Dynamic Web Services Composition

A Study of Services Composition Process to find an efficient and optimal solution by using AI Planning Concepts

by

Faisal Mustafa

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Doctor of Philosophy

April 2014

Copyright Statement

Acknowledgements

My sincere appreciation and regards goes to my supervisors Prof. Lee McCluskey and Dr. Gary Allen for their help and guidance. I would like to take this opportunity to thank my principal supervisor Professor Lee for his patience, guidance and appreciation throughout this thesis.

In addition, I appreciate the assistance of Ms Julie Wilkinson and ARTFORM research group. I would like to acknowledge the financial, academic and technical support of the University of Huddersfield.

To my parents for their support, prayers, encouragement and dreams, for staying with me in every hardship and showing me the candle of kindness for every single arduous abstraction of life.

To my two little flowers Subhaan and Eiman for spending some of their best life moments apart from me while allowing me to focus on this thesis.

Last, but by no means least, to my wife who deserve the most thanks for her endless encouragement, constant patience and prayers.

—

Abstract

Emerging web services technology has introduced the concept of autonomic interoperability and portability between services. The number of online services has increased dramatically with many duplicating similar functionality and results. Composing online services to solve user needs is a growing area of research. This entails designing systems which can discover participating services and integrate these according to the end user requirements.

This thesis proposes a Dynamic Web Services Composition (DWSC) process that is based upon consideration of previously successful attempts in this area, in particular utilizing AI-planning based solutions. It proposes a unique approach for service selection and dynamic web service composition by exploring the possibility of semantic web usability and its limitations.

It also proposes a design architecture called Optimal Synthesis Plan Generation framework (OSPG), which supports the composition process through the evaluation of all available solutions (including all participating single and composite services). OSPG is designed to take into account user preferences, which supports optimality and robustness of the output plan. The implementation of OSPG will be configured and tested via division of search criteria in different modes thereby locating the best plan for the user. The services composition and discovery-based model is evaluated via considering a range of criteria, such as scope, correctness, scalability and versatility metrics.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Preliminaries

During the last 20 years, the Internet has emerged as a ubiquitous aspect of society, situating computing function at the heart of business, personal, entertainment and educational activity [134]. There are well-established arrangements for utilizing online resources through the daily use of e-tools, which interact with digital libraries and integrated databases; making the World Wide Web (W3C) increasingly the first stop for information [39]. The Internet provides an enormous amount of information and it is regarded by the majority as one of the greatest inventions of the 20th century. Currently the internet is growing exponentially in size, processing power and software sophistication, making it the fastest growing technology in the world [93, 99, 167].

According to W3C, *web services are pieces of software, designed in a particular way, to facilitate interaction amongst machines, and to provide a machine processable arrangement for other applications to interact with each other* [27, 140]. The current web was initially designed for the use of human beings, with the system (hardware resources) not understanding the contents (information) of services [77].

The need for composing existing web services into composite services (for user support) is also increasing but full automation (services integration) is not possible without providing a description to the system about the meaning of the basic contents [158]. It is only possible to develop the aforementioned concept of W3C if the system (computer) is able to understand the inherited meanings of the concepts [75]. But as discussed, the contents of the current web (or online available services interfaces) are only understandable by human operators, and not by systems (machines) [75]. As such, it is very difficult to adopt the characteristics of automatic integration processes [144]. For example, it will be difficult to automate an annual pay online management system (as shown in Figure 1.1) which generates reports after integrating and collaborating via different online services containing information such as tax, bank details, employee record, extra allowances and national insurance contribution. Such composing environments are difficult to develop as a system [1] cannot understand the inherited meaning of all used concepts, which are in use by different services [163]. The pay online management system needs a high level of effort and technical skills to integrate, as all services are using different concepts and technical terms. For example without differentiating and considering the different instances of pay (pay, salary and wages), all participating services are unable to communicate. As all services are using different concepts, so semantic similarity between entities is not possible unless the environment supports services composition.

[1] Software, Hardware and Information processing devices

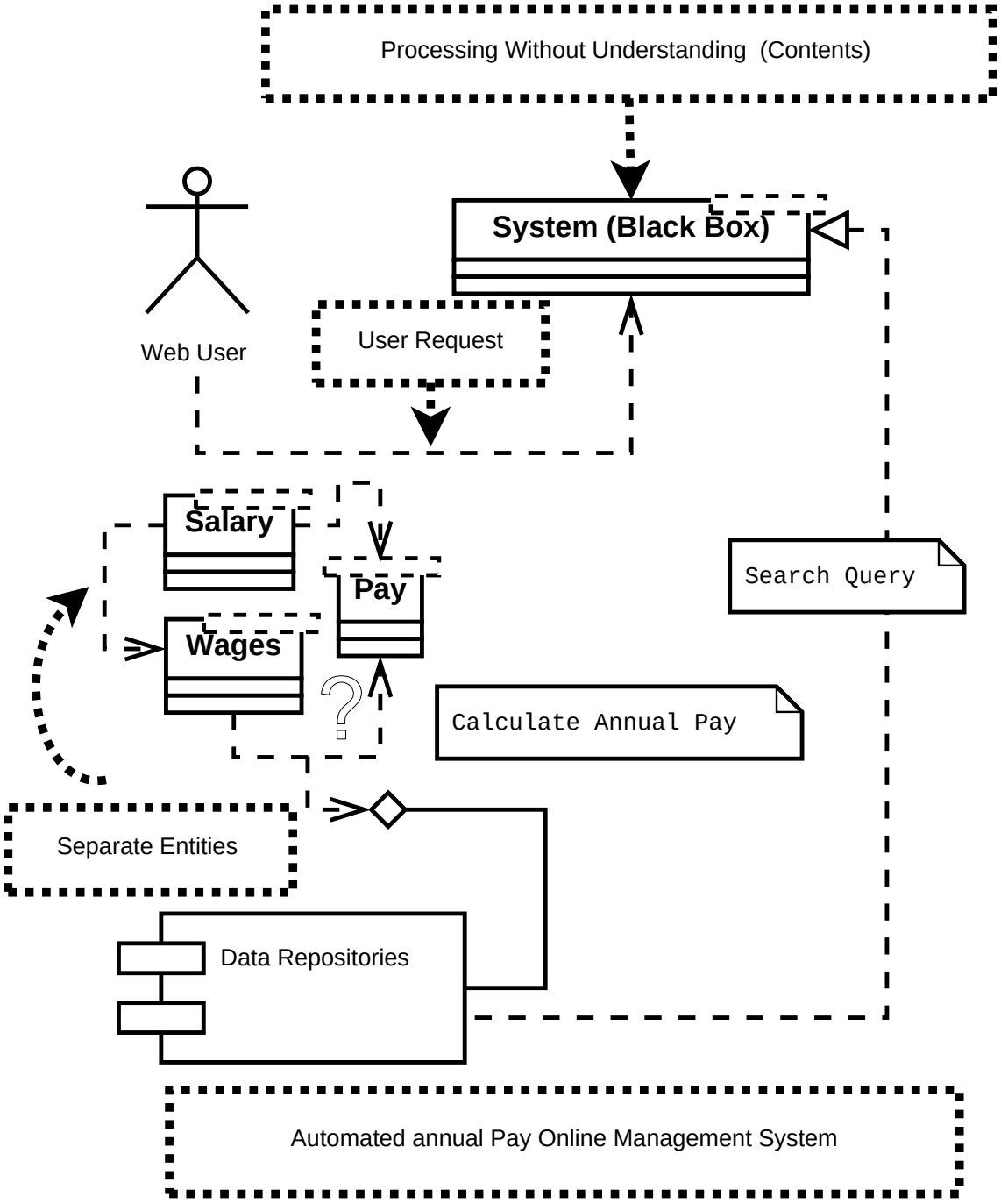

Figure 1.1: Current Web Contents Based Queries

Figure 1.1 shows the high level abstraction of inner class communication in objects where, essentially, the system cannot differentiate between the available constraints. By considering, firstly, the three separate entities (pay, salary and wages), the calculation process leads to an error for the annual pay calculations (dotted lines

show the information flow and dependency of all involved entities in the system). Secondly, there are ontologies (models with logical mapping) required in order to operate/process such applications [2]. At this stage, the development of such an application is carried out by providing static binding (logical or causal connections) between concepts which is a time consuming method [55]. As discussed in the previous paragraph, by making the system aware of the basic concepts' meanings, the integration of the services should then be possible, and carried out without the operator's intervention [58]. The environment will be able to understand and coordinate, to solve simple or complex tasks [21]. The discussed scenario is representing a very basic example, but for millions of available online services the same concept is forcing towards non-automatic or human operated environments based developments [21].

1.1.1 Key Research Issues

The main aim of this research is to explore the possibility of automatic dynamic web services composition and to provide more scalable, optimal, and robust final solutions according to user requirements. Solutions already exist that support the web services composition process, and that are being developed through the use of different methods [138]. However, for dynamic web services composition, AI (Artificial Intelligence) planning-based models have been utilized in previous research, that provide more scalable, and efficient results [23, 59, 67, 92, 137]. AI planners can be used in order to solve the web services composition problem, through the production of a sequence of actions that perform a particular task [89, 92]. This is because planners are able to perform any goal-oriented task within a web services domain if there is a procedural description available to achieve that goal.

There is, however, a requirement for a planning environment which can accommodate and support the services composition methods. This is the main reason for the AI planning-based approach being chosen. Though this problem has existed for the last few years, there is still no standard framework available that can be adopted by businesses, or other organizations, where the requirement is to obtain results following the analysis and composing of multiple services.

As discussed, a framework can be designed for the discussed composition approach, but it is difficult to put it into a working context, due to variations of development methods, communication protocols, compatibility and services security issues [16]. It is very difficult to adopt, or accept, a common framework, until all the major leading stakeholders in the IT industry (Microsoft, Amazon, Oracle, IBM, Apple etc.) adopt a common development model. The common model needs a unique framework, language, protocols and ontological structure for communication, and for inter communication of composite services [185].

1.1.2 Example Scenario

To explain the process of web services composition, let's consider a simple example of holiday planning. Travel planning is a simple, easily evaluative and traditional example of a web services composition problem, discussed by different research groups in the services composition area. Every year, thousands of web users around the world browse different websites in order to plan their holidays. There is the facility for customers (through web-based forms) to provide their basic requirements, such as tour time, food preferences, budget, and travel arrangements.

After basic input values have been entered (inserted), the system will attempt to match the desired preference-based results, and will present them to the user. The results will be analyzed by the user, or tour operators (staff), with the best option

finalized through manual comparison of the search results. There is a common terminology used while searching and analyzing the plan, *Packages*. Some tour operators will offer tailor-made packages, where travelers will have some of their preferences catered for, but must also accept, and pay for, additional, undesired facilities. Some websites provide a matched package in a few seconds, once the user has provided their desired location and preferences. These holiday offers are made, not only by a particular holiday planning company or travel agency, but by different stakeholders such as hotels, car rental firms, travel insurance companies and banks (Details provided in [73, 95, 97, 110, 116]).

Currently, the aforementioned offers are found through searching a static database, using search engines, or provided interfaces [8]. The system uses static repositories, with the traveler not getting the full advantage of services from other providers, which may have been uploaded or updated recently. The above problem is not the only problem of this kind. There are a number of contextual domains where the services composition process is required, but faster internet, technological advancements, and an ease of application access normally removes these types of problem from the sight of the common user [147]. The development of well-arranged data repositories has led to compiling of knowledge, which has helped to make developmental decisions quicker and easier [66]. There are many information repositories in which search facilities are in need of services composition, some examples including police investigation systems, weather forecasting applications, bio-medical research, and financial management institutions [2]. These are not the only examples, but with rapid growth of online solutions, and new technological concepts emerging, the need of services composition has increased.

Presently, the composition process is handled either manually, or by using distributed technologies-based, domain-to-domain static composition methods [163].

[2] Details provided in [68, 73, 95, 97, 110, 116, 129, 188]

The current methods are still useful, however the automatic services composition methods can provide a more effective, accurate and timely response.

As discussed previously, the basic layer of development, to support the Dynamic Web Services Composition (henceforth DWSC), is to implement the semantic web-based contents, which should enable the system applications to interact with each other [177]. However, the main critique comes from the fully-automated, semantic contents-based DWSC process, as changing the existing web appears to be a complex task. The DWSC research publications provide evidence that full automation is not impractical [41, 84]. The services composition task implementation is also being challenged by the amount of available services conversion processes, as it is not a feasible approach to convert the huge repositories of non-semantic-based contents available on the web. After development of semantic contents, the next stage is to locate these services, using a search procedure. During the discovery process, planner or software agents have to consider all related services, and available composite plans, before presenting the final solution to users.

In the aforementioned travel plan example, the planner has to consider all services. These can be singular or composite services [62], where a composite service is a service whose implementation calls upon other services, or consists of related sub-interfaces. This is as opposed to an atomic service, whose implementation is self-contained, not invoking any other services. Occasionally, there is a tailor-made package available, so that when the planner considers services, it has to consider, also, all partial plans e.g. travel company's packages for partial fulfilment of the actual output. This proposed approach can help clients (user) to obtain precise, accurate and updated access to results, as well as allowing potential clients access to the functionalities of businesses.

1.2 Research Question and Contribution

Research led activities related to services composition are moving towards the dynamic features of the SOA (Services Oriented Architecture). The researcher's main goal is to approach the services composition problem according to the demand of the fully automatic environments. *How can a normal user also take advantage of all available web services and packages without practical knowledge of the details of the services*, is the main focus of the current work. Considerable attention has already been given towards web services composition through consideration of the different approaches [177]. Still, however, no one addresses the full workflow context (end-to-end) to support composition processes, by considering available plans and composite services [144]. The current work will also look at user preferences for the composition process, and how to provide user's final results through evaluation of the desired constraints. It is very important to consider the user preferences before the composition process, in order to gain full advantage from the services composition process [159]. Along with user preferences, the final results will be analyzed, by conforming the input values, for optimal results, and ultimately to provide the user with choice-based options [101].

The main objective of the current thesis is to explore the AI planning-based solutions for the Dynamic Web Services Composition (DWSC) process, with an incorporated knowledge acquisition approach. A significant number of publications are already available in the dynamic web services composition area i.e. how the planner will locate services of the user's choice, but still, no discussion about the consideration of services by selecting composite existing services has been made. The main problem with all available AI planning-based solutions is that they can only be applied to one particular set of services, or similar domains [126, 183].

All implemented models consider only a number of services within one particular

domain [2], but in the current thesis, the composition framework has been developed to consider, not only similar kinds of domains, but other similar problems. Practically, it is not possible to test the solution of DWSC by considering different domains however where possible, domains problems and related solutions will be discussed in the current thesis. The obtained results will be compared with the similar efforts in DWSC area. For the rapid and dynamic discovery of services' support it is necessary to consider QoS (Quality of Services) features. In the proposed framework model, known as OSPG (Optimal Synthesis Plan Generation), the QoS constraint has been considered, along with user inputs. Within the main contribution to the Dynamic Web Services Composition (DWSC) model, for web services research context, the current work will also address the following research questions.

- ***RQ1: Which mechanisms are required in order to automate the Dynamic Web Services Composition process? Further, is it possible to develop a framework which can support the composition process without user (operator) involvement)?*** The main aim of this thesis is to analyze the requirements for supporting the workflow planning for web services composition that will address ongoing problems. Currently, the main problem with all available DWSC solutions is that all models work very well for deterministic domains where the static interfaces are designed to support that particular domain, but do not provide fully automated functionality for the non-deterministic domains [20]. Even by using existing interfaces, a high level of expertise from the technical user is required in order to carry out a single task.

  Initially, the available solutions for reducing the complexity for the user will be analyzed and a new and common framework will be proposed to improve the automated DWSC platform. Some of the related solutions will be dissected,

and this new common framework will be established in order to support the DWSC process. As discussed above, the AI-planning-based solution will be selected, however, research tasks will be actioned in order to establish alternative options and techniques for services composition. For example, it is possible to solve the DWSC problem by using a common structural model and languages-based platforms provided via the distributed web technologies (such as RMI, CORBA and JINI [3]) [135]. Due to the complexity of the proposed solution, it is very difficult to test this proposed framework within an online environment and by selecting different domains [31]. However, the system will be tested offline by using the OWLS-XPlan platform [85]. Two domains will be considered and within each one, a range of separate test cases will be applied and the results will be evaluated according to proposed concepts.

- ***RQ2: How should the framework be designed to consider functional and non-functional attributes when providing a final solution, according to user requirements?*** There are a number of input requirements for the composition process, which can be differentiated through consideration of their functional and non-functional attributes [94]. The composition process cannot be observed without some of the inputs, while some of the inputs are not required when narrowing down to the final solution. There is a requirement for the DWSC process to plan for a high level framework and a structural model, whereby it will be possible to accommodate the user's preferences during the final results of the services' composition [158]. The DWSC process model requires a common approach when considering the appropriate and relevant inputs for the discovery process. The proposed work

[3](Remote Method Invocation,Common Object Request Broker Architecture, JINI (pronounced GEE-nee; construction of distributed systems in the form of modular co-operating services)

will present a planning-based, intelligent solution for the search process and discover services by considering the different possible combinations of input values.

- ***RQ3: How will participating services be analyzed and considered for the composition process in order to establish optimal results?*** The selection of appropriate services, and provision of optimal results to the user, is also one of the requirements of the DWSC process. After the discovery process, the next step is to find the best solution according to the user preferences. The combined results of several services are also contained within some of the participating services in the composition process (as discussed in the travel plan example). There are different combinations of singular, or composite, web services available for the planner to analyze. The methods will be formulated to consider those combined results, if they are comparatively more efficient than the integrated multiple services (composite services) while the current work will also include the methods of optimum selection process.

The number of online, available services are increasing, and many services are shaping up to provide the same functionality. It is very difficult, therefore, to choose the best services. The web services research community introduced different ways to address this problem, however, the available solutions can provide a search functionality from a static repositories, without analyzing the services description. There is a requirement for the domain-independent framework, which will carry out the composition task without involvement the of the user.

1.2.1 Research Methodology

To address the problems described above, the existing approaches will be analyzed. Instead of *re-inventing the wheel*, the previous state-of-the-art techniques will be employed in the current work [84, 86, 88, 104, 157]. Due to the nature and complexity of the system, a mixed set of approaches will be used in order to analyze, and design, the integrated artefact-based model for the services composition. As discussed, previous successful efforts will be analyzed such as SHOP2, OWLS-XPlan and OWLS-MX [84, 86, 88, 104]. The discussed concepts and platform will be utilized as a basic testing system to evaluate the framework. The discovery process will be investigated, while the dataset of the European Union IST programme CASCOM [4] and planning environment of the SCALLOPS [5] project will be utilized. The proposed framework results will be compared with existing outputs for evaluation purpose however due to nature of the proposed model and extra user preferences based functionality (the exact output (results) providing environment), is not available.

For the services discovery process, AI-planning-based algorithms are being studied so that self-regulating services composition facilities may be provided, without involving the user during the composition process. There have already been many attempts to compose services by using an AI-planning-based approach [188], however in the current work, the user preferences consideration-based solution is adopted (the details are available in chapter 4). The required solution for the different types of possible input preferences from users, and the possible solution through conforming of the user requirements, will be further tested. Our proposed framework

[4]CASCOM is a Specific Targeted Research and Innovation Project (no. FP6 - 511632) supported by the European Union's IST programme . It has a total funding of 2.7 M

[5]Project of German Research Center for Artificial Intelligence (DFKI) further details available on (www-ags.dfki.uni-sb.de/)

model which is called OSPG (Optimal Synthesis Plan Generation) will address user preferences and will be able to provide an optimal solution.

1.2.2 Thesis Outline

As discussed in the previous section, the main contribution of this research project is to provide a conceptual workflow model that includes all pre-compiled web services packages during the Dynamic Web Services Composition (DWSC) process. AI-based applications have existed for years that have a marvelous reputation for solving complex mankind problems [28, 153]. There are different AI-based automated environments available, and in this thesis the planning-oriented solution will be employed. However, during the analysis phase, the available distributed web development technologies will be considered. The current work has been divided into four stages.

- The first stage explores the limitations of the distributed web development technologies (RMI, JINI, CORBA) and the related, proposed solution for Dynamic Web Services Composition (DWSC). Concurrently, the main focus has been on researching the current problems, and any unsolved issues in the previous research, while the proposed framework model will be finalized to support the DWSC process.

- In the second stage, the available planning environments requirements and the final outputs, will be observed in order to find gaps in the implementation process, and to understand how the planning process can support the DWSC process.

- The third stage demonstrated that the proposed, developed model, OSPG (Optimal Synthesis Plan Generation), is the abstracted framework. It can be

used in any related solution for services composition, where the automated environment can perform any task on its own after receiving the initial and final stage parameters from the web user.

- The final stage of the system will be observed through analysis of the final results, and by conforming to user preferences. The system will be evaluated by comparing the results (findings) with the available, state-of-the-art, environments in the DWSC area.

The thesis is structured around the following chapters, with the above research questions mapped into the respective chapters.

- **Chapter 2** addresses the review of the literature to develop the basic concepts of semantic content and ontological representation requirements, while the leading services composition solutions are also discussed in detail. Finally the requirements for the DWSC process are taken for the proposed model of composition.

- **Chapter 3** supports the proposed solution by providing the AI-planning-based discovery mechanisms that are already utilized in different attempts. Further, how to approach the optimal solution through the use of existing, modified techniques has also been well thought out.

- **Chapter 4** provides the basic requirements of the DWSC, and how the proposed framework will take into consideration the features of QoS, and user preferences. The chapter will also contain the basic expectations of users (*common user*) of the automated services environment. The technical implementation considers user preferences, and conforms to user desires for the services discovery. The conformance and contingency planning requirements will be discussed in conjunction with user preferences.

- **Chapter 5** explains the implementation details, where the practical activities-based results are discussed. In the same chapter the final outcome will be compared with the related work in the DWSC area.

- **Chapter 6** concludes the research framework by explaining the main outcomes, and by discussing the suggested directions of the futures in the DWSC area.

1.2.3 Concluding Remarks

History has shown that with every new development, there are issues that mankind could not have perceived at its onset [36, 60, 122]. The emerging concept of the semantic web is still in its infancy, as there is a collaborated effort required to implement and change the current structure from all services providers and developers [43]. The inspiring example of services composition can be fully automated by introducing the semantic context awareness, and the attached services description. The services automatic composition will play an important role, one which will reduce the current time consumption and incorrect results-based, manual search procedures.

The progress in the AI-based-planning solution is one of the driving forces that makes the DWSC process an achievable goal and that provides a new direction of research for web researchers. With the help of AI planning-based solutions, it is possible to provide solution and to plan according to users' choices. The overall DWSC process will be meaningless if the final results are not in accordance to user requirements. The user preferences need to be considered before finalizing any plan and as discussed in the last section, by using an AI planning heuristic, it is possible to conform to user requirements, resulting in the best use of data repositories and resources.

In the next chapter, the basics of the DWSC process are discussed, while related

work in the services composition area by using the AI planning-based approach, is explained, which will lead to the proposed design of the required framework.

Chapter 2

Background and Related Work

2.1 Motivation

The Internet has become part and parcel of everyday life. The World Wide Web (WWW) has made a contribution in every sphere of life (e.g. socio-economic and environmental) [180]. It has provided connection in the form of various networking sites (e.g; Facebook, Twitter, MySpace and Flickr), and helped to spread knowledge by new communications techniques [47]. This has resulted in the world becoming a global village. Various entertainment applications such as computer games, music, animations and real time videos have paved the way for a cherished life for youngsters and old people alike [98].

The Internet has emerged as one of the tremendous achievements and innovations in the field of computing, attracting many people, irrespective of age [51]. Everyone has started to think about personal web pages, businesses have uploaded their websites, and educational establishments have launched teaching activities on the Internet [98]. Researchers in web based resources development are particularly interested in new techniques and directions to produce new applications [102]. There is a multidimensional advancement of application development in the computing

field, involving home desktop and laptops, normal phones and i-phones, RPG [1] and MMORPG [2] games [149].

Modern computing fields such as Cloud computing, Grid computing and Volunteer computing have provided web researchers with many directions in which to advance their research and bridge the gaps among the different application domains for inter-communication [10, 32, 40]. Different applications have to be identified for inter-communication by using common protocols and unrelated deterministic behavior such as location, platform and tools [176].

With the help of current developments, progress in computation devices and speedy apparatus, it is possible to integrate the online applications and to enhance the capability of services for effective solutions [32, 132]. Here, an effective or optimal solution means providing users with final results after finding the best solution according to user requirements. In order to achieve optimal solutions we have to develop common protocols and models [121].

The current chapter is structured as follows, Firstly, the basic concepts, literature review and related work is discussed. The second section is about services composition procedures. The third section will explain the requirements of Dynamic Web Services Composition by considering issues and related research questions to support web services.

2.1.1 Service Oriented Computing

The Service Oriented Computing (SOC) concept has been around since 1999, However, during the last couple of years, it has been receiving more attention from the automated web research community [132]. The main idea behind the concept of SOC is to utilize the software capabilities from different domains and to provide

[1] Role-Playing Game

[2] Massively Multiplayer Online Role-playing game

on demand solution to users which will improve the machine-to-machine interaction [145]. The exploratory research topic of SOC raised many questions in the existing paradigm of intercommunication [10]. The different research directions were suggested along with a few efficient solutions for the integration of distributed environments. The developed software's environment integration and automatic intercommunication were the key priorities of SOC, but still the current research is not up to a level to adopt a fully autonomic model where services can be discovered by other systems or applications and will provide results to users [82, 112, 145].

One of the leading areas of research in SOC is the composition of web services to obtain an efficient solution from available web services [120]. *A Web service is a software system designed to support inter-operable machine-to-machine interaction over a network* [27]. The current setup of the web is now providing a vision of a well planned structured repository of resources. The available services are further multiplexing with available online resources and providing effective solutions [6]. With the introduction of cloud computing concepts (SaaS,PaaS,IaaS)[3], the web repositories are being populated and businesses are competing by improving their services for customers [130].

As the services repositories' sizes are increasing, the search functionality needs a better coordinated communication scope to return respective queries or results with minimum time and cost [168]. The improved search functionality will result in the common user being able to easily search for the best solution and indeed they will get an optimal solution. [2, 23, 88]. To improve the search facility, the requirement is to design the services by using contents which are easily interpretable by the software processes. In recent attempts, there is improvement in the designing side of the services to enable the intercommunication in services [105]. Currently, the following web variations and characteristics have been discussed in the literature.

[3]Software as a Service, Platform as a Service, Infrastructure as a Service

2.1.2 Definitions

- **Syntactic Web**: In the first generation of the web, the services only contained basic information regarding the contents. The user has little control, as they can only change the interfacing options (font selection, layout, color and use of plug-ins) but the contents are not interpretable by the system [12].

- **Semantic Web**: The main objective of the semantic web is that the contents of the web are machine processable. The information which is available on the internet is not interpretable by machines at the moment. Sir Tim Berners- Lee gave us the idea of the World Wide Web (WWW) and after years of its utilization in different directions of technology, a new vision has been proposed which is the semantic web [37]. The semantic web can be defined as, *is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation* [44, 111].

- **Pragmatic Web**: The Pragmatic web introduced the concept of the next generation web, where the concept of automatically extracting (or manipulating) information by software agents or planners, and to utilize contents is carried out without humans' involvement. Information retrieval will involve a simple search, for example last year's temperature (or weather conditions) on one particular day, to provide answers for complex queries by applying data mining rules or simple heuristics based solutions, which are not possible to solve manually in real time [165].

For future developments, the new vision of the semantic web which was again proposed by Sir Tim Berners- Lee is a fantastic approach, but after more than a decade of effort, is still in its infancy [151]. There are many reasons behind this as discussed in recent a survey by Pew Research Center's Internet and American

Life Project, but one reason is that people are not aware of the impact and application of the *Semantic web and Pragmatic web* [9]. The one possibility to introduce this concept, which is discussed by Sir Tim Burners-Lee, is to develop a critical mass for the semantic services which will include the development of data bridging (patches), semantic data and common ontologies [9]. The development of critical mass, which will be helpful for the enhancement and awareness of semantic services based composition models.

At the stage of the literature review of conducting and analyzing different models, it was observed that there is a requirement for an automated ontology [4] building environment to integrate semantic data [37, 55, 59]. The environment can be a part of the system or should be integrated with services repositories. The semantic services model on its own cannot provide support for DWSC as there is also a requirement for semantic data, domain ontologies and the planning environment to support the composition process [96].

> The Semantic Web will bring structure to the meaningful content of Web pages, by creating an environment where software agents roaming from page to page, can readily carry out advanced tasks for users [111].

.

The academic research community realized this fact and is trying to explore the experimental evidence for utilization of characteristics provided by the Semantic Web and Pragmatic Web as discussed the in last paragraph [172]. By analyzing and considering semantic contents, applications can communicate easily by following any provided rules [109]. These rules show the connection and link between concepts, known as ontologies. The use of ontologies make it possible to invoke other services

[4] An ontology is a formal specification of a shared conceptualization and allows user to organize information into taxonomies of concepts while maintaining a relationship between concepts

on the internet [109]. There are two types of methods to invoke and combine other service. The first type known as Static Web Services Composition (SWSC),suggests that services should be composable if the details of the different bridging services are provided up front. In the second type known as Dynamic Web Services Composition (DWSC), the services should be considered without existing knowledge by following certain standards applicable to all services.

> Let us change our traditional attitude to the construction of programs. Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do [63].

2.1.3 Static Composition Methods

Static composition environments are further divided into manual, and semi-automatic types of services composition [142]. The manual methods are where services are obtained from a repository of prearranged similar services. Semi-automatic methods, on the other hand, provide final results through consideration of user requirements [155]. The main difference between DWSC methods and semi-automatic services composition is that the latter can operate on only preset (arranged) services' databases, and have consistent input and output values [166]. The methods were being utilized in syntactic web based environments, the syntactic services interaction model concept assuming that parties may connect without prior knowledge of the participants, by following URL links and observing some fundamental rules [166]. In the web service-oriented interaction style, the particular web service request takes the form of a complete XML (query) document, and will provide the result on screen, with acknowledgement in the form of an email, or any other type of real-time response. During execution, web services do not contain a huge programming

framework in memory [187].

In the semi-automatic services composition method, the web services provider and the client may use different platforms and programming languages. Web services also hide the details of the application to the application interaction, hence the distributed application development moves away from a tightly coupled system to a loosely coupled one. Web services improve program integration by enabling program-to-program communication, while an automated, dynamic composition of web services requires a fairly rich, machine-understandable, description of services, and relationships between their basic concepts (ontology) [6].

2.1.4 Common Static Composition Methods Development Technologies

Static composition is the process of considering user requests in a predefined, acceptable format, where either the user request is obtained by using an advertised format, or the user must enter values manually, to initiate the composition process [178]. The protocol (SOAP) provides basic interfacing and the communication channel between the two applications. The protocol also established and agreed on the RPC (remote procedure call) interaction model and the acceptable data encoding format [178]. In the static composition process, the applications must follow binding rules, the process being unable to start with incomplete information. The SOAP specification defines the messaging framework for exchanging data across the internet, SOAP being, fundamentally, a one-way communication model that ensures a coherent message is transferred from the sender to the receiver [136]. SOAP messages have several parts.

- Envelope: (start and the end of the message)
- Body: (contains data)

- Header: (optional attributes of the message)
- Attachment: (attached documents)
- RPC Interaction: (structure of interaction)
- Encoding: (representation of data)

SOAP messages can be analyzed and subsequently related services will be mapped to carry out the services composition process. Universal description discovery and integration provide (UDDI) a registry service where a web service provider can register and publish their web service [61]. The UDDI registry consists of the following three parts. White pages contain contact information, human readable information and web service related information including business name, address, contact information, and known identifiers such as tax ids, customer ids or postcode. This information allows others to discover the service based on its business identification [61, 100].

The UDDI yellow pages contain a register of keywords that characterize the service, information that describes a web service using different categorizations, including industrial categorizations based on standard taxonomies [170]. Green pages actually contain technical information such as web service rules and a description for application invocation. Green Pages also include references or pointers to specifications for Web Services [100].

The UDDI registry has two kinds of clients: businesses that want to publish a service and clients who want to obtain single or composite services. Two APIs (Application Programming Interfaces) are described by this UDDI specification: the Inquiry API and the Publishing APIs [170]. The Inquiry API locates information about a business, the business offers, the specifications of services and information about what to do when failure occurs. The Inquiry API does not require authen-

ticated access and is subsequently accessed using HTTP but the UDDI detailed format does not support semantic descriptions [100].

The technologies of Microsoft's SOAP Contract Language (SCL) and IBM's Network Accessible Service Specification Language (NASSL) were combined to form the basics of WSDL(Web Services Description Language) . Although the technology is still under development, nearly all web services provide support for WSDL, and most development tools auto-generate WSDL files [100]. WSDL is divided into three major elements:

- Data Type definition
- Abstract operations
- Service bindings

SOAP, WSDL and UDDI based environments provided the basic functionality for services composition but their basic capabilities are insufficient for the DWSC process. OWL-S (Ontology Web Language) provides the description of semantic web services, and the details of its services enable planners or agents to locate, execute and compose services automatically [161].

Service Profile contains information about what a service does; goals which a service can achieve and the requirements to execute the particular service. With the help of OWL-S, it is now possible to specify the inputs and outputs which are required by a service before the start of the actual execution process [175]. For the services composition problem, OWL-S also offer non-functional properties to be considered along with main requirement expectations from the DWSC process

Service Model contains information about the use of the service and how the requester can communicate with the service.

Service Grounding contains the details of communication protocols and message format to integrate the service with the service model and operation details [175].

2.1.5 Dynamic Web Services Composition Methods

The design time services composition methods are discussed by many researchers, the concept being that services have to be discovered and composed according to user requirements at the time of design [12, 108, 182]. At the development stage, the services description has to be considered, and the set of services arranged in such a way, that the composite service will provide the user with a final solution. By following the aforementioned methods, it is possible to develop composite services for each type of user query, and in order to be successful, the user requirements must be well known in advance, the method not being suitable for on-demand services composition methods [12, 108, 182].

To apply these techniques for accurate results, the requirement of DWSC is to introduce semantic and ontological concepts to propose a web service model. Otherwise, dynamic web service composition will provide the same concept as any other distributed computing applications or static composition model [179]. The setup will provide a platform for the client and web services to find each other without prior knowledge of each other. Hence, the act of looking up the capabilities of a Web service and composing it with others may be done simultaneously, rather than using inflexible, static binding [179].

These issues can be addressed by using the DWSC process, where the individual services are considered as atomic entities. Each service atomicity enables the service to present itself for the composition process.

There are two types of further development directions to develop dynamic web

services composition models.

- The autonomic environments to compose services dynamically.
- The algorithms to support the discovery and composition process.

The above mentioned examples are some of the available solutions, however there is a requirement to develop algorithms for the discovery and composition process. The next section will explain the available solutions and will lead towards the composition requirements.

2.1.6 Common Dynamic Composition Methods Development Technologies

The number of services available over the web increases dramatically [179], and can be created and updated on the fly [178]. To achieve this dynamistic goal for web services composition, the AI Planning community provides a number of planning based solutions for the discovery and composition of web services [76]. The research based publications explain that while generating a plan, how the planner will locate any required services and how user preferences would be considered [58]. But as of now, no one has discussed the selection of services by searching existing composite packages based services (pre-compiled services) which is an important part of our research [84].

Considering existing pre-compiled web services during the composition process is a very important issue because there are many services around the web and each one has a limited functionality. In many cases, a single web service is not sufficient to respond to the user's request and often services need to be combined through a pattern of composition to achieve a specific goal. In some cases one service contains the combined result of other services. Instead of searching and getting results from

individual services it is more efficient and effective to consider only one particular service which holds a composite plan.

Mostly, services composition solutions are based on HTN (Hierarchical Task Network) such as SHOP2 (which is domain independent) [58]. HTN planning techniques have existed since the 1980s, and are utilized in planning problems of complex domains. In the HTN-planning domain, the goal state is typically achieved by using a set of operators (which contain the details of the procedure for decomposition of the main task) [69]. Along with operators, there are also attached methods that will help to decompose the main task into subtasks [157] (details in subsection 2.2.7). SHOP2 is a HTN based solution by Evren Sirin which is domain independent [181]. The planning process proceeds by task decomposition. The set of methods and operators are available to decompose the main task into smaller subtasks. The SHOP2 environment works by progressing towards planning steps of the composition task in three stages.

- At the first stage, the domain is formulated by processing OWL-S files and by replacing the operators of complex services with simple services.

- At the second stage, the main problem of services is transformed into a planning problem with the help of attached operators and methods. Here the complex task will be identified and the tasks which are further decomposable will be identified.

- At the third stage, SHOP2 will provide the planning steps to compose the services [188]. The main advantage of the SHOP2 environment is that it provides the sequence of steps, which later on can be executed in some order. The sequence of steps based solution will provide a breakthrough which will help to integrate the other applications by recognizing the steps of composition.

In order to start the planning process, SHOP2 requires knowledge about the domain (knowledge of operators and method) [181]. In SHOP2, the planner works very well, where the required information is available to decompose the main domain into subtask and the pattern of its hierarchical structure. The detailed knowledge about the process and subprocesses within the domain and the dependency among different processes should be available for a developer or planner to analyze. The planning solutions can be best supported if the detailed knowledged of the domains are available in advance which is not the case in the services discovery and composition problem. Here detailed knowledge means the input, output, facts (attached with methods) and preferences based requirements. Planners usually store a solid predefined hierarchical structure to decompose the main task. In SHOP2 environments, the planner's main objective is to perform simple (primitive) tasks instead of directly achieving one particular goal. Main tasks are divided into subtasks and the process will carry on until we reach a primitive condition which is directly addressable by using common methods [124, 181].

