the current cost of a non-solution plan, Eq. (1) cannot be used. This is because Eq. (1) computes the expected utility of plans that achieve the goal, provided that all their actions are successfully executed. However, partial plans do not yet achieve the goals. On the other hand, using only the second part of Eq. (1) is not satisfactory, since partial plans with very low success rate will be computed to have low cost. To alleviate this problem, we introduce the notion of expected cost for partial plans. Expected cost is defined as the cost that would be incurred if the partial plan was to be executed. This cost originates from the actions of the plan, as well as from failing to achieve the goal, which is equal to the utility that was not gained because the goal was not achieved, that is, U. So, any action a_i successfully executed incurs cost c_i, whereas any failing action incurs cost U. Thus, the expected cost of a partial plan $[a_1, a_2, \ldots, a_k]$ is defined as:

$$\mathrm{E}\,Cost = \sum_{i=1}^{k} \left((1 - p_i) \left(\prod_{j=1}^{i-1} p_j \right) \left(U + \sum_{j=1}^{i-1} c_j \right) \right) + \left(\prod_{i=1}^{k} p_i \right) \left(\sum_{i=1}^{k} c_i \right) \qquad (2)$$

Eq. (2) is a sum of two complex terms. The first term concerns the case where the first $i - 1$ actions were successfully executed, but action a_i failed. The second term concerns the case where all actions of the partial plan were successfully executed, thus not incurring the cost of U. Eq. (2) favors partial plans with high probability of success and low-cost actions. Eq. (1) and (2) are equivalent. Indeed, if they are both applied to a solution plan, their results differs exactly by the constant *U*.

Again, similar to the computation of the expected utility in Eq. (1), plans with different orderings of the same actions may have different expected costs, with the plans having the actions with the highest failure probability or/and the lowest cost earlier in the ordering having lower expected cost. **MAPPPA2**'s A* implementation subsumes identical plans, that is, plans with the same sets but different ordering of actions, by always retaining only the plan with the lowest expected cost.

Eq. (2) can be computed in an incremental way. Assume that Eq. (2) has been applied to a partial plan with k actions resulting in value $ECost_k$. If the plan is extended by another action, a_{k+1}, the expected cost of the new plan, let's say $ECost_{k+1}$, can be computed as:

$$\mathrm{E}\,Cost_{k+1} = \mathrm{E}\,Cost_k + c_{k+1} \cdot \left(\prod_{i=1}^{k+1} p_i \right) + U(1 - p_{k+1}) \prod_{i=1}^{k} p_i \qquad (3)$$

According to Eq. (3), the extra expected cost by appending action a_{k+1} to the partial plan $[a_1, a_2, \ldots, a_k]$ is equal to the probability of action a_{k+1} to succeed times c_{k+1}, plus the probability of action a_{k+1} to fail times U, provided in both cases that all previous actions were successfully executed.

Concerning the estimate of the cost to the goal, **MAPPPA2** employs two well-known heuristics, h_{max} and h_{add}. In order to compute them, the initial probabilities of the determinized actions are ignored, considering them as having 100% probability of success. This leads to a reduction of the cost values returned by the heuristic, thus h_{max} retains its admissibility (h_{add} is not admissible in any case). Thus, with the above definitions of expected cost of the current path and the heuristic estimate of the cost to the goal, A* with h_{max} generates weak plans in decreased order of expected utility (equivalently, increased order of expected cost), whereas A* with h_{add} generates weak plans more efficiently (that is, faster), but without any optimality guarantee.

6.1.1.2 Merging alternative weak plans into a contingent plan

The final contingent plan, having the form of a decision tree, is formed by prioritizing and merging the alternative determinized weak plans. Their prioritization is performed through their expected utility, as it has been defined in the previous section. However, since there are interactions between the alternative plans, continuously selecting the plan with the highest expected utility to be added to the contingent plan does not constitute an optimal strategy. Thus, the resulting contingent plan may be a plan with very high expected utility, however there is no guarantee that it is the optimal plan in terms of expected utility, given a set of alternative weak plans.

In the decision tree's generation function $GenerateDT$ (Alg. 1), the algorithm's input are the alternative determinized weak plans, with their deterministic actions mapped back to their non-deterministic counterparts; these plans are sorted by their expected utility in decreasing order (Alg. 1, line 1) and the first action of the first plan is added to the decision tree, provided that the expected utility of the best plan is non-negative (line 4). In case the current action is non-deterministic, a left branch is added to the current node of the decision tree, which is created by a recursive call to the $GenerateDT$ function, using as an argument the alternative plans that are compatible with the failure of the current action a (lines 9-11). Then, the algorithm iterates over the next actions of the current plan (lines 12-16). Leaf nodes are characterized either by *SUCCESS*, in case they are the last node of a plan, or by *FAILURE*, in case they represent the absence of an alternative plan after a failure of a non-deterministic action (lines 17-19).

It is important to note that in line 4 of the $GenerateDT$ function, only plans with non-negative expected utility are considered. In case there is no available plan with non-negative expected utility, $GenerateDT$ returns a decision tree with a single *FAILURE* node. This does not mean that determinized weak plans with negative expected utility are useless. Such a plan cannot be selected as the best plan in line 4 of $GenerateDT$, however after $FilterPlans$ has been applied to the remaining

plans (lines 10 and 15), which may remove their actions, the expected utility of the remaining actions may become non-negative. In any case, due to line 4, the expected utility of the overall decision tree never decreases by merging new plans, since new branches have always non-negative expected utility, compared to failure branches that have zero utility.

$FilterPlans$ (Alg. 2) is called with a set of plans (the alternative plans for the current branch of the decision tree) and an action as its inputs. The action is either the last determinized action of the current branch of the decision tree, or its negation, with $\neg a$ denoting the alternative determinized action that resulted by the same non-deterministic action as a. $FilterPlans$ performs two tasks: First, it removes the indicated action from each plan it considers. Second, it removes from P any plan that includes the alternative determinization of the indicated action.

$GenerateDT's$ output is a combination of weak solutions; as such, it may produce strong contingent solutions, if all branches lead to *SUCCESS* nodes. However, in general, there is no guarantee that the resulting contingent plan will be strong. Furthermore, for problems that only have strong cyclic solutions, the proposed algorithm is not suitable.

The expected utility of a contingent plan in the form of a decision tree can be computed recursively as follows:

- The expected utility of a *SUCCESS* leaf node is equal to U, whereas the expected utility of a *FAILURE* leaf node is equal to 0.
- The expected utility of a non-leaf node with a non-deterministic action a assigned to it and k of its outcomes having been added in the decision tree, each one of them occurring with probability p_i and incurring a cost c_i, is equal to:

$$\mathrm{E}U = \sum_{i=1}^{k} p_i(\mathrm{E}U_i - c_i)$$

where EU_i is the expected utility of the decision tree that has as root the i^{th} outcome of action a.

Function $GenerateDT$

Input A set of plans P

Output A decision tree containing all the plans

1. $Sort(P, EU())$
2. $p = P \rightarrow first\_plan();\ P = P - \{p\}$
3. $root = new\ DTNode()$
4. **If** ($p \neq$ null and $EU(p) \geq 0$)
5. $a = p \rightarrow first\_action()$
6. $n = root$
7. **While** $(a \neq null)$
8. $n \rightarrow action = a$
9. **If** $non\_deterministic(a)$
10. $n.LeftBranch = GenerateDT(FilterPlans(P, \neg a))$
11. **End If**
12. $a = a \rightarrow next\_action()$
13. $n.RightBranch = new\ DTNode()$
14. $n = n.RightBranch$
15. $P = FilterPlans(P, a)$
16. **EndWhile**
17. $n \rightarrow action = SUCCESS$
18. **Else**
19. $root \rightarrow action = FAILURE$
20. **EndIf**
21. **Return**($root$)

Algorithm 6.1. Function ***GenerateDT*** - Generation of the decision tree.

Function *FilterPlans*

Input A set of plans P and an action a

Output A simplified set of plans than are compatible with a

1. **For** (Plan p in P)
2. **If** a in p
3. delete *a* from *p*
4. **End If**
5. **If** $\neg a$ in p
6. delete *p* from *P*
7. **End If**
8. **Endfor**
9. **Return** *P*

Algorithm 6.2. Function ***FiliterPlans*** - Simplifies plans and removes plans that are not compatible with an action

6.1.2 Simple Contingent Use Case Scenario Plan Example

Section 4.2.6 MS-UC4 was presented , a simple dummy problem tailor-made to test **MAPPPA2**; Figures 6.1-6.6 present the alternative plans, as they are produced by A* with h_{max} for MS-UC4 using the algorithms introduced in Section 6.1.1, whereas Fig. 6.7 presents the contingent plan. The problem consists of a (single) initial state s_0 and the goal state G (each state is represented by a circle). Actions a_2 and a_7 are deterministic (denoted by a straight line), while the rest of the actions are probabilistic (denoted by a dotted line); for each probabilistic action a_i, $i \in \{1,3,5\}$, its (single) effect is achieved with a probability $prob_{\alpha_i} = 0.8, \forall i \in \{1,3,5\}$, and with a probability of $1 - prob_{\alpha_i} = 0.2$, it fails and no effect is achieved (the current state does not change). Actions a_4 and a_6 have two different effects, each having a different probability of being produced when executing the related action, with $prob_{\alpha_{41}} = 0.9$, $\text{prob}_{\alpha_{42}} = 0.1$ and $prob_{\alpha_{61}} = 0.8$, $prob_{\alpha_{62}} = 0.2$.

The cost associated with each action is shown opposite it, e.g., $cost_{\alpha_{11}} = 4$, $cost_{\alpha_7} = 2$. After the all-outcomes determinization process, the actions available to the deterministic planner are $a_{11}, a_{12}, a_2, a_{31}, a_{32}, a_{41}, a_{42}, a_{51}, a_{52}, a_{61}, a_{62}$ and a_7. Of these, a_{12}, a_{32}, and a_{52} are ignored as they do not produce any effect, whereas a_2 and a_7 remain as-is, since they were already deterministic. A* with h_{max} generates all the solutions to the problem, which in order of decreasing expected utility are denoted with $Plan_1$ through $Plan_6$. For this example we assume that U = 100. As such, the alternative plans, with their expected utility, are the following:

- $Plan_1 = [a_{11}]$ (Fig. 3a) with EU = 76.80
- $Plan_2 = [a_{2,}a_{31}]$ (Fig. 3b) with EU = 75.60
- $Plan_3 = [a_{41}, a_{51}, a_{61}, a_7]$ (Fig. 3c) with EU = 50.00
- $Plan_4 = [a_{41}, a_{51}, a_{62}]$ (Fig. 3d) with EU = 7.38
- $Plan_5 = [a_{42}, a_{61}, a_7]$ (Fig. 3e) with EU = 6.52
- $Plan_6 = [a_{42}, a_{62}]$ (Fig. 3f) with EU = 0.6

Fig. 6.7 presents the form of the contingent solution; Circular nodes are chance nodes that lead to alternatives over which the planner has no control, i.e., they denote non-deterministic actions. Triangular nodes are end nodes; either success ones ("S"), or failure ("F") ones. The edges depict the actual executed outcome of the action. Having sorted the plans by their expected utility, $Plan_1$ is inserted first into the decision tree. The successful execution of a_1 is added as the first right branch in the tree. The left branch of a_1 corresponds to its failure, and requires the computation of the possible plans for insertion.

Since no other plan contains a_1, all alternative plans can be inserted. Moreover they all retain their expected utility, thus $Plan_2$ is added; since a_2 is deterministic, it leads to a_3, with a_{31} achieving the goal. In case a_{32} occurs, $Plan_3$ is employed (again with its original expected utility, since none of the actions of the remaining plans has been executed), with a_4 being its first action. The branch with the successful execution of a_4, that is a_{41}, continues with $Plan_4$ as the only alternative plan, whereas the branch with a_{42} continues with $Plan_5$ as the primary plan and $Plan_6$ as alternative. If a_{51} of $Plan_3$ fails, we have a failure node, whereas if a_{61} of $Plan_3$ fails, there is the alternative of a_{62} from $Plan_4$ that achieves the goal. If a_{61} of $Plan_5$ fails, $Plan_6$ with a_{62} continues and achieves the goal; otherwise, $Plan_5$ continues with a_7, which is deterministic and achieves the goal. The final decision tree can be seen in Fig. 6.7; its overall expected utility is equal to 94.71 .

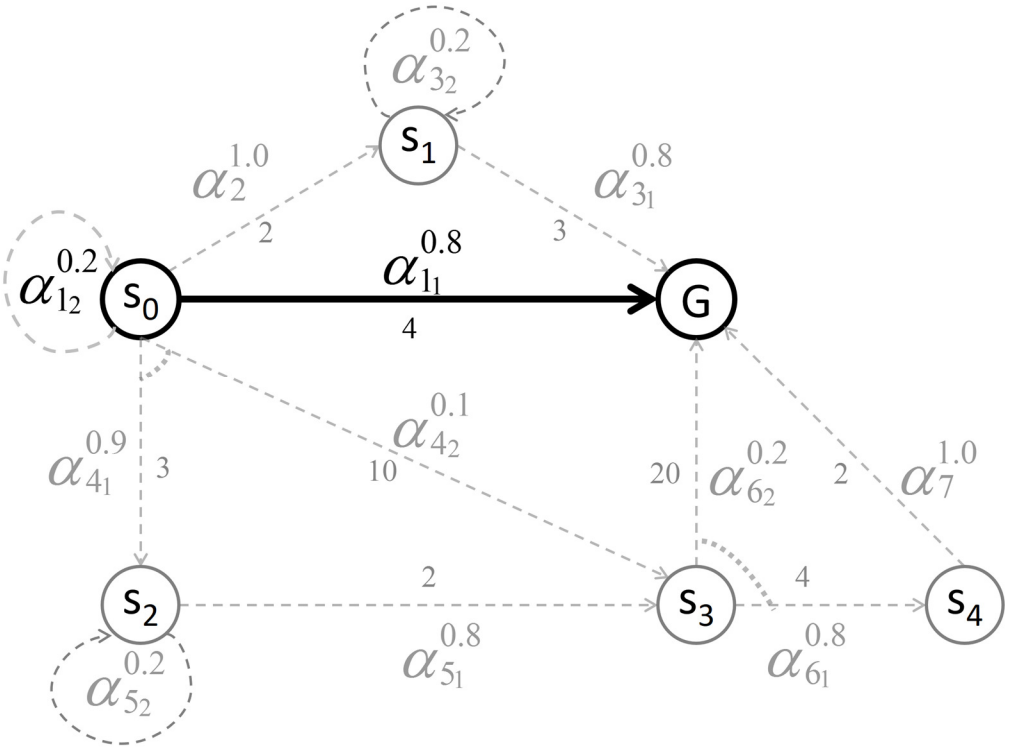

Figure 6.1: $Plan_1$ for MS-UC4.

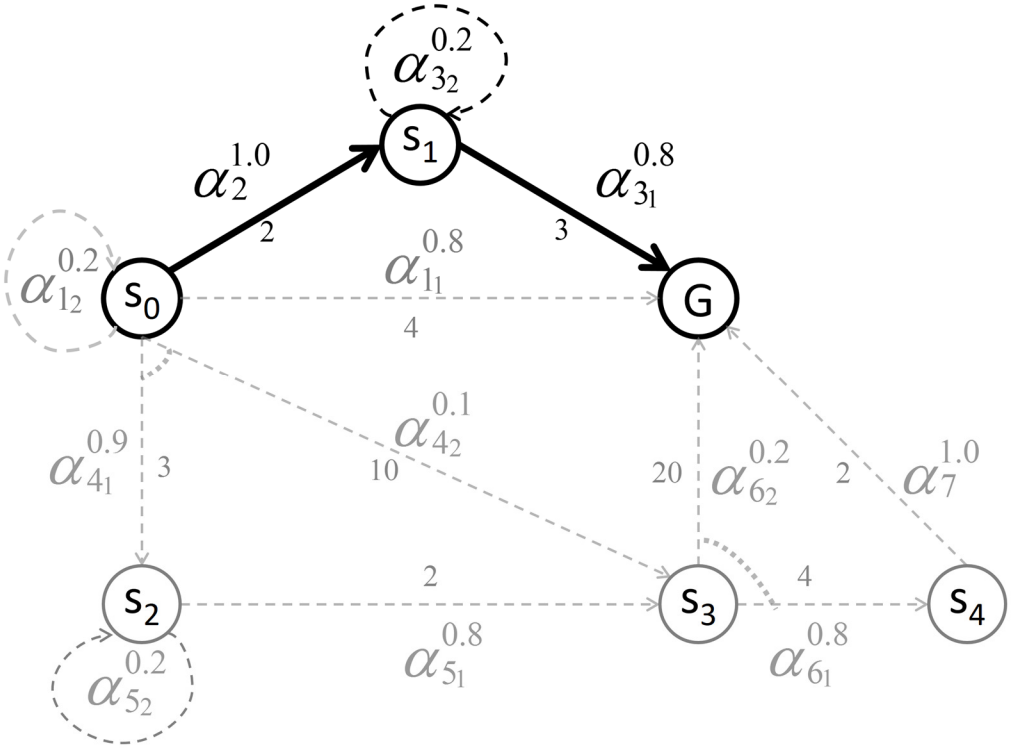

Figure 6.2: $\boldsymbol{Plan_2}$ for MS-UC4.

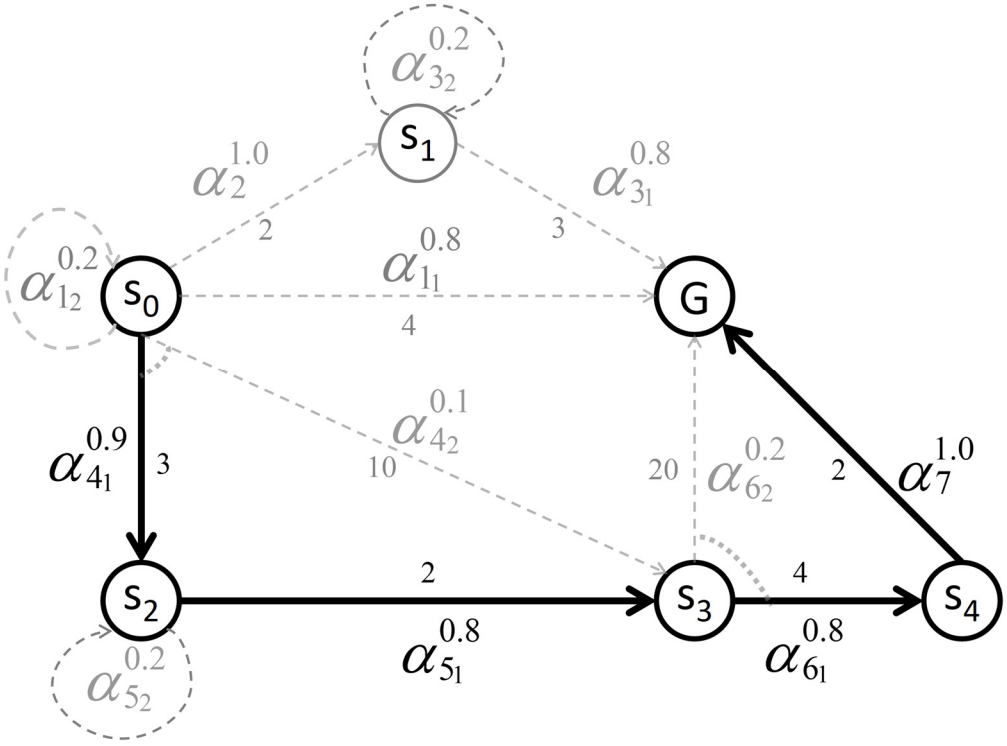

Figure 6.3: $\boldsymbol{Plan_3}$ for MS-UC4.