The HTN based model consists of a network of nodes and with every node one method is attached which actually shows a description of the node and its conjunctive and disjunctive relationships. Each node represents a complete sub task or composite task. Due to the dynamic nature of services composition problems there are limitations of the HTN based structure as discussed in [49, 74, 113]. The main problem with HTN based planning for DWSC is that the planner has to follow HTN hierarchical steps that have a predefined structure. However in DWSC problems, planners may observe changes during runtime so it will be difficult to traverse firstly all the nodes before reaching to the actual required goal. The following example will explain a few limitations which will be useful in explaining towards our research context.

In DWSC examples any online service at a (or any) node will provide the final output either from its own domain or by considering other nodes (services). One particular task may only need a single service from a node (either side of a leg) of a HTN. For example in a travel planning domain it might be the case that some services have to use a currency converter which may be attached to any other node. If we are using HTN, the planner has to start again using breadth first search for a currency converter which will result in session time out or waste of time before composition. Session time out is also an undesirable output for service composition. To address this kind of issue, we tried to use pre-compiled service structures. For any particular domain we can address such problems by using one related method attached to every node. But if we are looking for generalized AI based DWSC solution then it will be undesirable option.

In our current research, we are proposing a compiled web idea to adopt either at an individual service level or a complied services (packages) level. the proposed solution is that related services must be at one particular node or on an adjacent node. By adopting this kind of structure results will be more reliable and the composition process execution time will be faster.

Figure 2.1 provides the high-level abstraction of user requirements from the DWSC process which shows the response to a user query after composing different types of services. An intelligent software environment is required to provide final results to the user without the further involvement of the user.(Details in chapter 3)

In the WSC process, all the above mentioned requirements are not known in advance. The queries from the users are also non-deterministic as the user can search for any kind of solution while the solution can be obtained by exploring different ontological relations, semantic dependencies and attached constraints. The HTN based composition solutions to find the required goal (such as SHOP2) are not af-

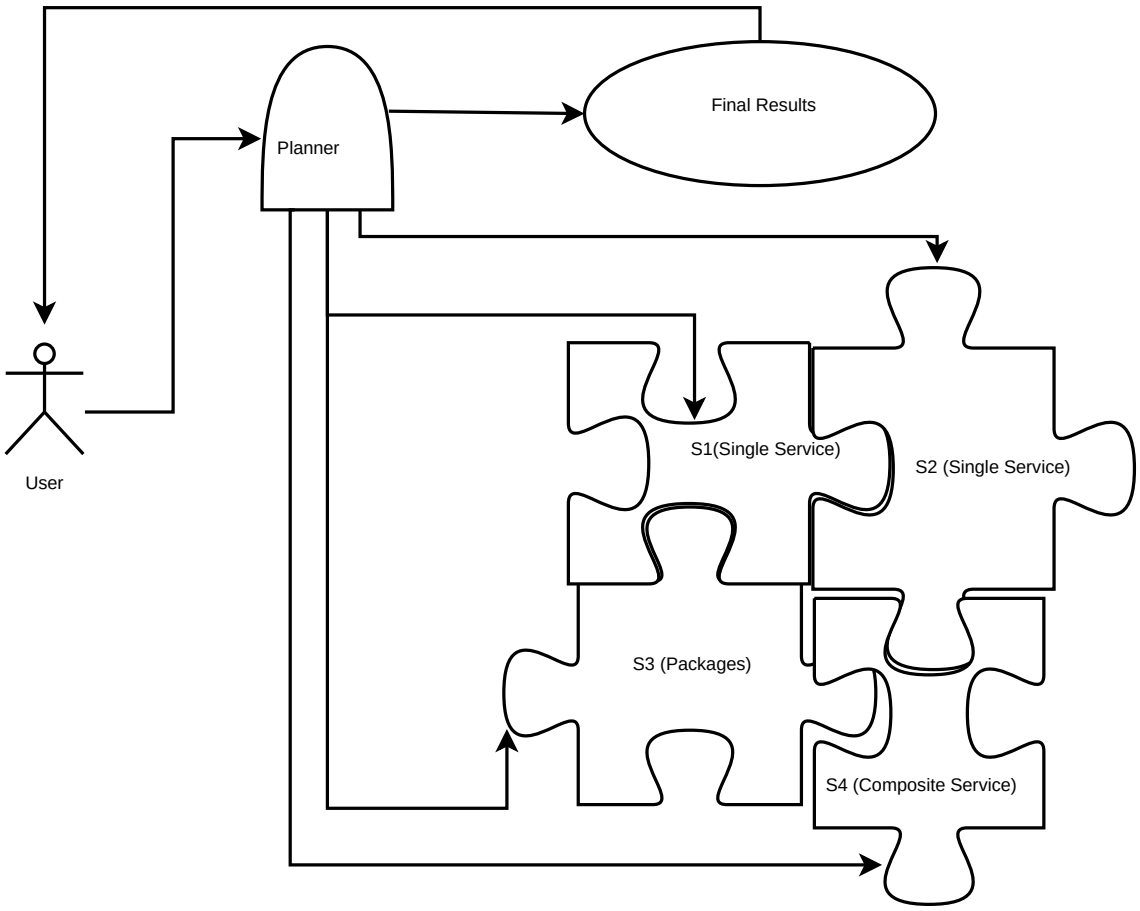

Figure 2.1: The Web Services Composition Process

fordable for such kinds of dynamic WSC solutions, as none of them are designed to adopt the patterns of non deterministic domains. However, for small domains where the sequence of services arrangement is well known in advance it is possible to utilize HTN planners. Along with all the above factors a human expert is required to analyze and tailor the decomposition requirements. As discussed before, the requirement of DWSC is that planners have to find a solution by taking different actions described in methods but without following the predefined pattern of composition. Here the predefined pattern can be a HTN structure or domain dependent rules, which can only support a particular domain.

After considering the above mentioned issues, a system is required which will be able to meet the above mentioned requirements and which can provide a solution for any type of non deterministic request without human expert involvement. At this stage of discussion, Figure 2.2 shows the basic functionality of the current setup which has been utilized by different research groups to locate services from UDDI [66, 127]. There were many attempts after the IPC (International Planning Competition) award winning attempt of SHOP2 to address the few deficiencies within the environment by using AI planning techniques and to enrich the final goal steps. The following are related attempts and helpful examples to formulate the solution for the current work.

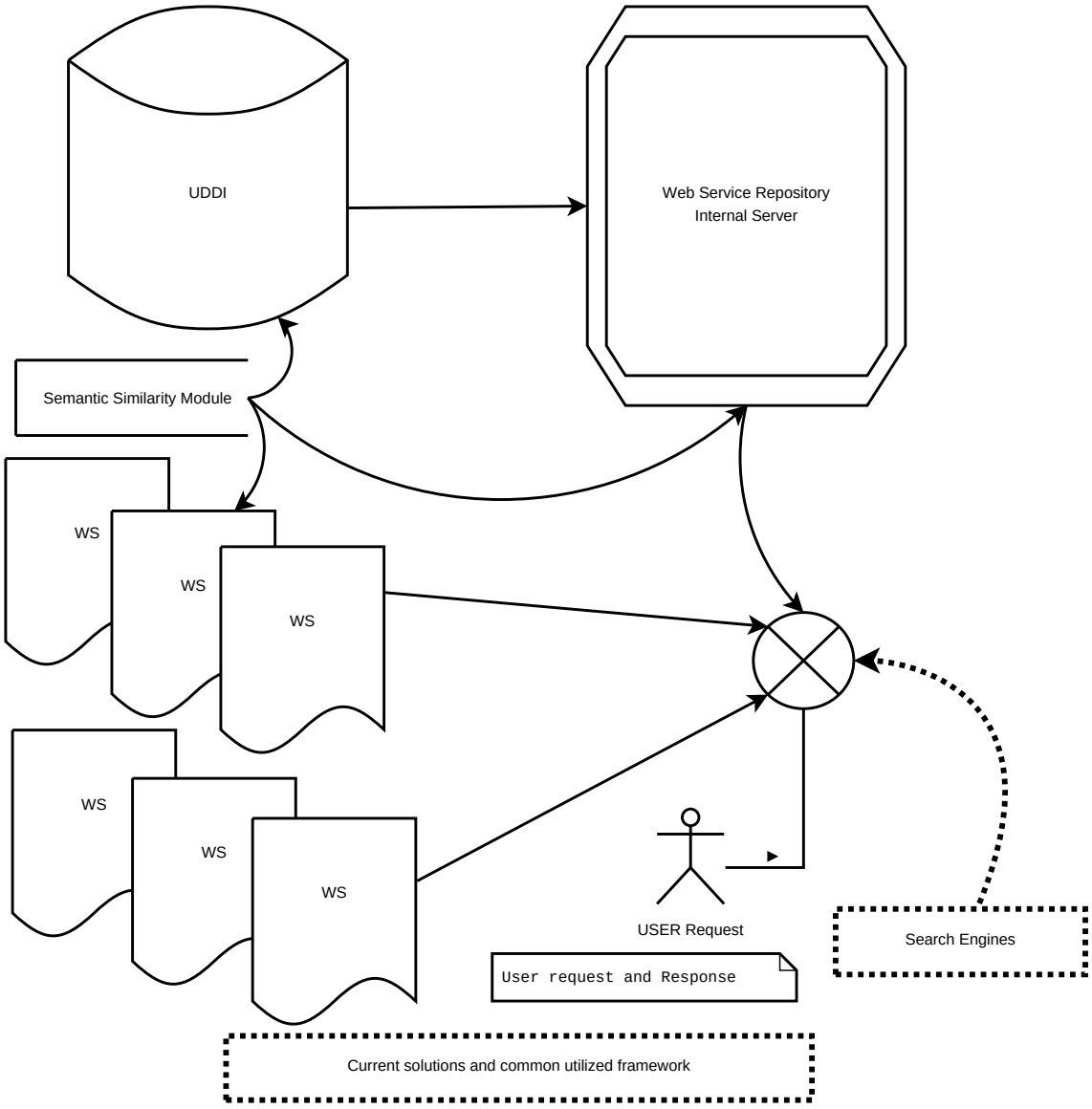

Figure 2.2: The Service Discovery Process

John Gekas and Maria Fasli presented a methodology for automatic web service composition that views the composition search space as a dynamic hyperlink environment of scalable size, and uses the environment's link connectivity structure in order to guide the composition process effectively [56].

Evren Sirin team considered user preferences and provided a framework to include user preferences along with related required goals however the system will try to search for the goal according to user preferences as much as possible. They

presented SCUP (social media monitoring service) with integrated algorithm to decompose the domain by using task decomposition method [92, 101]. Shelia A. Macllraith and her team ha also worked in the user preferences area [158].

Mark Carman, Luciano Serafini and Paolo Traverso introduced a semantic type matching algorithm, and an interleaved search and execution algorithm that together allow for basic automated service composition [33].

Similar work has been presented, known as 'PORSCE II', which provides a composition environment through consideration of the semantic contents, and the attached information of the services [67]. However, while the authors suggested that user preferences will be considered, the required functionality was not tackled. PORSCE II contains a visual interface. All other systems required domain knowledge before starting the process of composition, while the requirements of the services composition process are to dynamically adopt the changes and new constraint values. By considering the updated services that best provide a solution, according to the user's requirements. The three main advantages of PORSCE II are discussed by the authors. The exploitation of semantic services (which is the requirement of future developments) is seen as the main advantage. The second is that the system is not designed for one particular set of domains, in which the system has no power of analysis, or rigidly sticks to a predefined pattern. Also the system is integrated with a re-planning module that can help in case of any service failure [67]. PORSCE II supports a full composition process, and can consider only available services by evaluating the attached semantic description. The distance formula based evaluation metric has been used in the PORSCE framework. (Details of results and comparison are available in Figure 5.8).

One of the most prominent efforts in the direction of an advanced DWSC was attempted by using XPlan, developed by Marcus Schmidt [86]. XPlan is the hy-

brid AI-planner, the services matching being supported by the external component OWLS-MX, which provides a process level support for the discovery of the services. To enhance the semantic composition of the services, and to verify the IOPE (Input, Output, Preconditions and effects) of all involved services during the composition process, the OWLS-XPlan environment utilized the OWLS-MXP [85, 86, 88]. For the proposed framework evaluation and development purpose, the XPlan and OWLS-MX environment will be utilize.

The PDDL (Planning Domain Definition Language) was utilized which was proposed by McDermott in 1998 [4]. PDDL is a formal specification language used for encoding knowledge about actions in a specific application (called a 'domain description'). The domain description is written in the form of generalized operator schema. A planning problem to be solved is represented separately by an initial state of the problem, and a set of goal predicates to be achieved. Different versions of PDDL are available depending on the expressiveness of the domain description and problem languages. We will use PDDL3 which enables the encoding of user preferences. PDDL3 features can be utilized to consider the QoS (Quality of Service) factors, but in current work QoS constraints are also mapped as a preferences based factor.

As our proposed model is similar to OWLS-XPlan and PORSCE II, so where possible the outputs will be compared. The available OWLS-XPlan converts all OWL-S, OWL services and anthologies to PDDL 2.1. Hence the planner can use PDDL (Planning Domain Definition Language) as input which includes descriptions of objects, the initial stage, predicates and the final stage to achieve. The requirement of the proposed solution is to consider the user's strong and soft constraints along with the input requirements. To encode the user strong and soft constraints the PDDL 3.0 language is suggested [57, 157] and used in the current work. The

strong constraints represent the user main preferences which must be considered during plan execution while soft constraints are user desired goals. The soft constraints are preferences which the planner can suggest to user. As the OWLS- XPlan environment is designed to accept PDDL 2.1, the hard-coded user preferences in PDDL 3 format are used as input for the testing domains.

In similar attempts, the composition process is logically divided into vertical and horizontal composition based separate instances [65]. The vertical composition is, where the planning of an appropriate combination of services is involved to fulfil a certain task, while horizontal composition deals with finding the appropriate services for the composition process. The proposed attempts in the services composition area show that most of the related work is on vertical composition [65]. As discussed, the requirement of the composition process is that systems have to consider the constraints attached with the composition request otherwise the user is not getting any kind of advantage from the composition process. The authors work is on the horizontal side of the composition process and proposes a formalization process of web services problems by using techniques from Constraint Optimization Problems (COP). The Constraint Satisfaction Problem is a process of finding a solution which will satisfy all of the available constraint in the domain [65].

The second concept which is utilized in the related work is combinatorial Optimization, which is well known since 1960 [189]. The research was conducted by considering the combinatorial Optimization concept on a number of planning issues such as optimum assignment, shortest spanning tree, transportation, and the traveling salesman problem [7, 103]. *Combinatorial optimization problems can be defined as to search for the best solution from a set or related items by considering different variables in combination* [29]. The constraints attached with any domain represent a set of variables and the methods which are taking the satisfaction behavior of the

variables [103]. Here satisfaction behavior means that the possible factors which must be included in the composition process along with a set of limitations and restrictions which are not required to achieve the goal or to satisfy the individual variables condition [7, 186] .

By analyzing the CSP and COP theoretical perspective, in the current proposed work the user preferences are taken as a strong and soft constraints, where strong constraints will be dealt as a functional quality while soft constraints are taken as non functional parameters. The services composition problem is also considered by many researchers an NP-hard problem as an exhaustive search algorithm is required to find the solution which requires a high computation time [64]. The composition task where a user inputs constraints, parameters and variables has to be considered in combinatorial problem [64]. In the composition process all the input parameters represent flexible constraint satisfaction problem. The SWS (Semantic Web Services) planners can be categorized into two main types i.e; static and dynamic planners. The static composition planner examples are GOAL, AGORA-SCP and MetaComp. SHOP2, OWLS-XPlan and PORSCE II are among the examples of dynamic planners.

Service discovery and combining semantic contents based services together to provide efficient and intelligent solutions to users is one of the key points in the design of the proposed framework.

Figure 2.3 shows, the structure and flow to address the DWSC, where the AI-planning, and Distributed technologies based, development concepts are further divided to provide insight with various directions to approach the DWSC model. The Distributed technologies based approach represents different research directions such as Middleware Approaches, CORBA, JINI, RMI, Pattern Based and Agents Technologies. There is a lot of research available in this area, but AI based composition

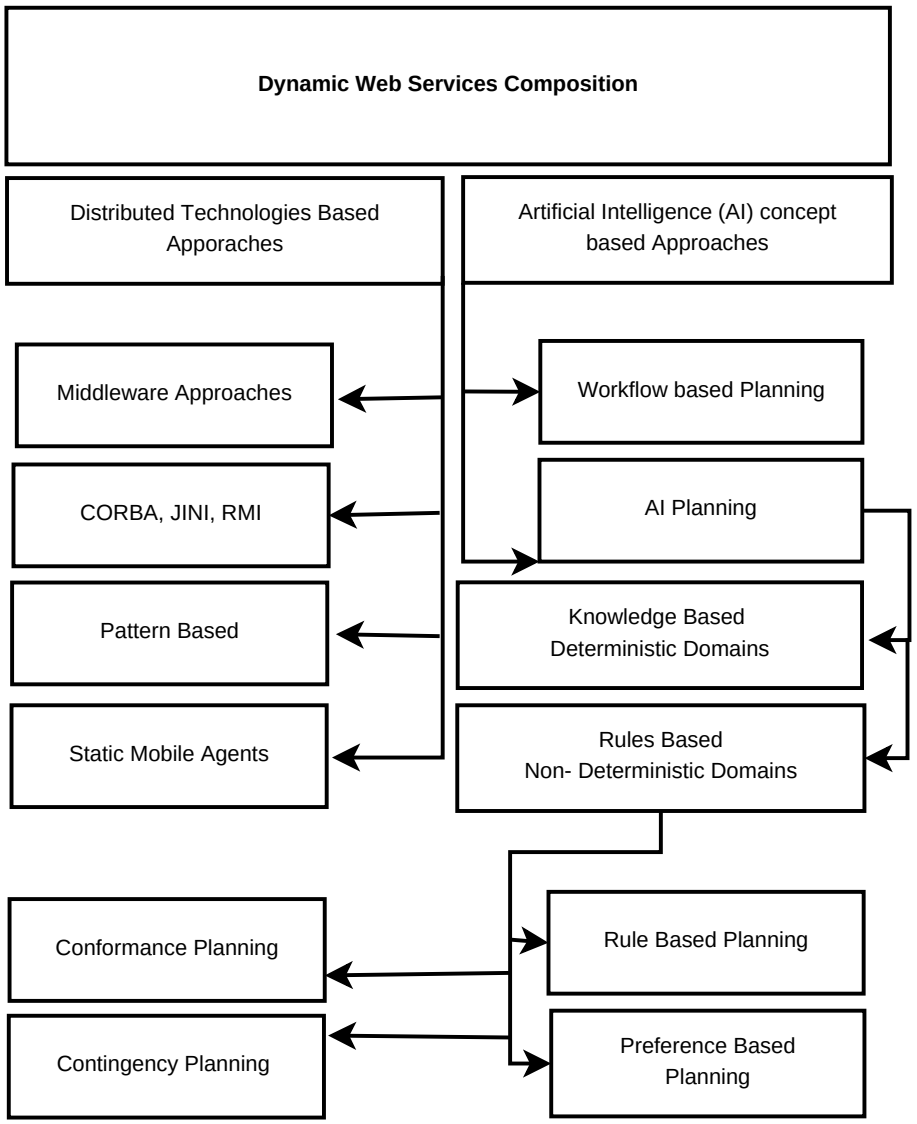

Figure 2.3: Dynamic Web Services Composition Structure

approaches are very popular in web community. The main reason behind this fact is the wide acceptance and reliance on AI based applications to support complex environments. The Workflow based planning and AI planning based approach provide further web services directions under the AI based approach. As discussed, there are many implemented environments available to support workflow planning. The workflow web services composition approaches are explained in [2**?**]. The AI planning approaches have two further direction for deterministic and non-deterministic domains. The complexity of the overall composition process depends on the main type

of domain (deterministic and non-deterministic). For non-deterministic domains there are further concepts to support the web services composition process for example Rule based and Preference Based approaches to include the functional and non functional requirements. while Conformance and Contingency based planning based techniques may provide a reliable solution according to user requirements.

2.2 Services Compositions Procedures and Methods

The services composition process is to locate and integrate services according to user requirements by using services syntactic, semantic and pragmatic contents. A fFew years ago, BPEL4WS [5] and WSDL [6] based solution were utilized, where the process of composition started by considering syntactic level details of services [178]. However, web services composition has been studied in the following three directions. The composition environments can be divided in to the following types by considering their structural and binding relation.

- Services Composition by using Distributed Web Development Based Technologies
- Workflow Based Web Services Composition Techniques
- AI (Artificial Intelligence) Planning Based Web Services Composition Techniques

The process of services discovery and composition has received much attention in recent years and there is ongoing research on related application development. The following sections show the details of the distributed concepts.

[5] Business Process Execution Language for Web Services

[6] Web Services Description Language

2.2.1 Services Composition by Distributed Web Development Based Technologies

Web services repositories are being populated day-by-day, as new services are added to repositories and along with new services existing services, functionality is being updated as well. For example if someone is interested in finding cheap hotels in Paris, they either have to visit the website which they already know or try to search the required web services by using search engines. The search engines will locate the services by using keyword based search techniques or by looking up by description in a web services registry. The discovery process is to locate the service which can provide the desired results while the composition process is to combine different services to achieve the required goal. UDDI (Universal Description, Discovery and Integration) is the prominent means of providing registry services. UDDI holds the services' information and also publishes information about services for service discovery.

It is very important to find the required services according to user requests which is only possible with a centralized database or coordinated access to decentralized database. At this point of the discussion it seems to be that UDDI is the most effective solution but there are a few limitations of UDDI due to which dynamic service discovery is not possible. The usage of XML for descriptions and consideration of classification and Identifier Systems makes UDDI based environments tightly coupled with static interfaces [131] so the dynamic integration of services is not possible without a powerful agent or planner. The planner will have to analyze and provide final services after composition [54].

2.2.2 Web Services and Distributed Computing

A few years back in the time span of the web services development concept, It was claimed that web services are adopting the concept of distributed architecture because they share many characteristics with other distributed computing architecture, such as CORBA, Distributed Smalltalk, RMI, JINI, μORB or DCOM [11, 124].

Technologies from distributed computing normally have a tightly coupled relationship between client and server. During the development phase of such applications, the main task is divided into client and server API sub modules (Application programming Interfaces) and sub processes so the interaction of different processes provides client/server environment. In distributed object technologies, main development activities will be around domain dependent constructs such as following object reference call procedure, defined data structures, language specific protocols.

During the design phase of software, developers have to check the compatibility of software with existing available applications and to think about the management of software interaction with all existing applications. Therefore it cannot inherently take advantage of the existing available services. For example the RMI (Remote Method Invocation) stub forward request to skeleton after using special routing mechanisms within the same or different networks and then waiting for a response from the server [124].

Along with other technical details developer must also consider different used protocols, sockets, routing, security, proxies etc. JINI comes with a concept of standard interfaces and lease oriented connection style therefore before calling another service, the user needs to know about its standard interface. The service request then takes the form of a method call defined by a name and a set of input and output parameters [124]. It then waits for a response in real time. In both the above major distributed computing technologies we must need detailed knowledge

of the available services and about all involved overheads to combine them. The main question arising at this point is how we can reduce tight coupling and static binding between these components [124].

The above mentioned limitations of the distributed technologies shows that a lot of extra effort is required to compose the services by bridging the individual interfaces [124]. It is therefore very difficult to compose services according to user requirements by using distributed Computing technological concepts and without considering semantic contents. According to previous research AI planning based techniques are being suggested for Dynamic Web Services Composition task [124].

2.2.3 Workflow Based Web Services Composition Techniques

In a workflow based composition, the user has to build a composition sequence model where all the involved services in the composition process will be combined to meet the final goal. There are two different ways to integrate services; the static and the dynamic methods. In the static method, the user knows in advance about the dependency of different elements within services. The pre-planned static model works very well for deterministic domains (see explanation in section 3.3). The graph-based structure plan is utilized for the connectivity of different services, by specifying their input/output dependencies. eFlow and Polymorphic Process Model (PPM) are examples of the workflow composition techniques [144], and in such a situation, the domain knowledge, and all the associated actions to be taken by the planner or software agent, are very clear.

In workflow based composition, the software agent, planner or administrator (if manual composition is required) proposes the services linked, or the pattern of composition with other services. Manual scripting is required for the basic links and the conversion of some of the parameters. Static or manual words were used for

such an environment, with these models working only for deterministic domains. A universal, or common, solution cannot be provided for similar domains, and a unique, ontological description is required for each domain. Such models were utilized in the industry by business partners (manufacturers, distributors, sellers and buyers of the same industry) for their interlink communication, the BPEL and WSDL languages were proposed for the workflow based approach. But, as discussed, the dynamistic feature of the services composition requirement is not possible with this model.

2.2.4 Planning Based Solution Analysis

The main problem with the above techniques (distributed and workflow based technologies) is that they can work only on one particular design pattern of services. All implemented models consider a number of services within one particular domain (deterministic), but in web services composition, the problem scenario is different, and at any instance of time, new services can be uploaded or updated, and the services structural model may differ. Further more, it may contain composite results of different services [48].

In the current work, one of the goals is to consider all the available composite and partial plans during the composition process. During the discovery process, the planner must consider all respective services, and also the available composite plans. As with booking a holiday plan, we must consider all available services and also the travel company's packages for partial fulfilment of the actual output (a composite service is a service whose implementation calls for other services. This is as opposed to an atomic service, whose implementation is self-contained, not invoking any other services). Sometimes, a broker synthesis plan is also available, so our idea is that when the planner considers the services, it must consider all partial plans as well. This approach may help clients (the users) to gain precise, accurate and updated

results. Further more, it may help to provide businesses with a utility for their business functionalities to be accessed by all prospective clients, without the need to spend huge budget on marketing of products or services.

As discussed in chapter 1, the contents, which are displayed by the system, are understandable by humans, but not by a system. Today, after many years of progress, there is a substantial use for the system's power and capabilities; however it could be more user-friendly if the system were to understand what it was doing. The expectation is not to act like a robot, or a human being, but to exhibit autonomic behavior to such an extent that the system can understand what it is doing, and how to bring forward the solution. Users have to adopt the methods to retrieve the information from the online web repositories, by searching manually, by using software scrapers, or by using web crawlers. If the system can understand the user query, then it is possible to assign some tasks to the system, which can provide an answer promptly.

In Artificial Intelligence (AI), *the process of formalizing a sequence of actions to achieve the final goal is called planning* [70, 190]. The set sequence of actions is called the plan, which itself provides instructions (step-by-step) to achieve certain goals. The starting point, with initial inputs, final goals, and any other constraint values attached, are requirements of the planning process.

2.2.5 AI (Artificial Intelligence) Planning Based Web Services Composition Techniques

Previously it was assumed that AI-planning-based techniques are only effective as a solution where physical operations are required, such as moving objects from one point to another (blocks world) or in providing a sequence of steps for robots to achieve any kind of goal. All attempts had a common basic functionality, where

the composition goal was achieved by taking different possible actions into account e.g; solving with the help of heuristic functions through consideration of a common set of beliefs and facts. The required planning task for a web services composition can be very simple, for example to search for flights after 6pm on Sunday, or very complex, such as booking a hotel near the football stadium, if the flight is affordable (not more than £100) on a Friday afternoon.

A few years ago, Heuristic Based Planning methods were introduced to address the abstract actions, such as searching for services for travel plans, route planning, patients' appointments, and supply chain management.

The Heuristics Based Planner (HSP) works by considering the composition problems, and mapping it into the search problem by encoding the facts and distributing them in the search space [25, 26]. The search space will be searched later by using the heuristic function, which itself is derived from the problem encoding. The use of the AI-planning-based solution for the composition of services was first utilized for deterministic domains (facts and constraints available in advance) in 2005. A similar type of the heuristic function was used in puzzle games, where, by observing the player's taken steps and common rules of play, the system was guided to play in different modes (low, intermediate, and high levels). The same functionality was employed in the industrial processing of products, such as objects that have to reach a certain goal by taking actions, such as up, down, right and left, by changing the discussed state description [139]. In the history of HSP, the first attempt was to enable the planner to search from an initial state to a goal state, using a heuristic function. In the next version (HSP2.0), the support for forward and backward searching was included.

The HSP planner has been utilized in different areas of AI-planning, but the heuristics search methods fail to get much interest from services composition re-

searchers. The heuristic search algorithms provide a sequence of actions that are possible to execute in the sequential form. However, they don't support parallel execution, and, as such, the composition process cannot progress on uncertain, initial conditions, and the alternative paths [119]. In that case, the planner has to execute the planning step by using a parallel approach- the plan will be unsuccessful with the sequential approach. This is because the user will proceed with booking the tour if all preferences are successful (such as the hotel being near the stadium, and the flight being after 6pm on a Friday). There is a requirement of the Heuristic evaluation function for the DWSC problem, as this will be able to analyze the output requirements and generate a plan to meet the goal conditions.

The AI research community offers the solution to the DWSC (Dynamic Web Services Composition) problem domains by considering, and taking, web services input as a planning problem. The overall process of providing action steps for the planner depends on the initial state expression, and the required goal. For the current research task, and to find the planning-based solution, the concept of forward and backward search options were initially utilized. In the forward search planner, it must start from the initial state (I), with the recognition that from the initial point the planner will move to the next new states by taking action (A). All actions taken by the planner to move from the initial state (I) to the final state (G) are the planning steps.

The concept of forward search was utilized in different planning-based experiments, and the work of Jorg Hoffman is the most important [70]. The Fast-Forward (FF) planner was the successor to HSP, providing the extra functionality of heuristic evaluation, and a method of identifying nodes that can be useful to reach the goal state (G) [70]. Meanwhile, the method of forward searching improved through consideration of the newly proposed methods (evaluation) of FF and HSP.

Heuristic evaluation provides a way to extract an efficient solution from a number of plans. In backward searching, the planner carries out the same steps, by moving from the goal state to the initial state, but if all required states and actions, with attached methods, are available in the search space, the planner can find the optimum plan. It does this by searching all graph nodes, and by applying a forward or backward solution. The simple search mechanism (using forward and backward) for the services composition task is not very effective, even for the benchmark examples configured by the leading environments for the DWSC process [24, 70, 85]. In different cases, the complexity of the problem is due to the uncertain nature of the search domains.

The dynamic composition method required no further user involvement (technical services integration) besides the provision of inputs at the beginning. However, some tasks are very complex, where systems sometimes have to take action by considering the different rules, or by analyzing different solutions. Comparatively, the workflow-based planning solution is easy to develop, but to address the DWSC task it is necessary to execute the services in parallel, and finally, to integrate results for final output. The user payment methods for air ticket bookings, and appointment booking options for patients in nearby hospitals, are examples of situations in which the final goal cannot be achieved by a sequential execution of the services. For complex, real-time composition domains, it is difficult to complete the composition process (within a constant polynomial time interval), as, in sequential execution, one service has to wait for the other service output, resulting in extra time being taken.

2.2.6 Forward, Backward State Space and Rule based planning

State space-based planning is also called progression planning, as the planner will progress towards the final goal. The description of the related preconditions and effects will help to locate a required plan within a desired timescale, and the planner will proceed by stepping towards the solution [148]. The state space algorithm works by comparing and analyzing the next connected nodes with the node under consideration. It is one of the easiest techniques to develop and adopt, however extra time is required to traverse the entire search space, and it may be difficult to find an optimal plan, the search direction in the search space being either forward or backward [186]. The concept of service composition, by consideration of the attached rules, with each service has been studied under rule-based planning. The best model of rule-based planning is SWORD [142]. In such a composition, the rule-based expert system is applied to search for the related services. All efforts in this area are well suited to semi-automatic composition, because of their deterministic nature, and it is possible, therefore, to set up a logical flow for similar domains [71, 104, 126].

2.2.7 Hierarchical Task Network based planning

In HTN domains, the tasks are normally in three states, which are: the primitive task, non-primitive tasks (compound tasks) and goal tasks. The compound tasks, which are further decomposable into the primitive task, are directly addressable to solve, while the goal task represents the final stage to achieve during the planning process [114]. In services composition scenarios, the best possible compositions are especially important for interactive systems, providing service composition services online integrated environments, where long delays may be undesirable. SHOP2

(Simple Hierarchical Ordered planner) is one of several examples of HTN-based web services composition planner [126], and provides a sequence of actions in the same order in which they will later be executed. This is one of the features, desired in some domains, where a fewer number of services participate in the composition process, and also depend on each other [188]. To address the above problems, and to search for available services, a combination of HTN and a graph-based plan is required (subsection 2.2.8), as discussed in [84, 104, 139]. The requirement is that the planner must traverse (or find) all available services, firstly that provide results for the required query. Then, following discovery, the planner must establish the best solution. Each node has one attached method that keeps track of all suitable services. After considering all nodes of a planner, it will move to the descendant best state.

As discussed before, the use of the HTN approach works very well with loosely coupled sets of web services, as the execution of sub-interfaces within domains perform independently [92, 125]. The final steps from the planner will be generated through use of a standard approach in HTN, which is not applicable for all domains. This is due to the fact that in the services domain, sometimes even the designer has a less than close knowledge of how the decomposition process will occur at certain points of the observed domain.

It is both time consuming and expensive to use only the HTN-based planning approach for non-deterministic domains [58]. The Employee Vetting Procedure (EVP) example (details in chapter 3), where new employees have to undergo security checks for a new job, is one of the best examples to use here to understand the HTN-based domain sub tasking concept. As with other HTN planning domains, EVP also consists of many subtasks. The planner has to analyze the main domain, and, according to the employee service history, provide steps for the plan execution. After identi-

fying the primitive tasks, planners have to consider composite services that contain multiple results (e:g; overseas criminal record check). If the employee had experience of working overseas, then it is a waste of time and resources to consider services such as credit checks, media search and professional body membership subtasks. It is also necessary to check the packages to save cost. It has been observed that for full dynamistic features (without human assisted composition) and for an efficient plan from non-deterministic domains, the planner should search for services by analyzing and considering composite services.

2.2.8 Graph based Planning

The graph planning method was first introduced by Blum and Furst in 1997 [146]. Later, Kautz and Selman concluded further research, introducing a mapping concept of how classical planning domains can be addressed by executing the parallel plan concept, thus providing a new direction of research with in the planning community [81]. The flow of the sequence of services for execution is called the control flow [119]. The control flow execution process is only valid if all services are considered in sequence, and the planner must consider (Analyze and Execute) all services one by one. As discussed, however, sequential search is not the most efficient method for a planning task, because of its longer execution time, and dependency on the previous state. In the services composition paradigm, the requirement is to handle the situation if the initial input, or intermediate results, are missing, and, after resolving such issues the next problem will be uncertainty of the new variables [80]. The discussed issues are not possible to solve by using available techniques, such as rule-based planning options.

The planning graph provides a level map to proceed within the domain, and provides a way of overcoming the problems of rule-based planning. In rule-based plan-

ning, the planner is bound to follow the rules attached with each method of execution, whereas, in forward or backward state planning, the planner should move from node to node, either forwards or backwards, by considering the optional branches. For a large domain, it is not practically possible to traverse all of the branches, and to consider all possible paths [186]. To overcome the limitations of the planning techniques for real-time, and the complete domain, the graph planning concept was introduced. In graph-based planning techniques, each level contains a set of 'literals' that will be either true or false. The searching process attempts to locate the actions to support the top level goal literals, before trying to find the supporting actions needed to execute the chosen actions [30]. The process will continue until the procedure will reaches the first level.

In Dynamic Web Services Composition (DWSC) problem the initial state, final state and required constraints to achieve one particular goal are very important, as all the next planning steps depend on these parameters. There are different methods proposed to compose these services such as (BPEL4Ws, WSDL) but they provide static, domain dependent and linear composition where human intervention is also required. In such models, the problem also arises whenever web content related changes appeared in the interface specification or any other type of changes, for example a simple change in the input values of a service can cause problems in composition. To overcome the above mentioned limitations, rich semantic contents based services are required which are machine readable and will be addressable without knowing technical binding aspects. The semantics of data dependency, control structure and a connectivity graph will help to locate the services and combine them dynamically.

2.2.9 Enforced Hill Climbin bg d g algorithm

In the composition task the planner can provide sequences of actions if the description of the input, actions and final goal is provided. To find the solution from the search space, there are different methods of search available such as greedy search and hill climbing. After the popularity of hill climbing, a more efficient algorithm was developed called Enforced Hill Climbing. The FF planner utilized the Enforced Hill Climbing (EHC) algorithm which is a modified form of hill climbing. Before going into the details of enforced hill climbing let's consider the definition of a plan Φ and the full notation can be expressed as

$$\Phi \doteq \{S_i, S_g\} \tag{2.1}$$

where S_i shows the initial state while S_g is the goal state. The planner has to search for the next states of actions in order to find S_g. One of the limitation of hill climbing which seems to be the main requirement for DWSC applications is that greedy search where the planner is only considering services inputs at the start of the process. While in DWSC the requirement is to consider runtime variable values which will result in optimal and realistic outputs after successful composition.