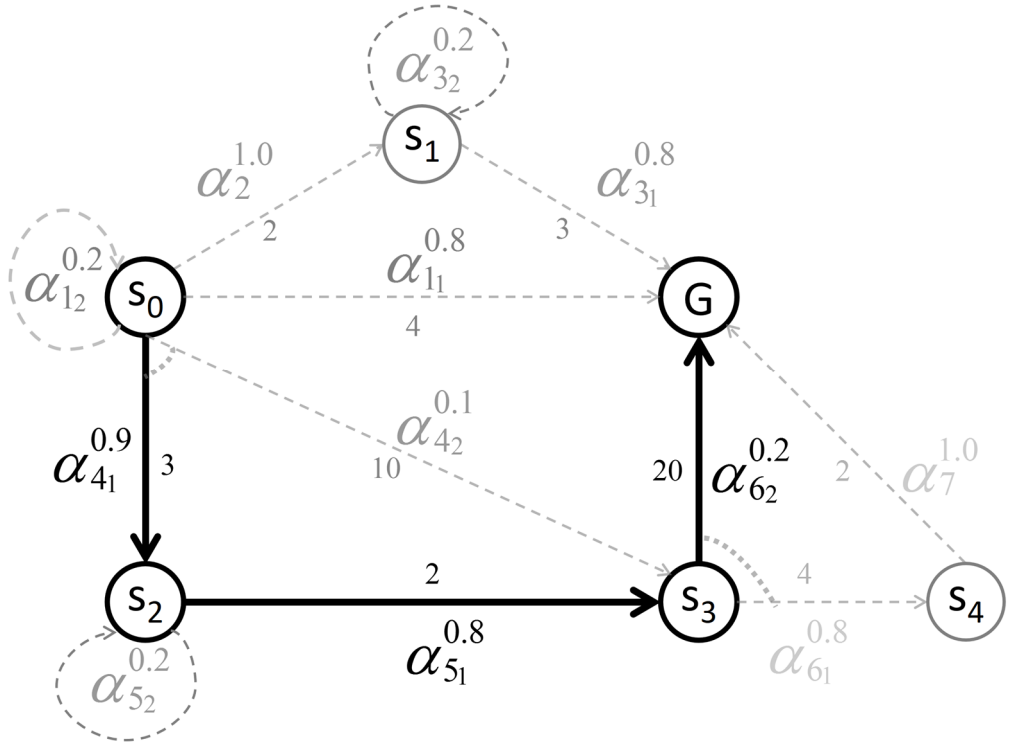

Figure 6.4: $\boldsymbol{Plan_4}$ for MS-UC4.

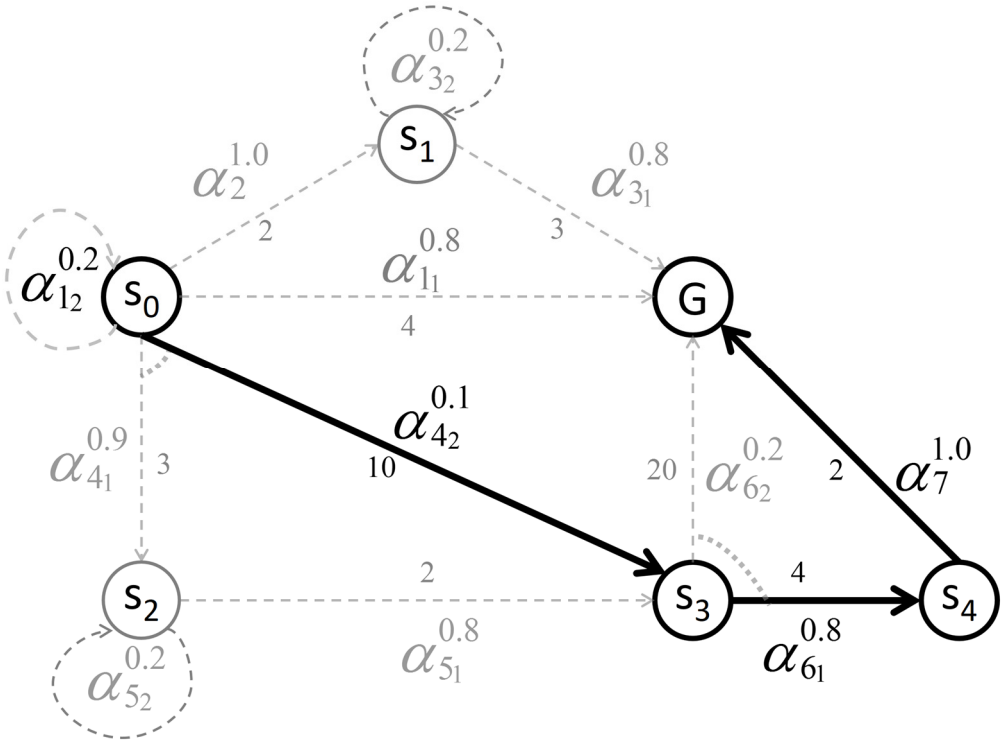

Figure 6.5: $\boldsymbol{Plan_5}$ for MS-UC4.

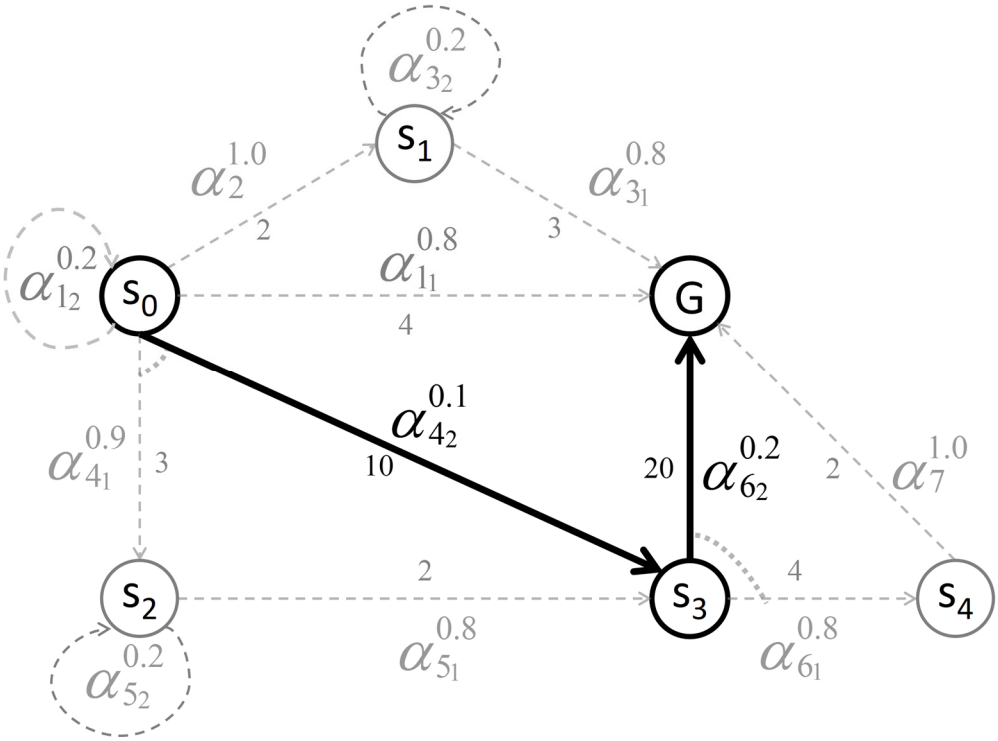

Figure 6.6: $\boldsymbol{Plan_6}$ for MS-UC4.

Figure 6.7 presents the form of the contingent solution; Circular nodes are chance nodes that lead to alternatives over which the planner has no control, i.e., they denote non-deterministic actions, with the ones that can potentially lead to the goal being depicted in grey. Triangular nodes are end nodes; either goal ones ("G"), or dead-end ("D") ones. The edges depict the actual executed outcome of the action. Having sorted the plans by their expected utility, $Plan_1$ is inserted first into the decision tree. The successful execution of a_1 is added as the first right branch in the tree. The left branch of a_1 corresponds to its failure, and requires the computation of the possible plans for insertion.

Since no other plan contains a_1, all plans except $Plan_1$ can be inserted; moreover they all retain their expected utility. $Plan_2$ is added; since a_2 is deterministic, it leads to a_3, where the previous procedure is followed again. Following the insertion of a_4 and its outcome a_{42}, $Plan_1$, $Plan_2$, $Plan_3$ and $Plan_4$ are rejected, as they contain outcome a_{11}, a_{31}, a_{41} and a_{41} respectively, which correspond to actions that have already been executed with different results. $Plan_5$ and $Plan_6$ are valid, and since a_{42} is now considered to have happened, only their remaining portions need to be added, i.e., $[a_{61}, a_7]$ and $[a_{62}]$ respectively. As such, their expected utilities have changed to 75.2 and 16.0 respectively for $Plan_5$ and $Plan_6$ and $Plan_5$ is chosen as the right branch of the tree. After the executions of the left branch at that point (a_{62}), only $Plan_6$ is valid and, as it contains no more actions, we reach a goal state.

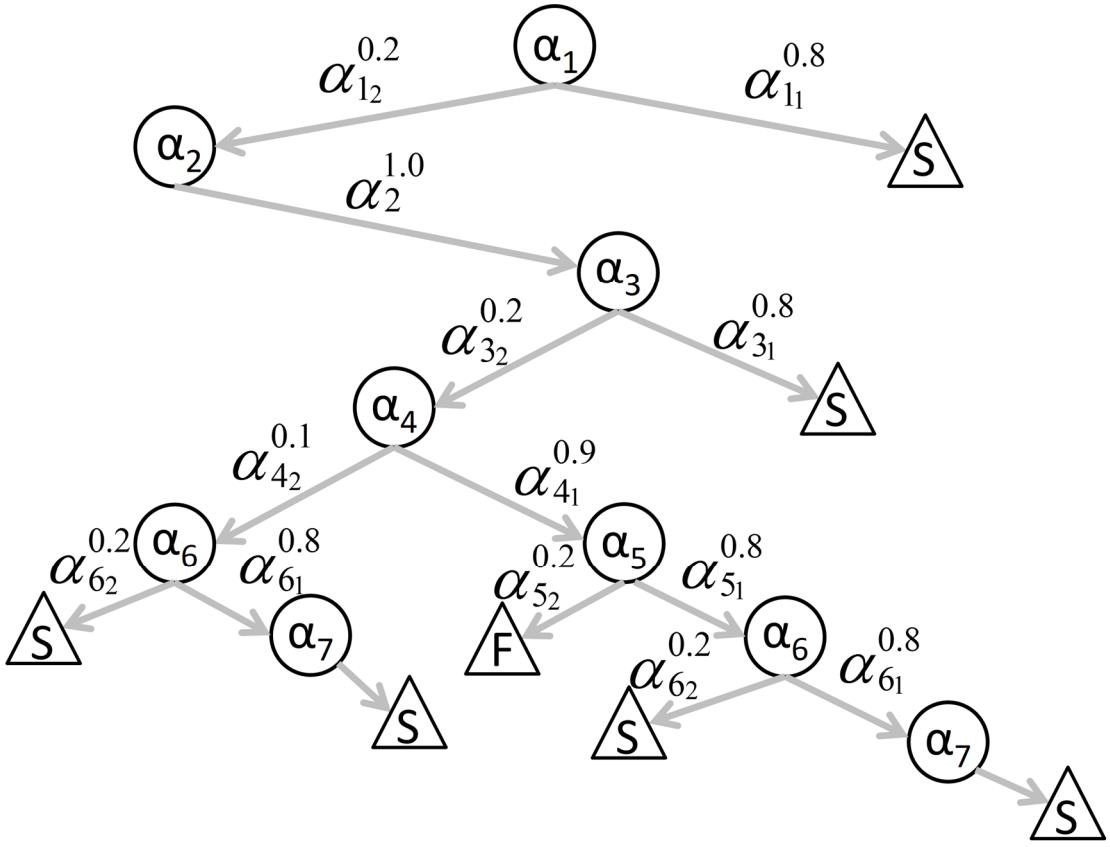

Figure 6.7: Decision tree for MS-UC4.

6.2 Experimental Evaluation of Web Service Composition Methods

In this section we evaluate the automated WSC aspect of the implemented system based on the use case scenarios that were presented in Sections 4.2.3-4.2.7. In particular, we quantitatively evaluate our deterministic WSC module against existing planning approaches; then we present an extensive empirical evaluation of the non-deterministic WSC algorithm that was presented in Section 6.1.1, and showcase its progress in terms of efficiency from its original implementation to the current one.

6.2.1 Evaluation of Deterministic WSC Algorithm and Translation Process

In order to test the automated WSC module of **MADSWAN** and the correctness and efficiency of the deterministic version that is integrated in it, we empirically compared the implemented deterministic algorithm in two domains. This section evaluates the translation and planning process, both in relation to each other, and relative to existing systems.

The deterministic algorithm that has been integrated in **MADSWAN** is a straightforward implementation of A*; in this implementation we used a simple heuristic, h_{sim} , equal to the percentage of achieved goals so far, i.e., $h_{sim_i} = \frac{|s_g - s_i|}{|s_g|}$, $0 \leq h_{sim_i} \leq 1$. Moreover, in the deterministic setting, the web services are assumed to not have a cost associated with their execution; that is, during planning, the algorithm only tries to optimize the number of web services comprising the solution. The approach is evaluated against an existing WSC system, PORSCE II and an existing PDDL-compliant planner, POND. We used two domains, one taken from the evaluation of (Hatzi et al., 2011), in order to provide direct comparison with PORSCE II, and one corresponding to the movie search scenario (MS-UC_1). The experiments were run on a PC using a Dual-Core Intel i5 processor running at 1.6GHz and allowing at most 4GB memory.

The former domain is a modified version of the evaluation domain from (Hatzi et al., 2011) and is similar to - but simpler than - MS-UC_2; the problem describes a user that desires to purchase a book through an electronic bookstore. He knows the book title and its author, and can provide his credit card information, and shipping address. The results of the output composite WS should be the book's purchase, as well as shipping dates and customs cost for it. This scenario does not feature non-determinism or any choices on behalf of the user. The user is presumed to have a credit card that he can use to purchase a book, and the book is always in stock.

According to (Hatzi et al., 2011), there are three different versions of this domain, named *P-x*, "*P*" symbolizing the PORSCE system and *x* representing the number of WSs participating in the problem. Since the WSs are translated to PDDL actions, each version is increasingly more complex than the previous ones, the first consisting of 10 WSs (*P-10*), the second of 100 (*P-100*) and the last of 1000 ones (*P-1000*). In the first two problems instances, the optimal plan length, i.e., the least amount of WSs needed to achieve the desired goal, is 6, whereas the last version has an optimal plan length of 5.

All of the aforementioned planning domains and problems are available at (Markou, 2014a). MS-UC_1 was used as the second evaluation domain; specifically a version of it with 21 WSs taking part in the WSC process in total. The optimal plan length for MS-UC_1 is also 6.

POND, presented in detail in Section 2.4, can generate plans using various search algorithms, specifically A^*, AO^*, LAO^* (Hansen & Zilberstein, 2001). In the current experimental setup two versions of POND version 2.2 were tried, namely with the A^* and AO^* search algorithms; the first was chosen as it is based on A^*, similarly to the integrated deterministic WSC module in **MADSWAN.** The second, as in experiments that are not reported here for reasons of brevity, POND-AO^* exhibited better performance than POND-EHC. As aforementioned in Section 3.3.4.1, PORSCE II relies on two alternative planning systems, JPlan and LPG-td. For the needs of the current evaluation, **MADSWAN** was compared against LPG-td, as (Hatzi et al., 2011) concluded that its performance was by far superior to that of JPlan. LPG-td is a sub-optimal anytime planner based on stochastic local search and planning graphs; its search space comprises action graphs, that is, particular sub-graphs of the planning graph representing partial plans. The experimental results are presented in Table 6.1. The system with the best performance in each problem is highlighted in bold, and the optimal plan length (*Opt*) is shown next to each problem. In all cases, the time reported is in seconds and represents the total time needed to reach a solution. It should be noted that, as POND is a strictly planning approach, these times do not include the necessary time to translate the original problem to a planning one; all methods are assumed to have available the planning description of the problem and its domain.

The experiments indicate that the number of WSs available for WSC is a crucial factor in regard to the planner's efficiency, even if not all services are necessarily useful for the achievement of the goal. The results in Table 6.1 indicate that, generally, as the number of WSs participating in each problem increases, so does the time required to solve it, although not linearly. This assumption is corroborated by the results of the comparison between MS-UC_1 and *P-10*, the two problems that comprise just a few WSs. MS-UC_1 requires more time than *P-10* for all versions of POND, and almost the same time for **MADSWAN,** but a little less time for LPG-td. This is not surprising, as the two problems share a common optimal plan length and MS-UC_1 comprises almost double the WSs than P-10.

Moreover, although all the other problems in the experiments have a larger optimal length, *P-1000* appears to be by far the most difficult problem in the test set. The increase in the required time to solve *P-1000* for POND-A^* is almost a thousand times more than for the second most difficult problem, *P-100*, a hundred times more for POND-AO^*, and tenfold for LPG-td and **MADSWAN**. Although the number of WSs comprising a particular problem is important, though, its difficulty is not dependent solely on this factor; e.g., **MADSWAN** manages to solve *P-100* faster than it does *P-10*. This fact can be mainly attributed to the complexity of the problem itself. The preprocessing phases of both LPG-td and all the versions of POND search for useless actions in the domain and allow the planners

Table 6.1: Comparison of planning times for POND, LPG-td and **MADSWAN**

	Optimal plan length	POND-A*		POND-AO*		PORSCE II		MADSWAN	
		Length	*Time*	*Length*	*Time*	*Average Length*	*Average Time*	*Length*	*Time*
MS-UC1	6	6	0.0068	6	**0.0032**	6.04	0.0040	6	0.0169
P-10	6	6	0.0020	6	**0.0016**	6	0.0066	6	0.0172
P-100	6	6	0.0028	6	**0.0026**	6	0.0124	6	0.0117
P-1000	5	5	2.5404	5	0.2742	5.76	0.1256	5	**0.0996**

* The time measurements are in seconds; for the stochastic planner, LPG-td, they represent the median values of 100 runs.

to ignore them during search (or simply prune them at parsing time). For this reason, both planners report that *P-10* consists of 10 (relevant) actions, whereas *P-100* of just 6.