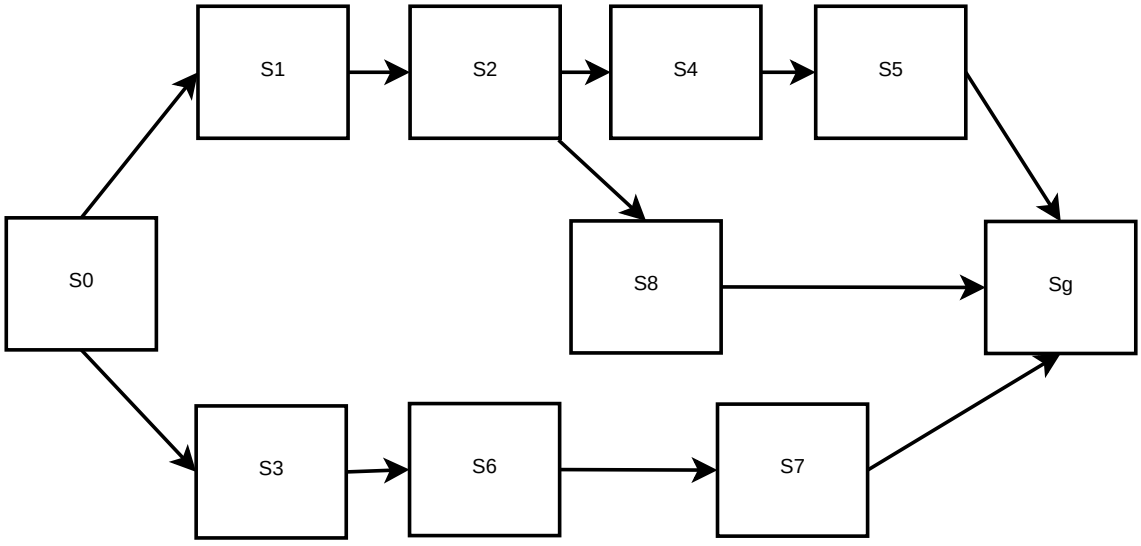

Figure 2.4: Enforced Hill Climbing

Figure 2.4 shows the methods of Enforced Hill climbing, If the sequential search method is used, the planner will find and analyze the services while the search sequence step will be in the following order.

Path 1 $(S_0, S_1, S_2, S_8, S_g)$

Path 2 $(S_0, S_3, S_6, S_7, S_g)$

Path 3 $(S_0, S_1, S_2, S_4, S_5, S_g)$

At the first instance path (1) will be selected and the planner will try to find the final goal. The process will be terminated if the composition goal is successful. If the enforced search mechanism is applied then the planner will check the alternative paths by considering path (2) and (3). At the end, the final efficient plan will be considered by comparing the results all of the paths. However it is possible to solve this problem with control heuristics ueing the Hill Climbing Algorithm.

In Enforced Hill Climbing the breadth-first search technique is utilized. where the results of the first state are taken firstly as the best results and later are replaced by the best services results. The repetitive process will continue searching until the last node is found. If the best solution search will not progress or the planner is not achieving the goal state the process will be terminated. There is also the extended EHC searching concept which is also a requirement of the DWCS (Dynamic Web Services Composition) process where the successor step nodes and branches are also being observed until the final goal is not achieved. For the services composition task it is necessary to strengthen the search process in depth search as there are a number of services available in the search space, which can provide alternative

solution. And the alternative plan can be a efficient solution for the required goal. The problem can be solve by using a constructed graph consisting of facts (literals) and activity (services invocation functions) nodes [119].

As discussed above, the requirement of the web services composition process is that the planner will follow the enforced hill climbing concept to reach a given goal [16]. As discussed previously EHC will not only consider the neighboring nodes, but in fact consider successor states later on for the optimal plan [16]. For the proposed concept it is also a requirement that services should be adopted by analyzing the functionality of services. As at any stage the service will be available in any of the two forms i.e; single services based inputs and composite services based output. It is very important to consider the services' functionality from the successor states and if a comparatively better service is available then replace the exiting service with the newest one. The final results will be in a proper format and the overall process will execute in a real and acceptable time. The requirement is to develop the search algorithm and heuristics to support the DWSC.

The advantage of the EHC concept is that it can support contingency and conformance planning [139]. There are many reasons behind this fact in the literature review but common reason to adopt the EHC concept is that the user preferences, selection of optimal plan by considering all options, conformance and contingency planning are the main goals.

2.3 Research Directions and Problems of Composition Process

Recent progress in the field of web services makes it possible to publish, invoke and locate web services, yet dynamic web services' composition is still a very difficult

task. The following development and implementation factors make the services' composition process more difficult.

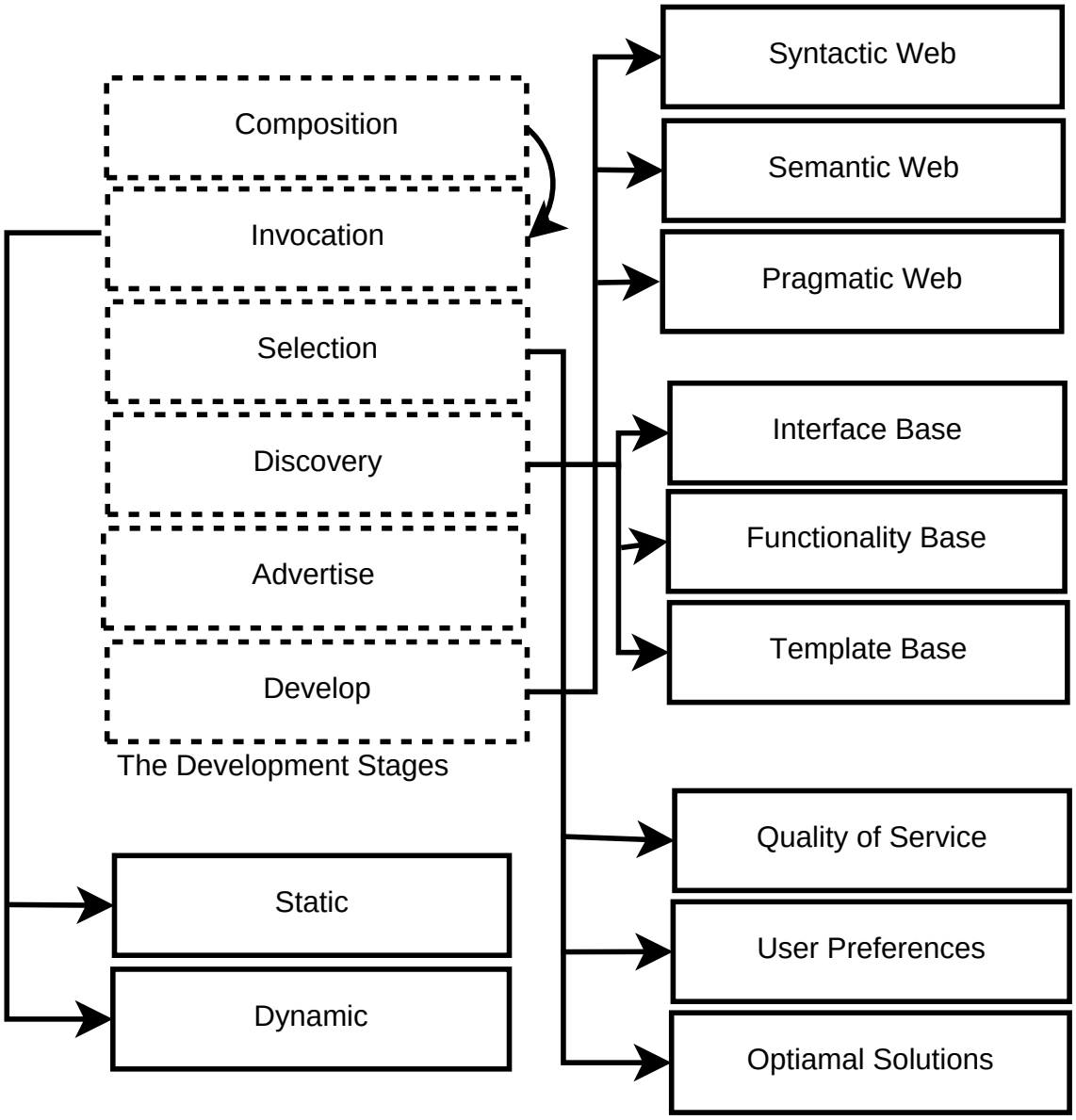

Figure 2.5: The Development Process

- Firstly, it is very difficult to analyze services (even manually) from the web services' repository (the UDDI) and integrate them to get specific required outputs.

- Secondly, web services' contents are going to be changed routinely to fulfil the user requirements. At selection time before composition, the system must be

able to analyze and select up-to-date services. There has been considerable research aimed towards getting updated web services at the time of composition, but techniques still fall short of the requirements of dynamic composition.

- Thirdly, web service suppliers are using different conceptual models to describe their services. To enable an automated dynamic composition process we need one common structure (model) of available services so that a service can easily invoke other services without any technical overhead. Figure 2.5 provide an overview of the development stages of web services, where the development stages show further design and structural variation of services to support the composition process. (further details in section 2.3)

A number of approaches addressing the problem of automatic web services' composition have been presented recently, both on academic and industrial sides [56, 68, 143]. Despite these approaches, many issues remain unresolved. The following are some of the analyzed issues (from the literature review) within the DWSC paradigm which need to be addressed to achieve the DWSC model.

1. How to compute all effects, side or after effects on the main domain of every action included in the composition process [89].

2. the execute-ability problem requires determining whether the precondition of all actions selected for composite service can be satisfied if given incomplete information about the world. However, this is unlike normal planning domains where operator semantics are fully specified. The execute-ability problem is discussed in detailin subsection 3.2.1, however usually six type of information are predicted to be missing from the input values [89].

    - Preconditions on input parameter values

- User preferences based constraints and consideration of different variations.
- Preconditions on prior operation invocation.
- QoS based constraint and considered parameters (strong or soft constraints).
- Knowledge of instance values in the output document.
- Real world precondition changes and post effects.

3. The projection problem requires determining whether a certain goal condition is satisfied after the execution of all component services given incomplete information about the current state [89]. (details in chapter 4)

4. Functional and non functional requirements to support the services composition problem [124, 176].

5. The conformance and contingency planning concepts to support the web services' composition problem.

6. QoS issues such as throughput Capacity, Latency, Response Time, Availability, Reliability, Reputation and Execution cost.

7. How to specify data storage requirements and data distribution. The arrangement of the required data storage and required structured is discussed in detail in the current chapter.

8. The data flow in services and the way of forwarding messages to related services. How to manipulate messages and route from/to components. (explanation in section 5.3).

9. Server-tailored or client-tailored approach. (FLC Or PLC details in subsection 4.1.3).

As discussed before, The number of services available over the web increases dramatically and can be developed, updated and taken off from the web on the fly. There is a requirement of the dynamic repositories which are able to update themselves according to new emerging changes in the existing services. In the next stage of the composition process the selection of services from the updated repositories is a very important task. There are three types of rules to consider, in both static or dynamic composition, as illustrated in Figure 2.5.

- In Template Based, a specific template either needs to be created, or acquired from a repository. A user has to locate the respective template first (in the static composition process) before composing services. This is a time consuming process as well.

- In Interface Based, on the basis of inputs and outputs through interfaces, the user obtains a services reference and these composite services after the composition process provide final results. This is a highly adaptable method but functionality is not guaranteed. During the composition process, some time we get similar interfaces, but after composition we get undesirable outputs (final results). Automated composition tools mostly used the interface base selection concept.

- In Functionality Based Composition, along with pre-conditions and post-conditions, the user has to provide first-order logic (formula representing the logic) into the interface information. The above mentioned individual rules are adaptable if we are interested in manual selection. In the context of our interface and functionality based (two rules combination) approach, the problem of dynamic selection can be solved by using the proposed framework model as discussed in chapter 4.

2.3.1 Concluding Remarks

In the current chapter, the related work and the used common services composition approaches are discussed (Static and Dynamic Methods). The research questions and composition problems in the related work are analyzed and the proposed research direction is originated. The AI-planning based approaches and related work provided the basic concepts for the proposed framework model. The next chapter will explain the AI-planning based requirements for the DWSC process.

Chapter 3

Web Services Composition and AI planning Support

3.1 General Discussion

Web Services Composition is a very important contemporary topic in the AI Planning community [169]. Emerging Semantic concepts provide awareness of the support to the future of the web, and web researchers use different techniques to facilitate internet users by using web semantics [117]. The AI community plays a vital role in access to web services and service composition related issues [143]. There are already many solutions available; however, most of these work very well for one particular domain (where the solution was initially tested) [164].

AI (Artificial Intelligence) based techniques have been employed for decades to generate planning oriented solutions and allow users to become conversant with an automatic approach to find the best solutions [58]. In modern times, the overall progress made in planning solutions demonstrates a systematic cohort of application development and management [184]. By applying planning techniques, developers can progress very efficiently by developing web based applications, automated hard-

ware and integrated environments for communications. However, there is a gap in the research, which leads to the basic problem that a system is required to answer a query without interpreting or knowing the basic contents [139]. In this case the term contents means data that is used by systems or complex environments. The non-semantic contents do not take full advantage of the high specification processing systems that are currently available [160].

With the recent development and progress on the hardware side, it is possible to develop fully automated, distributed and complex applications based on semantic contents [118]. These will be able to analyze the contents of a user request and provide answers that will be the exactly correct solution according to the user requirements.

3.2 The Requirement of Dynamic web services Composition

In chapter 2, the concepts related to services composition and the available planning environment were discussed in detail. In the current chapter, the proposed framework preliminaries concepts and requirements will be explained.

The current static solutions available for services composition are unable to handle such queries from users, where the services are arranged by simply framing them together in a predefined format [58]. This results in having to trawl through internet search engines. It is an unacceptable solution to rely purely on search engines, the tools being developed by invoking search engines, or by simply using the basic functionality of search engines. For example, Google alone was used 4.4 billion times by users in October 2007 [152]. The one reason to arrange services in a standard pattern is that individual services are sometimes down. For example, in April 2002,

eBay suffered a 22-hour server-outage which affected most of its online auctions, costing five million US dollars in lost revenue, and more than five billion dollars in market capitalization [152]. Amazon's main server was down for a three hour outage on June 29th, 2010, and following this, the service failed to display product information. For a business market leader such as Amazon, it represents a huge loss, with annual financial revenue of around 27 billion dollars, and sales at an average of 51,400 dollars per hour [154].

The proposed approach (discussed in chapter 1 and explained in the rest of the thesis) may help clients (users) to obtain precise, accurate and updated results. Further more, it may help to provide businesses with a utility for their business functionalities to be accessed by all prospective clients, without the need to spend huge amounts on marketing of products or services.

3.2.1 The Execute Ability Problem

Web services are sometimes portrayed as *silver-bullet* solutions to integrated web applications, because they have the potential to replace the role of the original web and relational database-related technologies. Web service technology enables application-to-application interactions over the web, since any interaction with a web service involves sending and receiving messages [52, 125]. One way to describe a particular service is in terms of its preconditions on input parameters values, pre-condition on prior operations invoked, and its output conditions and effects. There are two steps for the services composition process; the first is to get requirements from the user and then provide a sequence of instruction for the execution of services. In the web services composition process, the two terms choreography and orchestration are very important [125].

Web services choreography is to do with the interactions of services with clients

(users), and determines the specification of operations, states and conditions, which control how the interaction occurs [73]. Following the temporal constraints output by the choreography process should result in the completion of a useful function. As stated by Michael Hu [73], web service choreography permits the description of how web services can be composed, how rules and association in the web services can be established, and how the state, if any, of composed services is to be managed [42, 52]. The World Wide Web Consortium introduced the Web Services Choreography Description Language (WS-CDL) which captures the interaction within the participating services. The choreography model also helps to determine control-flow dependencies, message correlation, time constraints and transactional dependencies [42, 125].

On the other hand an orchestration defines the sequence and condition in which one web service invokes other services in order to carry out any specific task, i.e an orchestration is the pattern of interaction that a web service planner must follow in order to achieve a goal [124]. On the basis of the above discussion we can say that the dynamic composition model requires four additional layers Semantic, Ontological, Choreography description and Orchestration concepts. The Choreography Model and Orchestration Model provide us with a comprehensive solution for basic issues such as precondition, effect and post condition [125].

3.2.2 Data Distribution and Quality of Services

As discussed, Service-Oriented Architecture (SOA) is a recently defined paradigm for organizational models of systems, aimed at simplifying large business operations using existing services. SOA's main manifestation is in the area of web services. Although there is plenty of controversy about how SOA will manifest itself in the context of web services technologies, the issue of the quality of services will always

be central to the argument [125, 127].

Businesses will have to have secure web services and will have to be able to guarantee that messages arrive at their intended destination and are processed reliably [125]. During execution of a composed web service process we also require output variable values from different services (data servers). If some parameters are missing or due to any reason not available for the next service then the process will fail. Web service standards and technologies are composed of two major types of application interaction patterns on the basis of their database interaction access: Centralized Dataflow and Decentralized Dataflow. If our focus is towards dynamic service composition then in both approaches there are some limitations [125].

In Centralized Dataflow, data between component services is passed through the composite services and in that situation bottleneck problems occur, affecting throughput and response time. On the other end in Decentralized Dataflow components, services exchange data directly with various data base servers. The result is that the distribution of network traffic among all the services involved improves loading characteristics on the composite service, improving in particular throughput and response time. Both of the models have their own advantages in distributed computing environment [125].

The Decentralized Dataflow seems to be very efficient for dynamic services composition but in some situations (for example in the Hu's Police case study example discussed in [73]), it will affect QoS factors such as latency, execution cost and capacity. For automatic web services composition the centralized data model can be used by adding a middleware extension support to avoid tight coupling between services. This type of extension will be automatically added to the composition model if the common UDDI (Universal Description Discovery and Integration) or WS-Coordination and WS-Transaction is being utilized. The proposed framework model

will result in performance improvement, lower time response and higher throughput maintainability. The web services QoS requirement refers to both functional as well as non-functional quality aspects of web services. The overall performance of web services depends on the complexity of the application, as well as the network, messaging and transport protocols (e.g; SOAP.HTTP)[125].

As discussed in the previous chapter, previous research efforts in the services composition area have divided the process of composition of services into static and dynamic as shown in Figure 3.1. Static composition is purely manual or semi-automatic where the user problem must be defined in advance by mapping the user requirement into a static predefined template. The requested input variables are then compared with the available services. There are many examples of static composition for example getting insurance quote, searching holiday packages etc. The current search engines are using the same type of search functionality. In the dynamic composition process automated tools and execution engines have been used to select, analyze a user query and tack together web services interfaces to provide a response to a user. Although it is time consuming step, but from a user perspective this composition will continue to be considered as a use of a simple service, even though the services themselves are composed of several web services from different domains [1]. In the following sections the process of composition and related research models in both areas are discussed in detail.

3.3 AI and Dynamic Web Services Composition

In Artificial Intelligence, automated planning is the task of automatically selecting and organizing actions in a sequence in order to meet the user defined goal. The existing approaches for planning solutions are studied under two, main, broader approaches. The first is the workflow-based planning approach, where the prearranged

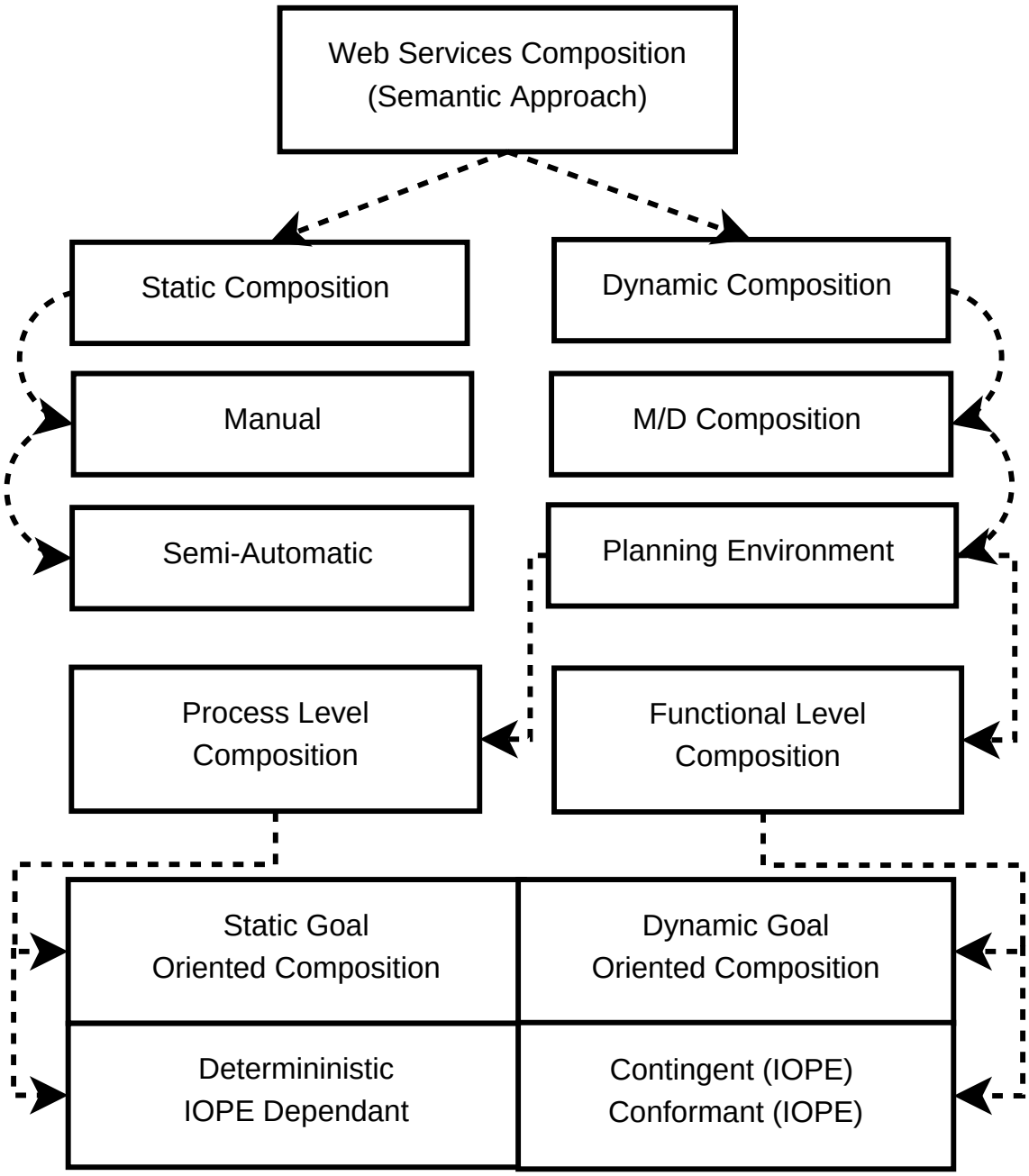

Figure 3.1: Types of Semantic Services Composition Process

and accepted flow of actions is required. In dynamic planning, however, the planner decides the steps of the actions, after analysis of the real time situation.

Example 1

To understand and explain the problem in more detail, let's consider the logistic domain. In logistic domains, the suppliers have to provide the delivery of the products to a number of stores in an area (perhaps one part of the county). Different types of items are ordered by stores, and it is the responsibility of the Head Office to deliver all items on time. The short time limit, or date of expiry, is attached to some daily consumable items such as cold perishables (milk, drinks, bread etc). If the delivery depot is a very long distance from the store, then there are many constraints (variables) to consider before finalizing the plan, such as order details, distance, expiry date of the items, time span, saving costs, and other delivery options to save time and cost. The Head Office must ensure the timely delivery of all items by considering the items' shelf life; otherwise the unavailability of daily consumable can work against the overall sales and reputation of the store. If the lorry breaks down on the way, or due to any other reason the delivery is delayed, the logistic department will have to plan for the recovery and redelivery of the products. Currently, this problem would result in the waste of a lot of resources and money of superstores such as Tesco, Asda, Sainsbury and Morrisons, who operate all over the United Kingdom, but who have delivery depots in only specific locations. The concepts of deterministic and nondeterministic domains will be discussed in the next section with in the context of logistic domains.

Example 2

For a discussion of the concepts and approaches towards services composition, the second example presented here is the arrangement of the tour plan. For example, a person is interested in planning a tour for a rugby match, and particularly looking

for a hotel near the stadium that will be less costly. The constraint attached with the flight services is that the user wants to catch a flight after 6pm on Friday, and is also interested in booking the ticket at the end, if the other preferences are matched, such as a flight within the allocated budget, a hotel near to the stadium and a match ticket. Although, the above mentioned domains contain different concepts, and are varied in terms of the representation model, there are some common issues where an autonomic environment to solve such problems is required. In both examples, there are preferences attached with the domains, some rules which have to be followed, and some quality of services-related issues to consider while generating the plan. Along with the similarities, there are key differences between both concepts. In the logistic domain example, the issues could be the pool of services, the variable values, and the connection between the services that are well known in advance (such as store locations, distance, transport information, product information, stock and order details). In the tour plan example, there is no information about the behavior of execution available in advance. All we can know about the process is established following execution of the plan, where the composition process will respond with the final plan. Still, however, the user is unaware about the number of services, the selection criteria and the individual response of services. To conclude, the logistic example represents a concept of a deterministic domain, while the tour plan is non-deterministic. Both concepts are studied in planning domains, but by using different techniques, as discussed in the following workflow and AI-planning-based techniques.

3.3.1 Services Selection Procedure

The first step for a planner is to generate a plan to fulfil the user request by analyzing preferences. There are different planners available to be utilized for the composition process and provide different functionalities. As the focus of the current work is

on the selection of services and to provide the user an optimal solution, we tried to concentrate on the services selection procedure. For current work, different planners' functionalities and final goal achievements procedure have been analyzed and where possible the features of each planner have been reviewed by comparing then with the feature set required for the proposed framework. As a lot of work has already been done in the services composition area, using different planners such as FF and Sim planner [85]. The planner XPlan is utilized in the current work to generate a plan. XPlan is widely adopted as a services composition planner in different frameworks [84].

In the current work, the preferences for web services are considered in two sub variances as strong and soft constraints along with some extra choices (preferences planning) to evaluate the conformance planning (explained later in the chapter). The strong constraint preferences are user choices which must be considered before finalizing the plan. Preferences which are also user choices which the system has to consider but which are not essential, will be discussed under soft constraints. For example, the planner can try to find the solution by considering the strong and soft constraints but in case the soft constraint are not met, system has to respond by considering just the strong preferences. All the AI planning based research efforts have a requirement for considering user defined preferences. For example, in the blocks world domain, all the same colored blocks are arranged in a tower or a block cannot be placed on top of a fragile block [45]. In the DWSC example, the requirement is to consider user preferences (such as flight must be booked before the match ticket are purchased) or the user likes to travel on Friday after 6pm. The preferences which *must be* required by the user (such as flight ticket purchase) are considered as strong constraints and user preferences (e:g; travel on Friday after 6pm) are considered as soft constraints.

3.3.2 Preference Based Planning

The main importance of the automation of the service composition task is to provide users an efficient solution by considering all the requirements within real time [58]. The results of previous academic research on WSC (Web Services Composition) shows that all the presented models tried to achieve the final results according to exact user requirements (functional). For a successful composition process, the planner will have to match the user required output with the available solutions by considering functional and non functional requirements. Although the requirement of the automated DWSC process is to provide the solution according to user requirements, sometimes the user is interested in searching for different available solutions by providing some preferences (functional and non functional). The planner will have to provide an efficient plan by considering user main requirements and preferences. The results will be arranged in a chronological order of selected factor by the user (such as more economical or most strongest desired options). To compose services according to user requirements or preferences is the key concept in the DWSC paradigm and in the current thesis proposed framework.

There is a significant amount of work done on the consideration of preferences by Shelia A. Macllraith and her team [158]. But until today there is no complete automated framework available for the DWSC which will provide an end to end complete workflow model to achieve an optimal plan (by considering functional and non functional constraints). The objective of the current part of the work is to propose a framework which will consider using preferences and provide optimal solutions by conforming user preferences, and provide a complete workflow model to locate services and compose, according to user functional and non functional requirements.

The preference based function expresses the logical constraint flow values [156]

and makes it possible to define expressions such as always, sometimes, sometime-before or sometime-after. The authors discussed two types of preferences which are precondition preferences and temporarily extended preferences (TEP) [14]. The preferences are user desired goals to achieve along with the final goal while TEP are again preferences but showing the sequence of actions or steps to follow while formulating a plan. In PDDL 3 the preferences can be expressed as follows

$$(preferenceBookFlight(always - until(not(occ(bookHotel))))) \tag{3.1}$$

The services composition task can be achieved by analyzing semantic contents. If services are developed by considering semantic contents then it is possible to map the user preferences and to provide users with the best solution. The user preferences can be qualitative (QoS) or quantitative (e:g; cheap package) in nature but before the composition task the preferences should be considered to be part of input variable values.

After finalizing the planning steps, the first requirement of the composition process is to locate the web services (Discovery Process). In the plan generation process, a few composite services provide better and faster results, instead of the planner gathering these outputs from different, semantically enriched services. Should a planner have to consider each service (node), during the quality of services consideration, and then, at the end, compare the results of several services with available packages, it will lead to time and cost problems.

3.3.3 Conformant Planning

Conformance planning is a branch of planning to explores nondeterministic domains and to finalize a solution by taking some action under uncertain condition [38]. To find efficient solutions, the planner analyzes the different states and tries to fill the

missing or required variable. Here the variable means at any state the required inputs for the process to complete or required by the next state. Comparatively it is harder to encode conformant problems (such as non deterministic domains) than classical domains where the final results or goals are achieved by a sequence of actions [72].

The conformant planning concept has been utilized in services composition to avoid the chances of failure during the composition process. In the current work, the concept of conformance planning is adopted at two different stages. Firstly, to include the missing parameters values and secondly, to conform the user preferences, to obtain the best search results.

The requirement of the composition process is that all services discovered for the composition process have rigorous input and output requirements and if any of the variable values are missing then the composition process will be unsuccessful. To avoid such problems the planner has to adopt some belief states or to consider some probabilistic input which will not halt the composition process. The planner will try to fill the missing values by analyzing the missing constraint if it does not get a response within the adjusted time [38].

The requirement of DWSC is that the process should be automatically completed and all services will participate in the composition process,which is not possible without knowledge of IOPE (Input, Output, Preconditions and effects) [19]. The missing constraint can be IOPE information or preference based constraints [88]. The IOPE constraint values can never be violated or replaced by constant values, but the preference constraint can be replaced by attached constant values [2]. As in the planning oriented composition process some of the included services are partially observable and the complete information about the pre or post effects is not available in advance which means there is a requirement of probabilistic planning where the incomplete

information will be completed first [46]. To address such kinds of nondeterministic problems researchers used conformance planning [38, 72, 78]. In all of the above mentioned problem it is very difficult to achieve the dynamic composition goal.

Conformance planning is the branch of planning for nondeterministic domains, where the planner is considering problem domain with different possible states to start [17]. Comparatively it is harder to encode conformant domains than classic problem domains where the results are achieved by a sequence of actions, but all states and set of actions are pre determined [50].

To discuss a practical example let's consider a services based scenario. All X number of services are participating to provide composed sequence based results for one particular problem. Let's assume all services are depending on each other services' executed results. The planner is selecting services in sequence, and all services have to participate during every required action. At the end the user can get the final result after the execution of all required services. Suppose service B is waiting for service A's results. The planner tried to execute service A but did not get any response. In this kind of situation most available planners provide unsuccessful results. The requirement of the DWSC is that the missing or failure results should be analyzed and where possible the missing constraint will be filled by probabilistic constraint. However the option should be given to the user for interaction at the process level [123].

$$\omega \equiv \Theta; \quad Z(missingvalue) \leftarrow Split; \quad \omega C_{pi} = \gamma_i; \tag{3.2}$$

For example if Θ represents the IOPE values at any stage and ω represents the intermediate value of the input at any stage of planning, C_{pi} represents the value of constraint then in Equation 3.2 firstly the values will be divided and then the constant value γ_i will be assigned to the input, which will take the composition process towards the final goal. However if the IOPE constraint value is missing then

the exception handling procedure of contingency planning will be adopted which is discussed in section 3.4.

As discussed before, plan generation is not only dependent on the initial stage and the final stage. The automated environments have also to consider the following limitations which are essential for the DWSC process

- *The structural model of new web services* as the new services are not developed by using same format. Due to this the number of input variables can be changed. To accommodate such changes, the proposed algorithms require a robust structure which will accept any changes and the selection process will update its functionality by analyzing the improved situation. As discussed, usually such changes are related to input requirements or some conversion task.

- Instead of following the predetermined composition path the planner will have to search for alternative solutions. This is one of the key features of the proposed model. There is a basic requirement for the composition task to find the optimal solution. As discussed, there are a number of services available to participate in the composition task, however the requirement is that the planner will have to consider all available services but if a service is not responding, will be able to cope with alternative solutions. This is only possible if there is a description of each individual entity (service) while the arrangement of the services is not different from domain to domain. There will be arrangement for the planner to select the alternative solution [119].

- The services' execution post effects must be determined in advance, as the composition process will fail if the participated service is producing unpredictable results which are not recognizable by the next service which is waiting for the

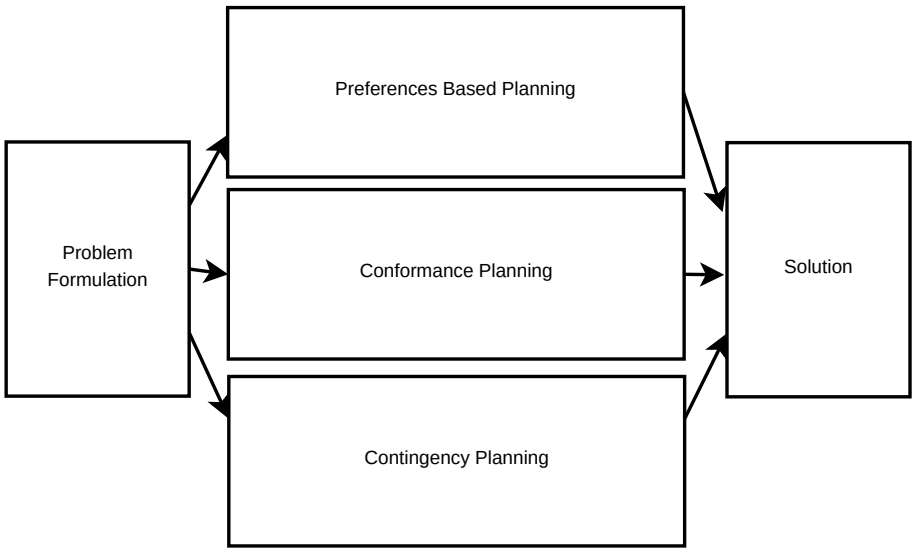

Figure 3.2: Planning Process.

first service to complete. Figure 3.2 shows a description of the required steps for problem formulation, where the planner or automatic environment has to consider preferences, find suitable services according to preferences and will follow a contingency procedure, in case of failure.

- The planning environment will be able to recognize the control flow patterns. As discussed before in ideal technical composition framework, services should be executed in sequential order, however this is a time consuming process. Some time this is not possible, as the service have to wait for the subsequent services to complete before it can start its own processing. In this case it is necessary to use sequential. If the services are not depending on each other and executable in a parallel fashion, the system has to recognize and adopt such changes, so the requirement is that the built in functionality is to recognize the control flow patterns.

The above mentioned concepts have a significant part in the services composition model. By reviewing and analyzing the different planning based attempts, we tried to propose our framework model which is also considering the above mentioned

limitations.

3.3.4 Conformant Planning and DWSC Requirement

In current work (DWSC), the concept of conformance planning is adopted to map the final results according to user preferences. The preferences will be analyzed before doing a semantic analysis of the available services. As discussed in the last section, there are two types of strong preferences, firstly where the user wishes the planner to consider certain facts and secondly where the user choices will be analyzed, to include some functionalities required from the composition process.

According to the above mentioned user preferences, the system will be able to provide optimal plan by conforming user preferences (details in the next section). The final results (after conforming preferences) will provide options to the users to select a solution of their choice [78].

In non deterministic domains the real time behavior of the participating services is not predictable in advance. In the above mentioned example of the logistic domain, the service S_i is depending on the execution of of S_{i-1}. This dependency can be the response from a GPS [1] service or information about products to deliver but the composition process cannot carry forward the plan without getting a response from services. There is a need to attach the planning heuristics at each node to overcome the problem of missing values or to find alternative nodes to include in the search space. To test the conformance planning facts, the semantic contents based mapping concept was being utilized. We argue that the constraint at any stages of failure during the composition process should be analyzed, and the results will be provided to the user by considering and providing a user choice based solution.

The node distance may also be considered to differentiate the related concepts,

[1] Global Positioning System

but for testing purposes we used a manual approach by taking and considering the aforementioned matching criteria. To explain the node distance concept, let's consider that X(0,0) represents the first node, and contains the conformance constraint by the user. X(1,1) and X(2,2) are the respective nodes at the next level and so on. The nodes at each level will be calculated by using the semantic similarity of the concept, so, if two concepts appear at the same level and both have values such as X (1,1) and X(1,2), both will be considered.

3.3.5 Examples of Static Composition Methods

- **Fusion** is a complete software package which provides a framework to specify user's basic requirements [173]. In the FUSION environment, the User Specification Subsystem (USS) is a graphical form-based interface that allows the user to (a) to specify his abstract requirements, and (b) convert this abstract specification into a more structured form suitable for consumption by the downstream plan generator subsystem in the architecture [173].

  User specifications include the set of methods to be invoked and a logical expression representing the user's satisfaction conditions. The Web Services Dynamic Plan Generator Subsystem (WGS) takes as input the set of parameterized method calls produced by the USS and generates a correct and optimal execution plan [173].