Preprocessing leads to simpler domains, that is, for POND (that includes such facts in its output), *P-10* comprises 15 state variables while *P-100* comprises only 6. As such, both planners spend less time actually searching for a valid plan; on the other hand, though, this means that for relatively easy domains, the vast majority of the total planner time is spent in the preprocessing phase. For example, PORSCHE-II, in *P-1000,* in some cases devotes more than 80% of its total solution time to preprocessing; for this problem **MADSWAN** manages to be faster than all planners, despite the fact that it spends almost all of its total time is spent on search. We attribute this to the fact that it is not a generic PDDL planner, and as such, has a-priori knowledge that the actions in the relevant problems are ground. Thus, in these problems, it seems that preprocessing is not particularly advantageous. In order to test this assumption, we also ran *P-1000* in FF; FF solves the problem in 0.2 seconds, of which it spends the entire time in instantiating the actions and practically no time in its plan search. These results seem to confirm our hypothesis.

In general, the problems seem to be almost trivial for all planners, with the exception of *P-1000* for POND-A*, which has a significantly worse performance for this problem, both compared to the other planners and to its performance in the rest of the problems. POND, as well as **MADSWAN**, find solutions of optimal length for all problems. LPG-td, however, being a stochastic anytime non-optimal planner, returns plans of slightly worse median length for MS-UC1 and (mainly) *P-1000*. **MADSWAN** needs more time than all versions of POND for the smaller problems (MS-UC1*, P-10,* and *P-100*), as well as than LPG-td for the two smallest ones. It is, however, the fastest in the most complex problem, *P-1000*. As aforementioned, we experimentally tested only the efficiency of the planners on the translated planning problems, and not on the whole process that **MADSWAN** typically follows, which includes the translation from WSC domains to planning ones. Table 6.2 presents the results of experiments regarding the average transformation time per WS description of a WSC domain comprising 10, 21, 100 and 1000 WSs. Comparing the results of Table 6.2 with the ones in Table 6.1

it is evident that a major bottleneck for WSC problems seems to be the translation process. That is, the results for **MADSWAN** show that even when the average transformation time per WS converges to its lowest value (0.1110 seconds), the transformation time of a single WS is larger than the solution of a problem containing the same number of WSs/actions (0.0996 seconds), i.e., the time required by the planner to find a solution is less than one hundredth of the total time needed for the entire WSC process. Furthermore, the experiments show a decrease in the average time required per WS translation as the total number of WSs in the set increases, with the necessary time to translate a WS registry analogous to the size of the entire OWL-S TC converging to approximately 0.1 seconds per WS.

Table 6.2 Translation times per web service for **MADSWAN**

WSs	**MADSWAN**
	Time
10	0.3974
21	0.2836
100	0.1774
1000	0.1110

a. All time measurements are in seconds and represent the median values of 100 runs.

6.2.2 Evaluation of MAPPPA2

In order to test the efficiency of **MAPPPA2**, the system was empirically evaluated using three domains; these include the two use case scenarios that were created to test the contingent planning capabilities of our system in Sections 4.2.6-4.2.7, i.e., MS-UC_4 and MS-UC_5, and, as in the previous sub-section, a variant of MS-UC_2. This problem instance is modified so that a subset of the WSs that comprise it also have non-deterministic effects. A probabilistic version of each domain was created, with costs attached to the WSs that comprise it, and a subset of the WSs being deterministic. Moreover, alternative WSs were added achieving the same results as the existing ones, with different probabilities and/or costs, as well as different preconditions, in order to facilitate the generation of alternative plans.[19]

[19] The created planning domains and problems, as well the WSs used are available at http://ai.uom.gr/gmarkou/Files/IJAITMapppaDomains/PDDL/IJAIT-MapppaDomains.zip

All domains have two versions, one with unary costs for all the WSs ($MS - UC_x^{un}$), and one with variant costs ($MS - UC_x^{var}$); in the latter case, The services' costs in each one start from 1 and are taken from an exponential probability density function, that is, $p(x) = e^{-(x-1)}$, for $x \geq 1$. The utility of achieving the goal is set to $U = 100$ for all problems. The three domains are characterized by different orders of complexity, with MS-UC_4 containing 7 WSs, MS-UC_2 containing 14 WSs and MS-UC_5 containing 20 WSs. All of the aforementioned planning domains and problems are available at (Markou, 2015).The experiments were run on a PC using a Dual-Core Intel i5 processor running at 1.6GHz, allowing at most 8 GB memory.

We examined three setup options for the evaluation; A* with h_{max}, A* with h_{add} and A* with $h = 0$ (that is, simple breadth first search), denoted by h_0. The experimental results for MS-UC_4, MS-UC_2 and MS-UC_5 are shown in Tables 6.3, 6.4 and 6.5 respectively. In all cases we refer to the domain instead of the problem, as the initial states/goals of the problems do not change; the differences between the two versions of each specific domain lie in their definition of costs. The tables show the time required to generate all possible non-subsumed plans in milliseconds, the required time to generate the first plan, the total number of weak plans generated and the expected utility of the final decision tree constructed by merging the alternative weak plans.

In general, the experimental evaluation showed that our approach is promising. The planner is able to produce multiple unique solutions to the problems at hand, even when the problem is computationally hard and its search space is large.

In the experiments, the planner managed to find all solutions in all cases, irrelevant to which heuristic had been used. Thus, the merging process resulted in the same contingent plans. The difference between the heuristics lies in the time required to generate the weak plans, as well as in the order in which they are produced. In regard to the time, when h_{add} or h_{max} is employed, the overall time involves the computation of the heuristics. Note that the two heuristics are computed for each state from scratch, in a forward direction. Concerning the order in which the weak plans are generated, with h_0 and h_{max} they are generated in decreasing order of their expected utility, whereas with h_{add} there is no such guarantee.

In order to demonstrate the anytime behavior of the proposed algorithms, Fig.6.8 presents the expected utility of the decision tree for various numbers of the best weak solutions used for its construction, using as an example $MS - UC_4^{var}$. As we can see, the expected utility takes very high values with only a few weak plans. Of course, this behavior also depends on the domain, with domains with high probability of failure requiring more weak plans to reach acceptable levels of expected utility and success probability for the final contingent plan.

Furthermore, Fig. 6.8 and Fig. 6.9 reveal the anytime behavior of the **MAPPPA2**. Since merging the weak plans requires significantly less time than finding them, the merging procedure can be employed

each time a new weak plan is found. Thus, the more time is available, the better contingent plans will be constructed.

Table 6.3. Evaluation results – MS-UC_4 (time in msecs).

	$MS-UC_4^{un}$					$MS-UC_4^{var}$				
Heuristic	Time	Time to first plan	Weak plans	Success probability	Expected Utility	Time	Time to first plan	Weak plans	Success probability	Expected Utility
h_0	10	6	6	99.28%	97.75	11	7	6	99.28%	93.79
h_{add}	15	3	6			16	16	6		
h_{max}	15	15	6			15	15	6		

Table 6.4. Evaluation results – MS-UC_2(time in msecs).

	$MS-UC_2^{un}$					$MS-UC_2^{var}$				
Heuristic	Time	Time to first plan	Weak plans	Success probability	Expected Utility	Time	Time to first plan	Weak plans	Success probability	Expected Utility
h_0	281	78	8	96.92%	90.92	278	78	8	96.92%	80.61
h_{add}	234	62	8			234	63	8		
h_{max}	249	78	8			234	63	8		

Table 6.5. Evaluation results – MS-UC_5 (time in msecs).

	$MS-UC_5^{un}$					$MS-UC_5^{var}$				
Heuristic	Time	Time to first plan	Weak plans	Success probability	Expected Utility	Time	Time to first plan	Weak plans	Success probability	Expected Utility
h_0	3494	296	4	88.00%	81.48	3385	359	4	88.00%	67.20
h_{add}	3666	468	4			3573	437	4		
h_{max}	3525	374	4			3463	437	4		

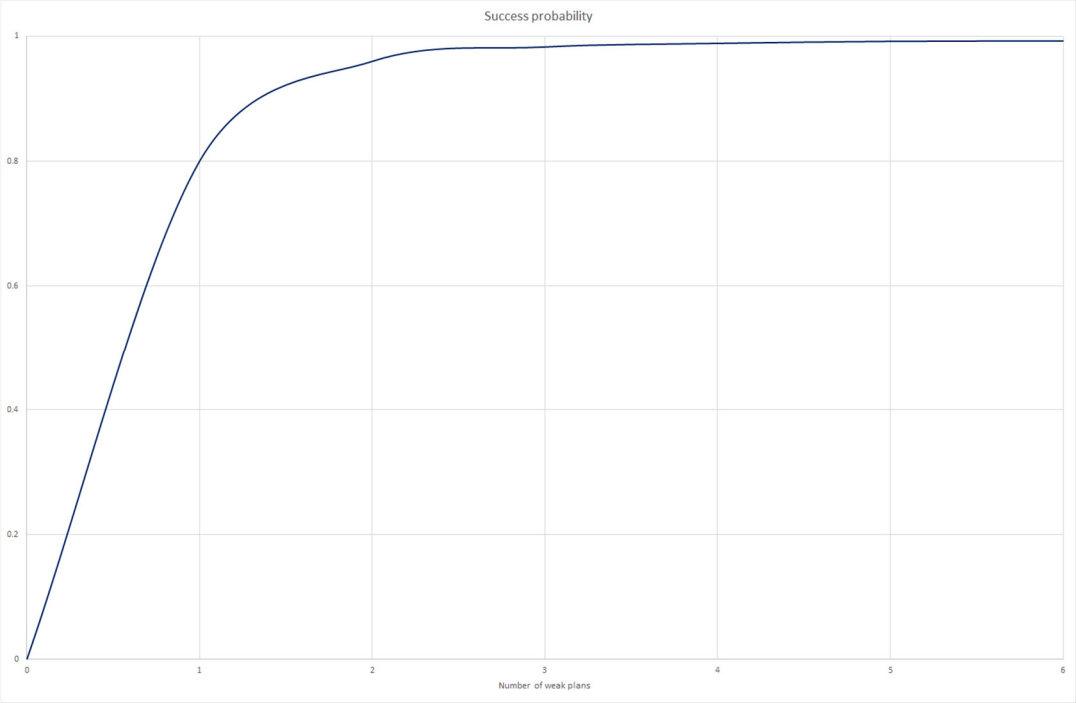

Figure 6.8: Success probability of the contingent plan as a function of the number of weak plans for $\mathbf{MS-UC_4^{var}}$.

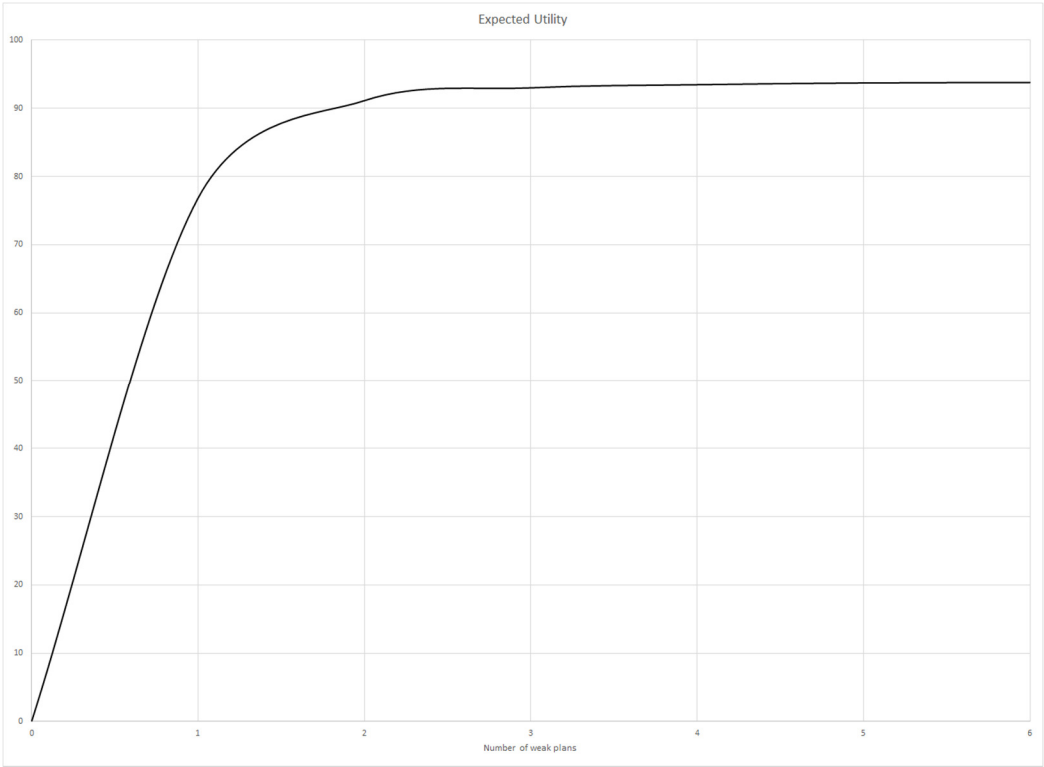

Figure 6.9: Expected utility of the contingent plan as a function of the number of weak plans for $\mathbf{MS-UC_4^{var}}$.

6.2.3 Discussion

This section presented an evaluation of the proposed algorithms to tackle the problem of automated WSC. Experimental results were presented in regards to a proof-of-concept deterministic planner for WSC, which was evaluated against existing planning approaches and proved to be competitive. The main focus, however, was the extensive evaluation of **MAPPPA2** in three domains.

MAPPPA2 it adopts expected utility as its single optimization criterion and combines it with established heuristics. Its A* implementation has been optimized to prune the paths that are subsumed by others, by checking whether all the previously added actions of the path already form another shorter solution. The experimental results reveal the effectiveness (in terms of expected utility and success probability of the contingent plan) and efficiency (in terms of the time needed to create the contingent plan) of the proposed methods, as well as its anytime nature.

6.3 Web Services and Automated Planning for Intelligent Calendars: A Case Study

This section promotes the automated creation of hybrid personal plans, comprising WSs and real human activities, to be supported by the next generation of intelligent calendar applications. A case study implementation is presented, utilizing existing state-of-the-art components as a proof of concept of how WSs can be utilized in real life applications. Although in the recent past the popularity of paper calendars has steadily declined in favor of web-based calendar applications like Google Calendar[20], such applications still do not provide any means for automated activity scheduling. In that way, users are forced to decide for themselves whether a particular activity has enough time to be scheduled alongside another, or whether the distance between the places where the two activities occur is a prohibiting factor.

This problem was effectively tackled in SELFPLANNER (Refanidis & Alexiadis, 2011),[21] the first system that automatically schedules personal activities into electronic calendars, using a combination of a greedy optimization algorithm, namely a modified version of the Squeaky Wheel Optimization (SWO) (Joslin & Clements, 1999), and stochastic local search in a post-processing phase. SELFPLANNER employs a rich model supporting temporal domains and preferences, locations, interruptible and periodic activities, binary constraints and preferences, etc. It also uses Google Maps Distance Matric API.[22] to compute the necessary travelling times between the activities' locations, as well as Google Calendar to present the outputted schedules.

(Bank et al., 2012) build directly upon this work, using SWO in addition to a set of calendar entity types that they propose; in specific, they discriminate between simple events, multiple choice events,

[20] https://www.google.com/calendar/
[21] Available at http://selfplanner.uom.gr
[22] https://developers.google.com/maps/documentation/distancematrix/

floating events and tasks. Moreover, they incorporate elements of psychology into the generation of the schedules, by defining preferences such as that there should be no wasted travel time between events, or that creative tasks should be split into multiple segments. (La Placa et al., 2009) follow a different approach, by utilizing HTN planning and focusing on a specific user target group. Their approach is directed towards people with cognitive impairments, e.g., Alzheimer's disease, and as such, HTN planning, which decomposes tasks into subtasks, is well suited as it resembles the way medical professional actually plan for their patients. Moreover, this degree of granularity is dependent on the specific patient, with information such as his impairment or personality being taken into account. Finally, (Berry et al., 2009) present Emma, a personalized calendar management tool, which simultaneously manages calendars from multiple sources, with the main aims of facilitating the coordination of groups of people, the negotiation of their meeting times and the (re)scheduling of various events.

These approaches highlight an obvious next step in intelligent calendar applications; that is to employ planning, instead of pure scheduling, to achieve the user's goals. Existing systems, such as the aforementioned, rely on the user to select the activities to be included in his plan, with the system providing only scheduling functionalities. We envision a situation where intelligent calendar applications (a) support action ontologies, with action descriptions containing preconditions and effects, (b) allow the user to set his goals, (c) know the user's state, and (d) employ automated planning to create plans that achieve the user's goals.

An even further step would be to extend intelligent calendar applications so as to take WSs into account. WSs can be considered as normal actions that can be used to achieve users' goals or other actions' preconditions. Incorporating them into the action ontology enables the substitution of human activities by WS calls, thus allowing for more flexible plans, more goals to achieve or just more free time. As an example of such a setting that is very common in our daily routine, consider this: a person may wish to attend a concert, and for that reason he may insert a personal activity in his schedule so as to reserve time and remember to buy tickets for it. To achieve this goal the user is required to physically go to a brick-and-mortar shop or buy his tickets online: both options require some of his time (obviously, buying the tickets online requires less time). A more efficient electronic calendar, on the other hand, would have searched for an alternative - automated – way of achieving such goals, so as to relieve the user from the burden of manually executing the necessary actions.

In this section we propose such an approach; it is based on the integration of WSs with an existing metric planner and an intelligent calendar application, particularly SELFPLANNER. WSs may be simple or composite; in the latter case, **MAPPPA2** is utilized so as to reduce the non-determinism underlying their execution. From the planning perspective, though, WSs are considered deterministic. As a result of the above process, the user's schedule contains only the human activities that cannot be completed through the use of WSs, thus relieving him of the extra burden to achieve the rest.

6.3.1 Motivating Example

Let us imagine a usual week in Bob's life, who uses a web-based calendar application to organize his time. Due to being self-employed, Bob needs to spend all working hours at his office, to which he commutes every day from his home. Once every week, he watches a movie and, although he prefers to go to the cinema, sometimes he watches the movie at home, depending on his work schedule. However, if the movie's duration is such that it will end later than 11 pm, he does not desire to watch a movie at all, as he has to sleep early. Moreover, this week, he will travel on a business trip abroad; at least a day before his trip, the day he usually watches a movie, he has to book his airplane tickets and hotel, as well as to buy a travel guide for the city he will visit.

In order to schedule these activities and insert them into a calendar, the user has to define their temporal domain and, if they are interruptible, the minimum and maximum allowed duration for their parts. Moreover, for each activity, the user has to declare whether it is periodic or not, as well as if it is bound to specific locations. For example, the user should define his daily work as a periodic task, with a temporal domain from 8 a.m. to 5 p.m., and a minimum and maximum duration of its parts – as he cannot work continuously, e.g., 30 minutes and 2 hours respectively. Watching a movie is also periodic but non-interruptible, and has a temporal domain set late in the evening, with a minimum and maximum duration of 90 minutes and 180 minutes respectively.