  The Plan Execution Subsystem (PES) takes as input the plan generated by the WGS, evaluates it and, maps it to executable code. The Verification Subsystem (VS) verifies that the service result meets the user specified satisfaction criteria. The Recovery Subsystem (RS), initiates a recovery process, which rolls back (to the extent possible) the effects of execution up to the point where the results are actually going towards the wrong direction, and finally

the verified result produced by the VS is input to the User Response Generation Subsystem (URGS), which is responsible for preparing and delivering the final response to the user. Debra VanderMeer also presents the Web Services Execution Specification Language (WSESL), a language that was developed to describe execution plans in the context of the FUSION services model [173].

- **Triana** was originally developed for data analysis and to support systems for analyzing gravitational wave signals [106]. Triana is an open source, distributed, platform independent Problem Solving Environment (PSE) written in the Java programming language. A PSE is a complete, integrated computing environment for composing, compiling, and running applications in a specific area. The Triana PSE is a graphical interactive environment that allows users to compose applications and specify their distributed behavior [106].

  A user creates a workflow by dragging the desired units onto the workspace and interconnects these by dragging a cable between them. Although the focus here is on the graphical interface, Triana consists of a complex set of interacting components that create the complete system or any subset. This federated approach gives Triana, the flexibility it needs to be able to be applied to many different scenarios and at many different levels. For example, it can be used as a workflow engine for grid applications; for connecting data driven grid components and managing the workflow between them, or as a data analysis system for image, signal or text processing applications; allowing a scientist to quickly apply algorithms to data sets and view results [106].

The following sections contain the description of dynamic composition approaches by different research groups.

3.3.6 Examples of Dynamic Composition Methods

- **eFlow :** Fabio Casati and fellows (from University of Trento) developed an environment (eFlow) to compose web services, which was one of the first services composition paradigm at a very early stage of the services composition process [35]. The eFlow composition model allows users to monitor, analyze, and update the process but technical ability is required by users to carry out simple tasks of composition. The eFlow system is modeled by creating a graph [34]. In the created graph, nodes represent the step by step execution of events while the graph arcs shows the dependency of services. QoS issues are not considered in eFlow however the user can change the sequence of execution of services and execution flow rules by manually changing the graph [82].

- **METEOR-S:** Abhijit Patil and fellows (from The University of Georgia) developed a semi-automatic tool for services composition called MWSAF (METEOR-S Web service Annotation Framework) [133]. The traditional composition life cycle is improved by adding semantic contents to discover services. Four different types of semantics characteristics are handled in MWSAF to support the discovery and composition process and to achieve the correctness goal [174]. The overall system was divided into three main components; Ontology store, the matcher library and the translator library. The ontology store contains ontologies DAML-S(DARPA agent markup language for services) and RDF-S(Resource Description Framework Schema) which can be stored by users in predefined domain folders. The translator library (WSDL2graph and Ontology2graph) contains programs to generate graphs before the execution of selected services. The matching library contains the algorithms to initiate the discovery process of related services by considering composition requests [133, 174].

- **ENQUIRER :** James Hendler and his research group (from the University of Maryland), utilized the HTN (Hierarchical Task-Network) and developed the algorithm ENQUIRER to support services composition [92]. The main advantage of ENQUIRER is that the process can start with an incomplete initial state. The ENQUIRER framework was based on SHOP2 [92].

- **User Preferences :** One of the leading efforts to consider user preferences is by Jihie Kim and Yolanda Gil (from the University of Southern California) where they explain how to formulate user goals at high levels of abstraction while the system is required to consider the user preferences and to finalize the solution [83]. The constraints are also considered and it was argued that choices made by users need to be checked at every step which will help to narrow down the search space and provide efficient solutions. The experiments and results showed that the system only considers linear composition and it is very difficult to handle alternative solutions by using a selective and iterative approach, which is also a requirement of the dynamic web services composition [83].

- **SWORD :** Shalil Majithia and David W.Walker present an architecture to facilitate automated discovery, selection, and composition of semantically described heterogeneous services using Semantic Web technologies [107]. Their framework has three main features which distinguish it from other work in this area [107].

  - Firstly, they propose a dynamic, adaptive, and highly fault tolerant service discovery and composition algorithm.

  - Secondly, their framework can distinguish between different levels of granularity of loosely coupled workflows.

- Finally, their framework allows the user to specify and refine a high level objective.

SWORD is a toolset that allows service developers to quickly compose base web services to realize new composite web services [142]. SWORD can compose information providing services (movies, theaters, restaurants, etc) and a class of other services (such as email and image conversion services) [142].

The key idea behind the SWORD framework is as follows:

1. Individual services are defined in terms of their inputs and outputs in an entity relationship based "world model". Given the inputs and outputs of the service, a rule is then defined which indicates which outputs can be obtained by the service given which inputs.

2. When a developer wishes to create and deploy a new composite service, they specify the inputs and outputs of the composite service in the world model and submit it to SWORD.

3. SWORD determines, using a rule engine, if the composite service can be realized using the existing services. If so, SWORD generates a composition plan for the composite service.

4. The developer can then view the generated plan and if appropriate, request that a persistent representation of the plan be generated. This representation contains the sequence of services that needs to be invoked to obtain the composite service outputs from its inputs.

5. When an actual request for the composite service is received, the service(s) specified in the plan are executed starting with the known inputs, in order to compute the desired outputs [142].

3.4 Optimal Synthesis Plan Generation (OSPG)

The Optimal Synthesis Plan Generation (OSPG) environment is a proposed framework for the dynamic web services composition process, which provides a solution by synthesizing a number of available results and following the attached heuristic requirements, provided by the user. The framework can be employed, however, for different domains such as weather forecasting, police web services, route planning, logistics, e-commerce and NHS Bio-medical research. The system's capability to work for the other domains is explained in the thesis by discussing the different domains.

The framework, Optimal Synthesis Plan Generation (henceforth OSPG), is proposed through analysis of the input requirements from the users and by addressing the limitations (discussed in chapter 2) of the current approaches. *Optimal* refers to choosing the best solution from a set of available alternatives and the word *synthesis*, suggests that the plan is generated by a combination of different optimal solutions and synthesizing concepts.

In the current research project, the aim is to consider user preferences along with the main goals of the composition process. These preferences may be for a particular service, or a number of services (such as for a flight booking, the preferences may be to book with a favorite carrier, one with the facility of Wi-Fi in flight, or that flies directly from X to Y airport). As all users (service requestors) are not technical, the objective is to design a framework that provides the user's final results without involving users in the technical complexities of the integration process. The requirement of the system is to recognize the requirements, and provide users with the best solution.

In some situations, it may be that all inputs and preferences are not in an acceptable format for the system to consider, or services are not available due to

attached restrictions (such as security, or membership). There are many real-world examples in which the dynamic services composition concept is required. In distributed systems, the applications have a message-oriented structure whereby the service requires a response from the domain interface, before providing any output or presenting itself for the composition process. Different services can be utilized by exchanging messages, according to predefined protocols. The requirement of the DWSC, however, is that the composer component must analyze and overcome on all of the technical overheads.

3.4.1 The Planning Environment Design

The purpose of the proposed framework model is to reduce the complexity and time needed to generate, and execute, a composition process and improve its efficiency by selecting the best possible services available at the current time, and also by considering any other QoS issues. The services registration process starts when a new service has been registered in a service repository [125]. The translator is required sometimes, if the service description is not in a particular format, or language conversion is required. The service descriptions are published for the users or software agents to consider when the service functionality is required, while the developer can improve the contents, functionality or services at any time. The requirement of the composition process, therefore, is that the updated contents will be considered at the time of request.

At a very basic level, the requirement of the composition process is that the system must search for, or select, related updated services. Abstractly, the process starts when the service composition request arrives at a web server [125]. The server will attempt to locate its own services database initially, and if such an interface-based composition already exists then an integrated result will be sent to the client.

Otherwise, the server will try to search through a web services database [125]. The concept of developing organizational own services database structure is a new one, as discussed in the recent services composition effort, and its main objective is to improve the QoS factors (discussed in detail in chapter 4). The extra services database setup is a requirement for the future, as the emerging concepts of cloud computing requires an extra layer of security. Further, if the small architectural blocks are in place, it will be easy to manage the single entities. The composition process environmnet can work on its own by communicating with external servers. However, as discussed, the organizational personal services database is a requirement for the future.

If the required services are unavailable in the inbound server, the web server will try to find the desired services through a matching engine (software agent or planner) from the web service database [125]. At the first instance, an evaluator module will evaluate these services on the basis of an interface-based search in the first round. In the second stage, the evaluator will apply functionality-based rules, according to the provided inputs. The selected services will be composed and the services address returned to the web server [125]. The composed solution results will be sent to the composition requestor, while a copy of this services integration will also be saved in a service repository (services database) for future use [125].

The above procedure for services composition is in its high-level abstraction form. A complete, integrated framework is required, which can automatically perform the discussed tasks while the requirement is to create methods for automatic discovery and the composition process [125]. In particular, the methods will be able to cope with any problems associated with the distributed, independent, and uncertain nature of the web. Individual service availability, reliability, and quality, are factors that make composition more difficult, though the discussed components, such as

evaluator, match-maker, and composer, are required to develop a logical concept, building environment that can work by analyzing the input states and required goals [125].

By using the AI-planning-based solution for web services composition, the issues related to services composition can be solved [163]. The planning-based composition solution will be helpful, not only for web services domains, but for different combinational optimistic problems. To understand the mapping of the normal domain into an AI-planning problem, let's consider an example of the Employee Vetting Procedure (EVP) domain. This domain is related to the vetting procedure for individuals (employees) to screen for access to rights and duties in the national security departments. The vetting process consists of an investigation to determine whether the previous background of the candidate is a matter of concern for the respective department, while pre-employment screening systems must consider different services that depend on the nature of the job. The various different services involved during the composition process are: previous employment history checking service, criminal record checks, qualifications check, medical check, and investigative services. The details of the vetting checks are available on Capitarvs website: http://www.capitarvs.co.uk/services.

Whenever an employer wants to start the vetting procedure, they should provide all related information. The composition system will generate a sequence of plans that will further invoke a number of related services. Let's map the EVP problem into a simple HTN planning domain. In the HTN planning domain, the objective of the planner is to perform simple (primitive) tasks instead of achieving one particular goal. In the EVP example, the planning environment must analyze and decompose the main task. In the second step, if the sub-tasks are still non-primitive after decomposition, then the planners search for further decomposition. Eventually,

the planner will achieve the final stage where all tasks are primitive. In the EVP domain, the main task will be divided into respective services, such as record checks, qualification checks or CRB (Criminal Record Bureau). A few services will be, again, compound services (like CRB or qualification), and then the planners will traverse towards other nodes before decomposing these services to primitive tasks.

3.4.2 The Procedural Context

The motivated research topic for integrating existing available web based solutions into composite services is also increasing, predominantly because new, and more useful, solutions can be achieved. To achieve this dynamistic goal, the first step is to introduce semantic web concept. *The Semantic Web aims to enrich Web services with a layer of machine-understandable metadata (data about data) to enable the machine to process information and services*[124]. Online services-based research is headed towards the discovery process of services, while, currently, the AI and Web development community are trying to achieve two goals service discovery, and service composition.

The second step is the requirement for ontological representation of concepts and knowledge, where the semantic relationship between the concepts will make it easy for the system to locate, and conform the required entities. Without considering the above steps, it's impractical to handle the composition project. There are numerous examples of where internet users gain optimal results after the composition of different services, such as, if a user wants to travel it is not only sufficient to book a flight, but also to reserve a hotel, rent a car, book tours to visit desired places in the specified time, and so on. Such composition is carried out manually nowadays, meaning that a user needs to execute all these services one by one, which can be both time, and effort consuming [124, 125]. For this reason, the notation of composite

services has been identified as a collection of services combined to achieve a user's request. In other words, from a user perspective, this composition will continue to be considered as a simple service, even though it is composed of several web services [124].

As the planner considers and takes services in the composition plan, the services are typically linked and executed, either in a serial or parallel pattern. To understand these kinds of patterns let's consider an example. In a ticket reservation process, a person wants to buy a ticket to watch a rugby match but only if he can get a hotel reservation (near the stadium) and an airline ticket. The two services (hotel reservation and airline ticket) will be executed in a serial pattern, and the next service will be invoked on the basis of the output of the previous two services. If the first two services provide positive results, then the next service will be invoked, otherwise the plan will be unsuccessful.

In the above rugby match example, if the parallel execution pattern is adopted then the composition process will be unsuccessful, with some of the required results not available, such as if the match ticket and hotel room is reserved but no airline ticket is available. In this particular example, the serial pattern is adopted, which means the user can get the desired results a time consuming process. The parallel (Split Join) process can be utilized by using planning-oriented solutions, where the sequence of steps can be handled by the planner, resulting in error-free, and timely response-based, updated results.

3.4.3 Preferences for Services

The proposed solution works by taking inputs such as, the initial point, what to achieve, constraint values and under what conditions it needs to achieve then. Here the input values are constant parameters, such as input variables (such as dates, id

numbers, amount to spend, starting postcode, etc) while the goal state parameters are what is expected the planner will achieve (travel plan, appointment schedule, route plan etc). The constraint values are user provided conditions, and by considering the conditions the planner must generate a solution. The constraint values are domain dependent, as, for different domains, different types of constraint are required. In the current work, the services composition constraint values are bundled with the actual input, while the final output results will be located by analyzing constraint values. As discussed, the user constraints can be divided into three different types for the DWSC process.

- User Preferences for Goal (UPG)
- User Preference for Quality of Services (UPQ)
- User Preferences for Process (UPP)

The UPG is to get the user, or desired, preferences for the final results, and these must be considered to achieve the final goal. The UPQ are about the preferable parameter values for quality oriented descriptions of the service, while in the UPP, the user can specify how the process of composition can be achieved; i.e; in which sequence the plan is required, and what steps or ways to consider before or during execution. There is a significant difference between UPG and UPP, as UPG is focused more on the final goal, while UPP is geared towards the planning process and procedure.

To achieve the required results, the planner must consider the constraints attached with each initial stage of the planning step. To discuss this further, there will be user interaction at the start of the process, and while all input parameters are satisfied, the user has no need to interact during the execution process. As discussed, there is no technical expertise or skills required from the user, and in its

simple form, the environment can work by simply publishing the interface where the user has to provide all the details. After processing, the user will obtain the final results. The proposed framework model is designed by keeping in mind the future development in the area of web services. The model provides support to conform user goals, to map the values and to provide users with an optimal solution that is not just the result of a simple search, but also shows intelligent behavior. The next step is to map the problem into a planning problem. But before approaching this stage the system has to tailor the inputs according to the required outputs. Lets assume that (I) shows the initial input value for the plan and (G) is the goal to achieve by considering individual constraint values.

ψ represents extra offered functionality according to user requirements (details in section 5.3)

$$\Phi = I, G, \psi \tag{3.3}$$

while ψ results should be finalized by following the discussed (UPG, UPQ, UPP) concepts, i.e;

$$\psi = P_{UPG} \bigoplus P_{UPQ} \bigoplus P_{UPP} \tag{3.4}$$

The planner will try to get the required solution by finding the required web services, and combining services. The process of the search for appropriate services will be initiated by the planner, and as discussed before that, different methods of search can be used. As in the proposed model, the DWSC domain is employed, so it is a better idea to discuss the concepts from a composition point of view. When the planner looks for a service, it matches the basic input of the service, or in a combination of semantic or pragmatic matches, the planner will try to locate the services. The available services provide results in different types. Some services provide just one output, and some provide a composite output

$$\Delta = [compositeoutput] or [singleoutput] \tag{3.5}$$

Here one output means the service will just return one type of variable which is not the final output but will be used for the final output(intermediate results) i.e;

$$(\Delta) \equiv O \in Xi \tag{3.6}$$

where Xi is a set of outputs and O represents single output.

There are two types of outputs are available, the services which are providing the single output and the services which are providing composite outputs. The planner has to decide which service to include in the planning process. In OSPG, the search method will run in three different cycles. In the first cycle the planner will try to locate the services which are providing single inputs. All the related services will be considered if they are offering single results and the services will be composed to provide final results. In the second cycle the planner will try to locate the available packages and try to find the cheapest package. In the third cycle the planner will try to find all the services providing partial results (Details in chapter 4). At the moment for testing process, the cyclic process is adopted which is time consuming cycle as all the services will be checked first and the missing services according to the goal requirement will be filled by selecting the best single services. For packages, the OSPG takes taking variations of user preferences and selecting services as discussed in chapter 4.

As discussed in the last paragraph, services will provide outputs which will be either single output or one service is providing a output from multiple services. Usually, the services are available in the following two separate settings.

- Services With Published Description
- Services With Compiled Results (Packages)

The Services with published results can be addressed in a smarter way as single services and the planner can select these services by matching them with their pub-

lished syntactic, semantic or pragmatic characteristics. The composite or compound services can be mapped into a HTN (Hierarchial Task Network) planning task. To produce a plan, the main task can be divided and primitive tasks can be handled as discussed in chapter 2. For non deterministic domains, it is difficult for the planner to identify and locate composite services (packages). For example if a travel package is available with some extra output the planner will selecting for the composition process. It has also been problematic issue to decide that the compiled services (henceforth packages) can either be taken on their own or with a combination of other services. There are also some limitations of the services composition process, even if we consider single services. In the following example scenario, the use of single services which is the basic requirement of the DWSC is explain in theoretical context.

3.4.4 Example Scenario

To take an example of route planning, consider that a PhD Student is driving from Huddersfield to London for a one-day trip to attend a conference. The Satellite Navigation System will provide the directions from Huddersfield University to Kings College, London, with the conference scheduled to start at 9am. The student heads towards London by allowing an extra 2 hours for his journey time, but due to an accident on the M1 near Milton Keynes, he has to drive 20 miles per hour for some time, and even 10 miles per hour at times. There is no alternative route plan if the student is interested in continuing to use the directions of the Satellite Navigation System, even though there are signs displaying the approximate time of travel to central London. In such a situation, there are only two alternatives for the student one is to wait, which could result in a late arrival. The other is to catch a train. There are information services available that can help in re-planning

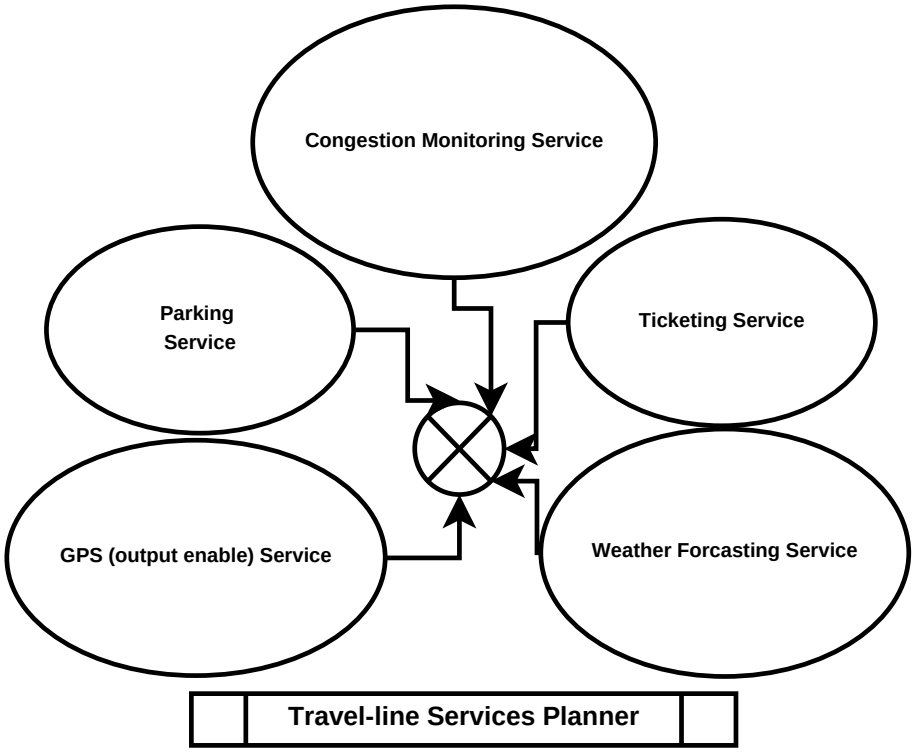

Figure 3.3: Travel line planner.

the tour plan, and can provide alternative options, either on PDAs, smart phones or satellite navigation systems. With the help of services composition applications, the planning environment can be employed to search for the best plan. Figure 3.3 illustrates the travel domain services and connectivity concept.

In the current proposed model, the discussed travel problem is mapped into a planning problem. Along with the main goal of reaching the destination on time, there are some strong and soft constraints attached with the problem domain. For example, the student would prefer that if there is congestion, or any other interruption, then the travel line services planner will run, providing alternative routes or travel options. The student is also interested in if the planning process updates itself two miles before the coming junction, to enable safe, advanced planning, and to decide which option is the best. The plan would be, for example, park your car in car park X, near the train station, and catch a train from Milton Keynes to London.

The online services are all around, such as congestion monitors for road traffic updates, congestion and parking charges, train journey planners and weather forecasting updates. It is possible that the new planning steps will be presented to the user during the course of travel, for example if he is travelling by car, it will cost him £40 (including congestion, parking and petrol charges, along with the estimated time to reach the destination) while if he continues his journey by parking his car in Milton Keynes it will cost him £20 (including parking charges, train ticket price, along with the total time saved information). If we look closely at this real world scenario, there are preferences or constraints that must be met in the planning process, such as time (a strong constraint) to reach the place before 9am. An example of a soft constraint is that the user can walk inside the city, but not more than half a mile. The constraint and preference for composition is discussed in more detail in chapter 4.

To further analyze and to explain the concept of preferences (UPG,UPQ,UPP) as explained in the last section, the time constraint is attached to the user's final goal (UPG). If the time efficient, or cheapest, plan is requested by the user then it will represent an example of UPQ. If the user wants to change the plan sequence by, for example, taking the next junction (Luton) to catch a train instead of travelling as far as Milton Keynes, the planner must adopt the changes, according to the user requirements. In the current example, the planning steps will be changed, instead of the initial starting point of Milton Keynes, to Luton.

(:goal

(and (Reach-by Time-x KCL)

(Preference Conferplan1

(imply (route-congestion))

(sometime (initiate (travel-line-service)))

```
(Preference Conferplan2
(sometime-after (terminate
(plan(travel-line-service)))
(Initiate (before -next-junction
(2-miles))))
(Preference Conferplan3
(sometime (occ (prefer-walk (not (half-mile
in-city)))))
)
(Preference Conferplan4
(always (not (occ (travel-by bus)))
))
```

The Conferplan1 preference suggests that the plan from the travel line service will be executed if there is congestion during the journey at any point. It will call the composition of the services plan by consulting with services such as the congestion monitoring service, parking service; GPS Output enabled service, train operator services, and weather forecasting services. The Conferplan2 preference is to complete the process of updating the plans, and providing options two miles before the junction. In Conferplan3, the person is ready to walk in the city if the overall distance is less than half a mile. The concept will be applied to search for direct trains by walking to the nearest train station. Conferplan4 shows that the user doesn't want to use the bus.

Although some of the required services, mentioned above interfaces are available in the UK, it is difficult to adopt such a complete model due to a lack of resources and integration of heterogeneous systems. However some of the options are adopted in the Navigation systems for route planning, to calculate the best possible solution.

The next step will be to analyze the planning steps and if any of the required values are missing to try to fill them. The replanning sub function will be called if the planner does not find an optimal plan which will reduce the search space and try to take the system goal approach into the nearest state (as shown in Figure 4.1). However, the contingent part is not analyzed in detail, as our priority is extended towards the dynamic discovery of services according to user preferences.

3.4.5 Concluding Remarks

The main objective of this analysis is to address pre-compiled services by considering their limitations. It is infrequent that one web service provides all the user required results. Services have to be composed from various domains to achieve the desired results. The applications of services composition are in travel, banking and traffic control systems as these are some of the domains which are developed with outstanding results. The efficiency of the composition process depends on the discovery of the services from different domains.

In the current chapter, the different concepts were discussed for the required services discovery and composition process. The discussion includes the requirements of OSPG and how planning concepts can support the process. Current chapter describes the AI planning concepts to support the DWSC process and consideration of different (type of) services. The next chapter will explain the proposed framework requirements to carry out the composition process by evaluating and analyzing the available searched results.

Chapter 4

Dynamic Web Services Composition (DWSC) Framework

4.1 Proposed Framework Model Support

To support the Web Services Composition (WSC) process, the requirement from the available online resources is to integrate them in such an arrangement whereby the services (and processes within service) can interact with each other automatically [58]. As discussed in chapter 2, the web research community has been working in two different directions to achieve the service discovery and composition goal, by using Workflow and AI planning techniques [16]. The Workflow approach has its own merits and disadvantages (discussed in subsection 2.2.3). The AI planning based solution is being utilized in this thesis. By following AI composition concepts, the overall process of services composition can be divided into two sub processes, *Services Discovery* and *Services Composition*(as discussed in subsection 2.2.4).

After a client request for the composition process, service discovery is the finding of all related services from all of the providers. By considering user preferences (for

example tour time, food preferences, budget, and travel arrangements)[1], services should be discovered for the composition process at the first stage of the process in order to achieve a particular goal, after which the process of composition will be initiated. If the users demand is known well in advance then it is easy to compose the services and provide the user final results. However to map the user's request in real time is a complex task [157]. To achieve this dynamic goal, AI planning professionals have provided a number of planning based patterns for the discovery and composition of web services, in which the results have explained how a planner locates the required services and will provide final output to the user [171]. Planning based solutions have been successful at solving complex domains with well-defined goals by taking simple actions. As discussed in the previous chapter, an end-to-end services automatic (environment) integration mechanism is required to enable services to find other related services and after integration, it would be able to automatically establish intercommunication links for DWSC process.

The proposed and developed framework - Optimal Synthesis Plan Generation (OSPG) as shown in Figure 4.1 is similar in structure to other previously proposed models [67, 88, 126], but will provide a compact and efficient context, where the planning environment accepts input from a user and provides an optimal plan. The goal is achieved by searching for the required services from a pool of single services, composite services and services contains packages. The planning environment provides the functionality to generate an optimal plan, while the integrated discovery process will finalize the services. There is a requirement of the Dynamic WSC (DWSC) process, to provide results to the user after evaluating the functional and non-functional requirements and to identify an optimal plan. At present, there are many automatic (or semi-automatic) environments available to support the DWSC process; though it is still a very difficult task to integrate these services given their

[1]As mentioned in subsection 1.1.2

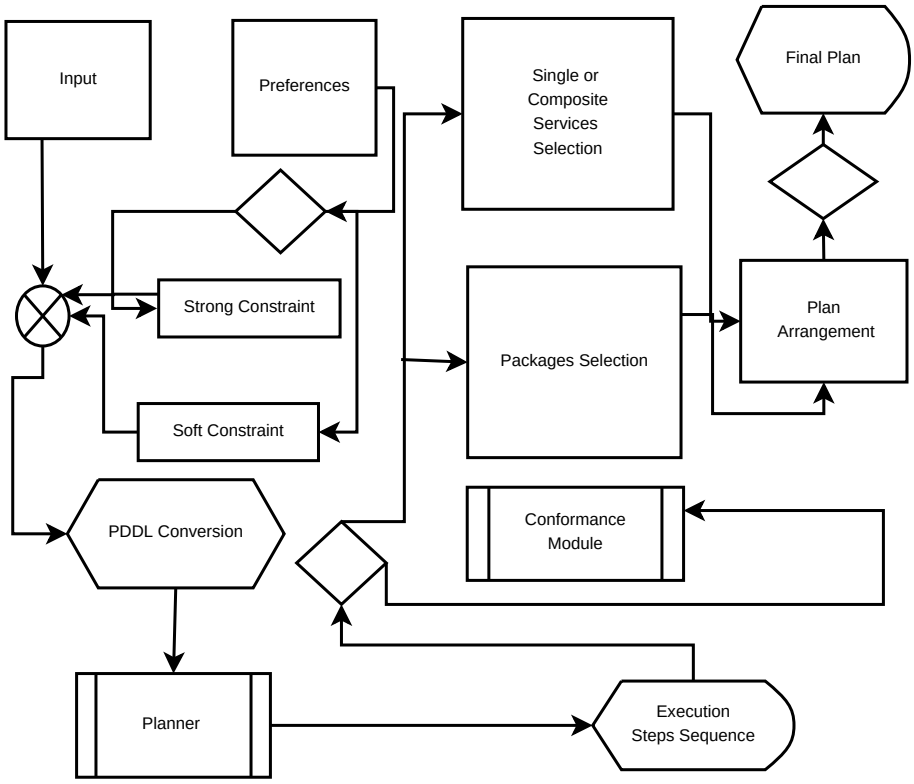

Figure 4.1: Planning Process For Web Services

different designs and developmental approaches. To test and evaluate the OSPG, a tool-based environment with integrated planning tools and intercommunication interface is required. Although the complete end-to-end framework model is proposed for the DWSC process, the main focus of the thesis is to evaluate user preferences, consideration of optimal solution and to explore the characteristics of conformance planning to strengthen the DWSC process.

The OSPG framework carries forward the ongoing research by extending the discovery process and utilizing user requirements which are complex in nature to provide an optimal plan. A gap does exist in research on the selection of services; no one has discussed the selection of services by equating a combination of single services, composite services and services which contains packages.

As discussed in chapter 1, the significant contribution of this research is to look closely at the user preferences in order to achieve the desired end product i.e; the op-

timal plan. For the output it argues that the planner should find the best results by incorporating the results of single and composite services (partial and complete packages). As discussed, the available approaches for web services composition problems works very well for one particular domain, where the framework is actually tested, or developed for [18] while an automatic environment is required which will be able to support different domains. In the next section, different level of compositions and requirements of the current system are explained while the subsequent section contains the details of different types of services.

4.1.1 Functional Level Composition (FLC)

During the services composition process the planner has to search for related services. The planner or agents select services by assuming services as a black box [without knowing the functionality, number of sub processes and IOPE (Inputs, Outputs, Precondition and Effect) requirements of the service] which means that the selection of services is carried out by evaluating services' published descriptions which are provided by the service provider [3]. The information directories attached with the service provide the functional and non-functional attributes. Functional attributes are attached details such as basic Inputs, Outputs, Precondition and Effect [23], while non-functional attributes contain information related to service details such as time, cost, QoS parameters and provider details. The service can be invoked by the service requestor in the semi-automatic composition process while can be searched by system if the fully dynamic model is implemented [3].

The planning based services composition tools some time rely on Functional Level Composition (FLC) but often the final results are not very appealing from an optimization and efficiency point of view and the composition process does not get the full advantage of all newly updated services from the providers. Sometimes the

advertised information about the service is not correct or the composition requirements are not in the defined format [95]. This means that there is the possibility that the searched services do not provide the desired output as the services' published information was wrong. The FLC compositions strategies are useful when the service is able to provide multiple functionalities. In that case, instead of searching sub methods within the services, the planners can carry forward the composition process by relaying on the services' published information.

4.1.2 Process Level Composition (PLC)

As discussed, the drawback of FLC is it does not allow the checking of functionality other than the basic input, output or description of the output which is not interpretable by automated services. For static composition (explained in chapter 3), it is possible to discover services by matching functional and non-functional attributes. However for the dynamic services composition process, it has been suggested to use Process Level Composition (PLC) [163]. In PLC the internal behavior and provided functionality of the services parameters are considered and then utilized accordingly [162]. For the PLC based concept and technical development, services providers have to provide a description of the internal application process and behavior by using OWL-S or WSMO. For example, OWL-S and WSMO provide the description of services at two levels. OWL-S addresses the main functionality of the service and WSMO provides the interaction model of the service with external interfaces. The composition process has to consider the featuresprovided by OWL-S and WSMO.

4.1.3 Proposed System Concept

As discussed, both methods (FLC and PLC) were being utilized in planning based attempts at web services composition however this also depends on the type of re-

quest for a service and to decide which method to use [97, 141]. One type of request from user is where the requestor will provide all the constant parameter values in advance and wait for the final results (see experimental details in section 5.1). For example, the services composition request for a car insurance quote where the user has to provide all details in advance and wait for final results. The composition environment will locate services and offer a few quotations to the user after composing different services. Secondly there are requests where the where requestor has to initiate the process and during execution of services, the user will again be requested to select some further options. For example, in the rugby match example, the user provided all the required inputs. After considering the request, the system will try to find the required solution and if the exact required solution is not available, the user will be presented with different options and then choose their preferred solution. According to the proposed solution plan and requirements of the DWSC process, both types of request are expected from the user. For the above mentioned two reasons, in the OSPG framework, both methods are employed. The PLC method is considered for single services while the FLC method is used to locate composite services or services (which contains packages). The details of services selection and composition for both concepts (FLC or PLC) are available in section 5.3.

As discussed, the requirement of the WSC problem is to map the problem into planning problem. Figure 4.2 shows the mapping of the services into a planning problem. The same mapping procedure is utilized in other planning based attempts [157, 163]. The suggested procedure can also support both Server-tailored or client-tailored approaches (Details in [5, 6]). The requirement of OSPG is to consider user preferences and then provide a solution by conforming preferences so in the final goal the user preferences parameters are added, along with other considered parameters values. The External Interface (User Preferences) in the figure, indicate the user

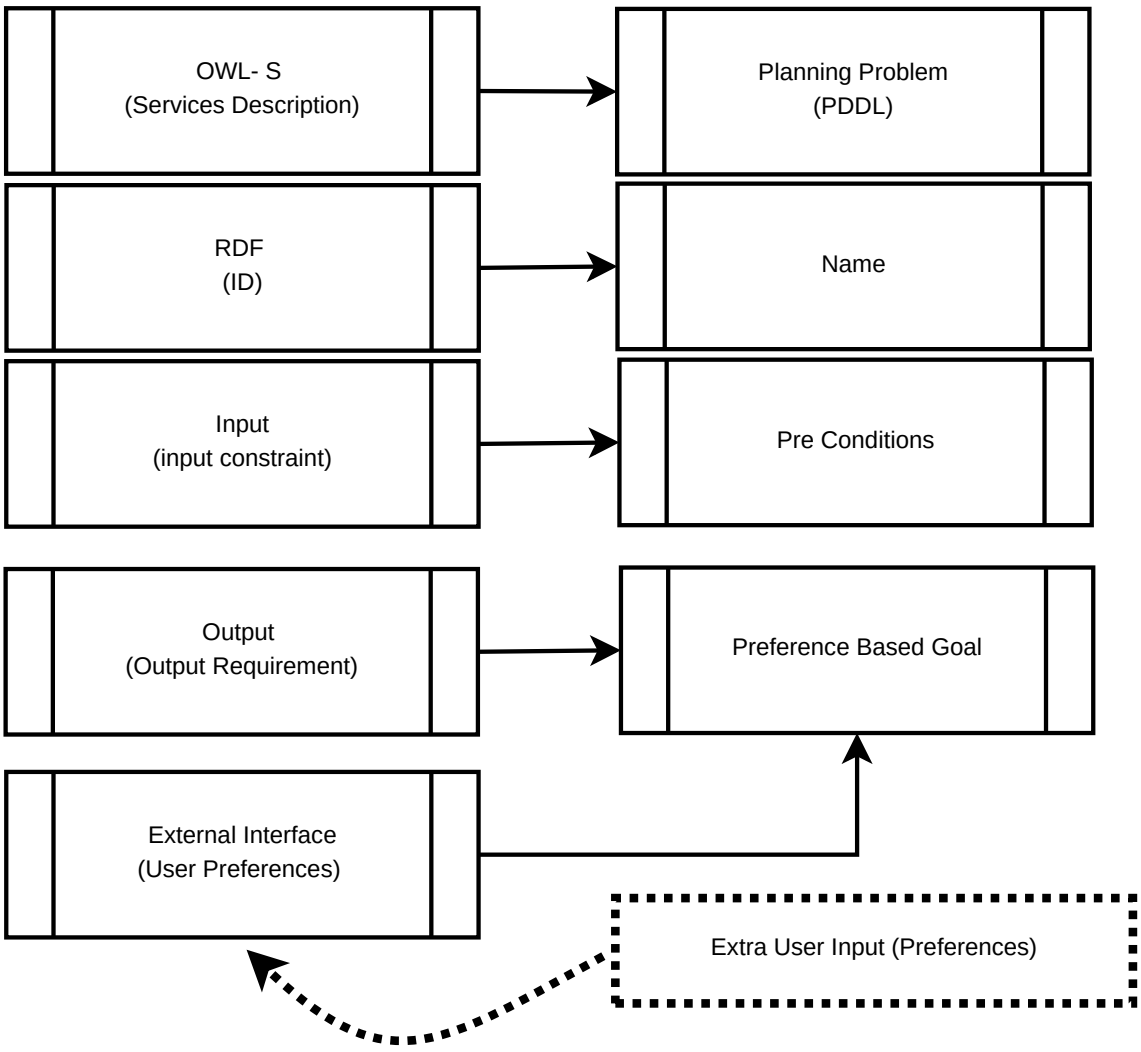

Figure 4.2: Service Conversion into Planning Problem

preferences based input, which are required to instantiate the DWSC process.

4.2 Services Analysis and Composition Process

As discussed, three types of services are required for the composition process, the services which are providing single results, services with composite results and services with packages. Before going into the details of the discovery and composition process, it is important to discuss about the type of available services.

Single Services: In OSPG framework, for the evaluation of services composition process, we consider the single service with a single output; where the service

is providing just one functionality. Like for the travel plan, the hotel services which provides a booking confirmation at any particular date, should be considered as a single services.