Since the user's work requires his physical presence, only a human activity can be placed in his calendar. However, for the rest of the aforementioned tasks, a combination of human and WSs' activities can be inserted. The user may have to drive to the cinema, choose among the available movies there and buy the tickets himself. Alternatively, he may ask for a list of the available movies to be emailed to him, book the tickets online and then, having saved considerable time, travel to the cinema later. Another option altogether would be to rent a movie online and watch it at home. As to the user's business trip, again, there are various alternative activities; the user may visit a single tourist agent to book his tickets and hotel; or prefer an online reseller. He may also require another trip to a bookstore to purchase his travelling guide or purchase it online and have it sent at home.

Since these options create a complex problem, containing a multitude of constraints and preferences, a schedule manually created by a human is usually highly inefficient. Applications such as SELFPLANNER tackle this problem; however, they only deal with human activities and, as such, they cannot take advantage of the opportunities that are offered by the use of WSs. In the aforementioned scenario, since a human activity requires the user to first purchase a movie ticket himself, in certain situations he may not have had the time to do so, and he would have had to watch it at home instead. Even worse, if he had to visit a travel agent and a bookstore, he may not have even had the time to watch a movie at home. With the introduction of WSs, the user saves time needed to perform the actual action of purchasing these services, as well as the time needed to travel between locations.

6.3.2 Integration of Intelligent Calendars with Web Services and Planning

WSs are assumed to be semantically annotated with their descriptions present in an online registry; in that way, the online registry presented in Section 5.1, along with the incorporated translation of the registered WSs to PDDL actions can be utilized. Moreover, the registry also contains composite WSs fortified against non-determinism, having been generated by **MAPPPA2** prior to the start of the scheduling process. Thus, the composite WSs used comprise multiple execution paths achieving the same result, and for this reason can only fail when all the execution paths fail.

Figure 1 presents the proposed system's architecture. Initially, an existing metric planner, namely LPG-td, is employed; this step is necessary in order to automate the planning process as the activities in SelfPlanner are normally entered by the users. Instead, in this case, in order to obtain the set of activities that achieve the desired goal a planner has to be utilized. LPG-td receives as input a planning problem containing both WSs and human activities. This problem comprises of the translation of the WSs from the registry, as well as a simplified version of the human activities; that is, the locations, durations and temporal domains of the activities, along with any preferences and constraints in regard to them are removed. The metric planner does not differentiate between the two types of activities; as a result, it also treats all WSs as deterministic ones, i.e., as if their intended output (the most probable one) is always outputted. Moreover, the planner has to take into account that web services are preferred to their human activities' counterpart. This is achieved by setting the cost of human activities higher than that of WSs.

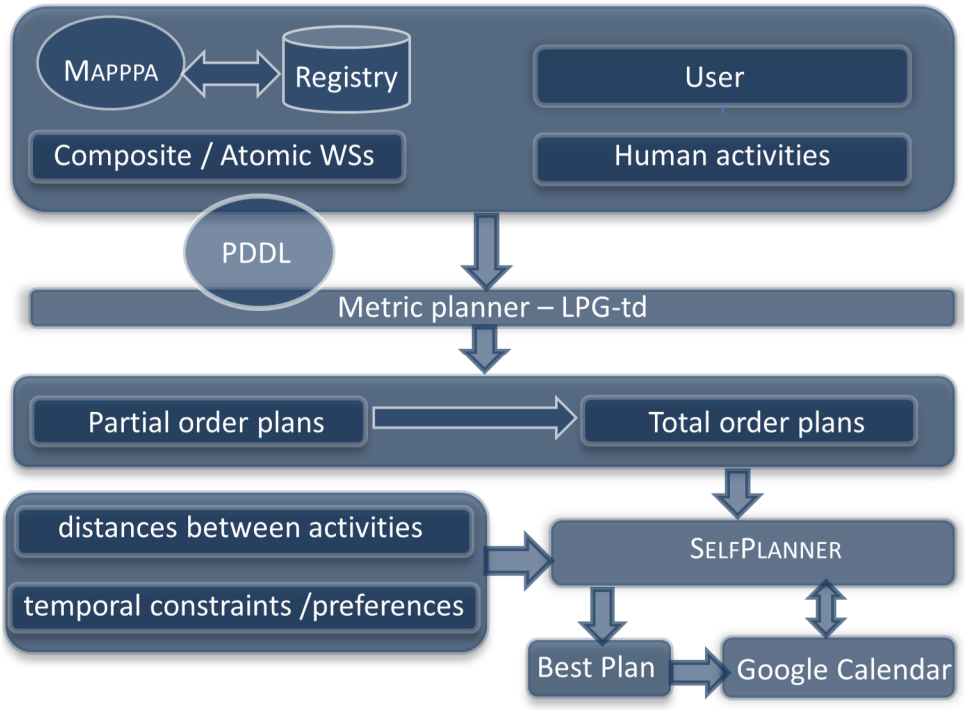

Figure 6.10: Proposed system's architecture overview.

LPG-td is capable of generating a sequence of alternative plans, each being an improvement – in terms of the plan's total cost – compared to the previous one; the plans generated by LPG-td allow for parallel actions, and thus, are very similar to partial order ones. The best generated plan by LPG-td is fed to SELFPLANNER so as to create a detailed schedule. In case a feasible plan does not exist, the rest of the previously generated alternative partial order plans are attempted to be scheduled in a similar process. In order to schedule the activities of the partial order plan, SELFPLANNER employs the information concerning temporal constraints and preferences (loaded from a separate activity definition file for a given problem instance)„ as well as the actual distances between the activities' locations as returned by the Google Maps Distance Matric API. WSs are considered to have an open temporal domain and are not related to a specific location, as they can be executed at any time and place. Moreover, they can be scheduled parallel to human activities. SELFPLANNER can generate multiple alternative plans and the human activities of the best schedule are uploaded into the user's Google calendar.

6.3.3 Discussion

This section presents the first steps towards the next generation of intelligent calendar applications. An approach comprising a contingent web-service composition system, a partial order metric planner and a scheduler to insert human activities into a user's web calendar is proposed. The contingent planner is used to create composite WSs, able to achieve complex goals with high probability of success; the metric planner is used to select an optimized set of activities to achieve a user's goal, favoring WS calls than real activities; and, finally, the scheduler automatically produces an optimized plan based on the user's constraints and preferences. In essence, the results of Sections 5.1 and 6.1 are used in conjunction with an existing external metric planner and a web based application in a modular fashion, following the WS principals. A motivating example along with a – still under work – implementation is presented. Future research includes further improving the current implementation, first by providing a graphical interface, as well as by integrating it with a WS execution platform. Also, we propose to perform an exploratory evaluation, by providing the - users with two schedules, one consisting solely of human activities and an equivalent one comprising both human and web service activities, and having them rating each one online. However, for the time being, the implementation serves as a proof that the integration of non-deterministic planning in combination with WSs into real world applications can be utilized to create better, more efficient, applications; in this case, in order to increase the users' capacity, their free time, as well as their scheduling options.

Finally, various research and implementation issues are still open; for example, we assume that the user's schedule should only contain the human activities that cannot be substituted by web services; this is achieved through setting the cost of human activities higher than that of web services. However, in some cases this may not be true; employing a web service may incur a (financial) cost that the user does not prefer over the benefit of not employing the respective human activity. Moreover, a web

service activity may not always provide exactly the same functionality or satisfaction as its human counterpart; such a case is present in our motivating example, in which a user prefers to go to the cinema to watch a movie, than watching it at home through streaming. In this case, however, if the problem is modeled so as to favor the human activity, it could be impossible to include it in the plan, thus providing as an alternative plan the use of a web service. Such problems require further investigation.

Chapter 7.

Conclusions and Future Work

Web service composition is a particularly active research area; in the latest years both academia and the industry have focused on providing efficient solutions. An aspect of web service composition, however, which has been not been studied in depth is non-determinism. Despite the fact that the domain is inherently non-deterministic, with web service descriptions being bound to real-world web services whose execution may not always be successful or have the desired result, the approaches that adopt such a formalism and try to tackle the relevant problem are scarce in the literature.

Similar to our previous work in (Markou & Refanidis, 2012), (Zuñiga et al., 2014) also argued that the architecture of a web service composition approach should be decoupled in accordance to the design principles of web services themselves; that such approaches should facilitate their use by non-expert users, especially as to the specification of the goal request or the selection of WSs; that the approach used to tackle the web service composition problem should be based on AI planning algorithms and be domain independent; and finally, they acknowledged the need for the inclusion of non-deterministic elements in the web service composition process. In this thesis we proposed an approach that adopts and conforms to all of the aforementioned principles. We presented a web-based application related to semantic web services that supports multiple phases of web service composition; the implemented prototype, **MADSWAN**, is to the best of our knowledge the first system of its kind that is also publicly available. The design and implementation of its modules were based on a modular architecture and make efficient re-use of existing open source software components that utilize the current web service standards. An extensive set of requirements for the application and its composition modules was defined as well as mockups for its intended GUI. The qualitative evaluation of the system shows that it complies with almost the complete set of requirements that had been set and that its finalized GUI is similar to the intended one.

A second major contribution of this thesis concerns the introduction of a new non-deterministic a web service composition algorithm, **MAPPPA2**, which assumes a fully observable probabilistic domain, based on an anytime contingent planning framework, and has been inspired by previous work in AI planning research field, specifically by determinization approaches. However, it differs from such methods, first in that it is specifically targeted to web service composition problems and, secondly, in that it takes the probabilities and the cost of the original non-deterministic actions into consideration while generating a contingent plan. In specific, the determinized problem is solved repeatedly by a deterministic planner, generating multiple plans which are then merged in a decision tree, using the initial non-deterministic actions as its decision nodes. Importantly, merging prioritizes the alternative plans according to their expected utility, which is updated each time a plan is selected for merging, by taking into account potential common web services between the newly merged plan and the remaining ones.

As it is evident from the above analysis, a third contribution of the thesis is the design of an evaluation set for web service composition approaches. The need for such a set arose from the relevant literature review that revealed not only that such a standard test set does not exist, but also that the majority of recent web service composition approaches utilize AI planning as their method of choice,

combining PDDL domains with OWL-S descriptions. Furthermore, the review of such approaches allowed us to identify the characteristics that differentiate the various web service composition approaches, along with their weaknesses and advantages. Along with an extensive survey of state-of-the-art non-deterministic AI planning approaches, we were able to identify the most promising lines of research for non-deterministic web service composition. The thesis presents a set of use case scenarios that can be used in a variety of such methods. This test set is based on an existing, open test collection of semantic web services and, as the scenarios and the web services that comprise them (along with their IOPEs) are described in detail, it can be easily be reproduced by and extended in other systems. Importantly, to the best of our knowledge, currently no other evaluation test set adopting non-deterministic elements exists in the literature. The quantitative evaluation of the web service composition algorithms that have been developed for the purposes of the thesis was based on this set of use case scenarios; the results of the evaluation show that the deterministic algorithm is competitive with existing web service composition and planning approaches. Moreover, the experimental results in regard to the non-deterministic web service composition algorithm revealed its effectiveness (in terms of expected utility and success probability of the contingent plan) and efficiency (in terms of the time needed to create the contingent plan), as well as its anytime nature.

Finally, as a proof of concept that non-deterministic web service composition planning can be utilized in real world applications to create better applications, an approach comprising **MAPPPA2**, an existing partial order metric planner and a scheduler to insert human activities into a user's web calendar was proposed. By reusing and incorporating existing modules, it has been possible to produce, with minimal effort, an efficient system towards the next generation of intelligent calendar applications.

7.1 Future Work

A first dimension of future work concerns the improvement of the underlying web service composition algorithms, deterministic and non-deterministic as well, including optimizing their implementation. Concerning **MAPPPA2**, one way of improving the algorithm's efficiency would be to optimize the current implementation; since, however, **MAPPPA2** does not depend on a specific method to generate the deterministic solutions that are then merged, the incorporation of an existing state-of-the-art deterministic metric planner would probably benefit its efficiency more.

A drawback of the current automated web service composition algorithms that we presented is that they only support exact matches of ontology concepts; we intend to fully support plugin matches, as well as subsume and sibling relationships in the immediate future. The incorporation of an efficient method of describing OWL-S QoS elements through a third-party extension also remains for future work.

As to the evaluation of the non-deterministic web service composition algorithm, **MAPPPA2** shows promising results, managing to solve all the problem variations of the use case scenarios in its evaluation in Section 6.2.2. However, this evaluation did not include the full set of use case scenarios

presented in Section 4.2; MS-UC_1 is deterministic; MS-UC_3 was excluded as it requires both the provision of preferences and the support of iteration control constructs, both aspects currently not supported by **MAPPPA2.** This fact, though, demonstrates the wide variety of approaches that the use case scenarios presented in Section 4.2 support. Moreover, this inability of **MAPPPA2** drives our future work, in the sense that we aim to support these aspects in future versions of the algorithm.

In relation to **MADSWAN**, despite being still in alpha version, it is largely fully functional. However, as mentioned in Section 5.2.3, the system currently does not comply with some of the composition requirements that were set, namely that the result of the manual web service composition module cannot be imported back into the registry as a web service. That is, the transformation from the visual workflow that the user creates to a valid OWL-S description file has not yet been implemented. As it is in alpha version, its source code is not yet ready to be publicly available, as in some cases it may still be unstable or present bugs. We plan to release a beta version of **MADSWAN** in a source code repository such as SourceForge[23] or GitHub[24] soon; in the meantime, the source code is available as-is[25].

Finally, as to the case study application presented in Section 6.3, despite it being a – still under work – implementation, we consider it to be the first step towards the next generation of intelligent calendar applications. It can be extended by providing a graphical interface to the application, as well as by integrating it further with a web service execution platform. The proposed approach still requires human intervention in several aspects of the process, such as the definition of the original domain that is inputted to the calendar application.

[23] http://sourceforge.net/
[24] https://github.com/
[25] http://ai.uom.gr/gmarkou/Files/MadSwanSourceCode/2014/

References

Akkiraju, R., 2007. Semantic Web Services. In J. Cardoso, ed. *Semantic Web Services: Theory, Tools, and Applications*. Information Science Reference. pp.191 -216.

Alamri, A., Eid, M.A. & El-Saddik, A., 2006. Classification of the state-of-the-art dynamic web services composition techniques. *International Journal of Web and Grid Services*, 2(2), pp.148-66.

Albore, A. & Geffner, H., 2009. Acting in Partially Observable Environments When Achievement of the Goal Cannot be Guaranteed. In *4th Workshop on Planning and Plan Execution for Real-World Systems in the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*. Thessaloniki, Greece, 2009.

Albore, A., Palacios, H. & Geffner, H., 2009. A Translation-Based Approach to Contingent Planning. In *Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*., 2009.

Albreshne, A. & Pasquier, J., 2010. Semantic-Based Semi-Automatic Web Service Composition. In *Proceedings of the Fith Libyan Arab International Conference On Electrical and Electronic Engineering (LAICEEE)*. Tripoli, Libya, 2010.

Alford, R., Kuter, U., Nau, D.S. & Goldman, R.P., 2014. Plan aggregation for strong-cyclic planning in nondeterministic domains. *Artificial Intelligence*, 216, pp.206–32.

Alrifai, M., Risse, T., Dolog, P. & Nejdl, W., 2008. A Scalable Approach for QoS-Based Web Service Selection. In *1st International Workshop on Quality-of-Service Concerns in Service Oriented Architectures (QoSCSOA '08)*., 2008.

Andrews, T. et al., 2003. Business Process Execution Language for Web Services. In *IBM Germany Scientific Symposium Series*., 2003.

Apt, K., 2003. *Principles of constraint programming*. Cambridge, UK: Cambridge University Press.

Arkin, A., 2002. *Business Process Modeling Language (BPML) Version 1.0*. [Online] Available at: http://www.bpmi.org .

Arkin, A. et al., 2002. *Web Service Choreography Interface (WSCI) 1.0*. [Online] Available at http://www.w3.org/TR/wsci/ [Accessed 13 May 2015].

Aslam, M.A., Auer, S. & Shen, J., 2006. From BPEL4WS Process Model to Full OWL-S Ontology. In *Proceedings of Posters and Demos at the 3rd European Semantic Web Conference (ESWC)*. Budva, Montenegro, 2006.

Bacchus, F. & Kabanza, F., 2000. Using Temporal Logics to Express Search Control Knowledge for Planning. *Artificial Intelligence*, 116(1-2), pp.123–91.

Backstrom, C. & Nebel, B., 1995. Complexity results for SAS+ planning. *Computational Intelligence*, 11(4), pp.625–65.

Banerji, A. et al., 2002. *Web Services Conversation Language (WSCL) 1.0*. [Online] Available at: http://www.w3.org/TR/wscl10/ [Accessed 01 June 2015].

Bank, J. et al., 2012. Turning personal calendars into scheduling assistants. In *Extended Abstracts of the Thirtieth CHI Conference on Human Factors in Computing Systems (CHI '12)*., 2012.

Bansal, A. et al., 2008. WSC-08: Continuing the Web Services Challenge. In *10th IEEE Conference on E-Commerce Technology and the 5th IEEE Conference on Enterprise Computing, E-Commerce and E-Services*., 2008.

Bartalos, P. & Bieliková, M., 2011. Automatic Dynamic Web Service Composition: A Survey and Problem Formalization. *Computing and Informatics*, pp.793–827.

Barto, A., Bradtke, S. & Sing, S., 1995. Learning to act using real time dynamic programming. *Artificial Intelligence*, 72, pp.81-138.

Baryannis, G. & Plexousakis, D., 2010. *Automated Web Service Composition: State of the Art and Research Challenges*. Technical Report. FORTH/ICS.

Battle, S. et al., 2005a. *Semantic Web Services Language (SWSL)*. [Online] Available athttp://www.w3.org/Submission/SWSF-SWSL/ [Accessed 01 July 2015].

Battle, S. et al., 2005b. *Semantic Web Services Ontology (SWSO)*. [Online] Available athttp://www.w3.org/Submission/SWSF-SWSO/ [Accessed 01 July 2015].

Battle, S. et al., 2005c. *SWSF Application Scenarios*. [Online] (1.0) Available at http://www.daml.org/services/swsf/1.0/overview/index.shtml [Accessed 01 July 2015].

Berardi, D., Calvanese, D. & De Giacomo, G..M.M., 2005b. Composition of Services with Nondeterministic Observable Behavior. In *Proceedings of the Third International Conference on Service-Oriented Computing (ICSOC)*., 2005b.

Berardi, D. et al., 2003. Automatic composition of e-services that export their behavior. In *1st International Conference on Service-Oriented Computing (ICSOC)*., 2003.

Berardi, D. et al., 2005. Automatic service composition based on behavioral descriptions. *International Journal of Cooperative Information Systems*, 14(4), pp.333-76.

Berman, K.A. & Paul, J.L., 2005. Chapter 24. Probabilistic and Randomized Algorithms. In *Algorithms: Sequential, Parallel, and Distributed*. Boston, Massachusetts, USA: Course Technology.

Berners-Lee, T., Hendler, J. & Lassila, O., 2001. The Semantic Web. *Scientific American*, pp.34-43.

Berry, P.M. et al., 2009. Evaluating user-adaptive systems: Lessons from experiences with a personalized meeting scheduling assistant. In *Proceedings of the Twenty-First Innovative Applications of Artificial Intelligence Conference (IAAI '09)*., 2009.

Bertoli, P. et al., 2001. MBP: aModel Based Planner. In *IJCAI'01 Workshop on Planning under Uncertainty and Incomplete Information*., 2001.

Bertoli, P., Cimatti, A., Roveri, M. & Traverso, P., 2006. Strong Planning under Partial Observability. *Artificial Intelligence*, pp.337-84.

Bertoli, P., Pistore, M. & Traverso, P., 2006. Automated web service composition by on-the-fly belief space search. In *16th International Conference on Automated Planning and Scheduling (ICAPS 06)*., 2006.

Bertoli, P., Pistore, M. & Traverso, P., 2010. Automated composition of Web services via planning in asynchronous domains. *Artificial Intelligence*, March 2010.

Birchmeier, P., 2007. *Semi-Automated Semantic Web Service Composition Planner Supporting Alternative Plans Synthesis and Imprecise Planning*. Diploma Thesis. University of Zurich.

Bizer, C., Heath, T. & Berners-Lee, T., 2009. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, pp.1-22.

Blake, M.B., Cheung, W.K.W. & Wombacher, A., 2006. WSC-06: The Web Service Challenge. In *8th IEEE International Conference on E-Commerce Technology and 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06)*., 2006.

Blum, A. & Furst, M., 1997. Fast planning through planning graph analysis. *Artificial Intelligence* , 90, pp.281–300.

Bonet, B., 2006. Bounded branching and modalities in non-deterministic planning. In *Sixteenth International Conference on Automated Planning and Scheduling (ICAPS)*. Ambleside, UK, 2006. AAAI Press.

Bonet, B., 2010. Conformant plans and beyond: Principles and complexity. *Artificial Intelligence*, 174(3-4), pp.245-69.

Bonet, B. & Geffner, H., 2000. Planning with incomplete information as heuristic search in belief space. In *Fifth International Conference on Artificial Intelligence Planning and Scheduling (ICAPS)*. Seattle, WA, USA, 2000.

Bonet, B. & Geffner, H., 2001. Planning as heuristic search. *Artificial Intelligence*, 129(1-2), pp.5-33.

Bonet, B. & Geffner, H., 2005. mGPT: A Probabilistic Planner Based on Heuristic Search.. *Journal of Artificial Intelligence Research (JAIR)*, 24, pp.933-44.

Bonet, B. & Geffner, H., 2011. Planning under partial observability by classical replanning: Theory and experiments. In *22nd International Joint Conference on Artificial Intelligence (IJCAI '11)*. Barcelona, Spain, 2011.

Bonet, B. & Geffner, H., 2014. Flexible and Scalable Partially Observable Planning with Linear Translations. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*., 2014.

Bouguerra, A. & Karlsson, L., 2005. PC-SHOP: A Probabilistic-Conditional Hierarchical Task Planner. *Intelligenza Artificiale*, 2(4), pp.44-50.

Bo, Y. & Zheng, Q., 2009. A Method of Semantic Web Service Composition based PDDL. In *Proceddings of the IEEE International Conference on Service-Oriented Computing and Applications (SOCA '09)*. Taipei, Taiwan, 2009.

Bozkurt, M., Harman, M. & Hassoun, Y., 2010. TR-10-01 *Testing Web Services: A Survey*. Technical Report. King's College London Centre for Research on Evolution, Search & Testing.

Brafman, R.I. & Shani, G., 2012. A multi-path compilation approach to contingent planning. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*. Toronto, Canada, 2012.

Brafman, R.I. & Shani, G., 2012. Replanning in Domains with Partial Information and Sensing Actions. *Journal of Artificial Intelligence Research (JAIR)*, 45, pp.565-600.

Bray, T. et al., 2008. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. [Online] Available at: http://www.w3.org/TR/REC-xml [Accessed 18 April 2015].

Bryant, R.E., 1992. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, pp.293-318.

Bryce, D. & Buffet, O., 2008. International Planning Competition Uncertainty part: benchmarks and results. In *Sixth International Planning Competition (IPC)*., 2008.

Bryce, D., Kambhampati, S. & Smith, D.E., 2004. Planning in belief space with a labelled uncertainty graph. In *Nineteenth National Conference on Artificial Intelligence (AAAI-04) Workshop on Learning and Planning in Markov Decision Processes*. San Jose, California,USA, 2004.

Bryce, D., Kambhampati, S. & Smith, D.E., 2006. Planning Graph Heuristics for Belief Space Search. *Journal of Artificial Intelligence Research (JAIR)*, pp.35-99.

Buffet, O. & Aberdeen, D., 2006. The Factored Policy Gradient Planner. In *Probabilistic Planning Track of the 2006 International Planning Competition - Part of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS 06)*., 2006.

Buffet, O. & Aberdeen, D., 2007. FF+FPG: Guiding a policy-gradient planner. In *Seventeenth International Conference on Automated Planning and Scheduling (ICAPS'07)*. Providence, USA, 2007.

Cabral, L., Li, N., Tymaniuk, S. & Winkler, D., 2011. *Evaluation Design and Collection of Test data for Semantic Web Services Tools v2 (D14.4)*. Deliverable. Seals - Semantic Evaluation at Large Scale. Available for download from: http://about.seals-project.eu/downloads/category/1-?download=74%3Ad14.4.-evaluation-design-and-collection-of-test-data-for-semantic-web-services-tools-v2&start=60.

Calvanese, D. et al., 2008. Automatic Service Composition and Synthesis: the Roman Model. *IEEE Data Engineering Bulletin*, pp.18-22.

Canfora, G. & Di Penta, M., 2006. Testing services and service-centric systems: Challenges and opportunities. *IT Professional*, 8, pp.10–17.

Casati, F. et al., 2000. Adaptive and Dynamic Service Composition in eFlow. In Wangler, B. & Bergman, L.D., eds. *12th International Conference on Advanced Information Systems Engineering (CAiSE 2000)*., 2000. Springer.

Chan, M., Bishop, J. & Baresi, L., 2007. *Survey and Comparison of Planning Techniques for Web Services Composition*. Technical Report. Pretoria: University of Pretoria, South Africa.

Chen, K., Xu, J. & Reiff-Marganiec, S., 2009. Markov-HTN Planning Approach to Enhance Flexibility of Automatic Web Service Composition. In *IEEE International Conference on Web Services (ICWS 2009)*. Los Angeles, CA, USA, 2009.

Chinnici, R., Moreau, J.J., Ryman, A. & Weerawarana, S., 2007. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. [Online] Available at http://www.w3.org/TR/wsdl20/#intro_ws [Accessed 18 April 2015].

CHOREOS, 2011. *CHOReOS: Large Scale Choreographies for the Future Internet*. [Online] Available at: http://www.choreos.eu/bin/view/Main/ [Accessed 14 September 2014].

Christensen, E., Curbera, F., Meredith, G. & Weerawarana, S., 2001. *Web Services Description Language (WSDL) 1.1*. [Online] Available at: http://www.w3.org/TR/wsdl.html [Accessed 18 April 2015].

Chung, M. et al., 2005. Improved Matching Algorithm for Services Described by OWL-S. In *Proceedings of International Conference on Advanced Communication Technology (ICACT)*. Gangwon-Do, Korea, 2005.

Cimatti, A., Pistore, M., M., R. & P., T., 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence Journal (IJAIT)*, pp.35-84.

Clement, L., Hately, A., von Riegen, C. & Rogers, T., 2004. *UDDI Version 3.0.2 - UDDI Spec Technical Committee Draft*. [Online] Available at: http://uddi.org/pubs/uddi_v3.htm [Accessed 29 July 2015].

Clocksin, W.F. & Mellish, C.S., 2003. *Programming in Prolog*. Springer-Verlag.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. & Stein, C., 2001. Depth-first search. In *Introduction to Algorithms*. Second Edition ed. MIT Press and McGraw-Hill. pp.540–49.

Cover, R., 2006. *Electronic Business XML Initiative (ebXML)*. [Online] Available at: http://xml.coverpages.org/ebXML.html [Accessed 31 May 2015].

Cui, L.-z., Li, J. & Zheng, Y., 2012. A Dynamic Web Service Composition Method Based on Viterbi Algorithm. In *IEEE International Conference on Web Services (ICWS 2012)*., 2012.

Curbera, F. et al., 2002. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing*, March - April. pp.86 - 93.

D'Mello, D.A., Ananthanarayana, V.S. & Salian, S., 2011. A Review of Dynamic Web Service Composition Techniques. In N. Meghanathan, B.K. Kaushik & D. Nagamalai, eds. *Advanced Computing - Communications in Computer and Information Science*. Springer Berlin Heidelberg. pp.85-97.

Dacosta, L.A.G., Pires, P.F. & Mattoso, M., 2004. Automatic Composition of Web Services with Contingency Plans. In *International Conference on Web Services Workshop (ICWS '04)*., 2004.

Davies, J..S.R..W.P., 2006. *Semantic Web technologies*. John Wiley.

De Giacomo, G. & Sardina, S., 2007. Automatic Synthesis of New Behaviors from a Library of Available Behaviors. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*., 2007.

Dearden, R. et al., 2003. Incremental contingency planning. In *Thirteenth International Conference on Automated Planning & Scheduling (ICAPS) Workshop on Planning under Uncertainty*. Trento, Italy, 2003.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T., 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6.

DeGiacomo, G., Mecella, M. & Patrizi, F., 2014. Automated service composition based on behaviors: the roman model. In *Web Services Foundations*. Springer.

Deng, S., Huang, L., Tan, W. & Wu, Z., 2014. Top-k Automatic Service Composition: A Parallel Method for Large-Scale Service Sets. *IEEE Transactions on Automation Science and Engineering*, 11(3), pp.891-905.

Digiampietri, L.A., Pérez-Alcázar, J.J. & Medeiros, C.B., 2007. AI Planning in Web Services Composition: a review of current approaches and a new solution. In *Proceedings of the sixth Encontro Nacional de Inteligencia Artificial (ENIA)*., 2007.

Dong, J., Sun, Y., Yang, S. & Zhang, K., 2006. Dynamic Web Service Composition Based on OWL-S. *Science in China Series F: Information Sciences*, pp.843-63.

Du, X., Song, W. & Munro, M., 2006. Using common process patterns for semantic web services composition. In *15th International Conference on Information Systems Development (ISD'06)*., 2006.

EBM Websourcing, 2014. *Petals BPM architecture overview,*. [Online] Available at: http://research.petalslink.org/display/petalsbpm/Petals+BPM+-+Open+source+BPMN+2.0+modeler [Accessed 20 May 2015].

Eid, M.A., Alamri, A. & El-Saddik, A., 2008. A reference model for dynamic web service composition systems. *International Journal of Web and Grid Services*, 4(2), pp.149-68.

Eiter, T. et al., 2003. A logic programming approach to knowledge-state planning, II: The DLVK system. *Artificial Intelligence*, 144(1-2), pp.157–211.

Feenstra, R.W., Janssen, M. & Wagenaar, R.W., 2007. Evaluating Web Service Composition Methods: The need for including Multi-Actor Elements. *Electronic Journal of e-Government (EJEG)* , 5(2), pp.153-64.

Fensel, D. et al., 2010. *WSMO-Lite: Lightweight Semantic Descriptions for Services on the Web*. [Online] Available at: http://www.w3.org/Submission/2010/SUBM-WSMO-Lite-20100823/ [Accessed 18 May 2015].

Fielding, R.T., 2000. *Architectural Styles and the Design of Network-based Software Architecture*. PhD thesis. Irvine, CA, USA: University of California.

Flanagan, D., 2006. *JavaScript: The Definitive Guide*. 5th ed. O'Reilly.

Foss, J., Onder, N. & Smith, D., 2007. Preventing unrecoverable failures through precautionary planning. In *Seventeenth International Conference on Automated Planning & Scheduling (ICAPS) Workshop on Moving Planning and Scheduling Systems into the Real World*. Providence, Rhode Island, USA, 2007.

Free Software Foundation, Inc, 2007. *GNU Affero General Public License*. [Online] Available at: http://www.gnu.org/licenses/agpl.html [Accessed 22 January 2015].

Fu, J., Ng, V., Bastani, F.B. & Yen, I., 2011. Simple and fast strong cyclic planning for fully-observable non-deterministic planning problems. In *Twenty-second International Joint Conference on Artificial Intelligence (IJCAI '11)*. Barcelona, Spain, 2011.

Furcy, D. & Koenig, S., 2005. Limited discrepancy beam search. In *Proceedings of the 19th international joint conference on Artificial intelligence (IJCAI'05)*., 2005.

Garey, M.R. & Johnson, D.S., 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, California, USA: W. H. Freeman and Co.

Geffner, H., 1998. Classical, Probabilistic and Contingent Planning: Three Models, One Algorithm. In *AIPS'98 Workshop on Planning as Combinatorial Search*., 1998.

Geffner, H. & Bonet, B., 2013. *A Concise Introduction to Models and Methods for Automated Planning*. 1st ed. Morgan & Claypool Publishers.

Gerevini, A., Saetti, A. & Serina, I., 2006. An Approach to Temporal Planning and Scheduling in Domains with Predictable Exogenous Events. *Journal of Artificial Intelligence Research (JAIR)*, pp.187-231.

Ghallab, M., Nau, D.S. & Traverso, P., 2004. *Automated Planning: Theory and Practice*. 1st ed. San Francisco, CA: Morgan Kaufmann.

Google, 2011. *Google Web Toolkit Overview*. [Online] Available at: http://code.google.com/webtoolkit/overview.html [Accessed 09 December 2014].

Gosling, J., Joy, B. & Steele, G., 1996. *The Java Language Specification*. Reading, Mass., USA: Addison-Wesley.

Gudgin, M. et al., 2007. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. [Online] Available at: http://www.w3.org/TR/soap12-part1/ [Accessed 18 April 2015].

Guinard, D., 2011. 19891 *A Web of Things Application Architecture - Integrating the Real-World into the Web*. PhD thesis. Zurich, Switzerland: ETH Zurich.

Gunzer, H., 2002. *Borland: Introduction to Web Services*.

Guo, R., Le, J. & Xia, X., 2005. Capability matching of Web services based on OWL-S. In *Proceedings of 16th International Workshop on Database and Expert Systems Applications (DEXA)*., 2005.

Haas, H., 2002. *Web service usage scenario: Travel reservation*. [Online] Available at: http://www.w3.org/2002/04/17-ws-usecase.html [Accessed 15 December 2014].

Hansen, E. & Zilberstein, S., 2001. LAO: A heuristic-search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1-2), pp.35-62.

Hart, P.E., Nilsson, N.J. & B. Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), pp.100–07.

Hatzi, O. et al., 2008. A Synergy of Planning and Ontology Concept Ranking for Semantic Web Service Composition. In Geffner, H..e.a., ed. *11th Ibero-American Conference on Artificial Intelligence (IBERAMIA)*. Lisbon, Portugal, 2008. Springer-Verlag.

Hatzi, O. et al., 2007. Visual Representation of Web Service Composition Problems through VLEPPO. In Velasquez, J.D., ed. *International Workshop on Intelligent Web Based Tools (IWBT07), in conjunction with the 19th IEEE International conference on Tools with Artificial Intelligence (ICTAI07)*. Patra, Greece, 2007.

Hatzi, O. et al., 2010. The PORSCE II Framework: Using AI Planning for Automated Semantic Web Service Composition. *The Knowledge Engineering Review*.

Hatzi, O. et al., 2011. An Integrated Approach to Automated Semantic Web Service Composition through Planning. *IEEE Transactions on Services Computing*.

Helmert, M., 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research (JAIR)*, 26(1), pp.191–246.

Helmert, M., 2009. Concise finite-domain representations for PDDL planning tasks. *Artificial Intelligence* , 173(5-6), pp.503-35.

Hennig, P. & Balke, W.T., 2010. Highly scalable web service composition using binary tree-based parallelization. In *IEEE International Conference on Web Services (ICWS '10)*. Miami, Florida, USA, 2010.

Hodges, W., 2001. Classical Logic I: First Order Logic. In *The Blackwell Guide to Philosophical Logic*. Blackwell.

Hoffmann, J., 2001. FF: The Fast-Forward Planning System. *AI Magazine*, pp.57 – 62.

Hoffmann, J., Bertoli, P., Helmert, M. & Pistore, M., 2009. Message-Based Web Service Composition, Integrity Constraints, and Planning under Uncertainty: A New Connection.. *Journal of Artificial Intelligence Research (JAIR)*, 35, pp.49-117.

Hoffmann, J., Bertoli, P. & Pistore, M., 2007. Web Service Composition as Planning, Revisited: In Between Background Theories and Initial State Uncertainty. In *Proceedings of the 22nd National Conference of the American Association for Artificial Intelligence (AAAI'07)*. Vancouver, Canada, 2007.

Hoffmann, J. & Brafman, R., 2005. Contingent Planning via Heuristic Forward Search with Implicit Belief States. In *15th International Conference on Automated Planning and Scheduling (ICAPS'05)*. Monterey, CA, USA, 2005.

Hoffmann, J. & Brafman, R., 2006. Conformant planning via heuristic forward search: A new approach. *Artificial Intelligence*, 170(6-7), pp.507-41.

Hoffmann, J. & Nebel, B., 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research (JAIR)*, pp.253 – 302.

Hoffmann, J., Porteous, J. & Sebastia, L., 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research (JAIR)*, 22(1), pp.215–78.

Hopcroft, J. & Ullman, J., 1979. *Introduction to Automata Theory, Languages, and Computation*. 1st ed. Reading, Massachusetts, USA: Addison-Wesley.

Horrocks, I. et al., 2004. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. [Online] Available at: http://www.w3.org/Submission/SWRL/ [Accessed 29 April 2015].

Huang, Z., Jiang, W., Hu, S. & Liu, Z., 2009. Effective pruning algorithm for QoS-aware service composition. In *11th IEEE Conference on Commerce and Enterprise Computing (CEC '09)*. Warren, Michigan, USA, 2009.

Hyafil, N. & Bacchus, F., 2003. Conformant probabilistic planning via CSPs. In *Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 03)*. Trento, Italy, 2003.

Information Society Technologies, 2006. *SUPER - Semantics Utilized for Process management within and between EnteRprises*. [Online] Available at: http://cordis.europa.eu/ist/kct/super_synopsis.htm [Accessed 17 May 2015].

iServe Team, 2010. *iServe Vocabulary*. [Online] Available at: http://iserve.kmi.open.ac.uk/wiki/index.php/IServe_vocabulary [Accessed 09 December 2014].