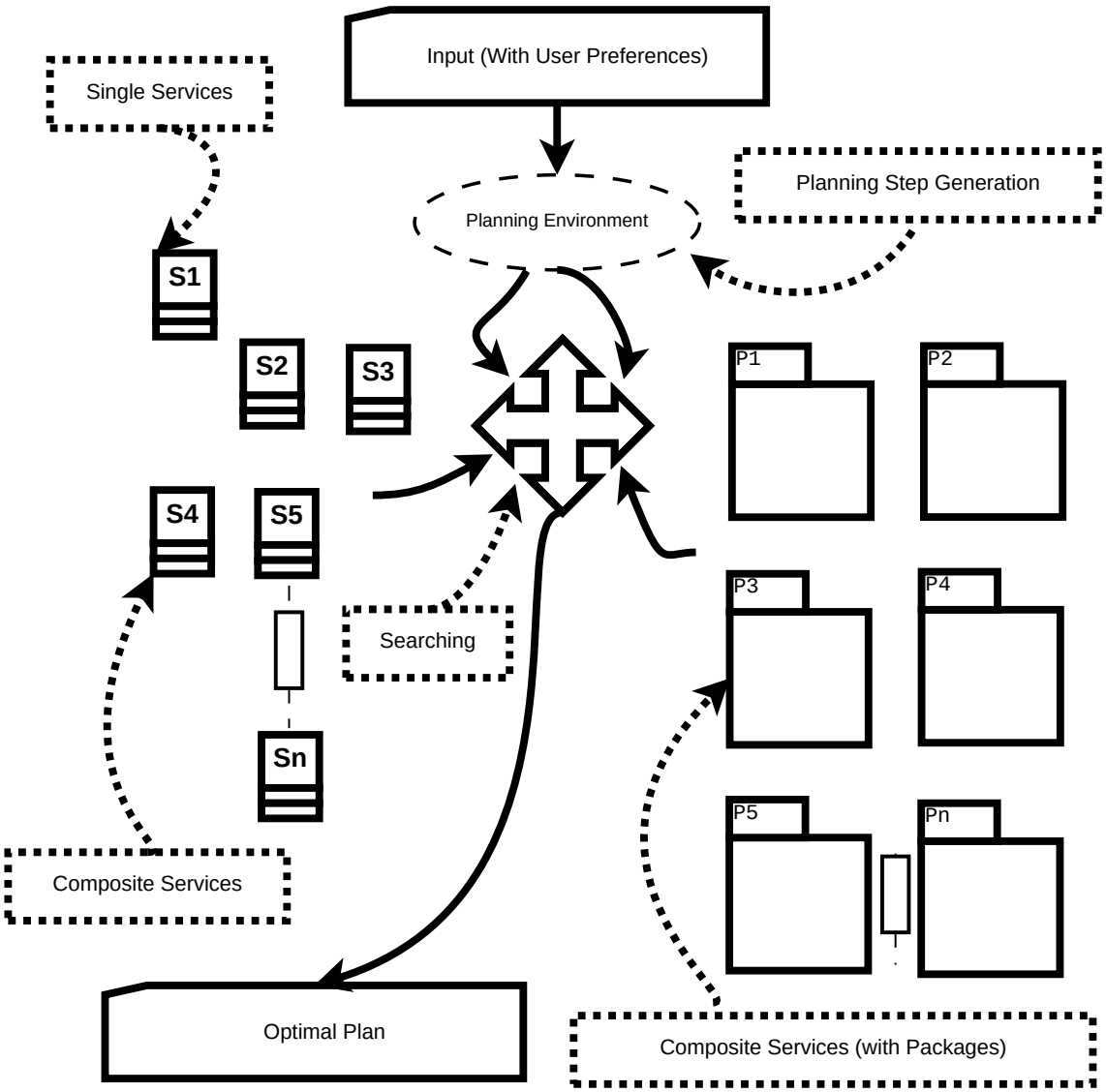

Figure 4.3: The process of OSPG

Composite Services: For the evaluation of the OSPG framework, the services which have not just one but few outputs according to required output, will be considered as composite services. Here the output means, the number of functionalities (outputs). For example in the rugby match tour plan example, the participating service in the composition process with hotel and taxi booking services results. But

not the flight booking service, which is also the requirement of the tour plan (flight service). If the services composition required three services, the functionality will be achieved by integrating single services with composite services. So the composite services provide full output required from composition process (Details in chapter 5).

Services with Packages: The advantage of the composition process is that it will facilitate users by providing an optimal solution. To search for the best solution, it is important that the system will consider all different combinations (different participating services results). For the composition process, the services contains packages which contain all the required services bundled together or providing extra functionality as well. Here extra functionality means that the service is providing all the required output plus some extra outputs (free travel insurance, music download and extra bonus loyalty points details in section 5.3).

To consider packages after taking user preferences into account is very important to achieve an optimal result. Figure 4.3 contain examples of three types of different services and services' consideration process, where (Sn) shows the number of services, (Pn) represents the number of composite services, composite services also contain packages while the optimal plan will be finalized after considering all such services. The next section explains the technical requirements and an in depth analysis of the required processes. The rugby match example will be used to clarify the following concepts (the scenario is discussed in chapter 3).

4.2.1 Technical Context

Definition (1): Web Service (Ws)

A web service is a three tuple w = (I, O, Int_X), where I is the basic interface, O is the output which the web service will return on invocation and Int_X is the interface description. The interface information will provide the basic information

to the related service execution and QoS (Quality of Services) related variables values which the planner will have to consider while executing plans for services composition. The interface contains the execution cost and any other information services. At a later stage the planner takes into consideration the Int_X description while planning the attribute values which will contribute towards an efficient plan.

Definition (2): Dynamic Web Service Composition problem (DWSC)

The DWSC problem can be defined as $(I, G, A, W_{si}, K_b, Q_s)$, where (I) represents the initial state, G is the goal state with all required outputs (O) by taking actions (A) while W_{si} represents the number of participating web services set in composition process. The actions are atomic and the final composition goal will be achieved by finding and composing appropriate services dynamically from the provided knowledge base (K_b). Here Q_s represents the QoS variable values. During the composition process it is very important to consider QoS factors. For example, in the rugby match scenario, user is expecting a cheap and efficient plan within a real time interval where I = [Req-Flight-Book, Req-Ticket-Book, Direct Flight, X Seat], G= [Flight-Book, Ticket-Book], A= [Step(i) search flight service and book flight and Step(ii) search ticket service and book ticket], K_b= [participating services repository], Q_s=[Direct Flight, X Seat].

Definition (3): AI planning based Web services composition by considering user preferences

User preferences are a range of possible characteristics or properties which are expected to be considered during the composition process. It is possible to express these properties before the composition process that specify under what conditions planners have to compose services or during the composition process what to avoid. The user preference based composition process within planning the environment

can be defined as $(P_{ri}, I, G, A, W_{si}, K_b, Q_s)$ where P_{ri} shows the preferences from the user. I and G represent the initial and goal states, W_{si} represent the set of number of services required for composition while K_b is a knowledge base which contains the basic heuristic and any constraint values attached with the services. The K_b also contains information about the alternative services which can be used in case the main service is not responding. If the overall services database is down then the contingent module will be called to obtain a simple plan according to user requirements without considering preferences constraint. Users can select the preferences which will be considered during composition task by the planner i.e; $P_{ri} = P_{r1}, P_{r2}, P_{r3}, ..., P_{rn}$. Suppose $\prod$ is a composition plan then

$$\prod = P_{ri} \bigcup I(\in K_b\,(W_{si}) \approx G) \tag{4.1}$$

Equation 4.1 shows that the final goal will be achieved by considering user preferences (P_{ri}) and initial state variables (I), the answer will be finalized after matching semantic contents and published information for the service in the K_b which will lead to the final goal G. When considering packages (P_{cki}) along with single services the following equation (Equation 4.2) will represent the overall process where the output will be finalized after analyzing packages. The notation ρ shows that both results will be compared first.

$$\prod = P_{ri} \bigcup I\,(\in K_b\,(W_{si})\,\rho\,P_{cki} \approx G) \tag{4.2}$$

Definition (4): Web services composition problem by considering packages (Pre-compiled Packages)

Packages are pre-compiled and linearly ordered set of services, which are composed by the provider, contain the individual services' descriptions in the interface (Int_X), and provide composite results. The interface for the particular type of services requires multiple variables to invoke the pre-compiled services.

Suppose Ω represents the pre-compiled available packages, $\Omega = P_{ck1}, P_{ck2}, P_{ck3}, ..., P_{ckn}$ R_{Ipi} represents the required input of the package and R_{Opi} shows the outputs which the planner will return by considering P_{cki} Hence

$$R_{Ipi} = R_{Ip1}, R_{Ip2}, R_{Ip3}, ...R_{Ipn} \text{ And}$$
$$R_{Opi} = R_{Op1}, R_{Op2}, R_{Op3}, ...R_{Opn}$$

The packages selection step will be achieved by the following pseudocode steps.

1. function (Φ): search_for ($\mathrm{P_{cki}}$) *(Initiating function)*
2. Inputs: Ω $\mathrm{R_{Opi}}$, $\mathrm{R_{Ipi}}$ *(Required Inputs and required outputs)*
3. Output: $\mathrm{P_{cki}}\,\alpha$ $\mathrm{R_{Opi}}$ *(Required Outputs)*
4. if $\forall$ $\mathrm{R_{Ipi}}$ $\in \Omega(\mathrm{P_{cki}})$ then *(Provided input matched with the packages interfaces)*
5. if $\mathrm{R_{Ipi\,(interface)}} \bowtie \Omega$ then (is acceptable) $\{\mid\}\rightarrow$ Ω is valid then

   (if the required input is part of the package and all required outputs are available in package then select the package(s))
   (the notation $\bowtie$ = Join of both literals $\{\mid\}\rightarrow$ = For all required outputs)

6. PUSH ($\mathrm{P_{cki,}}$ $\check{\mathrm{Z}}$)

   (step 6,7 and 8 shows the data structure of the matched package located will then be included on stack($\check{Z}$))

7. add [POP ($\check{\mathrm{Z}}_{\mathrm{i}}$)] $\rightarrow$ solution_tree *(inserting and adding in the solution tree($\check{Z}$))*
   else

8. delete_node (POP ($\check{Z}_i$)

   (Node will be deleted) endif

9. return (Φ)

The above pseudocode provides the basic criteria to locate the related packages according to user requirements. The first Line of code (Loc,1) is initiating a function Φ which will load the problem information. The discovery process requires the information about the available pre-compiled packages, initial input and output values. During the second step (Loc,2), the input parameter (constraints) will be assigned (initialized). The output will be the required package which match with desired output.

Loc,4 shows, if the given input is the same required to invoke the packages to start the service match process and if the set of all desired output values exists in the Ω then select that package. Here both concepts FLC and PLC can be utilized according to the requirements of the domain. However the PLC method is difficult to implement.

At (Loc,6): the package will be added to the stack if it matches with the required output R_{Opi} and in the next step the solution tree will be built up with the search results (with other similar packages). The selected packages of step 5 will represent one node and will be added to a tree. As discussed, (Loc,5) will match the set of all outputs with all the required packages. At the first instance, the system is matching all the packages against all the required outputs which means that the package will contain all the required output plus the extra outputs as well; which is mentioned in the Pseudocode with the symbol (The set of natural join mathematical notation $= \bowtie \{ \mid \}$). However to find the optimal solution we further divided the package selection into sub structures as discussed in the packages section.

4.2.2 Composite Services and Packages Exploration

To start with the example of the rugby match (the scenario is discussed in chapter 3), the planner will try to locate all available packages and compare them with the required output options provided by the user. In the current example, three types of the following results are expected after searching and considering all available packages.

- Packages with Exact Required output match
- Packages with Extra Results (including the required ones)
- Packages with Partial Results (some results are missing as required)

As discussed above, the requirement of our research plan is to locate, compare and include optimal packages (available pre-compiled services) and single services for the composition process. For packages selection, the planner will have to consider all the options of exact match, extra results and partial results. In our proposed model, the three different packages based concepts will be considered in the next section, but before going on let's look at the main pseudocode for the packages and services selection. The details of OSPG single services and package selection is explained in the following pseudocode.

1. Load pool_services *(load all available services)*
2. Start Traverse (Desired_Output) *(Initiating function for desired output)*
3. for i=1 ..n *(loop to consider all single services)*

   3.1 $Sol_w \leftarrow \forall \mathrm{R}_{op} \in S_{1...n}$

   3.2 $Sol_{comp} \leftarrow \{\mathrm{SORT}\ (Sol_w) \cup \Gamma_{\mathrm{x}}\}$

   $[where\ \Gamma_{\mathrm{x}} = (constraint\ attached\ with\ input\ for\ example\ preferences)]$

3.3 Continue_Build

3.4 Execute _final output (F_{Xn}) $\leftarrow$ Sol_{comp} *(where* F_{Xn} *will contain the results from single services)*

4. for j=1 ..n *loop to consider all packages*

4.1 $Sol_{wp} \leftarrow \forall R_{op} \in P_{ck1...n}$

4.2 $Sol_{compp} \leftarrow \{SORT\ (Sol_w) \cup \Gamma_x\}$

4.2.1 while $\forall R_{op} \nexists Sol_{compp}$ *do*

(In case of partial package, the planner will complete the package by searching and including the required single service(s), for example in the rugby match example if the flight and ticket services are available and the planner has to locate hotel services (single service) to fill the package, according to user requirements)

4.2.2 $Sol_{compp} \leftarrow$Select_Required R_{op} $(Sol_{comp})\}$

4.2.3 Goto 4.2.2 Until $Sol_{compp}(Complete)$

4.3 Continue_Build (Sol_{compp})

4.4 Execute _final output (F_{Yn}) $\leftarrow$ Sol_{compp}*(where* F_{Yn} *will contain the results from single services)*

5. $F_z \leftarrow$ Q_f [F_{Xn}] ||Qf [F_{Yn}]

(Both results obtained will be compared here)

5.1 Convert_format (F_Z)

(The final services will be forwarded to the planner to execute)

6. return (F_Z).

The first step (Loc,1) is to index the services and to load all available services. It is very important to check all the available services and to update the index of

web registries, There is an ongoing research (as discussed in chapter 2) on this topic to analyze the situation where more then one servers are available to provide the required services (registry) [112]. To establish connection before the composition process is also the main requirement of future applications for cloud computing (to initiate the links with all the available servers) [130]. At the second stage (Loc,2), the desired output will be analyzed while at the third stage the single services match will be analyzed.

For the single services matching process, to start with, the planner will match and locate services with single outputs (Loc,3.1). All the services which are providing partial results will be considered at this stage. In the next step (Loc,3.2), a sorting procedure will finalize the services by analyzing the constraint (Γ_x) values attached with each request. The constraint user preferences along with output are discussed in packages selections section but at this point, the service pool set obtained at (Loc,3.1) will be used to sort out the appropriate services. The same concept can be adopted for the QoS consideration to sort out services. Similar services (providing same result) will be compared on the basis of the user preferences based constraint or QoS factors. For example in travel plan for rugby match, firstly planner will find all services from pool of services providing single output like hotel, ticket price, availability and flight quotation.

During the second stage planner will find all services which provide composite output (packages) (Loc,4.1). The services are compared by considering the constraint vslue (Γ_x). The planner will analyze all packages and compared it with the final output, if the target is achieved then moving on third step (Loc,5). If the final output is not achieved (e.g; the complete package is not available), the planner will complete the package by searching the required single service(s) to achieve the final goal (required output). The (Loc,4.2.1) is showing the recursive instruction

sequence where the indicated missing services will be searched for which are not in the composed package. To continue with the example of rugby match, the planner will find related available packages and then will continue search and adding services (for example hotel or flight services) which are missing from desired package. After analyzing the available partial composite services, system will try to find the missing web services and try to finalize output of the package according to desired final output.

Finally the system will compare both outputs $[F_{\mathrm{Xn}}]$ and $[F_{\mathrm{Yn}}]$ (Loc,3.4,4.4). And return number of best services for composition process (sequence of steps). After selection process, the services will be composed and the user will get the desired result. In the above mentioned example planner will compare single out put combination of services and packages based output. By comparing both step results; user will be able to get the best rugby match plan. In this way we can save time by not traversing all the nodes and will get best results from selected services.

4.3 User Preferences Logic and Algorithms

Three types of the following forms of the package oriented structure are being proposed and evaluated by analyzing the results of the above mention algorithms.

1. Preferences Plus Package (PPP)
2. Complete Desired Package (CDP)
3. Partial Desired Package (PDP)

4.3.1 Preferences Plus Package (PPP)

Preference based planning is an important area in which to establish user preferences and interests [57, 157–159]. First-order language is suggested to support the syntax

and to take user preferences [22]. The concept of PPP (Preferences Plus Package) is useable in the classical and non-classical domains, where users have to provide their own choice of services selection and preferences in order to meet the final goal. *(PPP) defines the search and composition criterion to find an optimal plan by considering the user preferences, and providing the final output by conforming to the best results.* The main focus of this approach is that whenever the request for the desired composition package comes through, it will be bundled with the user's interest (preference or desired), ultimately providing the user with an optimal plan by analyzing different types of services (Single Services, Composite Services or Services with Packages)

The user preferences are divided into strong and soft constraints as discussed in chapter 3. The *strong constraint* are the user preferences to be met during the composition process, such as the User Preferences for Goal (UPG), User Preference for Quality of Services (UPQ) and User Preferences for Process (UPP). The *soft constraint* are the desired values from the user that the planner has to consider during the composition process in order to select different types of combinations for the user. The concept behind this setup is to consider functional and non-functional attributes for an optimal plan. To provide the user with a list of composed results by integrating all available services requires user input (preferences).

To consider the user preferences, the proposed solution is that the user should provide their preferences (which system will consider as strong and soft constraint) at the start of the process, along with the input and output requirements. The interface can be designed by asking simple questions, or a selection (choice-based), detailed form, where the selected method will be presented to map the user requirements and to locate the appropriate and beneficial services during the composition process(as shown in Figure 5.3). The question will be presented to the user to check the

user preferences, with the planner having to estimate (conformance planning-based actions) the best available solutions for the user, for example, if the user wants to make video calls on the phone, download music regularly, have free breakdown cover or mobile phone insurance etc. (Details are provided in section 5.3)

As discussed, the requirement of the dynamic web services composition process is that the user takes full advantage of all available services. The planning environment has to consider the user preferences and to search for the related services, and after the search process, the system will provide two types of results.

- Results which are showing the same output as the user requirements $\bar{O}p(i)$
- Results which are showing the same output as the user requirements plus an extra output (from Packages) which is not requested by user but was a part of the package during the discovery process $\bar{O}f(i)$

As discussed above, the system has to analyze the results, dividing them into two groups, the first of which, contains results which are the same as the requirement. The second group of sorted services provide the required output, plus an extra output, which means theses services contains results that are not requested by the user, but are included in the selected package; while the extra offered services will be analyzed in connection to the user preferences. The requirement is to check the extra output (results) and suggest to the user the advantages of the selected package, which will be included in the list by evaluating the user preferences. The following pseudocode provides a step-by-step selection process to search for the preference-based package.

1. *Load pool_services (load all available services)*
2. *Start Traverse(Desired_Output): (Initiating function for desired output)*

3. *for i=1 ..n (Loop to consider all required services)*

4. $Sol_{wp} \leftarrow \forall\, R_{op} \in P_{ck1...n}$ *(to select services which are containing results according to required output)*

5. $Sol_{compp} \leftarrow \{SORT\ (Sol_w) \models \bar{O}p\ (P_{r1...n}) \parallel \bar{O}f\ (S_{1...n})\}$ *(To consider the results offering same output* $\bar{O}p(i)$ *like user requirements and the results with an extra output* $\bar{O}f(i)$*)*

6. *Continue_Build (*Sol_{compp}*) (loop to consider all required services)*

7. *Execute _final_output (*F_{PPP}*)* $\leftarrow$ *(*Sol_{compp}*) (final output)*

8. *return(* F_{PPP}*) (Return final output)*

The above pseudocode shows an abstract view of the selection process. where all those services will be considered which are contain results according to required output (Loc,4). In Loc,5 the system will finalize the solution by considering Sol_{wp} and taking $\bar{O}p(i)$ and $\bar{O}f(i)$ into consideration. The process will continue until the (Sol_{compp}) is not completed. The final selection process of the extra offered services is discussed in next paragraph. The Figure 4.4 shows the selection sequence process which is logically represented by considering different levels. At level 1 and 2 user preferences will be considered in terms of (PPP, PDP, CPD), while at level 3 and 4 the constraint (Strong and Soft) will be analyzed. At level 5 and 6 the plan sequence will be generated by considering packages and services.

PPP can be divided in to two further types; Static Preferences Plus Package (SPPP), and Dynamic Preferences Plus Package (DPPP). The Static Preferences Plus Package is the selection of packages and services, made by considering the soft constraints, conforming to related concepts, and finalizing the composition process. The Dynamic Preferences Plus Package (DPPP), on the other hand, is the selection

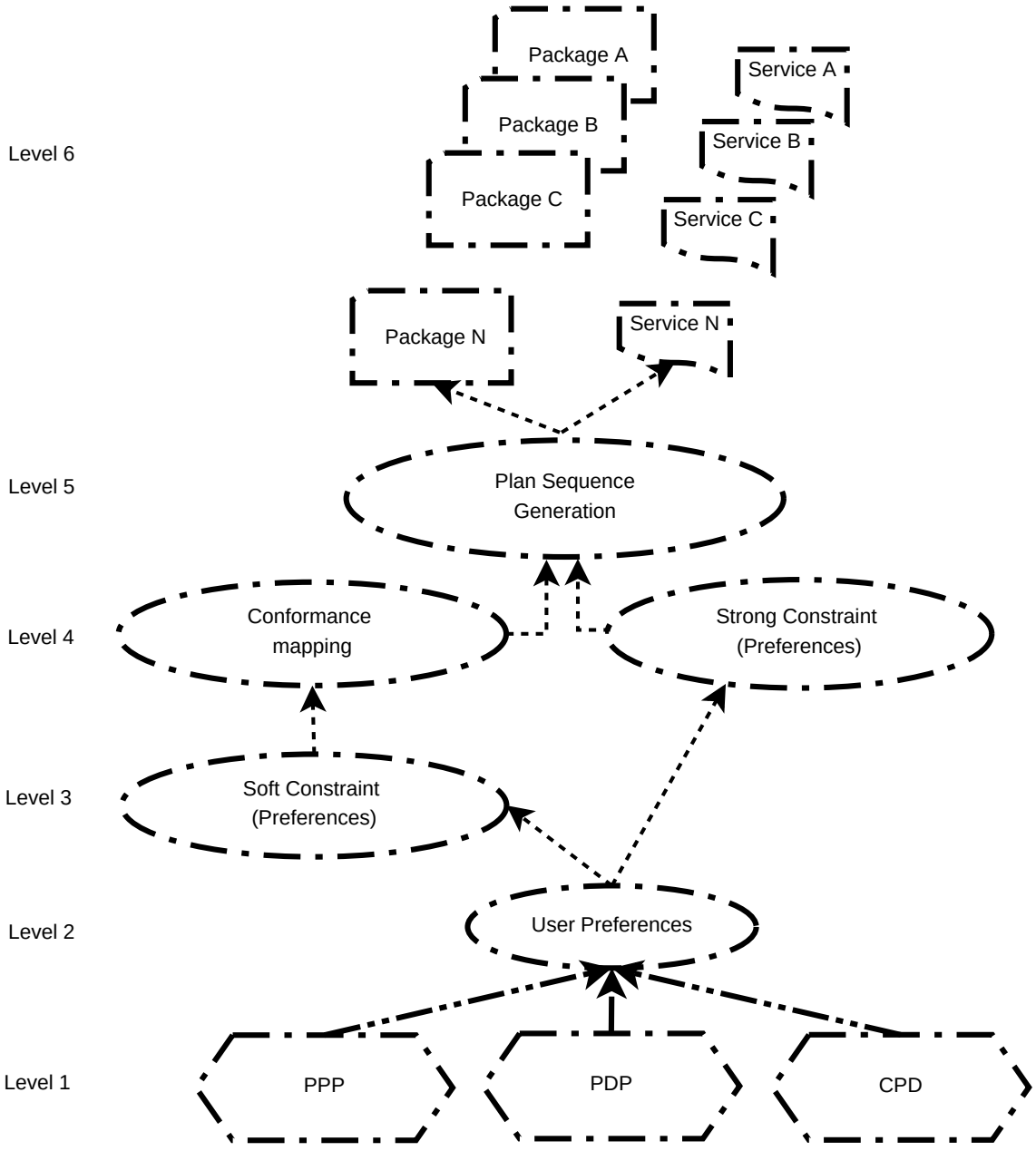

Figure 4.4: The User Preferences based Conformance Process

of packages and services made by considering the soft constraints, conforming to related concepts by the mapping user preference during the composition process (runtime), and finalizing the composition process. As it goes beyond the project domain to test this kind of scenario (DPPP), it was considered best to test SPPP where the system is not consulting with the user during the discovery process.

To research and implement the PPP, the main idea is to use the effectiveness of conformance planning in the dynamic services composition process. It is very important to provide the user with a plan by conforming to the user actions points, and to the desired goals. The concept can be utilized in design of computer games in order to capture the player's future motion, stock exchange trends and planning of routes. After analyzing such domains, the current framework is proposed to include the functionality of (PPP, CDP, PDP), which is a new concepts in services composition research, although the similar concept was utilized in planning applications.

4.3.2 Complete Desired Package (CDP)

The planning of services composition is based on user requirements. For example firstly, users must specify the input (such as the date, cash, and food preferences) and the desired goal (e.g; Tour plan). Then, the system will search for the complete desired package, as well as considering other available options by utilizing of single services, or partial packages to locate the optimal solution. There is only one difference between CDP and PPP. The Complete Desired Package returns an exact, efficient plan, while the PPP also considers the extra options available, by considering user preferences. To explain the above mentioned concept the Figure 4.4 shows the six level process. The following pseudocode will explain the CDP process in detail.

1. Load pool_services *(load all available services)*

2. Start Traverse(Desired_Output): *(Initiating function for desired output)*

3. for i=1 ..n (loop to consider all single services)

4. $Sol_w \leftarrow \forall \mathrm{R}_{op} \in P_{ck1\ldots n}$ (to select services which are containing results according to required output)

5. $Sol_{comp} \leftarrow$ {SORT $(Sol_w) \models \bar{O}p\ (P_{r1\ldots n})$}

6. Continue_Build(Sol_{comp})

7. Execute _final_output ($\mathrm{F_{CPD}}$) $\leftarrow Sol_{comp}$

8. Return ($\mathrm{F_{CPD}}$).

The same procedure as PPP is being utilized in the CDP algorithm however during the discovery process only the exact matched services will be considered. At (Loc,4), all the services will be analyzed which are providing different packages according to the required output. The solution will be finalized by considering the user preferences. At this stage only user preferences based packages are considered if the individual package contains the same output as the user requirements. However in PDP if the package is not providing the output in full then single services will be searched for to complete the package.

4.3.3 Partial Desired Package (PDP)

The Partial Desired Package is a type of package whereby part of the user requirement (output) is available in the form of a package. The package, however, does not provide the fully required output, so to achieve the main goal, the planner should add individual services to fulfil user requirements. The requirement of the composition process is to provide the user with a more economical plan, for example by consid-

ering the quality of services factors if the package is comparatively uneconomical. Services may then be added to fulfil the user requirements.

As discussed at the start of this chapter, the proposed model utilized functional level composition, as used by different DWSC approaches to locate packages. However, for the selection of single services, Process Level Composition (PLC) was adopted. By keeping PLC in mind, it was assumed that a single service interface (Int X), will provide all related information about what the service does, and the kind of outputs that the service will provide ultimately. For simplicity, in the PDP scenario, user's preferences were matched, with the available, related packages analyzed. The following pseudocode explains the scenario:

1. Load pool_services
2. Start Traverse(Desired_Output):
3. for i=1 ..n (loop to consider all single services)
4. $Sol_w \leftarrow \forall \mathrm{R}_{op(i,i+1..i(n))} \in P_{ck1...n}$ (to select services which are containing results accord- ing to required output)
5. $Sol_{comp} \leftarrow$ {SORT $(Sol_w) \models \bar{O}p\ (P_{r1...n})$}
6. Continue_Build(Sol_{comp})
7. Execute _final_output $(\mathrm{F_{CPD}}) \leftarrow Sol_{comp}$
8. while $\forall \mathrm{R}_{op}\ Sol_{compp}\ do$ (all related missing Results services.)
9. if $\forall \mathrm{R}_{op} \nexists Sol_{compp}\ do$
10. $Sol_{compp} \leftarrow$Select_Required R_{op} (Sol_{comp})}
11. endif

12. Goto Loc 4 Until $Sol_{compp}(Complete)$

13. Continue_Build (Sol_{compp})

14. Execute _final_output (F_{PDP}) $\leftarrow Sol_{compp}$

In the above mentioned pseudocode at Loc,4 the required output parameters will be considered. The required output will be matched with services providing composite services based output (packages). The requirement of the PDP is to locate the packages which are providing partial results. To locate the packages which are providing partial output, the required output parameters will be matched one by one and the packages which are not providing full results will considered. The searched packages will be analyzed at Loc,8,9 to search for services which are missing. The search will be continued for all the preferences and required outputs. The search process and creation of data structure view has already been explained in definition 4 and in chapter 3.

Services failure during the composition process is whimsical, due to the nature of services [107, 124]. It is also very hard to adopt exceptional handling techniques during execution of the composition process [78]. Due to the uncertain environment of the web services composition domain, it is inevitable that services which are not responding will be excluded [89]. In the available platforms for services composition, the planner is not responding or returning the correct plan if any of the services fail to respond. However, there are services available which can take over in case of services failure [101]. There is a requirement that similar services which can provide the same functionality can be considered or an exception handling facility can take the plan to a successful stage. For example, its services such as a van delivery service is not responding, due to the van service schedule, or the travel agency wants to obtain a ticket price in pounds sterling, while the conversion function-based service is not responding when attempting to convert from US dollars to pounds.

It is suggested that there is a need for a re-planning module, even if a service is not responding, planners have to increase their search domain and look for a new service which provides the same output [67]. As discussed previously XPlan will be utilized. XPlan also contains a re-planning module, which was one of the strong motives at the time of selection (for the DWSC process testing environment). If for any reason the plan is unsuccessful, it will initiate itself, regenerate a complete plan and forward it for processing. However, in the proposed current research context for re-planning (in case of failure), the search space is reduced by using a planning heuristic for minimal, optimal solutions, for example, searching for the CDP (Complete Desired Plan) plan (as discussed in the last section). The following chapter will explain the testing environment where all the discussed concepts will be utilized to find an optimal solution.

4.3.4 Concluding Remarks

In this chapter, the OSPG model requirements were discussed in detail. The user preferences were considered in terms of strong and soft constraints. The soft constraint analyzed further to divide the preferences in three different types (PPP, CDP and PDP). The procedure of services selection at FLC and PLC level is discussed in detail. The next chapter 5 contains the planning requirements, implementation details and will also explains the obtained results.

Chapter 5

Results and Evaluation

The composition of dynamic web services is not as simple as plan generation typically carried out in AI planning, where world states are not assumed to change unless under the effects of a specified action [71, 115, 138]. This is due to complete information (to instantiate the DWSC process) not being available well in advance, and given the fact that required domains DWSC process are constantly upgrading [143, 156]. The current progress in the development of different technologies and the emergence of a new web services' development standard are making it very difficult to improve intercommunication between services to support the services composition process [90].

5.1 Planning Requirements

As discussed in chapter 1, instead of reinventing the wheel, this research will analyze and adopt (where applicable) existing successful approaches The XPlan planning module and OWLS-MX matching module will be utilized (discussed in chapter 4 and explained in [84, 85, 88]. There is no exact matching framework available, as the

proposed system [1] is a new concept which will provide extra functionality to consider user preferences and will provide optimal results according to user requirements. The obtained results will then be tested and evaluated against the available benchmark examples. As discussed, the available benchmark examples do not provide the same solution (user preferences and conformance based results); however the current work is compared with available results and extra adopted functionalities are measured by using the accepted evaluative metrics.

This chapter will discuss the implementation requirements, the evaluation methods, and environment testing, concluding with a discussion of the final results.

5.1.1 Testing Environment

As discussed, with the help of AI planning-based environments, it is now possible to set up, or develop, an automatic system that has sensing and sagaciousness properties (it can act according to the current situation or requirements) [12, 76, 164, 169, 179]. For example, in the DWSC process example, at the first stage, the expectation is to receive or judge input values (user queries) in different forms (any format) and then to update and to convert inputs, according to the system input requirements. At the second stage, the system will be able to accept the input values and add new, missing, values, by using planning techniques (e.g. conformance planning). The system will collate all basic information from the user, and forward this to the planning environment. The chances of errors (final output) in such environments, are very small, as the concepts pyramid (ontologies) is organized and developed by operators (or automated software after analysis). As explained in chapter 3 and shown in Figure 5.1, in our current proposed approach, we have considered the developed approaches in the past and devised the new composition

[1] The system represents a complete interconnected environment which consists of planning and searching services modules as shown in Figure 5.1

framework which has the following phases.

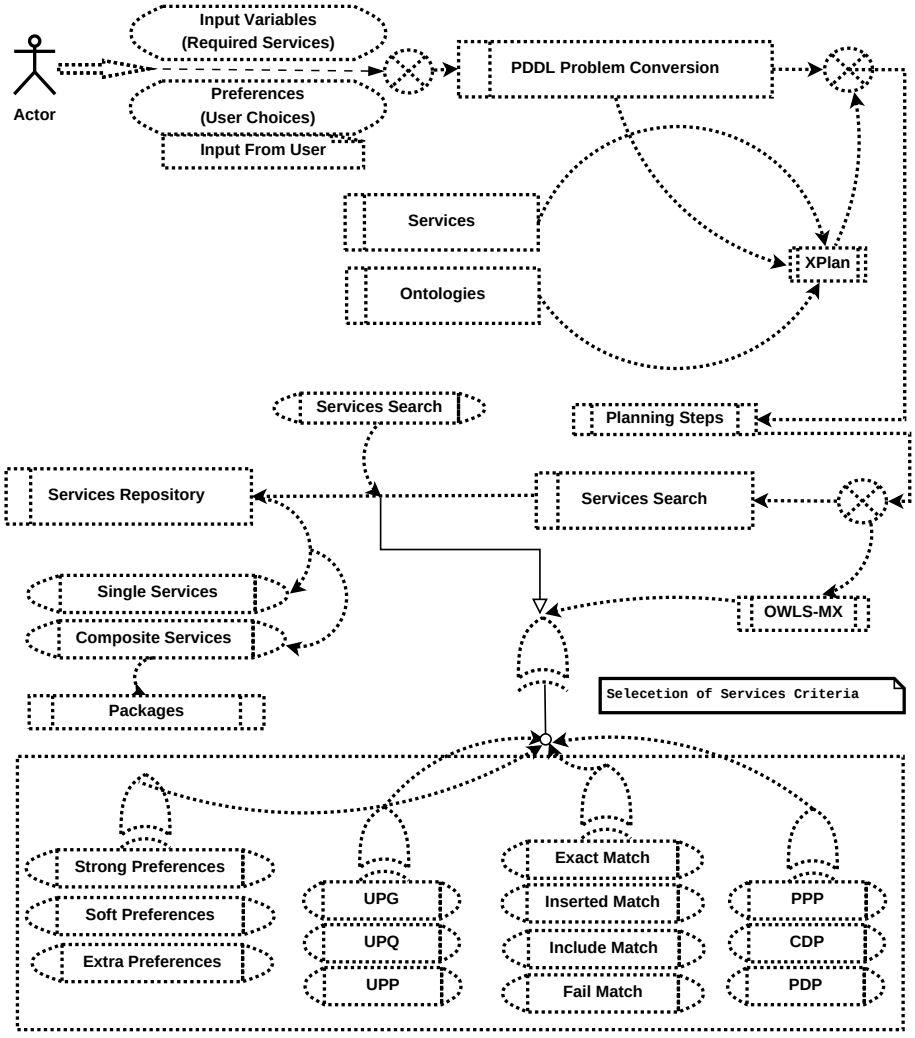

Figure 5.1: Full Procedural Flow Diagram (OSPG).

- Conversion of the actual problem into a planning problem.
- To generate the sequence of actions by using any planner
- To find an appropriate solution (sequence of steps to locate services)
- Analysis of the obtained services (Single and Composite Services)
- To consider the user preferences while planning (Mapping with Services Description)

- To select and provide an optimal solution (Packages and Conformance Planning option)
- To initiate the replanning process if the required services are not responding (Contingency Planning)

The current thesis also consider the discovery process of services from the pool of available services on the web, matching the capabilities of the services with the user's objectives, composing multiple services to fulfil all requirements. Finally, it also discusses and provides a composed framework according to discussed concepts.

The requirement of the DWSC process, by using the planning-based platform is mentioned in the following 11 steps. After obtaining input from the user, the constraint values are analyzed (strong and soft constraints). Figure 5.1 shows separate components for the required values (Inputs and Preferences), as in the current work the preferences are used for conforming the user's choices-oriented structure.

5.1.2 Implementation Approach (OSPG)

By considering and further analyzing the aforementioned concepts and our proposed approach (concepts followed from [1, 18, 21, 153, 158]), the main process of the OSPG can be divided in the following 11 steps, Figure 5.1 shows the setup of the OSPG environment.