Jensen, R. & Veloso, M., 2000. OBDD-based universal planning for synchronized agents in non-deterministic domains. *Journal of Artificial Intelligence Research (JAIR)*, 13, pp.189–226.

Jiang, W. et al., 2010. QSynth: A tool for QoS-aware automatic service composition. In *8th International Conference on Web Services (ICWS '10)*. Miami, Florida, USA, 2010.

Jiménez, S., Coles, A. & Smith, A., 2006. Planning in probabilistic domains using a deterministic numeric planner. In *Twenty-fifth Workshop of the UK Planning and Scheduling Special Interest Group (PlanSig)*., 2006.

Joslin, D. & Clements, D., 1999. Squeaky wheel optimization. *Journal of Artificial Intelligence Research (JAIR)*, 10, pp.353-73.

Kabanza, F., Barbeau, M. & St-Denis, R., 1997. Planning control rules for reactive agents. *Artificial Intelligence*, 95(1), pp.67–113.

Kaelbling, L.P., Littman, M.L. & Cassandra, A.R., 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, pp.99–134.

Kaldeli, E., 2013. *Domain-Independent Planning for Services in Uncertain and Dynamic Environments*. PhD thesis. University of Groningen.

Kapitsaki, G. et al., 2007. Service Composition: State of the art and future challenges. In *16th IST Mobile and Wireless Communications Summit, 2007*., 2007.

Kautz, H., Selman, B. & Hoffmann, J., 2006. SatPlan: Planning as Satisfiability. In *5th International Planning Competition (IPC 06)*., 2006.

Kavantzas, N. et al., 2004. *Web Services Choreography Description Language Version 1.0*. [Online] Available at: http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/ [Accessed 31 May 2015].

Keyder, E. & Geffner, H., 2008. Heuristics for planning with action costs revisited. In *18th European Conference on AI (ECAI-08)*. Patra, Greece, 2008.

Khadka, R. & Sapkota, B., 2010. An Evaluation of Dynamic Web Service Composition Approaches. In *4th International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing (ACT4SOC '10)*. Athens, Greece, 2010.

Kim, H.S. & Kim, I.C., 2006. Mapping semantic web service descriptions to planning domain knowledge. In *Fourth Int. Federation for Medical and Biological Engineering (IFMBE).*, 2006.

Kissmann, P. & Edelkamp, S., 2008. GAMER: Fully-Observable Non-Deterministic Planning via PDDL-Translation into a Game. In *Fully Observable Non-Deterministic (FOND) track of the sixth International Planning Competition (IPPC) - Part of the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 08)*. Sydney, Australia, 2008.

Kissmann, P. & Edelkamp, S., 2009. Solving Fully-Observable Non-Deterministic Planning Problems via Translation into a General Game. In *32nd Annual German Conference on AI (KI 09)*. Paderborn, Germany, 2009. Springer.

Klusch, M., 2011. *S3 Contest: Retrieval performance evaluation of matchmakers for semantic web services,*. [Online] Available at: http://www-ags.dfki.uni-sb.de/~klusch/s3/html/2011.html [Accessed 22 May 2015].

Klusch, M. & Gerber, A., 2006. Evaluation of Service Composition Planning with OWLS-XPlan. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IATW '06)*. Washington, DC, USA, 2006.

Klusch, M., Gerber, A. & Schmidt, M., 2005. Semantic Web Service Composition Planning with OWLS-Xplan. In *Proceedings of the 1st International AAAI Fall Symposium on Agents and the Semantic Web.*, 2005.

Klusch, M. et al., 2013. *5th International Semantic Service Selection Contest- Performance Evaluation of Semantic Service Matchmakers*. Summary Report.

Kona, S. et al., 2009. WSC-2009: a quality of service-oriented web services challenge. In *11th IEEE Conference on Commerce and Enterprise Computing (CEC).*, 2009.

Kona, S., Bansal, A., Blake, M.B. & Gupta, G., 2008. Generalized Semantics-Based Service Composition. In *IEEE International Conference on Web Services (ICWS)*. Beijing, China, 2008.

Kopecký, J. & Vitvar, T., 2008. *MicroWSMO*. [Online] (0.1) Available at: http://www.wsmo.org/TR/d38/v0.1/ [Accessed 22 July 2015].

Kopecky, J., Vitvar, T., Bournez, C. & Farrell, J., 2007. SAWSDL: Semantic Annotations for WSDL and XML Schema. *IEEE Internet Computing Magazine* , November - December. pp.60 - 67.

Korf, R., 1990. Real-time heuristic search. *Artificial Intelligence*, 42, pp.189-211.

Kuster, U., Konig-Ries, B. & Krug, A., 2008. OPOSSum - An Online Portal to Collect and Share SWS Descriptions. In *2nd IEEE International Conference on Semantic Computing.*, 2008.

Küster, U., Stern, M. & Knig-Ries, B., 2005. A classiffication of issues and approaches in automatic service composition. In *First International Workshop on Engineering Service Compositions at the 3rd International Conference on Service Oriented Computing (ICSOC-05)*., 2005.

Kuter, U., 2004. Pushing the limits of AI planning. In *Doctoral Consortium of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS)*., 2004.

Kuter, U. & Nau, D.S., 2004. Forward-Chaining Planning in Nondeterministic Domains. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI)*. San Jose, California, USA, 2004.

Kuter, U., Nau, D.S., Pistore, M. & Traverso, P., 2005. A Hierarchical Task-Network Planner based on Symbolic Model Checking. In *Fifteenth International Conference on Automated Planning and Scheduling (ICAPS '05)*., 2005.

Kuter, U., Nau, D.S., Reisner, E. & Goldman, R.P., 2008. Using Classical Planners to Solve Nondeterministic Planning Problems. In *Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 08)*. Sydney, Australia, 2008.

Kuter, U. et al., 2005. Information Gathering during Planning for Web Service Composition. *Journal of Web Semantics*.

Kuzu, M. & Cicekli, N., 2012. Dynamic planning approach to automated web service composition. *Applied Intelligence*, 36(1), pp.1-28.

La Placa, M., Pigot, H. & Kabanza, F., 2009. Assistive planning for people with cognitive impairments. In *Proceedings of the Workshop on Intelligent Systems for Assisted Cognition hosted by the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI '09)*., 2009.

Large Scale Distributed Information Systems Lab - University of Georgia, 2005. *Web Service Composition Standards*. [Online] Available at: http://lsdis.cs.uga.edu/proj/meteor/mwscf/standards.html [Accessed 30 May 2015].

Lausen, H., Polleres, A. & Roman, D., 2005. *Web Service Modeling Ontology (WSMO)*. [Online] Available at: "http://www.w3.org/Submission/WSMO/" http://www.w3.org/Submission/WSMO/ [Accessed 10 May 2011].

Leymann, F., 2001. *Web Services Flow Language (WSFL 1.0)*. [Online] Available at: xml.coverpages.org/WSFL-Guide-200110.pdf [Accessed 31 May 2015].

Li, G., Hai, H. & Hu, Z., 2008. A Flexible Framework for Semi-automatic Web Services Composition. In *IEEE Asia-Pacific Services Computing Conference.*, 2008.

Li, Y. et al., 2008. PASS: An Approach to Personalized Automated Service Composition. In *IEEE International Conference on Services Computing (SCC '08)*., 2008.

Lin, W., Kuter, U. & Sirin, E., 2008. Web Service Composition with User Preferences. In *Proceedings of the 5th European Semantic Web Conference on the Semantic Web: Research and Applications (ESWC'08)*., 2008. Springer-Verlag.

Lipovetzky, N. & Geffner, H., 2012. Width and serialization of classical planning problems. In *European Conference of Artificial Intelligence (ECAI 12)*., 2012.

Little, I. & Thiebaux, S., 2007. Probabilistic Planning vs Replanning. In *Seventeenth International Conference on Automated Planning and Scheduling (ICAPS '07) Workshop International Planning Competition: Past, Present and Future*. Providence, Rhode Island, USA, 2007.

Liu, L., Lecue, F., Mehandjiev, N. & Xu, L., 2010. Using Context Similarity for Service Recommendation. In *Proceedings of the Fourth IEEE International Conference on Semantic Computing (ICSC)*. Pittsburgh, PA, USA, 2010.

Long, D. & Fox, M., 2003. The 3rd International Planning Competition: Results and Analysis. *Journal of Artificial Intelligence Research (JAIR)*, 20, pp.1-59.

Luger, G.F. & Stubblefield, W.A., 1993. *Artificial intelligence: structures and strategies for complex problem solving*. 2nd ed. Benjamin/Cummings.

Luo, J. et al., 2006. Adding OWL-S support to the existing UDDI infrastructure. In *IEEE International Conference on Web Services (ICWS '06)*., 2006.

Macdonald, A., 2007. *Service composition with hyper-programming*. Technical Report. University of St Andrews. http://www.cs.st-andrews.ac.uk/~angus/docs/yawsa/final_report.pdf.

Majercik, S.M., 2002. APROPOS2: Approximate probabilistic planning out of stochastic satisfiability. In *AAAI Workshop on Probabilistic Approaches in Search of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*., 2002.

Maleshkova, M., Pedrinaci, C. & Domingue, J., 2009. Supporting the Creation of Semantic Restful Service Descriptions. In *Workshop for Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2) at the 8th International Semantic Web Conference*. Washington, DC, USA, 2009.

Maliah, S., Brafman, R., Karpas, E. & Shani, G., 2014. Partially Observable Online Contingent Planning Using Landmark Heuristics. In *24th International Conference on Automated Planning and Scheduling (ICAPS '14)*. Portsmouth, NH, USA, 2014.

Malu, P. et al., 2002. *ebXML Business Process Specification Schema (BPSS), Version 1.05*. [Online] Available at: xml.coverpages.org/ebBPSSv105-Draft.pdf [Accessed 31 May 2015].

Manzalawy, Y.E., 2009. *JPlan: Java GraphPlan Implementation*. [Online] Available at: http://sourceforge.net/projects/jplan/ [Accessed 21 July 2015].

Marconi, A., Pistore, M. & Traverso, P., 2008. Automated Composition of Web Services: the ASTRO Approach. *IEEE Data Engineering Bulletin*, pp.23 - 26.

Markou, G., 2014a. *Evaluation Domain and Problems*. [Online] Available at: http://ai.uom.gr/gmarkou/Files/EvaluationDomains [Accessed 25 May 2015].

Markou, G., 2014b. *MADSWAN Registry Channel*. [Online] Available at: https://www.youtube.com/channel/UCPGgczUsVhFTCHcCbKN_SrQ [Accessed 18 May 2015].

Markou, G., 2014c. *Use case scenarios' output files screenshots*. [Online] Available at: http://ai.uom.gr/gmarkou/Files/IJAIT/Scenarios(BPMNandScreenshots).rar [Accessed 24 May 2015].

Markou, G., 2015. *Non-Deterministic Evaluation Domain and Problems*. [Online] Available at: http://ai.uom.gr/gmarkou/Files/IJAITMapppaDomains/PDDL/IJAIT-MapppaDomains.zip [Accessed 03 September 2015].

Markou, G., Alexiadis, A. & Refanidis, I.., 2015. Web Services and Automated Planning for Intelligent Calendars. In *6th Italian Workshop on Planning and Scheduling (IPS '15)*. Ferrara, Italy, 2015.

Markou, G. & Refanidis, I., 2012. Towards an automatic non-deterministic web service composition platform. In *Proceedigns of the 8th International Conference on Next Generation Web Services Practices (NWeSP '12)*., 2012.

Markou, G. & Refanidis, I., 2012. Towards Automatic Non-Deterministic Web Service Composition. In *7th International Conference on Internet and Web Applications and Services (ICIW'12)*. Stuttgart, Germany, 2012.

Markou, G. & Refanidis, I., 2013. Composing semantic web services online and an evaluation framework. *International Journal on Advances in Internet Technology*, 6(3-4), pp.114-31.

Markou, G. & Refanidis, I., 2013. Mad Swan: A Semantic Web Service Composition System. In *Proceedings of the 10th Extended Semantic Web Conference (ESWC '13)*. Montpellier, France, 2013.

Markou, G. & Refanidis, I., 2014. Anytime Planning for Web Service Composition via Alternative Plan Merging. In *26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '14)*. Limasson, Cyprus, 2014.

Markou, G. & Refanidis, I., 2015. Cost-Sensitive Probabilistic Contingent Planning for Web Service Composition. *International Journal of Artificial Intelligence Tools (IJAIT)*, (Special issue on ICTAI 2014). To appear.

Markou, G. & Refanidis, I., 2016. Non-Deterministic Planning Methods for Automated Web Service Composition. *Artificial Intelligence Research (AIR)*, 5(1), pp. 14-35.

Martin, D., 2002. *DAML-S: Semantic Markup for Web Services*. [Online] Available at: http://www.daml.org/services/daml-s/0.7/daml-s.html [Accessed 20 May 2015].

Martin, D. et al., 2004. *OWL-S: Semantic Markup for Web Services*. [Online] Available at: http://www.w3.org/Submission/OWL-S/ [Accessed 12 May 2015].

McDermott, D., 1998. CVC TP-98-003/DCS TP-1165 *PDDL—The planning domain definition language*. Technical Report. New Haven, Connecticut, USA: Yale University.

McDermott, D.V., 2002. Estimated-Regression Planning for Interactions with Web Services. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems (AIPS)*. Toulouse, France, 2002.

McGovern, J., Tyagi, S., Stevens, M. & Mathew, S., 2003. *Java Web Services Architecture*. Morgan Kaufmann.

McGuinness, D.L. & van Harmelen, F., 2004. *OWL Web Ontology Language Overview*. [Online] Available at: http://www.w3.org/TR/owl-features/ [Accessed 29 April 2015].

McIlraith, S. & Son, T., 2002. Adapting Golog for Composition of Semantic Web Services. In *Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR-2002)*. Toulouse, France, 2002.

McIlraith, S., Son, T. & Zeng, H., 2001. Semantic Web Services. *IEEE Intelligent Systems. Special Issue on the Semantic Web*, March/April. pp.46-53.

McMillan, K.L., 1992. CMU-CS-92-131 *Symbolic Model Checking*. PhD Thesis. Carnegie Mellon University.

Mediratta, A. & Srivastava, B., 2006. RI 06002 *Applying Planning in Composition of Web Services with a User-Driven Contingent Planner*. Research Report. IBM Research.

Menasce, D.A., 2002. QoS issues in Web services. *IEEE Internet Computing*, November - December. pp.72 - 75.

Mesmoudi, A., Mrissa, M. & Hacid, M., 2011. Combining configuration and query rewriting for web service composition. In *9th IEEE International Conference on Web Services (ICWS '11)*., 2011.

Meuleau, N. & Smith, D.E., 2003. Optimal Limited Contingency Planning. In *Nineteenth conference on Uncertainty in Artificial Intelligence*. Acapulco, Mexico, 2003.

Meyer, H. & Weske, M., 2006. Automated Service Composition using Heuristic Search. In *Proceedings of the Fourth International Conference on Business Process Management (BPM 2006)*. Vienna, Austria, 2006.

Mrohs, B. et al., 2005. OWL-SF – A Distributed Semantic Service Framework. In *Proceedings of the Workshop on Context Awareness for Proactive Systems (CAPS 2005)*. Helsinki, 2005.

Muise, C., Belle, V. & McIlraith, S.A., 2014. Computing Contingent Plans via Fully Observable Non-Deterministic Planning. In *Proceedings of the 1st Workshop on Models and Paradigms for Planning under Uncertainty: a Broad Perspective, at the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*. Portsmouth, New Hampshire, USA, 2014.

Muise, C., McIlraith, S.A. & Beck, J.C., 2012. Improved Non-Deterministic Planning by Exploiting State Relevance. In *22nd International Conference on Automated Planning and Scheduling (ICAPS)*., 2012.

Mulligan, G. & Gracanin, D., 2009. A comparison of SOAP and REST implementations of a service based interaction independence middleware framework. In *Proceedings of the 2009 Winter Simulation Conference (WSC 2009)*., 2009.

Muscholl, A. & Walukiewicz, I., 2008. A Lower Bound on Web Services Composition. *Logical Methods in Computer Science*, 2, p.4.

Nam, W., Kil, H. & Lee, D., 2011. On the computational complexity of behavioral description-based web service composition. *Theoretical Computer Science*, 412(48), pp.6736-49.

Nau, D. et al., 2003. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research (JAIR)*, pp.379–404.

Nau, D.S., Cao, Y., Lotem, A. & Munoz-Avila, H., 1999. SHOP: Simple hierarchical ordered planner. In Dean, T., ed. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*., 1999. Morgan Kaufmann.

Nau, D.S. et al., 2001. Total-Order Planning with Partially Ordered Subtasks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*., 2001.

Nebel, B., Dimopoulos, Y. & Koehler, J., 1997. Ignoring irrelevant facts and operators in plan generation. In *4th European Conference on Planning*., 1997.

Nilsson, N., 1980. *Principles of Artificial Intelligence*. Tioga.

Novet, J., 2015. *Amazon Web Services posts $1.8B in revenue in Q2 2015, up 81% from last year*. [Online] Available at: http://venturebeat.com/2015/07/23/amazon-web-services-posts-1-8b-in-revenue-in-q2-2015-up-81-from-last-year/ [Accessed 24 July 2015].

OASIS, 2007. *Web Services Business Activity (WS-BusinessActivity)*. [Online] (1.1) Available at: http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec-os/wstx-wsba-1.1-spec-os.html [Accessed 28 July 2015].

OASIS, 2009. *Web Services Coordination (WS-Coordination)*. [Online] (1.2) Available at: http://docs.oasis-open.org/ws-tx/wstx-wscoor-1.2-spec.html [Accessed 28 July 2015].

Object Management Group, Inc, 2011. *Business Process Model and Notation (BPMN)*. [Online] (2.0) Available at: http://www.omg.org/spec/BPMN/2.0/ [Accessed 17 May 2015].

Oh, S.-C., Kil, H., Lee, D. & Kumara, S.R.T., 2006. WSBen: A Web Services Discovery and Composition Benchmark. In *IEEE International Conference on Web Services (ICWS '06)*., 2006.

Oh, S. et al., 2009. WSPR*: Web-service planner augmented with A* algorithm. In *11th IEEE Conference on Commerce and Enterprise Computing (CEC '09)*. Washington, DC, USA, 2009.

Oh, S.C., Lee, D. & Kumara, S.R.T., 2006. A Comparative Illustration of AI Planning-based Web Services Composition. In *Proceedings of the 2005 ACM SIGecom Exchanges*., 2006.

Oh, S.-C., Lee, D. & R.T., K.S., 2007. Web service Planner (WsPr): An Effective and scalable Web Service Composition Algorithm. *International Journal of Web Services Research*, 4(1), pp.1-23.

Onaindia, E., Sapena, O., Sebastia, L. & Marzal, E., 2001. SimPlanner: An Execution-Monitoring System for Replanning in Dynamic Worlds. In *10th Portuguese Conference on Artificial Intelligence (EPIA 2001)*., 2001.