1. Query (Input from user)
2. User Constraint Values (Input from user)
3. Problem Description (Input from user)
4. Problem mapping into PDDL (System [2])

[2]The system represents a complete interconnected environment which consists of planning and searching services modules as shown in Figure 5.1

5. Missing Parameters Input (System)

6. Planning Step Generation (Planner [3])

7. Analysis of Planning Steps (System)

8. Planning Solution Formulation (Conformance Planning Module)

9. Planning Solution Formulation (Contingent Planning Module [4])

10. Finalizing Planning Steps (System)

11. Forwarding Plan to System for Planning Step Execution and services search (System and OWLS-MX [5])

5.1.3 System Flow and Execution Context (OSPG)

The System [6] will initiate itself after receiving a request for required services from the user. The main inputs are considered in two variants i.e; input variables and user preferences. Here input variables represents the number of services required and the preferences which should be considered while finalizing a plan. The system will then finalize the sequence of steps and the services search process will begin. The details of the user choices-based constraint is available in chapter 4. The next step is to convert the services composition problem into the planning problem (explained in chapter 3). As discussed, the PDDL3.0 version is required since we take the user preferences into account. In OWLS-XPlan, there is an embedded module for the conversion of the problem, which accepts the PDDL format [86]. To include the user preferences, the problem domain, along with the input interface was modified

[3] OWLS-XPlan

[4] In case of failure system will provide basic solution

[5] Integrated services search and final results execution module

[6] System represents complete interconnected environment which consists of planning and searching services modules as shown in Figure 5.1

according to proposed framework requirements, where user preferences in terms of strong and soft constraint were considered. And on later stages, the system have to provide the sequence of actions that will be executed, in order to find the required services. As discussed, the required domain will be mapped into a planning domain, and there is a look up subroutine for missing parameters.

According to input requirements of XPlan (as discussed in chapter 1), XPlan requires both an initial state ontology and a goal state ontology as input at the start. XPlan interface provides different features to update the problem description level (as updated version of XPlan is available). But due to the complexity (extra functionality) of our current work, only manual step generation function (basic version of XPlan) has been utilized. However in future, the complete installable environment along with embedded module of preferences and composite services consideration will be designed for logistic and travel domain. For DWSC domains (such as the travel and logistic domain), the complete integrated system will be able to support end-to-end services selection and composition process. As discussed, the execution steps will be generated by the planner, while later the system will consider single services, composite services and services containing packages by taking the input values into consideration. During the planning for services, the user preferences will be matched by the required packages. The desired preferences plus packages will be presented to the user along with the complete and partial packages (details in chapter 4). This step of the sequence will be analyzed, and in case of the system failure the contingency module will be invoked. The final solution for the execution will be put forward, which will complete the composition process.

Due to the complexity of the task, it is not possible to test the framework in a real-time environment (online services registries and integration issues). But for a realistic and timely approach, the following assumptions are considered before

generation of the sequence of steps for the composition process.

- All services in the repository will use a standard, defined interface. The standard design interface should include the information such as service profile, service model and service grounding (defined interface details are provided in subsection 2.1.4). This is because the planner is unable to select the services, during the search process, that are not developed according to a predefined or acceptable format. In the current work, the system attempts to find related services and packages by considering the user input, and conforming to user preferences.

- When the services composition problem inputs are finalized and the required problem is put forward for the services composition task (to XPlan), there will be no option for changes in the input parameters (strong, soft or extra preferences), or services composition sequence options (e.g; when the input values will be inserted in the interface and the XPlan process module initiated), although the user can stop the composition process any time. To generate the output, the single services and composite services present in the repository will be considered, but there will be no changes possible after the system has finalized the services indexing. The system will be unable to respond to any effects or runtime changes, and as per the rule, the user submitted input/output options will not be tackled. However, the user can restart the planning process at any time.

- Due to the complexity of the system, the contingency planning module will be invoked, in the case of failure, which will consider the single services, and provide a composition sequence. The re-planning exception handling facility has been employed in the related work, which is both time consuming and can cause the system to enter a deadlock situation if the same failure occurs again

during the process.

5.2 Evaluation Methods

In the last few years, research dedicated to services mediation, discovery and composition and the autonomic characteristics of the dynamic web services composition process have grown; however there are only a few publications where the standard benchmark examples have been discussed and evaluated [91, 128]. Currently, there are some active research groups attempting to establish mature evaluation methods for the DWSC process, which are reviewed and adopted in the current research [18, 57, 91, 128, 158]. There are artificially synthesized data sets available where the existing approaches have been evaluated, but it is hard to find any common attempts where the same use cases (on same data sets) were tested against the same parameter values [91]. The observation of the real time composition process is also not possible. The available data sets are not large enough to establish optimistic results, and a major reason for this is that it is a very vast and complex topic, with many sub-topics that require further, in-depth research.

In the current work, the following research metrics questions were devised to evaluate the proposed context.

1. What benefits will the proposed approach bring in terms of considering preferences and locating services to find an optimal solution?

2. Does the proposed approach support the efficiency of search by considering the user desired factors?

3. Is the proposed approach more scalable as the new input parameters are introduced in the proposed framework (Strong, Soft and Extra Preferences based Constraint values)?

4. Can this proposed framework support the full automation?

As discussed in section 5.1, with reference to our proposed research context there are no similar evaluative measures available where the same inputs and expected output parameters are used. So the above mentioned questions are formulated with reference to the software evaluation approach GQM (Goal-Question-Metric) which was developed by NASA and which has been utilized in other DWSC research approaches [15, 91]. The following research metric are adopted to answer the above mentioned questions.

- **Scope**: As the proposed framework is designed to provide users with extra functionality, leading to an increase in problem representation difficulty level but to locate an optimal solution, the scope is considered as one of the evaluative metric. The services will be discovered by considering the user preferences, and by analyzing the best results. The user will be presented with the best-matched results, according to the basic preferences, and is the main goal of the current work. As compared to other related efforts, the current research contains preferences and preferences based selection of packages which increases the complexity level, so scope needs to be considered as an evaluative metric.

- **Correctness**: The discovered services by the planner are correct and relevant, to the required composition task. In the current work, since there is a functionality to include preferences with different variations, and as different types of services (single services, composite services and services with packages) are considered, it is very important to use the correct evaluative metrics to verify the final output of the system. In the related work, the provided solution (number of generated steps) is considered, while in our proposed work model, it is tested for correctness, due to the following contributing factors into the complexity level of the testing interface.

- Input Parameters (Required Services)

- Selection of services (Procedure)

- Conformance of results (for best results)

- User Strong, Soft and extra preferences based goal requirements

- Contingency planning in case of failure

At this stage the integrated environment is not fully automatic and manual coding and hard-coding is involved,so correctness is considered as an evaluative metrics.

- **Scalability**: The proposed framework is able to handle a growing amount of work in a capable manner as it is changed to accommodate preferences and different type of services . The overall composition task completion time of the system is observed by considering single services and composite services (packages). The capability of a system to deal with different type of requests in a specific period of time is represented under scalability [59]. In the current work, different use cases with functional and non-functional attributes are compared with benchmark examples. Generally in previous work, a single domain with different use case scenario has been used for evaluation purpose, however in current work two different types of use case scenarios are adopted to observe the composition process. As the current work include extra functionality of three different concepts (preferences, types of available services and conformance), it would be practicable to test both domains for evaluation of the created plans by the planner (OWLS-XPlan).

- **Versatility**: The important concept and aim of the services composition process is to enhance the framework support for different domains, in order to support the characteristics of applicability and flexibility. Due to the fact, the

thesis explains the characteristics and features of the proposed framework by comparing with similar existing approaches. At present, the system is tested against two domains but where possible we tried to include the examples of different domains. Which shows the capability of individual components to support different domains with maybe some extra technical changes.

5.3 Results and Discussion

Figure 5.2 shows the system implementation[7] and flow. As discussed, the main contribution of the current work is to consider user preferences and provide the best results. As mentioned the XPlan planning module and OWLS-MX matching module were utilized (developed for the CASCOM and SCALLOPS Projects [84, 86, 88, 104]). According to our proposed environment, we developed the algorithms and procedures to consider different type of services, preferences and selection of optimal results (details are available in chapter 3 and chapter 4). As discussed, the overall system (OSPG) consists of different sub modules while the developed and implemented modules required according to our environment are shown in Figure 5.2 (Red Colored Filled) and details of our contributions are as follows.

- Input requirements; XPlan accepts initial state and final goal state ontology while providing planning steps, we used the same planning steps generated by XPlan.

- Services Preferences; The algorithm was developed and implemented to consider Strong, Soft and Extra preferences.

- Services Match; At this stage, our selection of services criterion were included to consider single services, composite services and packages.

[7] Consists of integrated and developed (our contribution is shown in red color) modules

- Packages Selection; Further to the above, we also implemented a procedure to consider three types of packages (PPP, CDP, PDP).

- Evaluation of the discussed concepts in the proposed Model; to test and evaluate different concepts such as UPG, UPQ and UPP, we implemented different changes on services selection level, which were carried out manually by hard-coding information about participating services and selection criteria.

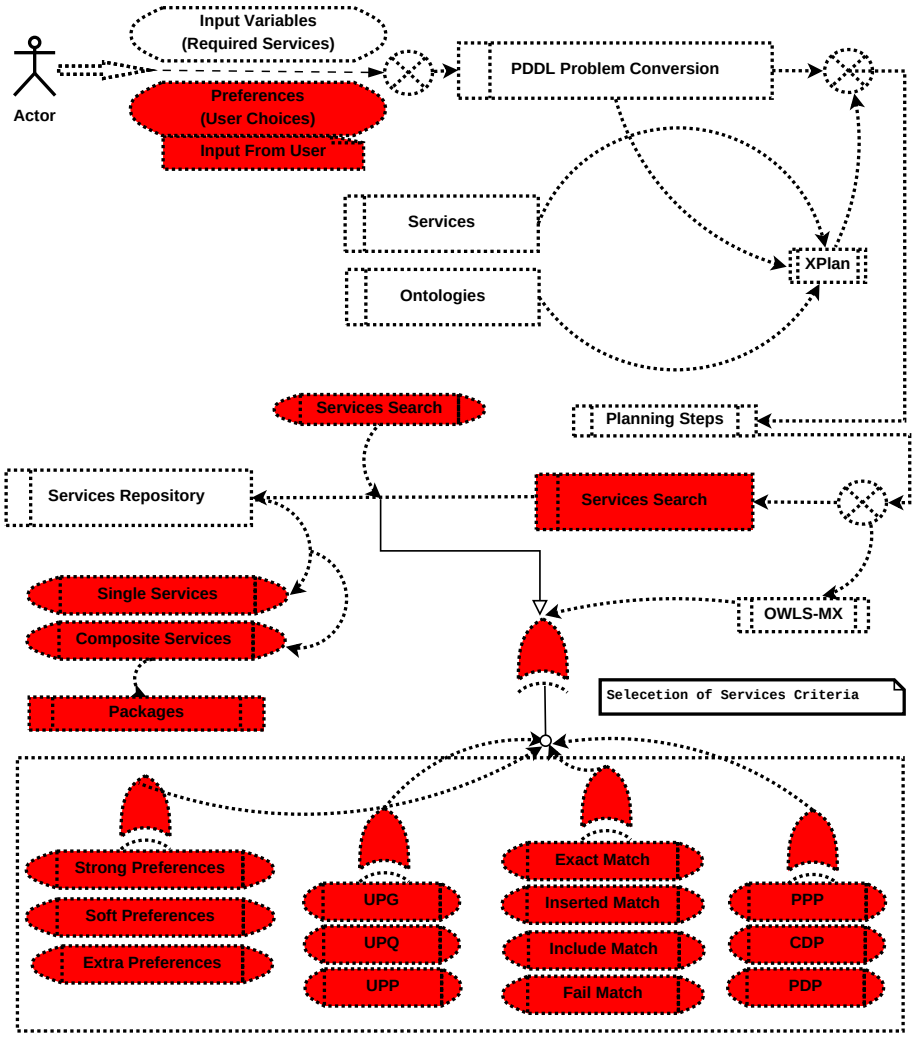

Figure 5.2: Full Procedural Flow Diagram (OSPG).

As discussed in the last section, it is not a fully automatic process (OSPG), as it is only designed to test the preference-based procedure calls from users, and to observe optimal results by considering different type of services. It can, though, be reckoned

and adapted for different domains but manual conversion and changes in services description required. Where possible, the current work includes examples (for an explanation of concepts) from different semi-automatic developed environments such as travel, logistics, employee screening and system hardware installation. For testing purposes, we used the same web services that were used in XPlan E-health domain, and a modified tour plan services set of the Sem-Web-Central project, accessible at the following link.
(http://projects.semwebcentral.org/frs/download.php/277/ SWS-TC-1.1.zip).

The main interface (Graphical user interface (GUI)) to interact with the system (as shown in Figure 5.3) is designed for the purpose of understanding the system and to give insight into the data flow information of the system with in the proposed framework. In proposed model different softwares (XPlan, OWLS-MX and developed modules (details in section 4.3 and subsection 5.4.1)) are integrated to achieve the composition goal but to get full advantage of system fully automatic environment (installable) is required to avoid the manual conversions and assigning values to different components (see the explanation in chapter 6). To examine the Quality of Services factor (QoF), time, cost and reputation factors were manually assigned to individual services as it is not supported by XPlan and developed modules. For example the cost factor has been utilized to compare the final outputs generated by composing single and composite services but is out of the scope of the current research task. For further details, non-functional attributes are explained in [79, 150].

To observe the proposed solution and validity of the planning steps, 15 different use cases were tested [8]. As discussed, the use cases are designed to observe the

[8]Only 15 use cases are explained here according to the proposed framework and to support the discussed concepts however for each use case different variation and instances were considered by changing different input parameters values and combination

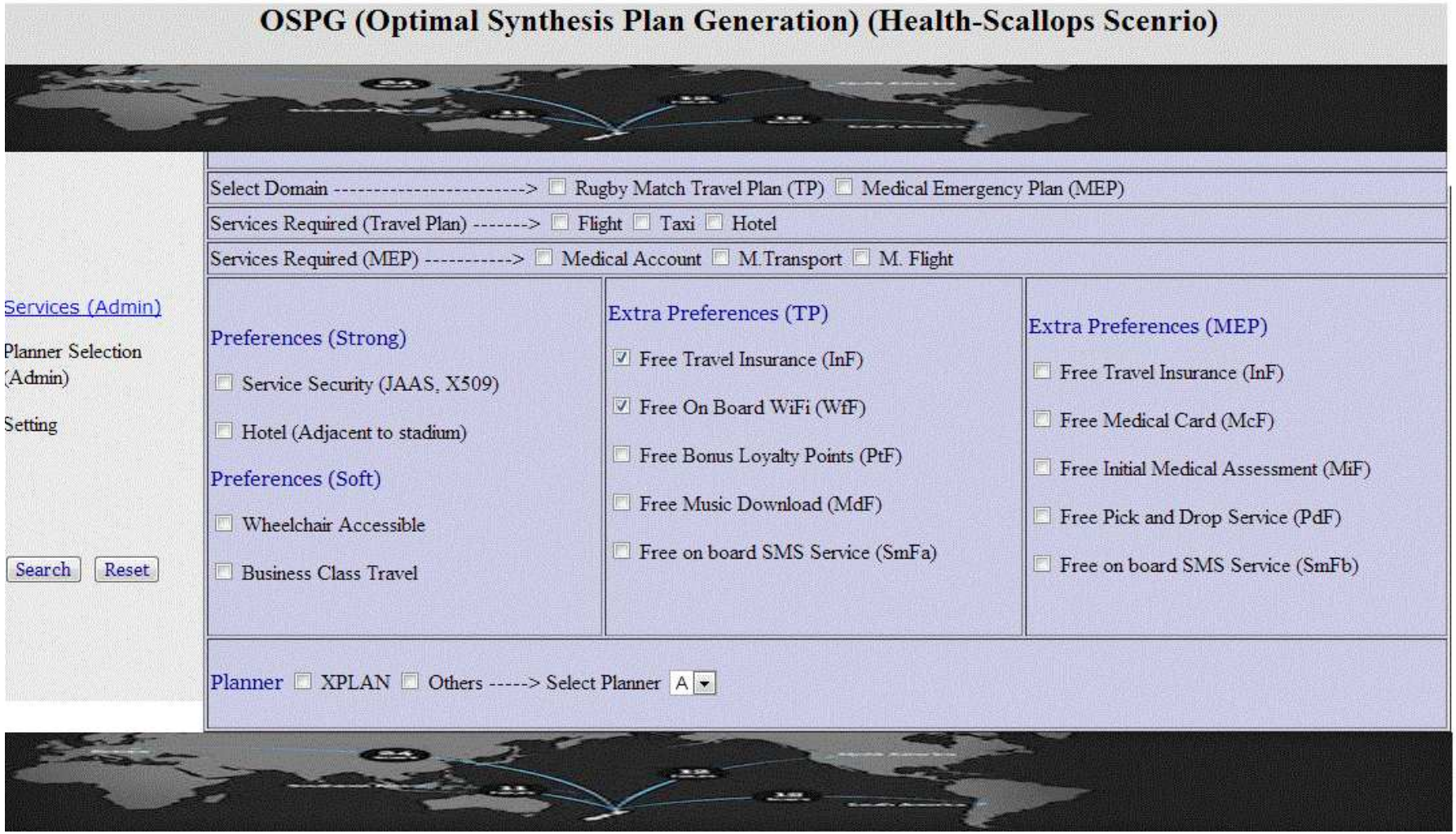

Figure 5.3: OSPG User Interface (OSPG User Interface)

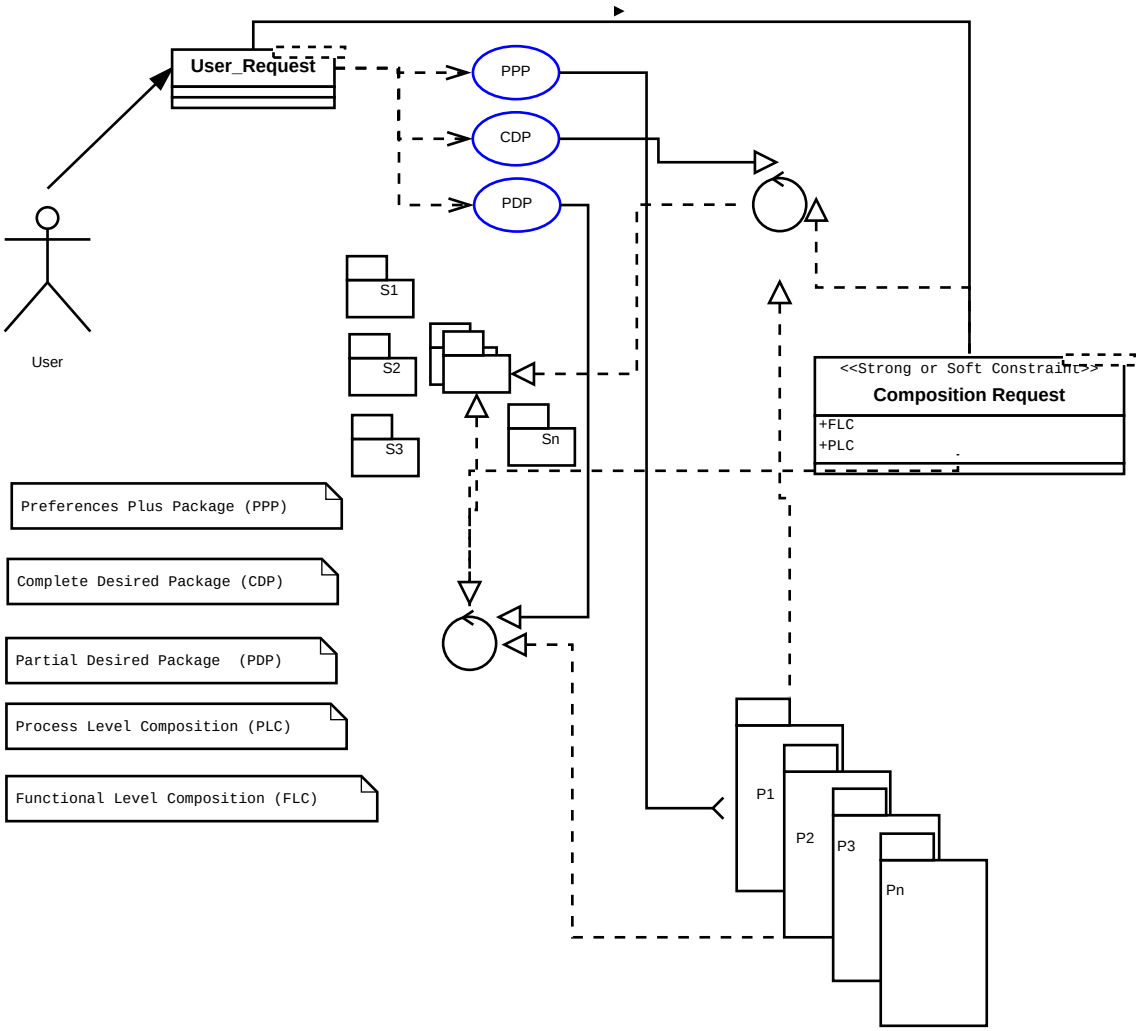

Figure 5.4: The System Flow Diagram

services composition process by considering PPP, CDP and PDP.

We considered the services by mapping user preferences. The free offered services have been added in packages for experimental purpose to test the PPP scenario. For example, in the travel plan, planners has to show the packages which are offering free travel insurance along with a series of questions to map user interests are a series of questions to ask (Details in subsection 5.4.1). Figure 5.4 shows the system sequence and selection of services by considering user preferences. The user's request is mapped to locate the services while the concepts of PPP, CDP, and PDP are utilized. The figure shows the single services and packages while the concept of

Strong, Soft constraint and Extra preferences parameters values will be utilized to compose these services, and to find an optimal solution. The dotted lines show the dependency of the FLC and PLC concepts to search for the required services, according to user requirements.

For the main interface design, and to update the existing structure, Eclipse version 3.4.0 with the GEF (Graphical Editing Framework) plug-in, Apache Tomcat server 6.0, and a UDDI server-based integrated platform, have been utilized [125]. All the experiments were performed on a window based platform (Pentium 2 GHz, 2 GB RAM, 32bit OS) and Windows Vista operating system. In all the experiments 8 different metrics were considered which are time (by considering and without considering Packages), number of packages, input variables, strong or soft constraint and number of services. As discussed, the main aim is to include the challenging factors in the services composition area. Due to the complexity of the system, the initial proposed model is tested against the available, tailor-made domains, by hard-coding the data sets interfaces and according to the desired platform requirements. To observe the time difference between the composition process by considering packages and without considering packages, more services have been added in the existing data set to support our proposed framework (obtained from Sem-Web-Central Link; http://semwebcentral.org/).

5.4 Travel Plan Domain

To support our proposed framework we devised a tailor-made domain. We used the same data set which was already used by different developers to observe the output of the proposed framework. Table 5.1 shows the results of the travel plan queries. The travel plan domain, with seven different use case scenarios and settings has been tested to verify the concepts of services composition. The following examples

will explain the different tested setup and results. 15 different queries were selected to explain the proposed framework, however the system was tested by changing different parameter values. The 15 examples/queries are explained as follows.

5.4.1 Example 1

In the first set of experiments, the service discovery and plan generation sequence has been observed. The main aim of the test is to check the selection of services by conforming user choices and providing user final results, according to user preferences.

The Preferences Plus Package (PPP) approach has been observed in the current query. As discussed in subsection 4.3.1, the most efficient plan is PPP which is considering strong and soft constraint while also containing user preferences. The time to generate a plan in PPP based scenario is comparatively low, if we compare it with CDP (Table 5.19) results. The preferences based strong constraint functionality is considered, which is *User Preference for goals* in current use case. The hotel booking service adjacent to the stadium is considered as a strong constraint for the first example which includes two tailor-made packages along with other single services. Input variables value shows the number of inputs and the services required to initiate the composition process while in the current query (example 1) flight, hotel and taxi based services package was requested by user. To provide these inputs, the user will check the three options on the provided web page along with the preferences (as shown Figure 5.3). For testing purpose the basic web page has been created however more options based form can be created and deployed to get the full advantage, according to domain requirements.

In current example, the user can also select free offered services, which are included to verify the concept of conformance planning in packages and to verify the

Table 5.1: OSPG Results (Travel Plan Results)

Options Queries	Time By Cond. Packages	User Preferences	Time without Packages	No of Packages	Input Variables	Strong Constraint Preference	Soft Constraint Preference	Number of total Services	Planning steps Successful (Y/N)
Query 1	41.52	3	34.57	2	3	Y	Y	241	Y
Query 2	37.31	3	25.76	2	3	Y	N	241	Y
Query 3	34.02	3	20.87	2	3	N	N	241	Y
Query 4	42.23	3	35.34	3	2	Y	Y	241	Y
Query 5	40.26	3	35.02	3	3	Y	N	241	Y
Query 6	50.67	5	43.19	4	3	Y	Y	241	N
Query 7	49.23	5	41.70	4	3	Y	N	186	Y

soft constraints based preferences. As discussed, that there are some services in the packages which may offered at no extra cost. The planner have to consider those packages, along with extra output which are considered, according to user preference (Soft Preferences). For example in the current query, free travel insurance, music download, and extra bonus loyalty points have been offered. During services discovery process, it is very important but difficult to consider non-functional properties. The non-functional properties are reliability, availability, cost and performance which might be taken into consideration while finalizing a final plan. The quality of overall plan directly depends upon the functional and non-functional properties of participating single services in composition process. In current proposed model, as our priorities are to design a framework model to support composition process but where possible, the functional and non-functional properties were also considered. Obviously it is not possible to consider,the reliability and availability factors are not possible to consider at this stage. As the overall test environment is offline, however we tried to analyze the cost and performance by manually assigning the executing cost.

1. The user will have to provide the details of the tour plans on the created web page (As shown in Figure 5.3). In current example (details in Table 5.2), user selected three services which are flight, hotel and taxi. To include the user preferences and services selection procedure, the user will also check the three different types of preferences along with strong, soft and preference for free offered services. The system (as an environment) will consider all the four different types of inputs to generate a travel plan for the user.

2. In the next step, the inputs will be considered in two separate instances which are Choice of Services (S) and Preferences (P). For example in the current query, *Choice of Services* input will be (Flight(f), Hotel(h), Taxi(t)) while

EXAMPLE 1.

Input Variables (Required Number of services) =3

(Flight, Hotel, Taxi)

User Preferences

Strong Constraint = Yes(Hotel service for composition will be adjacent to stadium)

Soft Constraint = Yes(User prefer Business Class travel)

No of Extra Preferences (Optional Free offered Services) =3

–Free Travel Insurance(InF)

–Unlimited Music Download(MdF)

–Bonus loyalty Card points(PtF))

Number of Packages Available to fulfil Request =2

Number of Services =241

Time

Time by Considering Packages =41.52

Time without Packages =34.57

Table 5.2: Example 1

preferences will be in the form of Strong, Soft and Extra preferences. As discussed the actual Preferences based inputs values are further divided into three different variations such as Strong (Hotel next to Stadium(HnX)), Soft (Business class travel) and free offered services to test the PPP package composition sequence. To include the free offered services in current example, the user has selected options Free Travel Insurance(InF), Unlimited Music Download(MdF), Bonus loyalty Card points(PtF).

3. The Choice of services (S) based input is assigned to XPlan to generate a planning steps. XPlan takes the OWL-S service file as an input along with the use case ontology. XPlan converts the provided services and ontologies into OWL-S and OWL services. The problem definition includes the initial state, final state and the user preferences.

4. XPlan will generate a sequence of steps (the plan generation and services search criteria is explained in chapter 3). In the current example, the steps will be as follows

   (a) Arrange taxi from home to airport

   (b) Flight Required

   (c) Taxi from airport to hotel

   (d) Hotel booking

   (e) Taxi from hotel to airport

   (f) Flight Required (Return)

   (g) Arrange taxi from airport to home

   In the travel plan generation process we assumed that match ticket reservation is already done as according to the framework requirements we have to test the

PPP (Preference Plus Package) which in the current query is that the hotel will be adjacent to the Stadium.

5. The XPlan output steps will take the composition process to the services selection stage. The related services will be identified. Firstly the Preferences (P) are considered and divided into soft, strong and extra preferences. OWLS-MX is utilized to search for the related services. The functionality of the OWLS-MX is discussed in chapter 2 while further details are explained in [53, 87]. The advantage of OWLS-MX is that it support signature (I/O) matching which can provide a faster and efficient approach to the services selection procedure.

   To finalize the selected packages for the composition process, we used the ontological similarities structure. This is where the related extra package results will be mapped with input requirements by conforming to user preferences. The process of semantic analysis of the user preferences (taken from input requirements) will be initiated at this stage. For example, by considering the basic input and output requirements of the user, the related node links will be analyzed to check the similarity of concepts (semantic). The searched concepts will be added in the graph by matching the degree of similarity. The graph will be constructed by taking the first concept (soft constraint) from the user preferences before searching for other related concepts. It does this by exploring the services semantic description. The length of the graph (nodes links) can be adjusted by the user. The graph is constructed by appending nodes in a sequence of semantic match; for example the concept at layer N has a higher degree of similarity then any node at layer N+1. The graph will be populated by following the above procedure, with further nodes added in the graph. As discussed, the semantic similarity-based matching function

works by considering the soft constraint. The first step will be to perform a semantic analysis of the soft constraint, with the related concepts selected and a graph of similar keywords-based constraints constructed. The number of keywords (concepts) selected will be taken and compared with the extra offer results-based services. In the final round, the selected packages, which have matching concepts, will be selected for the final output for the user. Here, the matching degree can be divided by evaluating the different types of relationships between concepts, however in the current arrangement the following matching criteria are being utilized. The OWLS-MX functionality is utilized to locate the services. OWLS-MX computes the degree of semantic matching for a given services by applying four different following filters.

- Exact Match $(C_a, C_b)\, is\, possible\, iff\, C_a = C_b\, or\, C_a \equiv C_b$
- Inserted Match $(C_a, C_b)\, is\, possible\, iff\, C_a \mid C_b$
- Includes Match $(C_a, C_b) is\, possible\, iff\, C_a \simeq C_b$
- Fail Match $(C_a, C_b)\, is\, possible\, iff\, C_a \neq C_b$

6. After receiving the matching required services, the system will analyze and consider the single output provided by participating services. In the current example the services Taxi, Flight and Hotel will be considered.

   For the simplicity of the current work, there are only a few matching services in the services pool however multiple services along with the same or with different functionality (Extra offers) can be included for efficient results. For example, there may be a cheaper offer available, to select a taxi from (Home to Airport) or from (Airport to Home). But in the current work it was assumed that taxi services covers both required options to compose the services. For services representation purposes, Figure 5.5 shows participating services for

composition. All the participating services will be included and the OWLS-MX module will return matching services according to related input and output requirements.

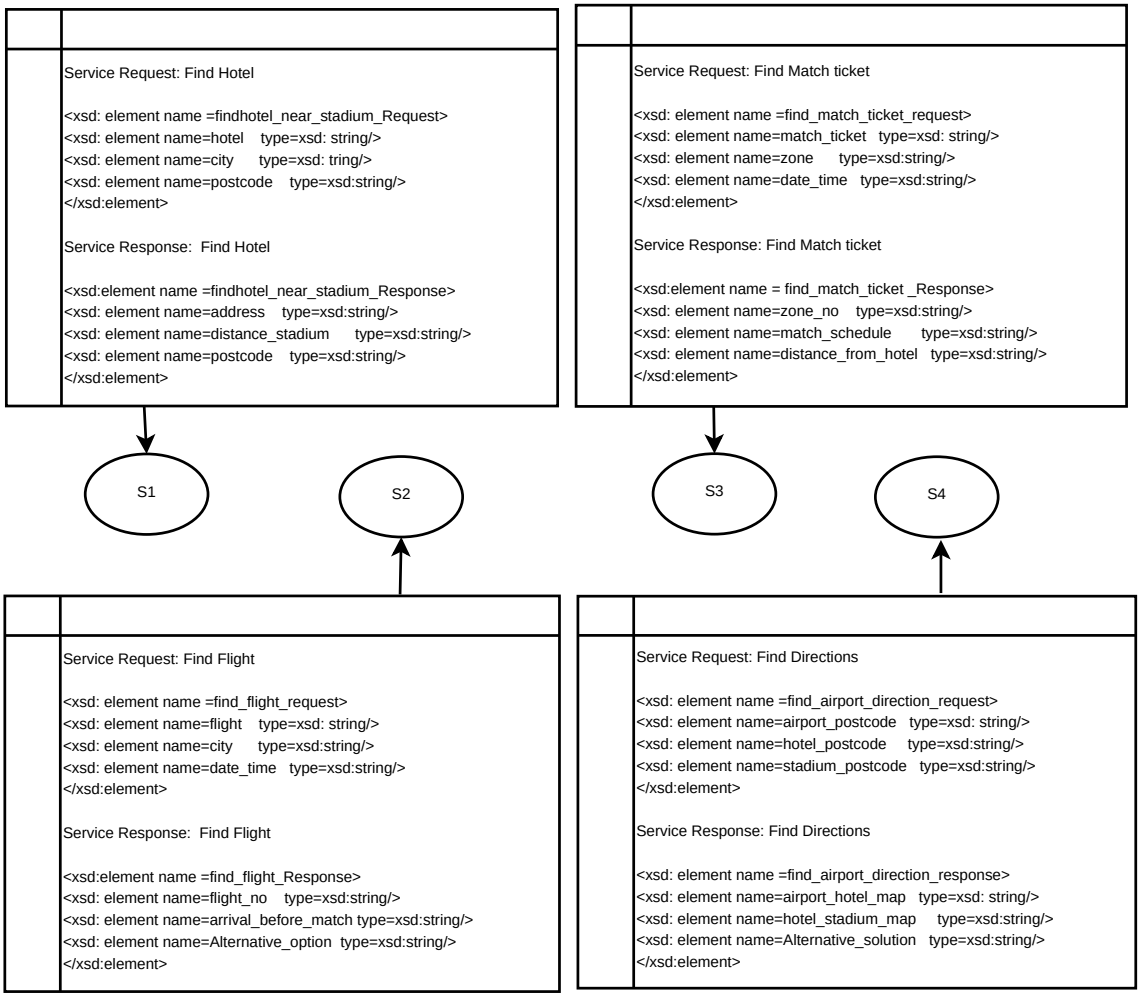

Figure 5.5: The match booking travel plan (Request and Response)

7. At the next stage, the packages will be considered, which provide the same or extra functionality. In the current example, 2 packages were present. The first package with exact (according to user requirement) and extra output, while the second with exact required output. Here the extra output refers to the optional free offered services. During the same step, the system will try to locate the partial output packages. At the time of testing, there are three partial packages which contain just two services. The system will try to make

these packages by locating single services.

8. The final results will be obtained after comparing both outputs i.e; the composed single service and packages. For the sake of the simplicity of the test bit, the cost was manually assigned to the services and tailor-made packages.

9. In the current example, the time by considering packages is increased by 7 seconds. Which is due to the fact that the planner will firstly consider all the single services, then packages and partial packages will be considered at the end. The more efficient approach can be adopted here but to evaluate the proposed framework concept and to support the proposed framework requirements the full algorithm is utilized. However it is possible to analyze the packages first, which can provide final result in less time and will leads to more productive solution. There is sometime requirement to consider the single services first. Like in Conference plan example as discussed in subsection 3.4.4, the user need a plan while traveling where complete package will not be useful. As the navigation device needs in plan to be update during the journey. In Employee Vetting Procedure related domain, it maybe possible that the user can select any options or services from vetting procedure where the package is not the economical option. The discussed approach will support intermediate planning in domain independent environment.

The selection procedure and details of conforming results are available in chapter 4. The final steps contain the results which are obtained from single services while two packages were presented to the user for the travel choice package which were not providing exact results according to user preferences but were reflecting user choices. The presented 2 packages contain free insurance and internet access for an extra charge of £1.50. Both packages are selected by considering user choices which were obtained at start of the process. The selected hotel service is next to

the stadium while the business class seat is reserved on the selected flight which show that service based choices were attached with services to test strong and soft constraints.

5.4.2 Example 2

In the second use case scenario (example), the discovery and composition procedure was observed without soft preference constraints within the same domain. The search functionality for CDP (Complete Desired Package) was presented by searching for a complete solution by considering strong constraints (details in Table 5.19).

In this example (details in Table 5.3), the hotel booking service adjacent to the stadium was considered as a strong constraint based preference. The free travel insurance, music download and extra bonus loyalty points were selected by user while the output package contains the extra services free insurance and internet access, however with some extra charge. The results shows that the overall time taken is less than example 1. The query outcome shows that that the system took less time (Approx 11.5 Sec) when the planner does not consider packages.

The strong and soft constraint based preference is included in the services design interface while extra preferences were included to locate the free offered services in the proposed framework. The same preferences can be included as part of the services description, but the main aim was to introduce the concept of different tiers of preferences e.g; user desires and interests which must be included (strong constraint) or which can be included (soft constraint) during the search process. For non-deterministic domains the preferences can be considered as a non-functional attribute, which are required along with other outputs to locate an optimal solution. To evaluate the proposed framework model and related concepts, the preference are considered in different forms (e:g; Strong, Soft and Extra), however different

constant values can be considered (functional and non-functional), according to domain requirements.

To validate the QoS factor, the time coefficient value is observed by checking the time taken to complete the process however this is not a desired evaluation factor in the proposed framework model. At the first instance, as the system consists of different integrated components and some manual conversion (e.g; to update assigned cost factors), it is not possible to improve the search quality after plan generation. The overall time taken can be reduced by adopting high spec systems [9], updating the data structures and improving the search functionality of algorithms. As discussed, the main aim is not to consider and evaluate the QoS factors but to observe the discovery and composition process to achieve an optimal solution and to evaluate the proposed framework. However other QoS factors can be included for optimal results.

[9](Pentium 2 GHz, 2 GB RAM, 32bit OS) and Windows Vista operating system were utilized at the time of experiments while we used a normal system to achieve the functionality of the server. However good coding practice, a high spec system and compatible server can help to minimize the time for the deployment of commercial DWSC environment

EXAMPLE 2.

Input Variables(Required number of services) =3

(Flight, Hotel, Taxi)

User Preferences

Strong Constraint = Yes(Hotel service for composition will be adjacent to stadium)

Soft Constraint = No

No of Extra Preferences (Optional Free offered Services) =3

–Free Travel Insurance(InF)

–Unlimited Music Download(MdF)

–Bonus loyalty Card points(PtF))

Number of Packages Available to fulfil Request =2

Number of Services =241

Time

Time by Considering Packages =37.31

Time without Packages =25.76

Table 5.3: Example 2

5.4.3 Example 3

In use case 3 (details in Table 5.4), the scenario has been tested by taking away any constraint values. i.e; strong constraints and soft constraints. The reason for this setup is to observe the variation in the results and specifically the change in the time taken to generate the composition sequence and selection of the services.

EXAMPLE 3.

Input Variables (Required number of services) =3

(Flight, Hotel, Taxi)

User Preferences

Strong Constraint = No

Soft Constraint = No

No of Extra Preferences (Optional Free offered Services) =3

–Free Travel Insurance (InF)

–Unlimited Music Download (MdF)

–Bonus loyalty Card points (PtF))

Number of Packages Available to fulfil Request =2 Relevant

Number of Services =241

Time

Time by Considering Packages =34.02

Time without Packages =20.87

Table 5.4: Example 3

The PDP (Partial Desired Package) approach has been observed and tested in this test bit (details in subsection 4.3.3). PDP is the Partial Desired Package, where the searched package is incomplete (a composite service with few results) and is not providing the result according to the required output. For our proposed framework, we assumed that PDP represents incomplete packages (composite services) with fewer outputs. In such cases, there is a requirement to complete the package by including single missing service, if the partial solution (with fewer number of services) is to fulfil the optimal solution according to user requirements. In the current scenario, the missing services were selected to complete the packages. Here the cost (manually assigned) was considered as an optimality factor, however other QoS factors can be considered, according to domain requirements.

5.4.4 Example 4

In use case 4 (details in Table 5.5), the experimental setup has been arranged to check the behavior of the system by changing the number of packages. The number of total services remains same but for the search process the user preferences have been changed to check the effect on the final output. The number of required services are reduced to two; as the taxi service is not required. The time difference is not significantly high, as there were many services available (participating services for the DWSC process) in the service pool which contains partial results. As previously discussed regarding the time coefficient factor, All the queries were tested by taking out packages, single services or number of inputs. The results of such queries (or queries) are not mentioned here, if the time difference is +/-2 sec, and the change haven't any significant impact on the overall results. But for the output analysis purpose, the system was tested by changing different combination of parameters.

As discussed in the last example, the system will try to complete the packages

(Partial Results / Composite Services) by considering and adding single services. In the current example, there are many participating composite services with partial results which further leads to an increasing time factor, as only 2 services are required in the current plan.

EXAMPLE 4.

Input Variables (Required number of services) =2

(Flight, Hotel)

User Preferences

Strong Constraint = Yes (Hotel service for composition will be adjacent to stadium)

Soft Constraint = Yes (User prefer Business Class travel)

No of Extra Preferences (Optional Free offered Services) =3

–Free Travel Insurance (InF)

–Unlimited Music Download (MdF)

–Bonus loyalty Card points (PtF))

Number of Packages Available to fulfil Request =3

Number of Services =241

Time

Time by Considering Packages =42.23

Time without Packages =35.34

Table 5.5: Example 4

5.4.5 Example 5

In use case 5, the system is tested to observe the output while considering an extra package and excluding the soft constraint (details in Table 5).

The aim of the query was to observe the composition process where the package contains some services which may or may not be required by user. The extra package contains the required services (Flight, Hotel and Taxi) while the unlimited music download offer based package has been selected by conforming user preferences.

In the same example, the two matching criteria exact match and inserted match have been tested. The matching engine is discovering the services by matching the input values. To verify the conformance planning requirements, the free offered services have been matched by comparing the required parameters and services offered values. The inserted match criteria return the extra number of offered outputs in the package or composite service (details are shown in Figure 5.2 and explained in chapter 3).

As discussed in the EVP(Employee Vetting Procedure) example in subsection 3.4.1, there are services which are not required but offered freely. For example, the medical history check is a free service which can be offered in an EVP package which is not required by user. However the system has to consider all of the solutions. There are services which are part of a package, but not required by the user for example the international qualification check service is required by foreigners but not for local residents. For the EVP domain, all the searched options will be presented to the user and at the final stage the user can select the most appropriate solution.

EXAMPLE 5.

Input Variables(Required Number of services) =3

(Flight, Hotel, Taxi)

User Preferences

Strong Constraint = Yes (Hotel service for composition will be adjacent to stadium)

Soft Constraint = No

No of Extra Preferences (Optional Free offered Services) =3

–Free Travel Insurance (InF)

–Unlimited Music Download (MdF)

–Bonus loyalty Card points (PtF))

Number of Packages Available to fulfil Request =3

Number of Services =241

Time

Time by Considering Packages =40.26

Time without Packages =35.02

Table 5.6: Example 5

5.4.6 Example 6

In this example, the number of extra preferences are increased, while the contingency planning concept is tested. The process had been observed by excluding the input constant value required by the service, while the system responded by providing just a CDP plan.

For example, in the current use case the free offered service was selected by the user, but when the system observed the services description of the package and didn't find the exact required output, instead of taking the system into a failure stage, the simple CDP plan should be returned as an output. As mentioned in subsection 5.1.3, the CDP plan consist of single services and will be represented to user, in case of any parameter value missing. The case is tested in different setting and the time values varies ranging from 50.67 to 49.23 by considering packages.

The time coefficient factor is increased, as the number of extra preferences factor is elevated. However as discussed before the main aim is to find the different possibilities and solution to fulfil the user request with an optimal solution.

The tailor-made examples and the testing scenarios are just for the evaluation purpose of the proposed framework. However it is still an abstracted model which can be employed according to the nature of the domain and still a lot of work is required to fully automate the planning sequence generation and services discovery process. In current example, the concept of contingency planning can be adopted in a more efficient way. Due to the complexity of DWSC process, the exceptional handling facility should be instantiated, while in current case, the system will at least try to provide the minimum solution.

EXAMPLE 6.

Input Variables(Required Number of services) =3

(Flight, Hotel, Taxi)

User Preferences

Strong Constraint = Yes (Hotel service for composition will be adjacent to stadium)

Soft Constraint = Yes (User prefer Business Class travel)

No of Extra Preferences (Optional Free offered Services) =5

–Free Travel Insurance (InF)

–Unlimited Music Download (MdF)

–Bonus loyalty Card points (PtF))

–Free on board WiFi (WfF)

–Free on board SMS Service (SmF)

Number of Packages Available to fulfil Request =4

Number of Services =241

Time

Time by Considering Packages =50.67

Time without Packages =43.19

Table 5.7: Example 6

5.4.7 Example 7

The query was updated by changing constraints to evaluate the effect of increasing user preferences and packages. The details of constant and non-constant inputs are explained in following table. The overall taken time was increased as there are four packages present.

The same scenario is also tested by changing the total number of services. The total number of services was decreased from 241 to 186 in the services repository to evaluate the correctness and scalability metrics (i.e; by decreasing the number of services to consider the time taken and by increasing the number of packages)

EXAMPLE 7.

Input Variables (Required number of services) =3

(Flight, Hotel, Taxi)

User Preferences

Strong Constraint = Yes(Hotel service for composition will be adjacent to stadium)

Soft Constraint = No

No of Extra Preferences (Optional Free offered Services) =5

–Free Travel Insurance (InF)

–Unlimited Music Download (MdF)

–Bonus loyalty Card points (PtF))

–Free on board WiFi (WfF)

–Free on board SMS Service (SmF)

Number of Packages Available to fulfil Request =4

Number of Services =186

Time

Time by Considering Packages =49.23

Time without Packages =41.70

Table 5.8: Example 7

5.5 Health-Scallops Domain

In the next set of experiments, Health-Scallops domain has been selected to verify the concepts and evaluate the time coefficient on different domain. The Health-Scallops domain is a similar domain to the travel plan but was selected for evaluation purpose due to two reasons. Firstly, there is a lot of information available for the data set which is developed by the Ministry of Health in Germany. Secondly, the same domain was utilized in OWLS-XPlan. The Health-Scallops domain contains the *Create Account* service, which is a common functional requirement of different web services before utilizing other parts of services to validate security. As discussed, we selected OWLS-XPlan to evaluate our work, which resulted in less number of technical changes to test the proposed scenario. To include the composite services and packages, the services were combined together and the tailor-made packages or preferences are not showing a realistic approach.

For next set of experiments the irrelevant services were excluded from the repository which results in less time required for indexing and searching required for the total number of services. The number of services from repository was reduced from 241 to 186. To observe the contingency planning concept, the services descriptions were updated and changed deliberately, to observe the time out process. After the time out occurred time, the process initiated itself to replace the service which is not responding (contingency planning). In current domain, the system responded by providing the basic solution with out considering user preferences and by just combining single services.

Realistically, the tailor-made packages are not required and usually not included in real life scenarios of such domains (like Health-Scallops Domain), however to evaluate the situation and verification before use of domain, it was necessary to include such example. Here the concept of services' security research emerged, where

Table 5.9: OSPG Results (Health Plan)

Options Queries	Time By Cond. Packages	User Preferences	Time without Packages	No of Packages	Input Variables	Strong Constraint Preference	Soft Constraint Preference	Number of total Services	Planning steps Successful (Y/N)
Query 8	38.02	5	41.08	4	2	Y	N	186	Y
Query 9	30.13	5	36.39	4	3	N	N	186	Y
Query 10	36.18	3	29.46	1	3	Y	Y	186	Y
Query 11	40.25	3	32.16	2	3	Y	Y	186	Y
Query 12	23.43	3	22.34	0	3	Y	Y	186	Y
Query 13	14.98	3	14.21	0	3	N	Y	186	Y
Query 14	15.21	5	14.04	0	3	Y	Y	186	Y
Query 15	14.55	5	13.21	0	1	N	N	186	Y

a lot of research is ongoing to verify the user request and to secure their provided information.

For the services' security purpose, the security code was manually assigned to services. The security code A represents, the Call back Handler service JAAS (Java Authentication and Authorization Service) and X509Certificate.

5.5.1 Example 8

In the current use case, the total number of services is 186 and only 2 services were required while 4 packages are present. There is only one service available to fulfil the Create Medical account (as discussed, the hospital registration service has usually one instance and only one provider can provide such a service). As there is only one service available in the services pool (Create Medical account), the time by considering packages is decreased as compared with queries 6 and 7. Time is decreased as there are no participating composite services (with partial results) to fulfil the request. The matching engine took less time while in queries (6 and 7), there are four (4) similar services available. The system was tested for this setup as there are examples where just one service can provide the functionality required by user. For example login, library, email and memberships services are unique where usually only one service can provide the required service. In Rugby match travel plan example, the discovery time is increased as there are multiple services available to provide the same functionality and the matching engine have to compare the services. Table 5.10 shows the details of example 8.

5.5.2 Example 9

In example 9 (see table 5.11) , the system is tested by eliminating the Strong and Soft constraints constant values. There are five (5)extra preferences while there are

4 packages present. Create Medical Account, Request Transport and Propose Flight services are requested. The time spent considering packages is 30.13 seconds as there is just one services for Create Medical Account and two services each for Request Transport and Propose Flight available in the services pool.

EXAMPLE 8.

Input Variables (Required number of services) =2

(Create Medical Account, Request Transport)

User Preferences

Strong Constraint = Yes(Security Code A (JAAS,X509 Certificate)))

Soft Constraint = No

No of Extra Preferences (Optional Free offered Services) =5

–Free Travel Insurance (InF)

–Free Medical Card (McF)

–Free Initial Medical Assessment (MiF))

–Free Pick and Drop Service (PdF)

–Free on board SMS Service (SmF)

Number of Packages Available to fulfil Request =5

Number of Services =186

Time

Time by Considering Packages =38.02

Time without Packages =41.08

Table 5.10: Example 8

EXAMPLE 9.

Input Variables (Required number of services) =3

(Create Medical Account, Request Transport, Propose Flight)

User Preferences

Strong Constraint = No

Soft Constraint = No

No of Extra Preferences (Optional Free offered Services) =5

–Free Travel Insurance (InF)

–Free Medical Card (McF)

–Free Initial Medical Assessment (MiF))

–Free Pick and Drop Service (PdF)

–Free on board SMS Service (SmF)

Number of Packages Available to fulfil Request =5

Number of Services =186

Time

Time by Considering Packages =30.13

Time without Packages =36.39

Table 5.11: Example 9

5.5.3 Example 10

In current query (details in table 5.12), there is only one package available, as compared to previous queries in the Health-Scallops domain. In the forthcoming queries, the system output will be analyzed by reducing the number of packages and later where there are no packages present. In the current use case scenario, there are 3 services requested while there are 3 extra preferences present. The time spent considering packages is 36.18 seconds and without considering packages is 29.46 seconds. The PPP proposed approach is tested in this scenario, where the user was presented with a package consisting of three services (Create Medical Account, Request Transport, Propose Flight). The package also contains the service with security code A and the provided transport facility has the wheel chair accessibility. The package also contain Free Initial Medical Assessment(MiF) and Free Medical Card(McF).

EXAMPLE 10.

Input Variables (Required number of services) =3

(Create Medical Account, Request Transport, Propose Flight)

User Preferences

Strong Constraint = Yes (Security Code A (JAAS,X509 Certificate))

Soft Constraint = Yes (The selected trasport (service) has Wheelchair accessibility)

No of Extra Preferences (Optional Free offered Services) =3

–Free Travel Insurance (InF)

–Free Medical Card(McF)

–Free Initial Medical Assessment (MiF))

Number of Packages Available to fulfil Request =1

Number of Services =186

Time

Time by Considering Packages =36.18

Time without Packages =29.46

Table 5.12: Example 10

5.5.4 Example 11

Query 11 in table 5.13 is same as the previous example however there are 2 packages available to compare the output time coefficient. There is no significant change, as there are two packages present, while the second package was representing an partial output. In the current example, the flight service was missing from the partial package, while the single service was located and attached to the second package. Both packages were presented to the user while the first package represents the cheaper option which the user can select after analyzing both options.

5.5.5 Example 12

To verify the consistency in input/output and to check the generated sequence of steps, the system is tested by generating a final plan while considering only single and composite services (with Partial Results). In query 12 (see table 5.14), no tailor-made packages are present while the package is generated by considering the composite and single services. Because no packages are present, there is a few second difference in time by with and without considering packages.

The query represent the concept of (PPP) where the package was generated by considering composite services Create Medical Account and Request Transport while the Propose flight service was added to complete the package. The user extra preferences were considered and the user is presented with a package along with two free services (Free Medical Card(McF) and Free Initial Medical Assessment(MiF))

EXAMPLE 11.

Input Variables (Required number of services) =3

(Create Medical Account, Request Transport, Propose Flight)

User Preferences

Strong Constraint = Yes (Security Code A (JAAS,X509 Certificate))

Soft Constraint = Yes(The selected trasport (service) have a Wheelchair accessability)

No of Extra Preferences (Optional Free offered Services) =3

–Free Travel Insurance(InF)

–Free Medical Card(McF)

–Free Initial Medical Assessment(MiF))

Number of Packages Available to fulfil Request =2

Number of Services =186

Time

Time by Considering Packages =40.25

Time without Packages =32.16

Table 5.13: Example 11

EXAMPLE 12.

Input Variables (Required number of services) =3

(Create Medical Account, Request Transport, Propose Flight)

User Preferences

Strong Constraint = Yes (Security Code A (JAAS,X509 Certificate))

Soft Constraint = Yes (The selected transport (service) has Wheelchair accessibility)

No of Extra Preferences (Optional Free offered Services) =3

–Free Travel Insurance (InF)

–Free Medical Card (McF)

–Free Initial Medical Assessment (MiF))

Number of Packages Available to fulfil Request =0

Number of Services =186

Time

Time by Considering Packages =23.43

Time without Packages =22.34

Table 5.14: Example 12

5.5.6 Example 13

To verify the system output when there are no strong constraint parameter present, the system was tested by only providing soft and extra preferences. The aim of the query (details in table 5.15) is to observe the system output while there is no Strong constraint value present. There are rare chances of such kind of setup in real life scenario, but some time user have a strong motive but may be have some preferences or options in mind (for example in logistics and route planning domains). In the current query, only single services and partial services were present, thats why the total time is decreased to 14.98 seconds. Although there are partial services available but the ontological represents was change where the matching engine was unable to locate and attach the single service.

EXAMPLE 13.

Input Variables (Required number of services) =3

(Create Medical Account, Request Transport, Propose Flight)

User Preferences

Strong Constraint = Yes (Security Code A (JAAS,X509 Certificate))

Soft Constraint = Yes(The selected transport (service) has a Wheelchair accessibility)

No of Extra Preferences (Optional Free offered Services) =3

–Free Travel Insurance(InF)

–Free Medical Card(McF)

–Free Initial Medical Assessment(MiF))

Number of Packages Available to fulfil Request =0

Number of Services =186

Time

Time by Considering Packages =14.98

Time without Packages =14.21

Table 5.15: Example 13

5.5.7 Example 14

To verify output results, in query 14 and 15 two more partial packages were introduced and there is no complete package present. In example 14 (details in table 5.16), there is the same setup to observe the PPP package but no complete packages were present. The time constraint value shows that the system take very less time to locate the services if there are no packages available. The results were almost same, when the Travel domain was tested with same variations (without complete packages)

5.5.8 Example 15

As discussed in the last query, that both examples 14 and 15 were tested without considering any packages. In query 15 (see table 5.17), there is only one service required which is to Create Medical Account while there are no strong and soft constraint Preferences available. The same service was duplicated in the services pool while the one service was assigned security code A. The aim of the query was to observe the sequence of steps while finding a single service. The query was generated by passing the preference parameter of security code A while the system took 14.55 seconds to respond. In such case, the system processing flow can be updated by including request analyzer module, which will result in more efficiency of results.

EXAMPLE 14.

Input Variables (Required number of services) =3

(Create Medical Account, Request Transport, Propose Flight)

User Preferences

Strong Constraint = Yes (Security Code A (JAAS,X509 Certificate))

Soft Constraint = Yes (The selected transport (service) has a Wheelchair accessibility)

No of Extra Preferences (Optional Free offered Services) =5

–Free Travel Insurance (InF)

–Free Medical Card (McF)

–Free Initial Medical Assessment (MiF))

–Free Pick and Drop Service (PdF)

–Free on board SMS Service (SmF)

Number of Packages Available to fulfil Request =0

Number of Services =186

Time

Time by Considering Packages =14.55

Time without Packages =13.21

Table 5.16: Example 14

EXAMPLE 15.

Input Variables (Required number of services) =1

(Create Medical Account)

User Preferences

Strong Constraint = No

Soft Constraint = No

No of Extra Preferences (Optional Free offered Services) =5

–Free Travel Insurance (InF)

–Free Medical Card(McF)

–Free Initial Medical Assessment (MiF))

–Free Pick and Drop Service (PdF)

–Free on board SMS Service (SmF)

Number of Packages Available to fulfil Request =0

Number of Services =186

Time

Time by Considering Packages =14.55

Time without Packages =13.21

Table 5.17: Example 15

5.5.9 Effectiveness of Results in conjunction with Evaluative Metrics

The system input parameters (required inputs) were changed to observe the variation in results and queries. The changes were made to observe the significant difference in the services selection procedure and required outputs. The discussed examples show some results which were obtained after running different cycles, by changing different parameter values, however only the queries with significant changes are considered and mentioned in this thesis. In current set of experiments, we tried to use both methods, by providing all the requirements up in front and analyze the plan behavior while just manually coding the strong and soft constraint at the time of sequence generation. It is important to observe the results by changing the variations, three different variations analyzed which are preferences, strong constraint and soft constraint.

As discussed Scope, Correctness, Scalability and Versatility are four evaluative metrics which are used to test our proposed approach. In the current thesis, we use a number of Web services and different types of preferences to measure the scalability and scope of the proposed approach while the changes in input values and results of two domains are utilized to prove correctness and versatility. The following two tables shows the results of the adopted strategies to measure the proposed evaluative metrics. The first table shows Travel Plan results (Example 1-7), related research questions and evaluative metrics. The next table shows the Health-Scallops domain results (Example 8-15), related research questions and evaluative metrics. The 2nd row shows the linked functionality and 1st column shows the relevant metrics. If the required example satisfy the relevant criteria, the box is filled with **X**. Some queries were specifically executed to proof the concepts and to support the required logic for scalability and versatility metrics (Examples 6, 12, 13, 14, 15).

Examples Evaluative Metrics	Example 1	Example 2	Example 3	Example 4	Example 5	Example 6	Example 7
Functionality	Preferences PPP	Preferences CDP	Strong \Soft Preferences	Packages	Extra Offered Services	Contingency Planning	Preferences and Packages
Scope	X	X	X	X	X	X	X
Correctness	X	X	X	X	X		X
Scalability						X	X
Versatility		X	X				X

Examples Evaluative Metrics	Example 8	Example 9	Example 10	Example 11	Example 12	Example 13	Example 14	Example 15
Functionality	Preferences PPP	Preferences CDP	Strong \Soft Preferences	Packages	Strong Soft	Soft Constraint	Partial Packages	Basic Plan (No Packages)
Scope	X	X	X	X	X	X	X	X
Correctness	X	X	X	X	X	X	X	X
Scalability	X	X		X	X			X
Versatility	X	X			X		X	X

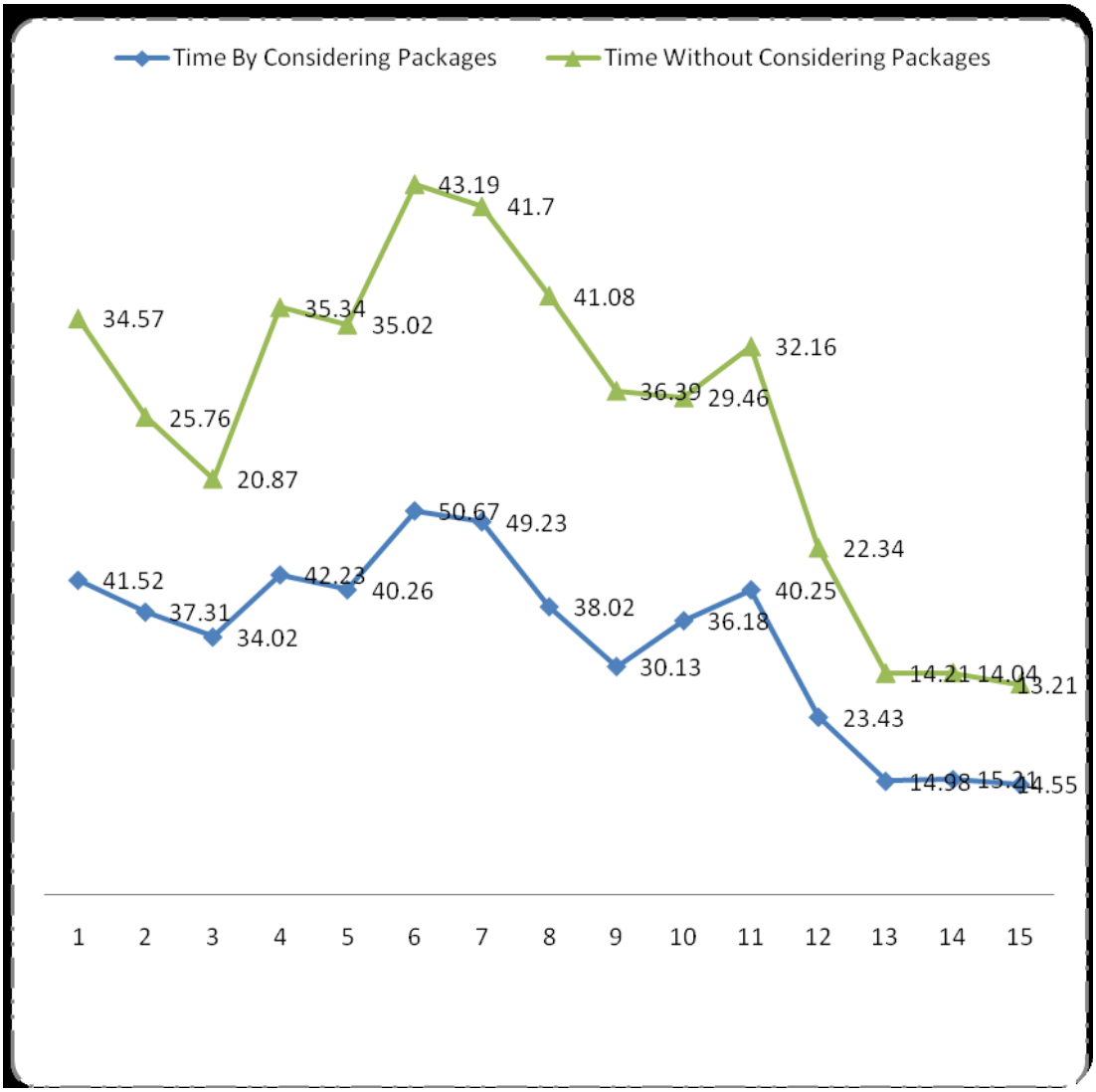

Figure 5.6: Comparison of Packages Results (Graph)

Figure 5.6 shows the variation of changes in the number of packages and other constraints. There is no significant observable time difference, while some variations of input values are observed by changing input constraints (strong and soft constraints and the number of packages). The bar chart of Figure 5.7 shows the overall comparison of all the queries by showing the impact of all constant and variable factor. The reason to arrange such a setup is to compare the results with a similar environment and to check the efficiency of the proposed framework and to answer the following research metrics questions which were proposed to evaluate the proposed model.

1. ***What benefits will the proposed approach bring in terms of considering preferences and locating services to find an optimal solution?*** The results (Examples 1-15) show that the system is able to support different types of preferences, two levels of support are considered, strong and soft constraints while the preferences are further supported by three types of packages (PPP, CDP, PDP).

2. ***Is the proposed approach support the efficiency of search by considering the user desired factors?*** The system is providing results after considering a number of options and factors from participating services (i:e; Single Services, Composite Services and Packages (Services containing compiled results from different services)

3. ***Does the proposed approach is more scalable as the new input parameters are introduced with the proposed framework (Strong, Soft and Extra Preferences based Constraint values)?*** The above mentioned results show that the system is able to support a growing number of constraints and inputs. OSPG is tested to support number of inputs (strong and soft preferences) and outputs (extra free services) to address user needs.

4. ***Can this proposed framework support full automation?*** The proposed approach can support full automation, as the system is designed to address a number of issues while there is the possibility to add extra features to support a number of functional and non-functional features.

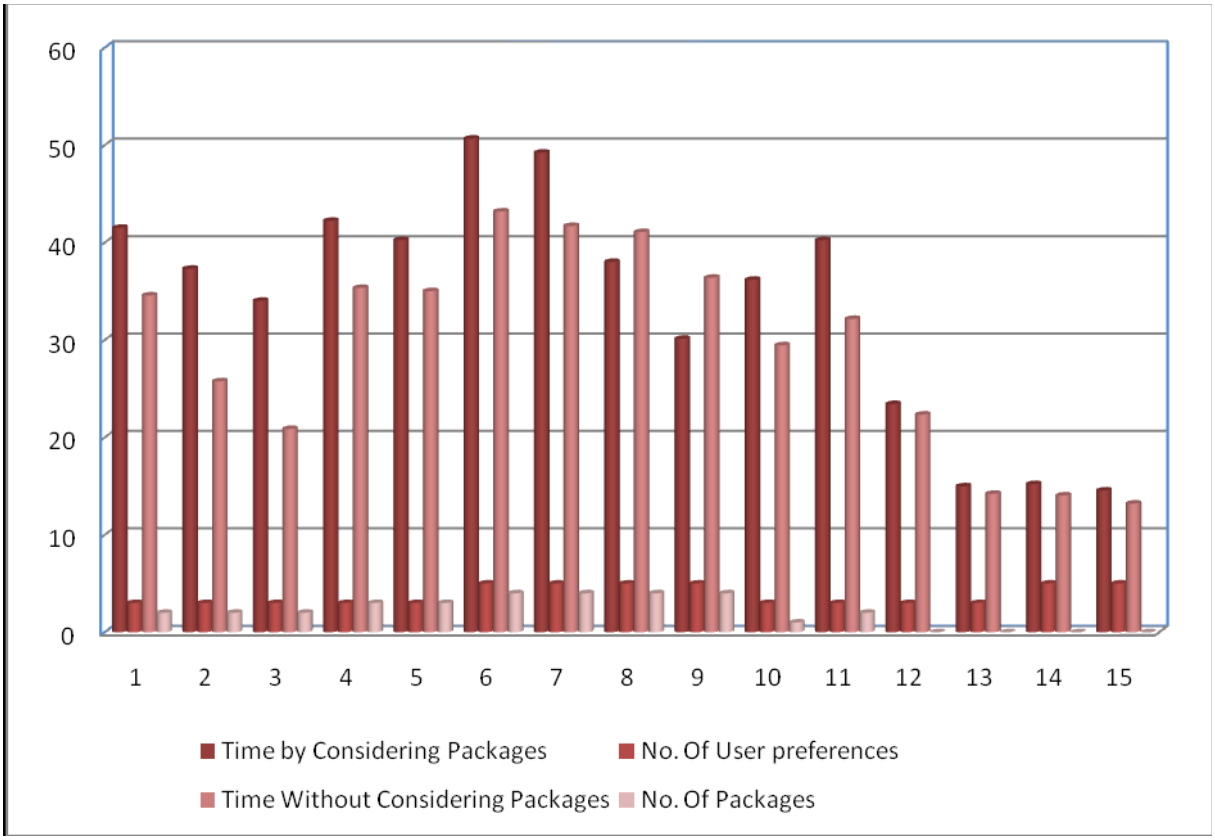

Figure 5.7: Comparison of overall Results (Graph)

5.6 Comparison with related work results

In the previous work, only 50 services were considered by OWLS-XPlan [84, 104], but in the current model, some 241 and 186 services were tested. For the sake of evaluation, the logic of including the composite services within packages was updated to accept the user preferences. The strong and soft constraint-based inputs have been observed in all instances.

Service	Inputs	Outputs
BookToPublisher	Book, Author	Publisher
CreditCardCharge	OrderData, CreditCard	Payment
ElectronicOrder	Electronic	OrderData
PublisherElectronicOrder	PublisherInfo	OrderData
ElectronicOrderInfo	Electronic	OrderInformation
Shipping	Address, OrderData	ShippingDate
WaysOfOrder	Publisher	Electronic
CustomsCost	Publisher, OrderData	CustomsCost

ımber of web rvices		10	100	500	1000
Preprocessing time		5857	6104	5875	5703
Total transformation time	X	4594	70062	350836	792109
	E	4531	75725	335477	796797
	C	4585	74688	728633	3901141
Transformation time per web service	X	459	700	702	792
	E	453	671	757	797
	C	459	746	1457	3901
Planning time (LPGtd)	X	1	13	16	17
	E	4	6	15	16
	C	3	5	16	16

Figure 5.8: PORSCE II and VLEPPO-based framework Results (Details in [67])

We compared the processing time results with the PORSCE II and VLEPPO-based framework by Ourania Hatzi and Ioannis Vlahavas [67] as shown in Figure 5.8. It has been observed that the current results in this thesis are comparatively better than previous dynamic composition efforts, where SHOP2, PORSCE and OWLS-XPlan were utilized. However, since there are no quantitative results available for the OWLS-XPlan, in the current work, the discovery and composition process have been observed in both directions i.e. with, and without preference or packages. The PORSCE II results present the number of services included in tests while the transformation time was discussed. The services are tested offline; however the time will be reduced if the same arrangement is tested online. Due to the limitation of the OWLS-XPlan' environment in supporting the selection of preferences-based packages selection, the separate block of control structures were modified to include the preferences, which again consumes the CPU processing cycles, slowing down the overall process. This can be set right by following coding standards, and by improving the discovering process sequence. The time difference is significantly high for the composition, by considering composite services, but as discussed, some patch-up coding to include preferences and interfaces of manual conversion process takes more time. It has been analyzed that the composition time increases directly if the number of packages increases. There is, on average, 3245 milliseconds (approximately) difference if the number of input parameters (variables) is increased to include all of the user's strong and soft preferences.

The experimental setup was adjusted to observe the results, however as discussed before, a high level of abstraction is required in the actual composition process. For good scientific evaluation, there is a requirement for development of an end-to-end, autonomic environment, that will take the input from the users and provide a final solution after communicating with all services located on different servers. However,

all related work completed in the services composition area is varied across different sectors. There is also a lack of benchmark examples in the DWSC area; whereas the strength of the current work is that different sector examples are discussed and evaluated. Instead of discussing the technical aspects of the environments, concepts are explained. It is important, however, to mention here that DWSC is difficult, but not an impossible process to adopt. The difficulty is the adaptation of the processes, by observing the domain requirements in real-time, as the requirement is that the framework should be able to support preferences and optimal services selection.

The proposed framework is useable by different applications and different examples from daily life are explained to support such concepts. In the current work, different variations were considered and a common model is presented which is usable for different type of setup and applications. As discussed to support the scope of the current setup, we presented the different possible answers to support user queries and to explain concepts PPP, CDP, PDP. Such kind of setup can be varied and utilized, according to domain requirements. The following tables (Table 5.18, Table 5.19, Table 5.20) shows the results of supported functionality of OSPG.

For a successful composition process, the system has to match the user required output with the available solutions by considering functional and non-functional requirements. In current work functional and non-functional properties have been analyzed, the Figure 5.7 shows the overall results of all queries. Number of test cases were performed, however just significant changed results are taken for discussion in this thesis. which are further divided for two domains.

By establishing this integrated environment, we are attempting to fill the gap between distributed network application development and autonomic environments [125] , as discussed in chapter 1. In the SOA paradigm, if we introduce communication links between distributed technologies and web services, then we will automat-

ically find an autonomous environment [125]. This environment will be one where applications can connect without prior knowledge of platform, human interaction and technical details, and can help to achieve loosely-coupled, platform-agnostic application properties [125].

5.6.1 Summary

In the current chapter, the implementation and evaluation methods are discussed. There are no common evaluation methods available for the dynamic web services composition, however, the existing methods are analyzed. The Scope, Correctness, Scalability and Versatility evaluation metrics were observed in the current work, with the obtained results explained. The user preferences types were considered, and the system was observed against user preferences (e.g. PPP, CDP and PDP). The 15 different use case scenarios are explained to understand the system flow and to prove the efficiency to solve services composition task.In the related work, the separate matchmaker modules are utilized for search process. In the AI-planning-based approaches, the search process (discovery) is normally ignored, or it is assumed that any matchmaker can be integrated within the planning environment [158].

We have examined the services composition task by using the AI-planning-based approach. To discuss the limitations, we analyzed and evaluated the proposed model by using theoretical and experimental approaches where possible. It is very difficult to test the problem for non-deterministic domains; however we tried to check the sub tasks, before comparing the accumulative final results with the related work (SHOP2, PORSCE II and VLEPPO and OWLS-XPlan).

Table 5.18: OSPG Results (Preferences Plus Package)

Options Queries	Time By Cond. Packages	User Preferences	Time without Packages	No of Packages	Input Variables	Strong Constraint Preference	Soft Constraint Preference	Number of total Services	Planning steps Successful (Y/N)
Query 1	41.52	3	34.57	2	3	Y	Y	241	Y
Query 4	42.23	3	35.34	3	2	Y	Y	241	Y
Query 6	50.67	5	43.19	4	3	Y	Y	241	N
Query 10	36.18	3	29.46	1	3	Y	Y	186	Y
Query 11	40.25	3	32.16	2	3	Y	Y	186	Y
Query 12	23.43	3	22.34	0	3	Y	Y	186	Y
Query 14	15.21	5	14.04	0	3	Y	Y	186	Y

Table 5.19: OSPG Results (Complete Desired Package)

Options Queries	Time By Cond. Packages	User Preferences	Time without Packages	No of Packages	Input Variables	Strong Constraint Preference	Soft Constraint Preference	Number of total Services	Planning steps Successful (Y/N)
Query 2	37.31	3	25.76	2	3	Y	N	241	Y
Query 5	40.26	3	35.02	3	3	Y	N	241	Y
Query 7	49.23	5	41.70	4	3	Y	N	186	Y
Query 8	38.02	5	41.08	4	2	Y	N	186	Y

Table 5.20: OSPG Results (Miscellaneous)

Options Queries	Time By Cond. Packages	User Preferences	Time without Packages	No of Packages	Input Variables	Strong Constraint Preference	Soft Constraint Preference	Number of total Services	Planning steps Successful (Y/N)
Query 3	34.02	3	20.87	2	3	N	N	241	Y
Query 9	30.13	5	36.39	4	3	N	N	186	Y
Query 13	14.98	3	14.21	0	3	N	Y	186	Y
Query 15	14.55	5	13.21	0	1	N	N	186	Y

Chapter 6

Conclusion and Future Work

6.1 Summary

Web services are self contained programs that can be executed through the standard, global protocols of the internet [13]. There are many services on the web and each one has a limited functionality [7]. In many cases, a single web service is not sufficient to respond to the user's request and as a result of this services should be combined through a pattern of composition to achieve a specific goal [2]. Due to constant changes in input/output parameter values, interfaces and networking issues, it is difficult to integrate and maintain these services dynamically [87]. The research topic addressed these issues and tried to conform user preferences by getting their basic desires for the composition process and presenting the user with an optimal solution (explained in section 5.1). To provide an optimal solution, single services and composite services with packages were considered.