Onder, N. & Pollack, M.E., 1999. Conditional, Probabilistic Planning: A Unifying Algorithm and Effective Search Control Mechanisms. In *16th National Conference On Artificial Intelligence (AAAI-99)*. Orlando, FL, USA, 1999.

Oracle, 2011. *UUID Class*. [Online] Available at: http://docs.oracle.com/javase/7/docs/api/java/util/UUID.html [Accessed 10 December 2014].

O'Reilly, T., 2007. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. *Communications & Strategies*, 65, pp.17-37.

Palacios, H., Albore, A. & Geffner, H., 2014. Compiling Contingent Planning into Classical Planning: New Translations and Results. In *Proceedings of the 1st Workshop on Models and Paradigms for Planning under Uncertainty: a Broad Perspective, at the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS 2014)*. Portsmouth, New Hampshire, USA, 2014.

Palacios, H., Geffner, H. &., 2009. Compiling uncertainty away in conformant planning problems with bounded width. *Journal of Artificial Intelligence Research (JAIR)*, 35, pp.623–75.

Paolucci, M. & Wagner, M., 2006. Grounding OWL-S in WSDL-S. In *Proceedings of the International Conference on Web Services (ICWS '06)*., 2006.

Papapanagiotou, P. & Fleuriot, J.D., 2011. A Theorem Proving Framework for the Formal Verification of Web Services Composition. In *Proceedings of the 7th International Workshop on Automated Specification and Verification of Web Systems*. Reykjavik, Iceland, 2011.

Papazoglou, M.P., 2008. *Web services: Principles and technology*. Pearson Educated Ltd.

Papazoglou, M.P. & Heuvel, W.-J., 2007. Service oriented architectures: approaches, technologies and research issues. *The International Journal on Very Large Data Bases (VLDB)*, 16(3), pp.389–415.

Pedrinaci, C. et al., 2010. iServe: a Linked Services Publishing Platform. In *Ontology Repositories and Editors for the Semantic Web Workshop at the 7th Extended Semantic Web Conference*. Heraklion, Greece, 2010.

Peer, J., 2005. *Web Service Composition as AI Planning: A Survey*. Technical Report. St. Gallen, Switzerland: University of St. Gallen. Available at: http://decsai.ugr.es/~faro/CDoctorado/bibliografia/refPlanning4SW/LinkedDocuments/webservice-composition-as-aiplanning-pfwsc.pdf.

PetalsLink, 2011. *ISTA3: 3rd generation of Interoperability for Aeronautics Sub-contracTors*. [Online] Available at: http://research.petalslink.org/display/ista3/ISTA3+Overview [Accessed 16 September 2014].

Pistore, M., Marconi, A., Bertoli, P. & Traverso, P., 2005. Automated composition of web services by planning at the knowledge level. In *19th International Joint Conference on Artificial Intelligence (IJCAI 05)*., 2005.

Pistore, M., Traverso, P. & Bertoli, P., 2005. Automated composition of web services by planning in asynchronous domains. In *15th International Conference on Automated Planning and Scheduling (ICAPS 05)*., 2005.

Ponnekanti, S.R. & Fox, A., 2002. SWORD: A Developer Toolkit for Web Service Composition. In *Proceedings of the 11th International WWW Conference (WWW2002)*., 2002. Elsevier.

Potti, P.K., 2011. *On the Design of Web Services: SOAP vs. REST*. Master's Thesis. College of Computing, Engineering & Construction, University of North Florida (UNF).

ProgrammableWeb, 2014. *ProgrammableWeb Research Center - Growth in Web APIs From 2005 to 2013*. [Online] Available at: http://www.programmableweb.com/api-research [Accessed 24 July 2015].

Provos, N. & Mazières, D., 1999. A Future-Adaptable Password Scheme - OpenBSD. In *Proceedings of the annual conference on USENIX Annual Technical Conference (ATEC '99)*. Monterey, California, USA, 1999.

Prud'hommeaux, E. & Seaborne, A., 2008. *SPARQL Query Language for RDF*. [Online] Available at: http://www.w3.org/TR/rdf-sparql-query/ [Accessed 28 April 2015].

Pryor, L. & Collins, G., 1996. Planning for Contingencies: A Decision-based Approach. *Journal of Artificial Intelligence Research (JAIR)*, pp.287-339.

Puterman, M.L., 1994. *Markov Decision Processes*. Wiley.

Rama Akkiraju, R. et al., 2005. *Web Service Semantics - WSDL-S*. [Online] Available at: http://www.w3.org/Submission/WSDL-S/ [Accessed 02 May 2015].

Ramírez, M., Yadav, N. & Sardiña, S., 2013. Behavior composition as fully observable non-deterministic planning. In *Twenty-third International Conference on Automated Planning and Scheduling (ICAPS '13)*. Rome, Italy, 2013.

Rao, J., Kungas, P. & Matskin, M., 2006. Composition of Semantic Web Services Using Linear Logic Theorem Proving. *Information Systems*, pp.340–60.

Rao, J. & Su, X., 2003. A Survey of Automated Web Service Composition Methods. In *Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*., 2003.

Refanidis, I. & Alexiadis, A., 2011. Deployment and evaluation of SelfPlanner, an automated individual task management system. *Computational Intelligence*, 27(1), pp.41-59.

Richardson, L. & Ruby, S., 2007. *RESTful Web Services*. O'Reilly Media.

Richter, S. & Westphal, M., 2010. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research (JAIR)*, 39(1), pp.127–77.

Rintanen, J., 1999. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research (JAIR)*, 10, pp.323–52.

Rintanen, J., 2002. Backward Plan Construction under Partial Observability. In *Sixth International Conference on Artificial Intelligence Planning Systems*., 2002. AAAI Press.

Rintanen, J., 2003. Product representation of belief spaces in planning under partial observability. In *Eighteenth International Joint Conference on Artificial Intelligence*. Acapulco, Mexico, 2003.

Rodriguez-Mier, P., Mucientes, M. & Lama, M., 2011. Automatic web service composition with a heuristic-based search algorithm. In *9th IEEE International Conference on Web Services (ICWS 2011), Washington DC, USA, 4-9 July, 2011*. Washington DC, USA, 2011.

Rodríguez-Mier, P., Mucientes, M., Vidal, J.C. & Lama, M., 2012. An Optimal and Complete Algorithm for Automatic Web Service Composition. *International Journal of Web Services Research*, pp.1-20.

Rodriguez-Mier, P., Pedrinaci, C., Lama, M. & Mucientes, M., 2015. An Integrated Semantic Web Service Discovery and Composition Framework. *IEEE Transactions on Services Computing*, Forthcoming.

Rosenberg, F. et al., 2010. Metaheuristic Optimization of Large-Scale QoS-aware Service Compositions. In *IEEE International Conference on Services Computing (SCC '10)*., 2010.

RuleML Initiative, 2011. *The Rule Markup Initiative*. [Online] Available at: http://ruleml.org/ [Accessed 29 April 2015].

Russell, S. & Norvig, P., 2003. Chapter 11: Planning. In *Artificial Intelligence: A Modern Approach*. 2nd ed. Upper Saddle River, NJ, : Prentice Hall. pp.375–459.

S3 Contest, 2011. *Retrieval Performance Evaluation of Matchmakers for Semantic Web Services*. [Online] Available at: http://www-ags.dfki.uni-sb.de/~klusch/s3/html/2011.html [Accessed 12 December 2014].

Schlimmer, J.C., 2002. *Web Services Description Requirements - W3C Working Draft*. [Online] Available at: http://www.w3.org/TR/ws-desc-reqs/ [Accessed 17 April 2015].

Schmidt, M., 2005. *Ein effizientes Planungsmodul fuer die lokale Planungsebene eines InteRRaP Agenten*. Master's thesis. Universitaet des Saarlandes.

Schuster, H., Georgakopoulos, D., Cichocki, A. & Baker, D., 2000. Modeling and composing service-based and reference process-based multi-enterprise processes. In *Proceeding of the 12th International Conference on Advanced Information Systems Engineering (CAiSE)*. Stockholm, Sweden., 2000. Springer Verlag.

Semantic Technology Institute International Working Group, 2009. *Minimal Service Model*. [Online] Available at: http://cms-wg.sti2.org/minimal-service-model/ [Accessed 09 December 2014].

SemWebCentral, 2008. *SAWSDL service retrieval test collection*. [Online] (4.0) Available at: http://projects.semwebcentral.org/projects/sawsdl-tc/ [Accessed 17 May 2015].

SemWebCentral, 2010. *OWL-S Service Retrieval Test Collection*. [Online] Available at: http://semwebcentral.org/frs/?group_id=89 [Accessed 28 October 2014].

Shani, G. & Brafman, R.I., 2011. Replanning in Domains with Partial Information and Sensing Actions. In *Twenty-second International Joint Conference on Artificial Intelligence (IJCAI-11)*. Barcelona, Spain, 2011.

Shani, G., Brafman, R., Maliah, S. & Karpas, E., 2013. Heuristics for Planning under Partial Observability with Sensing Actions. In *Heuristics and Search for Domain-independent Planning (HSDIP)*

workshop at the 23rd International Conference on Automated Planning and Scheduling (ICAPS '13). Rome, Italy, 2013.

Sheng, Q.Z., Benatallah, B., Maamar, Z. & Ngu, A.H.H., 2009. Configurable Composition and Adaptive Provisioning of Web Services. *IEEE Transactions on Services Computing*, 2(1), pp.34-49.

Sheng, Q.Z. et al., 2014. Web services composition: A decade's overview. *Information Sciences*, pp.218–38.

Shin, D.H. & Lee, K.H., 2007. An Automated Composition of Information Web Services based on Functional Semantics. In *IEEE Workshop on Web Service Composition and Adaptation (WSCA '07)*., 2007.

Sirbu, A. & Hoffmann, J., 2008. Towards Scalable Web Service Composition with Partial Matches. In *Proceedings of the 6th IEEE International Conference on Web Services (ICWS'08)*. Beijing, China, 2008.

Sirin, E., 2004. *Automated Composition of Web Services using AI Planning Techniques*. Master's Thesis. Department of Computer Science, University of Maryland.

Sirin, E., Hendler, J. & Parsia, B., 2002. Semi-automatic composition of Web services using semantic descriptions. In *Proceedings of Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS 2003*., 2002.

Sirin, E., Hendler, J.A. & Parsia, B., 2003. Semi-automatic Composition of Web Services using Semantic Descriptions. In *Proceedings of the 1st Workshop on Web Services: Modeling, Architecture and Infrastructure (WSMAI)*. Angers, France, 2003.

Sirin, E. et al., 2007. Pellet: A Practical OWL DL Reasoner. *Journal of Web Semantics*.

Sirin, E. et al., 2004. HTN planning for web service composition using SHOP2. *Journal of Web Semantics*, 1(4), pp.377-96.

Sniedovich, M., 2010. *Dynamic Programming: Foundations and Principles*. 2nd ed. CRC Press.

SOA4All, 2011. *Service Oriented Architectures for All (SOA4All)*. [Online] Available at: http://www.soa4all.eu/ [Accessed 08 December 2014].

Srinivasan, N., Paolucci, M. & Sycara, K., 2004. Adding OWL-S to UDDI, Implementation and Throughput. In *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*., 2004.

Srivastava, B. & Koehler, J., 2003. Web Service Composition - Current Solutions and Open Problems. In *Proceedings of the 13th International Conference on Automated Planning & Scheduling (ICAPS)*. Trento, Italy, 2003.

STI International, 2011. *Test Beds & Challenges Service*. [Online] Available at: http://testbeds-challenges.sti2.org/index.php/semantic-web-services.html [Accessed 12 December 2014].

Sutton, R.S. & Barto, A.G., 1998. *Reinforcement Learning: An Introduction*. MIT Press.

SYNERGY, 2011. *Synergy project*. [Online] Available at: http://www.synergy-ist.eu/ [Accessed 15 September 2014].

Tabatabaei, S.G.H., Kadir, W.M.N.W. & Suhaimi, I., 2008. An Evaluation of Current Approaches for Web Service Composition. In *International Symposium on Information Technology (ITSim 2008)*., 2008.

Teichteil-Koenigsbuch, F., Infantes, G. & Kuter, U., 2008. RFF: A Robust, FF-Based MDP Planning Algorithm for Generating Policies with Low Probability of Failure. In *Third International Planning Competition (IPPC-ICAPS '08)*., 2008.

Thatte, S., 2001. *XLANG: Web Services for Business Process Design*. [Online] Available at: www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm. [Accessed 31 May 2015].

The Workflow Management Coalition, 2012. WFMC-TC-1025 *Process Definition Interface - XML Process Definition Language*. Technical Report.

Thulasiraman, K. & Swamy, M.N.S., 1992. Acyclic Directed Graphs. In *Graphs: Theory and Algorithms*. John Wiley and Son. p.118.

Tilkov, S., 2010. *UDDI R.I.P.* [Online] Available at: https://www.innoq.com/blog/st/2010/03/uddi-r.i.p./ [Accessed 30 June 2015].

To, S.T., Pontelli, E. & Son, T.C., 2009. A conformant planner with explicit disjunctive representation of belief states. In *19th International Conference on Automated Planning and Scheduling (ICAPS 09)*. Thessaloniki, Greece, 2009.

To, S.T., Son, T.C. & Pontelli, E., 2011. Contingent Planning as AND/OR forward Search with Disjunctive Representation. In *21st International Conference on Automated Planning and Scheduling (ICAPS)*. Freiburg, Germany, 2011.

URI Planning Interest Group, W3C/IETF, 2001. *URIs, URLs, and URNs: Clarifications and Recommendations*. [Online] (1.0) Available at: http://www.w3.org/TR/uri-clarification/ [Accessed 19 July 2015].

Vaculín, R. & Sycara, K., 2007. Monitoring execution of OWL-S web services. In *Proceedings of the OWL-S: Experiences and Directions Workshop at the 4th European Semantic Web Conference (ESWC)*. Innsbruck, Austria, 2007.

van der Aalst, W.M.P., 2003. Don't go with the flow: Web services composition standards exposed. *IEEE Intelligent Systems*, pp.72 – 76.

van der Aalst, W.M.P., Dumas, M. & ter Hofstede, A.H.M., 2003. Web Service Composition Languages: Old Wine in New Bottles? In *Proceedings of the 29th Euromicro Conference*., 2003.

van Harmelen, F., Patel-Schneider, P.F. & Horrocks, I., 2001. *Reference description of the DAML+OIL ontology markup language*. [Online] Available at: http://www.daml.org/2001/03/reference [Accessed 20 May 2015].

Viterbi, A.J., 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2), pp.260–69.

W3C OWL Working Group, 2009. *OWL 2 Web Ontology Language Document Overview*. [Online] Available at: http://www.w3.org/TR/owl2-overview/ [Accessed 29 April 2015].

W3C RDF Working Group, 2004. *Resource Description Framework (RDF)*. [Online] Available at: http://www.w3.org/RDF/ [Accessed 28 April 2015].

Wagner, F., Ishikawa, F. & Honiden, S., 2011. QoS-aware Automatic Service Composition by Applying Functional Clustering. In *9th IEEE International Conference on Web Services (ICWS 2011)*. Washington DC, USA, 2011.

Wagner, F. et al., 2012. Multi-objective service composition with time-and input-dependent QoS. In *19th IEEE International Conference on Web Services (ICWS)*. Miami, Florida, USA, 2012.

Wang, P. et al., 2015. Automatic Web Service Composition Based on Uncertainty Execution Effects. *IEEE Transactions on Services Computing*, Forthcoming.

Weld, D.S., Anderson, C.R. & Smith, D.E., 1998. Extending Graphplan to Handle Uncertainty and Sensing Actions. In *Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference (AAAI/IAAI 98)*. Madison, Wisconsin, USA, 1998.

Wu, B., Li, Y., Wu, J. & Yin, J., 2011. AWSP: An automatic Web service planner based on heuristic state space search. In *9th IEEE International Conference on Web Services (ICWS 2011)*. Washington, DC, USA, 2011.

Wu, D. et al., 2003. Automatic Web Services Composition Using SHOP2. In *ICAPS '03 Workshop on Planning for Web Services*, 2003.

Yoon, S., Fern, A. & Givan, R., 2007. FF-Replan: A baseline for probabilistic planning. In *Seventeenth International Conference on Automated Planning and Scheduling (ICAPS '07)*. Providence, Rhode Island, USA, 2007.

Yoon, S. et al., 2008. Probabilistic Planning via Determinization in Hindsight. In *Twenty-third Conference on Artificial Intelligence (AAAI 08)*., 2008.

Yoon, S., Ruml, W., Benton, J. & Do, M.B., 2010. Improving Determinization in Hindsight for On-line Probabilistic Planning. In *20th International Conference on Automated Planning and Scheduling (ICAPS 2010)*. Toronto, Canada, 2010.

Younes, H. & Littman, M., 2004. *International Probabilistic Planning Competition*. [Online] Available at: http://www.cs.rutgers.edu/~mlittman/topics/ipc04-pt/ [Accessed 28 October 2014].

Younes, H. & Littman, M., 2004. CMU-CS-04-167 *PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects*. Technical Report. Carnegie Mellon University.

Yu, H.Q. & Reiff-Marganiec, S., 2006. *Semantic Web Services Composition via Planning as Model Checking*. Technical Report (CS-06-003). University of Leicester.

Zahoor, E., Perrin, O. & Godart, C., 2009. Rule-Based Semi Automatic Web Services Composition. In *Proceedings of SERVICES I 2009.*, 2009.

Zaremba, M., Moran, M. & Haselwanter, T., 2006. Applying Semantic Web Services to Virtual Travel Agency Case Study. In *Proceedings of the 1st Workshop SemWiki2006From Wiki to Semantics co-located with the 3rd Annual European Semantic Web Conference ESWC*. Budva, Montenegro, 2006.

Zeng, L. et al., 2004. QoS-aware middleware for Web services composition. *IEEE Transactions on Software Engineering*, 30(5), pp.311 - 327.

Zhao, H. & Doshi, P., 2009. Towards Automated RESTful Web Service Composition. In *7th IEEE International Conference on Web Services (ICWS '09)*. Los Angeles, CA, USA, 2009.

Ziaka, E., Vrakas, D. & Bassiliades, N., 2011. Translating Web Services Composition Plans to OWL-S Descriptions. In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence*. Rome, Italy, 2011.

Zou, G. et al., 2012. Towards Automated Choreographing of Web Services Using Planning. In *Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12).*, 2012.

Zúñiga, J.C., Pérez Alcázar, J.d.J. & Digiampietri, L.A., 2010. Implementation Issues for Automatic Composition of Web Services. In *Proceedings of the 2010 Workshops on Database and Expert Systems Applications (DEXA '10).*, 2010.

Zuñiga, J.C., Pérez-Alcázar, J..J. & Digiampietri, A..L., 2014. A loosely coupled architecture for automatic composition of web services applications. *International Journal of Metadata, Semantics and Ontologies*, 9(3), pp.241-51

Appendix A.

Use Cases Scenarios

Ap.A.1. The Movie Search Use Case

The WSs from the Communication domain of OWL-S TC4 that are relevant to the achievement of the user's goal in the movie search scenario are presented in Table Ap.A.1. The concepts that appear more than once are colored identically; those that share both a common input and a common output can substitute the use of another (fully or partially). For example, the "*titlecomedyfilmBFservice*", "*titlecomedyfilmMegaservice*", "*titlecomedyfilmservice*" WSs in Table Ap.A.1 are in essence identical and can be used interchangeably producing the same output.

We have added three WSs, with the following intention: "*filmtitleservice*" has the exact opposite functionality than "*titlefilmservice*", that is, it produces as output an ontological concept my_ontology.owl#Film based on the input books.owl#Title. As such, it can be used as input to most of the WSs in Table Ap.A.1. "*filmcomedyfilmservice*" was added so that the planner can circumvent the composition of "*filmtitleservice*" and the services that take as input a title and output a comedy film. As such, an optimal planner should always prefer to use it instead of creating a sequence of "*filmtitleservice*" then "*titlecomedyfilmservice*" to find a comedy film. Finally, "*comedyfilmpriceservice*" was added to be used in conjunction with "*filmcomedyfilmservice*" as an alternative to the sequence "*filmtitleservice*" then "*titlecomedyfilmpricequalityservice*" (both sequences include two steps, so using one or another is not – on its own – better). It should be noted that some of the ontological concepts in the following table are not relevant to the movie search scenario and the planner should ignore them during the WSC process; such examples are finance_th_web.owl#quality and my_ontology.owl#ActionFilm.

Table Ap.C.1: Semantic Web Services relevant to the movie search use case

	Name	Input(s)	Output(s)		
	comedyfilmactionfilmservice	-	Comedyfilm	Actionfilm	
Same	titlecomedyfilmBFservice	Title	Comedyfilm		
Same	titlecomedyfilmMegaservice	Title	Comedyfilm		
Same	titlecomedyfilmservice	Title	Comedyfilm		
	titlecomedyfilmmaxpricequalityservice	Title	Comedyfilm	Maxprice	Quality
	titlecomedyfilmpricequalityservice	Title	Comedyfilm	Price	Quality
	titlecomedyfilmrecommended pricequalityservice	Title	Comedyfilm	Recommendedprice	Quality
	titlecomedyfilmtaxedpricequalityservice	Title	Comedyfilm	Taxedprice	Quality
	titlecomedyfilmtaxfreepricequalityservice	Title	Comedyfilm	Taxfreeprice	Quality
	titlefilmservice	Title	Film		

Added or modified:

Name	Input(s)			Output(s)
filmtitleservice	Film			Title
filmcomedyfilmservice	Film			Comedyfilm
comedyfilmpriceservice	Comedyfilm			Price
TitleSavingservice	Title	Maxprice	Price	Database
	Recommendedprice	Taxedprice	Taxfreeprice	

Ap.A.2. The E-Bookstore Use Case

67 descriptions in the Education domain are related to the e-bookstore use case scenario, that is refer to specific kind of books, their authors, ISBN numbers or various publication services; these are presented in Table Ap.A.2

Table Ap.A.2: Descriptions relevant to books in the education domain of OWL-S TC

_author_CompJservice	novel_authorcommitting_service
_author_DMservice	novel_authorgenre_service
_publication_PPservice	novel_authortime_service
_book_Oracleservice	novel_person_Reserverservice
AcademicBookNumberOrISBNSearch	novel_person_Writerservice
AcademicBookNumberSearch	novel_userreviewauthor_service
academic-item-number_book_service	printedmaterial_price_service
academic-item-number_bookauthor_service	printedmaterialperson__service
book_author_EncSSservice	publication_author_service
book_author_service	publication_book_service
book_authorbook-type_service	publication_number_publication_service
book_authorprice_Novelservice	publication-number_book_Portalservice
academic-item-number_publication_service	publication-number_book_service
book_authortext_service	publication-number_bookauthor_service
academic-item-number_publicationauthor_service	publication-number_bookauthorpublisher_service
book__ShoppingCartservice	publication-number_currencypublication_service
book_person_Publisherservice	publication-number_edited-book_service
book_pricesizebook-type_service	publication-number_publication_service
book_publisher_service	publication-number_publicationauthor_service
book_readerreview_service	publicationperson__service
book_readerreviewperson_service	researcher-in-academia_publication-referencepostal-address_
BookFinder	researcher-in-academia_publication-referencepostal-address_service
BookSearchService	research-fellow-in-academia_publication-reference_service
isbn_book_service	romanticnovel_authorbook-type_service
isbn_bookauthor_service	sciencefictionbook_author_service
isbn_publication_service	sciencefictionbook_authorbook-type_service
isbn_publicationauthor_service	sciencefictionnovel_author_MyOntoservice
isbn_publicationpublisher_service	science-fiction-novel_authorbook-type_service
monograph_author_service	sfnovel_authorauthor_BookOntoservice
novel_author_BookOntoservice	short-story_author_service
novel_author_MyOntoservice	short-story_authorbook-type_service
novel_author_service	text_publication_service
novel_authorbook-type_service	

Furthermore, 29 SWS descriptions in the economy domain of both OWL-S TC4 and version 2.2. revision 1 relate to books. These are shown in Table Ap.A.3.

Table Ap.A.3: Book related descriptions in the economy domain of OWL-S TC

author_bookmaxprice_service	book_taxedprice_service
author_bookprice_service	book_taxedpriceprice_service
author_bookrecommendedprice_service	bookperson__service
author_booktaxedprice_service	bookperson_price_service
author_booktaxfreeprice_service	bookpersoncreditaccount__Beaservice
book_authorprice_Novelservice	bookpersoncreditaccount__service
book_authorprice_service	bookpersoncreditcardaccount__BShopservice
book_Cheapestprice_service	bookpersoncreditcardaccount__service
book_price_service	bookpersoncreditcardaccount_price_service
book_pricereviewbook_service	bookpersoncreditcardaccount_recommendedprice_service
book_pricesizebook-type_service	bookpersoncreditcardaccount_taxedfreeprice_service
book_recommendedprice_RegisteredUserservice	bookpersonOptional_price_service
book_recommendedprice_service	BookPrice
book_recommendedpriceindollar_service	bookusercreditcardaccount__service
book_reviewprice_service	

Table Ap.A.4 will analyze the IOPEs of some of the services mentioned in Tables Ap.A.2-3, as well as those that were added by us in order to facilitate the WSC process described in the e-bookstore use case.

According to the finance_th_web.owl ontology, there are two methods of payment (finance_th_web.owl#payment): cash and credit. Moreover, the finance_th_web ontology defines three different types of cards, namely credit, cheque and debit cards; these concepts, however, are only stated to be subclasses of finance_th_web.owl#card (and not of the credit concept), which in turn is a subclass of finance_th_web.owl#financial_instrument. That is, the ontology does not define them to be related to either the credit or the cash payment concepts. For this reason we will slightly alter this ontology and add the debit and cheque cards as subclasses of cash, and the credit card concept as a subclass of credit. The finance_th_web ontology as it will be used in our scenario is shown in Figure Ap.A.1 (as shown in the Protégé Ontology Editor[26]).

[26] http://protege.stanford.edu/

Table Ap.C.4: : Examples of IOPES of web services relevant to the e-bookstore use case

	Name	**Input(s)**	**Output(s)**		
Same	book_author_EncSSservice	Book	Author		
	book_author_service	Book	Author		
	book_authortext_service	Book	Author	Text	
	book_price_service	Book	Price		
	book_readerreview_service	Book	Reader	Review	
	book_sizebook-type_service	Book	Size	Book-Type	
Same	BookFinder	Title	Book		
	BookSearchService	Title	Book		
	isbn_book_service	ISBN	Book		
	novel_author_BookOntoservice	Novel	Author		
	publication_book_service	Publication	Book		
	Name	**Inputs**	**Preconditions**	**Outputs**	**Effects**
	book__ShoppingCartservice	Book	be-available		ShoppingCartRequestItems
	Added:				
	Name	**Input(s)**			**Output(s)**
	credit_card_charge_service	Credit_Card	Price		Retail_Sale
	debit_card_charge_service	Debit_Card	Price		Retail_Sale
	cheque_card_charge_service	Cheque_Card	Price		Retail_Sale
	credit_charge_service	Credit	Price		Retail_Sale
	purchase_service	ShoppingCartRequestItems	Retail_Sale	Address	PurchasedItems

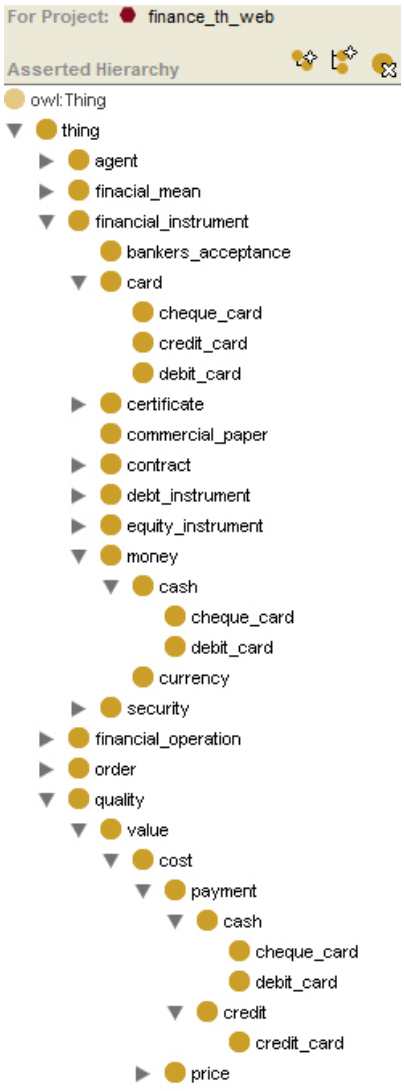

Figure Ap.A.0.1: The finance_th_web ontology

Ap.A.3. The Camera Purchase Use Case

28 descriptions in the Economy domain are relevant to the camera purchase scenario; these services either refer to cameras in their filename, or in their textual description in the OWL-S files:

Table Ap.A.5: Descriptions relevant to cameras in the economy domain of OWL-S TC

_aps-slrpricereport_Musuemservice	retailstore_slrtaxedprice_service
_cameraprice_MyShopservice	shoppingmall_analogpricecalendar-date_service
_camerataxedprice_service	shoppingmall_calendar-datepricecamera_service
_camerataxedpricedutytax_service	shoppingmall_cameraprice_service
_digitalstandardpriceprice_MediaMarktservice	shoppingmall_compactprice_service
_price_CannonCameraservice	shoppingmall_compacttaxedprice_service
_pricecamera_Wallmartservice	shoppingmall_digital-slrpricecalendar-date_service
CCP_service	shoppingmall_maxpricedigital-video_service
KodakDigCamera_price_service	shoppingmall_pricecellphonewithcamera_service
mercantileorganization_compactprice_service	shoppingmall_pricedigitalanalog_service
mercantileorganization_slrprice_service	shoppingmall_pricepurchaseableitemrange_service
PhillipDigCamera_price_service	shoppingmall_purchaseableitemprice_service
retailstore_compactprice_service	shoppingmall_slrprice_service
retailstore_slrprice_service	SRcamera_service

As mentioned in the description of the scenario, we have added or modified several of the SWSs in this domain in order to make our scenario more meaningful. We added two subclasses to the "RetailStore" class of Mid-level-ontology.owl; specifically, in order to modify the _pricecamera_Wallmartservice and _digitalstandardpriceprice_MediaMarktservice WSs to accept as input a specific store of their brand, we have added a MediaMarktStore and a WalmartStore concept as its subclasses, as seen in Figure Ap.A.2. We also modified the _pricecamera_Wallmartservice WS even further, so that it only specializes in selling only SLR cameras and not all cameras in general. Furthermore, we added a WS (Camera_Shopping_Cart_Service) that checks whether a specific camera (extendedCamera.owl#Camera) and its product code (extendedCamera.owl#ProductCode) are available (ontosem.owl#be-available) at a specific mercantile organization (SUMO.owl# MercantileOrganization), and if so, adds the product in the client's shopping cart (ShoppingCart.owl#Shopping CartRequestItems). For every description that outputs a camera model and its price, we added the product code concept as a required input. As such, we do not mention every modified service; only those we have modified more extensively, the ones that were not present in the domain and we added ourselves, and a sample of the rest of the modified services (the ones with the product code concept addition). These are presented in Table Ap.A.6.

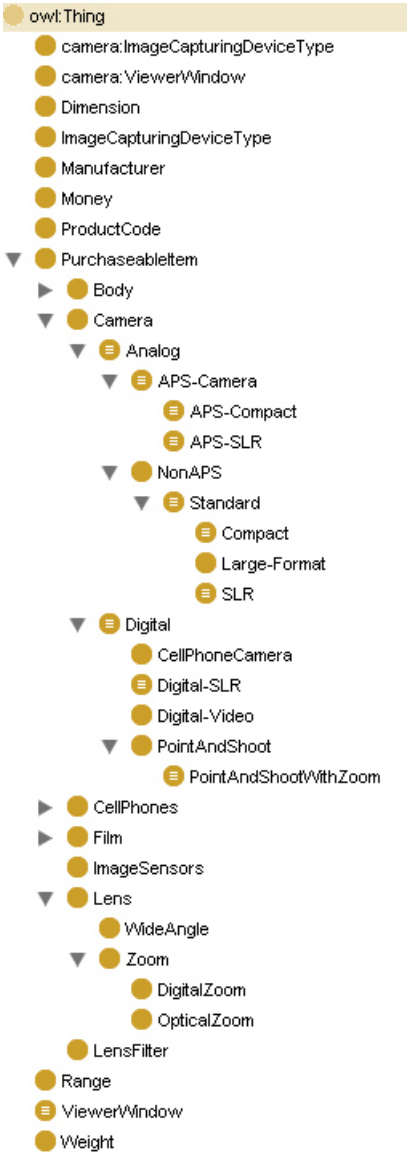

Figure Ap.A.0.2: The extended camera ontology

Table Ap.C.6: Examples of modified descriptions from the economy domain of OWL-S TC

Service	**Input(s)**				**Output(s)**			
Cameraprice_Myshopservice	ProductCode				Camera		Price	
Digitalstandardpriceprice_ Mediamarktservice	MediaMarkt Store		ProductCode		Digital	Standard	Price	Price1
Price_Cannoncameraservice	ProductCode				Price			
Pricecamera_Wallmart service	WalmartStore		ProductCode		Camera		Price	
CCP_Service	ShoppingMall		ProductCode		Compact	RecommendedPriceInEuro		
mercantileorganization_ compactprice_service	Mercantile Organization		ProductCode		Compact		Price	
retailstore_compactprice_ service	RetailStore		ProductCode		Compact		Price	
shoppingmall_cameraprice_ service	ShoppingMall		ProductCode		Camera		Price	
shoppingmall_slrprice_service	ShoppingMall		ProductCode		SLR		Price	
Service	**Input(s)**			**Precondition(s)**	**Output(s)**		**Effect(s)**	
Camera_Shopping_ Cart_Service	ProductCode	Mercantile Organization	Camera	be-available			ShoppingCart RequestItems	

Finally, Figure Ap.A.3 presents the extendedCamera ontology in which the relevant camera concepts are included, so that the relation between the different camera types is better understood. Based on the ontology, cameras are divided in two categories, analog and digital; Analog cameras are further divided in two sub-categories, APS (Advanced Photo System) cameras and non-APS cameras.

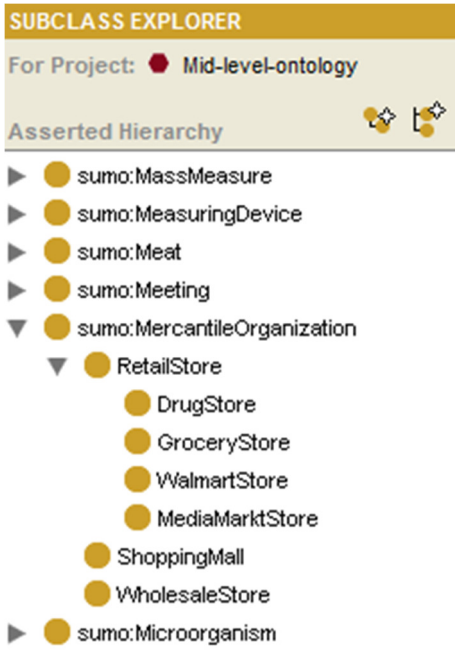

Figure Ap.A.0.3: The MercantileOrganization class and its subclasses

Ap.A.4. The Complex Contingent Use Case Scenario

54 descriptions in the Geography domain are relevant to the complex contingent use case scenario; these are shown in Table Ap.A.7.

As mentioned in the description of the scenario, we have added or modified several of the SWSs in this domain in order to make our scenario more meaningful. Here we do not mention every modified service; only those we have modified more extensively and the ones that were not present in the domain and we added ourselves, as shown in Table Ap.A.8.

Table Ap.A.7: Descriptions relevant to cameras in the Geography domain of OWL-S TC

addressDistanceCalculator	getLocationOfUSCity
addressGeocoder	getLocationOfUSZipcode
calculateDistanceInMiles	getLocationOfZipCodeWorldwide
calculateDistanceUsingSphericalGeometry	getMapOfAddress
calculateSunriseTime	getMapOfUSAddress
calculatorDistanceSphericalLawOfCosines	getPlaceOfAddress
checkAndLookupAddress	getPopulationDensityOfLocation
city_state_ZipCodes	getSunsetAndSunriseTime
findNearbyPostalCodes	getSunsetSunriseTimeOfLocation
findNearbyWikipediaArticles	getSunsetSunriseTwilightTime
findPlaceNamePostalCode	getTrafficInformation
gazetteerLookupLocation	getUSZipCodeLocation
geocodeUSAddress	getZipcodeForUSCity
getAddressOfLocation	getZipCodeInfo
getATMLocationsInCity	getZipCodeInfoWorldwide
getCoordinatesOfAddress	getZipCodeLocation
getDistanceBetweenCitiesWorldwide	getZipCodesWithinCityState
getDistanceBetweenLocations	googleGeocodingAPI
getDistanceBetweenPlaces	mileToKilometerConverter
getDrivingDirections	queryParserLocation
getGeographicAreaOfZipCode	real-time_geocoding
getListOfMatchingCities	real-time_geocodingStreetAddress
getLocationOfAddress	renderMapService
getLocationOfAddressWorldwide	searchFormattedAddress
getLocationOfAddressYahooMaps	searchRawAddress
getLocationOfCityState	usPostalCode_distance
getLocationOfCityWorldwide	uszipcode_distance

Table Ap.A.8. Web services added for the purposes of the complex contingent use case

Service	**Input(s)**		**Output(s)**		
destinationAddressGeocoder	Address		Latitude	Longitude	
distanceTomMileConverter	Distance		Mile		
pricecamera_Wallmart service	Latitude	Longitude	City	State	Country
getCountryCode	Country		CountryCode		

Index of Terms

N

O

P

Q

R

S

U

W

X

Y