In this thesis, the topic of Web Services Composition (WSC), its dynamic discovery process to locate services and search for optimal results, was studied. The main aim of the current work was to develop a framework model where the best services according to user requests should be discovered dynamically and participate in the

composition process. This was successfully achieved by proposing and evaluating [1] a new framework model which is more efficient and scalable for the composition task. The final results were evaluated and compared with the available benchmark examples.

A number of solutions are available for services composition which utilize different techniques and methods to compose the services [105, 134, 138, 144, 171]. The available approaches were analyzed and the proposed OSPG framework was presented. This provided a synopsis model of the dynamic web services composition concept by considering and conforming user preferences. In addition to the DWSC automation, the thesis explores the current issues surrounding services composition process by using AI planning methods.

The complete framework-based models were established to meet the user requirements while the algorithms to include and search for optimal solutions were developed and improved (where applicable). This was done so that the required, optimal solutions could be found through consideration of user preferences.

chapter 2 provides details of the related work and the available tools and technologies used to automate the services composition methods. The proposed framework can be divided into three dimensions and classified as a cross disciplinary research areas. The following three layers show three different areas in the SOC (Service Oriented Computing).

- Current web services, Structural and integrated coordination (workflow and AI planning).
- AI planning and automated optimization
- User support interfaces and preferences based automation.

[1] explained in section 5.1

To synchronize the three different areas, the system required an automated and coordinated planning based solution which will provide a channel for achieving a final goal.

In chapter 3, the planning requirements were discussed. The aim of the research is to work towards the solution of web service composition by using AI (Artificial Intelligence) techniques. Before moving towards an AI based composition framework, the research direction was focusing towards current issues as discussed in previous research. During the development process of DWSC and before reaching the end of the composition spectrum, research activities were carried out in two directions i.e; the theoretical side included literature reviews of related research on the semantic web, selection, discovery and composition. And the second, technical context of developed technologies, tools and features(such as WSDL, UDDI, SOAP, DAML, DAMLS and OWL-s) were studied along with the practical side, this was done by testing automated tools used by different developers.

chapter 4 investigated the requirements of the proposed model, the main focous was to design a common solution which will not only fulfil the requirements of web services but also be reusable for all similar applications. The main focus of the project was to provide a conceptual framework for services composition which will consider user preferences and include all the pre-compiled web services packages during the Dynamic Web Services Composition (DWSC) process. It is a clear from the discussion in chapter 4 that the dynamic services composition process can not be executed with the current structural (syntactic) web or current distributed technology approaches. The basic requirements for the required model were established. The essential demands from a current proposed system (such as FLC and PLC) were discussed in detail. The explanation of recompiled packages was provided in chapter 4.

The developed algorithms were presented by considering user preferences such as PPP, CDP and PDP. The investigation of the existing approaches for preference based planning demonstrated that the algorithm can support only strong constraints or soft constraints as some of them ignored the preferences approach and instead used a standard QoS constraint (such as cost, time etc) for the discovery of an optimal plan. We therefore explored the possibility of different types of preferences and divided user preferences into PPP, CDP and PDP. The search process and composition process was discussed in the same chapter.

In chapter 5, the system planning results were presented. The test bit contains a variation of inputs to verify the different combination of inputs, and two domains were tested to verify the user inputs.

6.2 Research Contribution

The OSPG model made several important contributions to the dynamic web services composition process. Three important issues were addressed in the current work which were proposed and explained in chapter 1.

- *RQ1. Which mechanisms are required in order to automate the Dynamic Web Services Composition process ? Further, is it possible to develop a framework which can support the composition process without user (operator) involvement ?* The user preferences based composition framework that heavily relies on four key concepts semantic similarity, ontologies, preferences and conformance planning, was implemented. The possibility of the proposed framework model to support different domains as discussed in chapter 1 without user involvement was discussed in this thesis. Two domains were tested and the results were evaluated through comparison with related work. Where possible different examples were discussed in the thesis to provide a deterministic characteristic

of the proposed framework model.

- *RQ2. How should the framework be designed to consider functional and non-functional attributes when providing a final solution, according to user requirements ?.* In existing work compiled by different research groups, it has been discussed that the services composition process main complexity is to analyze and adopt a number of different inputs, otherwise it is possible to use simple distributed technologies. The thesis contributed by using strong and soft constraint based inputs to address the different input parameter issue. The problem can be researched further to analyzed attributes or characteristics of provided inputs and by selecting the correct input from a number of possible inputs. To include different functional and non-functional attributes, different use case scenarios and results were mentioned in section 5.3.

- *RQ3. How will participating services be analyzed and considered for the composition process in order to establish optimal results ?.* Besides the contribution of user preferences, we also argued to find the optimal solution by dividing and considering different types of services. As discussed services were divided into three types; Single Services, Composite Services and Services which contain packages (to include PPP, CDP, PDP concepts) as discussed in chapter 4. The current approach improved the discovery process by analyzing the services' contents and by using the attached description. The attached description can help to find the required package instead of finding the single services and integrating them. At present, there is a requirement to provide intelligence power capabilities to machines so it will be possible for machines to work out intelligent solutions. We tried in the current work to conform the user preferences, and used a knowledge-based setup to conform user requirements. These types of research question and outcome have heightened our

understanding at a new research direction which was not discussed before in previous web services' composition approaches.

6.2.1 Practical Issues(Outstanding)

The QoS objectives and all factors need to be evaluated in conjunction with user preferences. In the current work during testing we tried to use the time constraint (as a quality of service factor) but all other factors such as cost, latency and response time need to be tested. For real time results, the UDDI services database needs to be linked with the OSPG to obtain the actual results.

The main advantage of the current developed system is that it can support different services composition requirements and structures. We addressed the main problem of DWSC in section 2.3, where services contain single or multiple results. Although we used simple examples to explain the concepts, the same model can be adopted for different domains such as weather forecasting, police web services, route planning, logistics, e-commerce and NHS Bio-medical research. The only disadvantage at this stage is that it represents a tailor-made platform (some integrated modules where a lot of manual conversion is required which can be easily improved by coding all modules and taking the system up to a fully automatic model). The interface then requires to be reprogrammed to receive the main inputs. As we discussed before, the replanning module could be investigated further to update and reconfigure itself due to configuration and compatibility issues. This module needs to be updated and tested against the required criteria.

6.2.2 Directions of Future Work

As discussed, a frontend interface is required to get the user requirements which will provide an easy and effective way to get user preferences. There is a requirement for

the user to interact during the composition process, to observe the composition process; and to change the process sequence which will be developed in future (similar examples were discussed in chapter 2).

It is necessary to store the user request for the services composition and response states, which later can help to provide instant answers to other requests without wasting time and resources searching again for the same query to fulfil the request. In addition, the web cache and proxy server platform based concepts can be applied to save incoming and outgoing messages for the composition process by adopting a local services repository.

The QoS based services composition was not analyzed in detail, as our priority was extended towards dynamic discovery, preferences and the composition process. However by keeping in mind the importance of a quality driven approach, the QoS parameters were embedded in the proposed framework.

Two different approaches were discussed in order to find the best results. The first was to analyze the appropriate services, according to user requirements, and provide an exact, final solution. The second was to find the alternative solutions and options through consideration of the combined results, provided by services. There is a requirement of the DWSC process to not only consider those composed services which vary according to exact user requirements, but to suggest a different solution(s) to the user, by conforming to user preferences. During the DWSC process, if the services fail to respond, or due to any reasons for failure, there is a requirement for the system to save itself, by reverting to the previous, acceptable and correct state. It should not re-plan the composition steps again, and have to select the second best solution in real time. There is also a question mark around the carrying out of the composition task in real time, by conforming to the user preferences, and contingency planning in case of failure. The proposed framework

can be extended to accommodate the aforementioned requirements for the DWSC process.

Bibliography

[1] Agarwal, S., Handschuh, S., and Staab, S. (2004). Annotation, composition and invocation of semantic web services. *Web Semantics Science Services and Agents on the World Wide Web*, 2(1):31–48. 65, 125

[2] Agarwal, V., Chafle, G., Dasgupta, K., Karnik, N., Kumar, A., Mittal, S., and Srivastava, B. (2005). Synthy: A system for end to end composition of web services. *Web Semantics Science Services and Agents on the World Wide Web*, 3(4):311–339. 4, 9, 19, 38, 72, 188

[3] Aggarwal, R. and Verma, K. (2004). Constraint driven web service composition in meteor-s by rohit aggarwal (under the direction of amit p. sheth and john a. miller). *Computing 2004SCC*. 99

[4] AIPS-98 Planning Competition Committee (1998). PDDL - The Planning Domain Definition Language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control. 35

[5] Al-Masri, E. and Mahmoud, Q. H. (2007). Qos-based discovery and ranking of web services. In *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, pages 529–534. IEEE. 101

[6] Alonso, G., Casati, F., Kuno, H., and Machiraju, V. (2004). *Web services*. Springer. 19, 23, 101

[7] Alrifai, M. and Risse, T. (2009). Combining global optimization with local selection for efficient qos-aware service composition. In *Proceedings of the 18th international conference on World wide web*, pages 881–890. ACM. 36, 37, 188

[8] Alrifai, M., Skoutas, D., and Risse, T. (2010). *Selecting skyline services for QoS-based web service composition*, page 11. Number October 2006. ACM Press. 6

[9] Anderson, J. Q. (2010). The fate of the semantic web. Technical report, Elon University, Pew Research Center's Internet and American Life Project, Washington. 21

[10] Antonopoulos, N. and Gillam, L. (2010). *Cloud Computing*. Springer. 18, 19

[11] Arno Puderarno, K. R. and Pilhofer, F. (2005). Distributed System Architecture, A Middleware Approach. In Tim Cox, editor, *Distributed System Architecture*. Elseevier. 41

[12] Arpinar, I. B., Zhang, R., Aleman-Meza, B., and Maduko, A. (2005). Ontology-driven web services composition platform. *Information Systems and E-Business Management*, 3(2):175–199. 20, 26, 123

[13] Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer networks*, 54(15):2787–2805. 188

[14] Baier, J. A. (2010). PhD thesis. Effective search techniques for non-classical planning via reformulation. 71

[15] Basili, V. R., Caldiera, G., and Rombach, H. D. (2001). The Goal Question Metric Approach. 2:1–10. 130

[16] Benatallah, B., Hacid, M.-S., Leger, A., Rey, C., and Toumani, F. (2005). On automating web services discovery. *The VLDB Journal*, 14:84–96. 5, 54, 96

[17] Bertoli, P., Cimatti, A., Roveri, M., and Traverso, P. (2000). Planning in Non-deterministic Domains under Partial Observability via Symbolic Model Checking. 73

[18] Bertoli, P., Helmert, M., and Pistore, M. (2009). Message-Based Web Service Composition , Integrity Constraints , and Planning under Uncertainty : A New Connection. *Artificial Intelligence*, 35:49–117. 99, 125, 129

[19] Bertoli, P., Hoffmann, J., Lecue, F., and Pistore, M. (2007). Integrating discovery and automated composition: from semantic requirements to executable code. *Web Services, IEEE International Conference on*, 0:815–822. 72

[20] Bertoli, P., Pistore, M., and Traverso, P. (2010). Automated composition of web services via planning in asynchronous domains. *Artificial Intelligence*, 174(3-4):316–361. 9

[21] Bidot, J. (2011). Using AI Planning and Late Binding for Managing Service Workflows in Intelligent Environments. *Artificial Intelligence*, pages 156–163. 4, 125

[22] Bienvenu, M., Fritz, C., and McIlraith, S. A. (2006). *Planning with Qualitative Temporal Preferences*, pages 134–144. Number Reiter. 113

[23] Biswas, D. (2007). Web services discovery and constraints composition. Technical report, IRISA-INRIA, Campus Universitaire de Beaulieu, Rennes, France. 4, 19, 99

[24] Blum, A. L. and Furst, M. L. (1997). Fast planning through Planning Graph Analysis. *Artificial Intelligence*, 90:281–300. 47

[25] Bonet, B. and Geffner, H. (1998). HSP: Heuristic search planner. In *The Fourth International Conference on Artificial Intelligence Planning Systems*. 45

[26] Bonet, B. and Geffner, H. (2005). mGPT: A Probabilistic Planner Based on Heuristic Search. *Journal of Artificial Intelligence Research*, 24:933–944. 45

[27] Booth, D. and Haas, H. (2004). Web services architecture. Technical report, W3C Working Group. 1, 19

[28] Booth, D., Haas, H., McCabe, F., and Newcomer, E. (2004). Web services architecture. *Group*, (February). 13

[29] Bosio, S. and Righini, G. (2003). A combinatorial optimization problem arising from text classification. *Electronic Notes in Discrete Mathematics*, 13(March 2003):22–25. 36

[30] Bryce, D., Kambhampati, S., and Smith, D. E. (2002). Planning graph heuristics for belief space search. 51

[31] Bucchiarone, A., Melgratti, H., and Severoni, F. (2007). Testing service composition. *In Proceedings of the 8th Argentine Symposium on Software Engineering*, pages 1–16. 10

[32] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616. 18

[33] Carman, M., Serafini, L., and Traverso, P. (2003). Web service composition as planning. In *In ICAPS 2003 Workshop on Planning for Web Services*. 34

[34] Casati, F., Ilnicki, S., jie Jin, L., Krishnamoorthy, V., and Shan, M.-C. (2000a). Adaptive and dynamic service composition in . In *CAiSE'00*, pages 13–31. 79

[35] Casati, F., Ilnicki, S., jie Jin, L., Krishnamoorthy, V., and Shan, M.-C. (2000b). eflow: A platform for developing and managing composite e-services. In *AI-WoRC'00*, pages 341–348. 79

[36] Castells, M. (2011). *The rise of the network society: The information age: Economy, society, and culture*, volume 1. John Wiley & Sons. 15

[37] Chen, H., Finin, T., and Joshi, A. (2003). An ontology for context-aware pervasive computing environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, 18:197–207. 20, 21

[38] Cimatti, A., Roveri, M., and Bertoli, P. (2004). Conformant planning via symbolic model checking and heuristic search. *Artificial Intelligence*, 159(1-2):127–206. 71, 72, 73

[39] Conti, M., Chong, S., Fdida, S., Jia, W., Karl, H., Lin, Y. D., Mhnen, P., Maier, M., Molva, R., Uhlig, S., and et al. (2011). Research challenges towards the future internet. *Computer Communications*, 34(18):2115–2134. 1

[40] Cunsolo, V. D., Distefano, S., Puliafito, A., and Scarpa, M. (2009). Volunteer computing and desktop cloud: The cloud@home paradigm. *2009 Eighth IEEE International Symposium on Network Computing and Applications*, pages 134–139. 18

[41] da Silva, E. G., Pires, L. F., and van Sinderen, M. (2011). Towards runtime discovery, selection and composition of semantic services. *Computer communications*, 34(2):159–168. 7

[42] Daniel, F. and Pernici, B. (2006). Insights into web service orchestration and choreography. *International Journal of EBusiness Research*, 2(March):58–77. 63

[43] d'Aquin, M., Motta, E., Sabou, M., Angeletou, S., Gridinoc, L., Lopez, V., and Guidi, D. (2008). Toward a new generation of semantic web applications. *Intelligent Systems, IEEE*, 23(3):20–28. 15

[44] Ding, Y., Stollberg, M., and Fensel, D. (2001). Semantic Web Languages Strengths and Weakness. 20

[45] Dinh, T. B. (2007). *Optimal temporal planning using the plangraph framework*. PhD thesis, School of Computing and informatic, University of Huddersfield . 69

[46] Domshlak, C. (2007). Probabilistic Planning via Heuristic Forward Search and Weighted Model Counting. *Artificial Intelligence*, 30:565–620. 73

[47] Dwyer, C., Hiltz, S. R., and Passerini, K. (2007). Trust and privacy concern within social networking sites: A comparison of facebook and myspace. *Americas The*, 123(4):339–350. 17

[48] El Hadad, J., Manouvrier, M., and Rukoz, M. (2010). Tqos: Transactional and qos-aware selection algorithm for automatic web service composition. *Services Computing, IEEE Transactions on*, 3(1):73–85. 43

[49] Erol, K., Hendler, J., Nau, D., and Tsuneto, R. (1995). A Critical Look at Critics in HTN Planning. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. 29

[50] Fabra, U. P. and Fabra, U. P. (2002). Compiling Uncertainty Away : Solving Conformant Planning Problems Using a Classical Planner (Sometimes). 73

[51] Fenn, J. and Raskino, M. (2008). *Mastering the hype cycle: how to choose the right innovation at the right time*. Harvard Business Press. 17

[52] Foster, H. (2006). A Rigorous Approach To Engineering Web Service Compositions. *Provider*, (January). 62, 63

[53] Fries, B., Klusch, M., and Sycara, K. (2009). Owls-mx: A hybrid semantic web service matchmaker for owl-s services. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(2):121–133. 143

[54] García-Sánchez, F., Valencia-García, R., Martínez-Béjar, R., and Fernández-Breis, J. T. (2009). An ontology, intelligent agent-based framework for the provision of semantic web services. *Expert Systems with Applications*, 36(2):3167–3187. 40

[55] Gardner, S. P. and Gardner, S. P. (2005). Ontologies and semantic data integration (Reviews). 10(14). 4, 21

[56] Gekas, J. and Fasli, M. (2005). Automatic Web Service Composition Using Web Connectivity Analysis Techniques. 33, 56

[57] Gerevini, A. and Long, D. (2005). Plan Constraints and Preferences in PDDL3 . Technical report, Department of Electronic for Automation, University of Brescia. 35, 112, 129

[58] Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated Planning: Theory and Practice*. Morgan Kaufmann ISBN 1-55860-856-7. 4, 27, 28, 49, 60, 61, 70, 96

[59] Gholam, S., Tabatabaei, H., Kadir, W. M. N. W., and Ibrahim, S. (2009). Automatic Discovery and Composition of Semantic Web Services Using AI Planning and Web Service Modeling Ontology. *International Journal*, 4(1):1–10. 4, 21, 131

[60] Goldstein, J. (1999). Emergence as a construct: History and issues. *Emergence*, 1(1):49–72. 15

[61] Graham, S., Daniels, G., Davis, D., Nakamura, Y., Simeonov, S., Brittenham,

P., Fremantle, P., Koenig, D., and Zentner, C. (2004). *Building Web services with Java: making sense of XML, SOAP, WSDL, and UDDI*. SAMS publishing. 24

[62] Granell, C., Gould, M., Grø nmo, R., and Skogan, D. (2005). Improving Reuse of Web Service Compositions 2 A Service Composition Methodology. *Information Systems*, (August). 7

[63] Grift, E. W. V. D. (2002). Literate Programming C has been around for 30 + years. (c). 22

[64] Hamadi, R. and Benatallah, B. (2003). A petri net-based model for web service composition. In *Proceedings of the 14th Australasian database conference-Volume 17*, pages 191–200. Australian Computer Society, Inc. 37

[65] Hassine, A. B., Matsubara, S., and Ishida, T. (2006). A constraint-based approach to horizontal web. *Language*, 147:130–143. 36

[66] Hately, A. and Clement, L. (2008). UDDI Spec TC UDDI Version 3 . 0 . 2 UDDI Spec Technical Committee Draft. pages 1–396. 6, 32

[67] Hatzi, O., Vrakas, D., and Bassiliades, N. (2010). The PORSCE II Framework : Using AI Planning for Automated Semantic Web Service Composition. *The Knowledge Engineering Review, Cambridge University Press*, 01:1–17. 4, 34, 97, 121, 181, 182

[68] Hioual, O. and Boufaida, Z. (2011). An Agent Based Architecture (Using Planning) for Dynamic and Semantic Web Services Composition In an EBXML Context. *International Journal of Database Management Systems*, 3(1):110–131. 6, 56

[69] Hoffmann, J. (2000). A Heuristic for Domain Independent Planning and its Use in an Enforced Hill-climbing Algorithm. In *Proceedings of the 14th Workshop on*

Planning and Configuration - New Results in Planning, Scheduling and Design. 28

[70] Hoffmann, J. (2001). FF: The Fast-Forward Planning System. Technical report, Albert Ludwigs University. 44, 46, 47

[71] Hoffmann, J. (2003). Towards Efficient Belief Update for Planning-Based Web Service Composition. Technical report, Albert Ludwigs University. 48, 122

[72] Hoffmann, J. and Brafman, R. I. (2006). Conformant planning via heuristic forward search: A new approach. *Artificial Intelligence*, 170(6-7):507–541. 72, 73

[73] Hu, M. (2003). Web services composition , partition , and quality of service in distributed system integration and re-engineering. *XML Conference*, pages 1–9. 6, 63, 64

[74] Ilghami, O., Nau, D. S., and Munoz-Avila, H. (2006). Learning to do htn planning. In *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling*, pages 390 – 393. 29

[75] Jeffrey, O. and David, M. (2003). The Vision of. *System*, (January):41–50. 2

[76] Jiang, W., Zhang, C., Huang, Z., Chen, M., Hu, S., and Liu, Z. (2010). Qsynth: A tool for qos-aware automatic service composition. In *Web Services (ICWS), 2010 IEEE International Conference on*, pages 42–49. IEEE. 27, 123

[77] Johnsen, F. T. and Gagnes, T. (2010). Agility and Interoperability for Semantic Service Discovery for Interoperability in Tactical Military Networks. *Control*, 4(1). 1

[78] Jorg Hoffmann, Marco Pistore, P. B. (2009). Message-BasedWeb Service Composition, Integrity Constraints, and Planning under Uncertainty: A New Connec-

tion. *Journal of Aritificial Intelligence Research (JAIR)*, 35(1):49–117. 73, 76, 120

[79] Karunamurthy, R., Khendek, F., and Glitho, R. H. (2012). A novel architecture for web service composition. *Journal of Network and Computer Applications*, 35(2):787–802. 134

[80] Kautz, H. and Selman, B. (1998). The Role of Domain-Specific Knowledge in the Planning as Satisfiability Framework. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. 50

[81] Kautz, H. and Selman, B. (1999). Unifying SAT-based and Graph-based plan. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. 50

[82] Khadka, R. and Sapkota, B. (2010). An evaluation of dynamic web service composition approaches. In van, M. S. and Sapkota, B., editors, *4th International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing - ACT4SOC 2010*, pages 67–79, Portugal. SciTePress. 19, 79

[83] Kim, J. and Gil, Y. (2003). Towards interactive composition of semantic web services. *ISWC 2003*, pages 61–62. 80

[84] Klusch, M. (2008). *Semantic Web Service Description*, volume 16, pages 31–57. Birkhuser Basel. 7, 12, 27, 49, 69, 122, 132, 180

[85] Klusch, M. and Gerber, A. (2006). Fast Composition Planning of OWL-S Services and Application. In *Proceedings of the 4th IEEE European Conference on Web Services (ECOWS), Zurich, Switzerland*, pages 181–190. 10, 35, 47, 69, 122

[86] Klusch, M., Gerber, A., Schmidt, M., and Gmbh, D. (2005). Semantic Web Service Composition Planning with OWLS-Xplan. *German Research*, pages 55—-62. 12, 34, 35, 126, 132

[87] Klusch, M., Kapahnke, P., and Fries, B. (2008). Hybrid semantic web service retrieval: A case study with owls-mx. 143, 188

[88] Klusch, M. and Kaufer, F. (2008). WSMO-MX : A Hybrid Semantic Web Service Matchmaker. *German Research*, 5:1–5. 12, 19, 35, 72, 97, 122, 132

[89] Koehler, J. (2006). AI for Service Composition Organization. *Information Sciences*. 4, 56, 57, 120

[90] Kokash, Natallia, B. (2007). Web Service Discovery Based on Past User Experience. *Culture*. 122

[91] Küster, U., König-ries, B., Jena, F.-s.-u., Jena, D., Birgitta-koenig-riesuni jenade, U. K., and Klusch, M. (2009). Evaluating Semantic Web Service Technologies : Criteria , Approaches and Challenges. (1992). 129, 130

[92] Kuter, U., Sirin, E., Parsia, B., Nau, D., and Hendler, J. (2005). Information gathering during planning for Web Service composition. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):183–205. 4, 34, 49, 80

[93] Labovitz, C., Iekel-Johnson, S., McPherson, D., Oberheide, J., and Jahanian, F. (2010). Internet inter-domain traffic. *ACM SIGCOMM Computer Communication Review*, 40(4):75. 1

[94] Lcu, F. (2009). *Optimizing QoS-Aware SemanticWeb Service Composition*, volume 5823, pages 375–391–391. Springer Berlin Heidelberg. 10

[95] Lcu, F., Delteil, A., and Lger, A. (2008). Towards the composition of stateful and independent semantic web services. *Proceedings of the 2008 ACM symposium on Applied computing SAC 08*, page 2279. 6, 100

[96] Lecue, F. and Leger, A. (2008). Causal link matrix and ai planning: A model for web service composition. Technical report, University Pierre and Marie Curie, Paris, France. 21

[97] Lecue, F. L. F. and Leger, A. L. A. (2006). Semantic web service composition based on a closed world assumption. 6, 101

[98] Lenhart, A. and Madden, M. (2007). *Social networking websites and teens: An overview*. Pew/Internet. 17

[99] Li, C. and Bernoff, J. (2011). *Groundswell: Winning in a world transformed by social technologies*. Harvard Business Press. 1

[100] Limthanmaphon, B. and Zhang, Y. (2003). Web Service Composition with Case-Based Reasoning. 17. 24, 25

[101] Lin, N., Kuter, U., and Sirin, E. (2005). Web Service Composition with User Preferences. *Networks*. 8, 34, 120

[102] Littlejohn, A. and Pegler, C. (2007). *Preparing for blended e-learning*. Routledge. 17

[103] Lovsz, L. (2005). Review of the book by alexander schrijver: Combinatorial optimization: Polyhedra and efficiency. *Operations Research Letters*, 33(4):437–440. 36, 37

[104] M. Klusch, A. G. and 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, Arlington VA, USA, pages 55–62. 12, 48, 49, 132, 180

[105] Madhu, G., Govardhan, a., and Rajinikanth, T. (2011). Intelligent Semantic Web Search Engines: A Brief Survey. *International journal of Web & Semantic Technology*, 2(1):34–42. 19, 189

[106] Majithia, S. and Taylor (2005). Triana: A Graphical Web Service Composition and Execution Toolkit. 78

[107] Majithia, S., Walker, D. W., and Gray, W. A. (2004). Automated web service composition using semantic web technologies. *International Conference on Autonomic Computing 2004 Proceedings*, pages 306–308. 80, 120

[108] Man, L., Dazhi, W., and Du Xiaoyong, W. S. (2005). Dynamic composition of web services based on domain ontology. *Chinese Journal of Computers*, 28(4):644–650. 26

[109] Martin, D., Burstein, M., Mcdermott, D., Mcilraith, S., Paolucci, M., Sycara, K., Mcguinness, D. L., Sirin, E., and Srinivasan, N. (2007). Bringing semantics to web services with owl-s. *World Wide Web*, 10(3):243–277. 21, 22

[110] Martinez, E. and Lesprance, Y. (2004). Web service composition as a planning task: Experiments using knowledge-based planning. In *Proceedings of the ICAPS-2004 Workshop on Planning and Scheduling for Web and Grid Services*, pages 62 – 69, Whistler, BC. 6

[111] Matthews, D. B. (2010). Semantic web technologies. Technical report, CCLRC Rutherford Appleton Laboratory, ISC Technology and Standards Watch, Ingenta. 20, 21

[112] Maximilien, E. M. and Singh, M. P. (2004). Toward autonomic web services trust and selection. *Proceedings of the 2nd international conference on Service oriented computing ICSOC 04*, page 212. 19, 111

[113] McCluskey, T. L. (2000). Object Transition Sequences: A New Form of Abstraction for HTN Planners. In *The Fifth International Conference on Artificial Intelligence Planning Systems*. 29

[114] McCluskey, T. L. and Kitchin, D. E. (1998). A Tool-Supported Approach to Engineering HTN Planning Models. In *Proceedings of 10th IEEE International Conference on Tools with Artificial Intelligence*. 48

[115] McCluskey, T. L. and Porteous, J. M. (1996). Planning Speed-up via Domain Model Compilation. In Ghallab, M. and Milani, A., editors, *New Directions in AI Planning*, pages 233–244 . IOS Press. 122

[116] McIlraith, S. and Son, T. C. (2002). Adapting Golog for Composition of Semantic Web Services. In *Proceedings of the 8th International Conference on Knowledge Representation and Reasoning (KR2002)*. 6

[117] McIlraith, S. A., Son, T. C., and Zeng, H. (2001). Semantic web services. *IEEE Intelligent Systems*, 16:46–53. 60

[118] Medjahed, B. (2004). Semantic Web Enabled Composition of Web Services Semantic Web Enabled Composition of Web Services Brahim Medjahed. *Communities*. 61

[119] Meyer, H. and Weske, M. (2009). Automated services composition using heuristic search. Technical report, University of Potsdam, Germany. 46, 50, 54, 74

[120] Milanovic, N. and Malek, M. (2004). Current solutions for web service composition. *IEEE Internet Computing*, 8(6):51–59. 19

[121] Mrissa, M., Ghedira, C., Benslimane, D., Maamar, Z., Rosenberg, F., and Dustdar, S. (2007). A context-based mediation approach to compose semantic web services. *ACM Trans. Internet Technol.*, 8. 18

[122] Mumford, L. (2010). *Technics and civilization*. University of Chicago Press. 15

[123] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT Press. 73

[124] Mustafa, F. and McCluskey, T. L. (2008). Dynamic web services composition: current issues. In *Proceedings of Computing and Engineering Annual Researchers' Conference 2008: CEARC08*, pages 48–54, Huddersfield,UK. University of Huddersfield. 29, 41, 42, 57, 63, 86, 87, 120

[125] Mustafa, F. and McCluskey, T. L. (2009). Dynamic web service composition. In *Proceedings of the 2009 International Conference on Computer Engineering and Technology - Volume 02*, ICCET '09, pages 463–467, Washington, DC, USA. IEEE Computer Society. 49, 62, 63, 64, 65, 83, 84, 85, 86, 137, 183, 184

[126] Nau, D., Cao, Y., Lotem, A., and Munoz-Avila, H. (2000). SHOP and M-SHOP: Planning with Ordered Task Decomposition. Tech. Report CS TR 4157, University of Maryland, College Park, MD. 8, 48, 49, 97

[127] Newcomer, E. (2002). *Understanding Web Services: XML, WSDL, SOAP, and UDDI*. Addison-Wesley Professional. 32, 64

[128] Oh, S.-C., Lee, D., and Kumara, S. R. (2008). Effective Web Service Composi-

tion in Diverse and Large-Scale Service Networks. *IEEE Transactions on Services Computing*, 1(1):15–32. 129

[129] O.Hioual and Z.Boufaida (2011). An agent based architecture (usingplanning) for dynamic and semantic web services composition in an ebxmlcontext. volume 3, pages 110–131. IJDMS. 6

[130] Padhy, R. P., Patra, M. R., and Satapathy, S. C. (2011). X-as-a-Service: Cloud Computing with Google App Engine, Amazon Web Services, Microsoft Azure and Force.com. *Computer Science and Telecommunications*, 2(9). 19, 111

[131] Paolucci, M. and Srinivasan, K. S. N. (2005). Semantic web service discovery in the owl-s ide. Technical report, Carnegie Mellon University. 40

[132] Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. (2007). Service-Oriented Computing: State of the Art and Research Challenges. *Computer*, 40(11):38–45. 18

[133] Patil, A., Oundhakar, S., Sheth, A., and Verma, K. (2004). Meteor-s web service annotation framework. In *In Proceedings of the 13th International Conference on the World Wide Web*, pages 553–562. ACM Press. 79

[134] Paul, S., Pan, J., and Jain, R. (2011). Architectures for the future networks and the next generation internet: A survey. *Computer Communications*, 34(1):2–42. 1, 189

[135] Pautasso, C. and Alonso, G. (2005). *From web service composition to megaprogramming*, volume 3324, pages 39–53. Springer-Verlag Berlin. 10

[136] Pautasso, C., Zimmermann, O., and Leymann, F. (2008). Restful web services vs. big'web services: making the right architectural decision. In *Proceedings of the 17th international conference on World Wide Web*, pages 805–814. ACM. 23

[137] Peer, J. (2004). A PDDL Based Tool for Automatic Web Service Composition. *Language*. 4

[138] Peer, J. (2005a). Web service composition as ai planning - a survey. Technical report, University of St. Gallen. 4, 122, 189

[139] Peer, J. (2005b). Web Service Composition as AI Planning a Survey ? *Language*, (March). 45, 49, 54, 61

[140] Pinninck, A. P. D., Dupplaw, D., Kotoulas, S., and Siebes, R. (2007). The OpenKnowledge Kernel. pages 376–381. 1

[141] Pistore, M., Roberti, P., and Traverso, P. (2005). *Process-level composition of executable web services: "On-the-fly" versus "once-for-all" composition*, volume 3532, pages 62–77. SPRINGER-VERLAG BERLIN. 101

[142] Ponnekanti, S. R. and Fox, A. (2002). SWORD: A Developer Toolkit for Web Service Composition . Technical report, Computer Science dept. Stanford university. 22, 48, 81

[143] Qiu, L., Lin, F., Wan, C., and Shi, Z. (2006). Semantic web services composition using ai planning of description logics. In *Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing*, pages 340–347, Washington, DC, USA. IEEE Computer Society. 56, 60, 122

[144] Rao, J. and Su, X. (2005). A survey of automated web service compositions methods. Technical report, Norwegian University of Science and Technology, Norway. 2, 8, 42, 189

[145] Rehan, M. and Akyuz, G. A. (2010). Enterprise Application Integration (EAI), Service Oriented Architectures (SOA) and their relevance to e- supply chain formation. *Journal of Business*, 4(13):2604–2614. 19

[146] Rintanen, J. (1998). A planning algorithm not based on directional search. In *Proceedings of the 6th International Conference on Principles of Knowledge Representation*. 50

[147] Rodriguez-Mier, P., Mucientes, M., and Lama, M. (2011). Automatic web service composition with a heuristic-based search algorithm. 6

[148] Russell, S. and Norvig, P. (2002). *Artificial Intelligence: A Modren Approach*. Prentice Hall. 48

[149] Sánchez, J., Salinas, A., and Sáenz, M. (2007). Mobile game-based methodology for science learning. In *Proceedings of the 12th international conference on Human-computer interaction: applications and services*, HCI'07, pages 322–331, Berlin, Heidelberg. Springer-Verlag. 18

[150] Schmeling, B., Charfi, A., and Mezini, M. (2011). Composing non-functional concerns in composite web services. 134

[151] Sequeda, J. (2009). Towards a Query Language for the Web of Data (A Vision Paper). *Processing*. 20

[152] Sheng, Q. Z., Maamar, Z., Yu, J., and Ngu, A. H. H. (2010a). Robust Web Services Provisioning Through On-Demand Replication. *Structure*. 61, 62

[153] Sheng, Q. Z., Yu, J., and Dustdar, S. (2010b). *Enabling context-aware web services: methods, architectures, and technologies*. CRC Press/Taylor & Francis. 13, 125

[154] Simmonds, J. (2011). PhD thesis. Dynamic Analysis of Web Services. 62

[155] Sirin, E., Hendler, J., and Parsia, B. (2003). Semi-automatic composition of web services using semantic descriptions. *WSMAI*, 3:17–24. 22

[156] Sirin, E., Parsia, B., and Hendler, J. (2004). Template-based Composition of Semantic Web Services. 70, 122

[157] Sohrabi, S., Baier, J. A., and McIlraith, S. A. (2009a). Htn planning with preferences. 12, 28, 35, 97, 101, 112

[158] Sohrabi, S., Baier, J. A., and McIlraith, S. A. (2011). Preferred explanations: Theory and generation via planning. In Burgard, W. and Roth, D., editors, *AAAI*. AAAI Press. 2, 10, 34, 70, 125, 129, 184

[159] Sohrabi, S., Prokoshyna, N., and Mcilraith, S. A. (2009b). Web service composition via the customization of golog programs with user preferences. In *Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos*, pages 319–334. Springer-Verlag. 8, 112

[160] Sollazzo, T., Handschuh, S., Staab, S., Frank, M., and Stojanovic, N. (2002). Semantic Web Service Architecture Evolving Web Service Standards toward the Semantic Web. *Information Sciences*, (iii):425–429. 61

[161] Srinivasan, N., Paolucci, M., and Sycara, K. (2004). Adding owl-s to uddi, implementation and throughput. *Proceedings of Semantic Web Service and Web Process Composition*. 25

[162] Srivastava, B. and Kambhampati, S. (1997). A Structured Approach for Synthesizing Planners from specifications. In *Proceedings of the 12th IEEE International Conference on Automated Software Engineering*. 100

[163] Srivastava, B. and Koehler, J. (2003). *Web service composition-current solutions and open problems*, volume 35, pages 28–35. Citeseer. 2, 6, 85, 100, 101

[164] Strunk, A. (2010). Qos-aware service composition: A survey. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 67–74. IEEE. 60, 123

[165] Sullivan, A. R. . J. (2003). The Pragmatic Web: Agent-Based Multimodal Web Interaction with no Browser in Sight . Technical report, Computer Science Department,University of Colorado. 20

[166] Sycara, K., Paolucci, M., Ankolekar, A., and Srinivasan, N. (2003). Automated discovery, interaction and composition of semantic web services. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1):27–46. 22

[167] Tidd, J. and Bessant, J. (2011). *Managing innovation: integrating technological, market and organizational change*. John Wiley & Sons. 1

[168] Toch, E., Gal, A., Reinhartz-Berger, I., and Dori, D. (2007). A semantic approach to approximate service retrieval. *ACM Transactions on Internet Technology*, 8(1):2–es. 19

[169] Tompkins, J. A. (2010). *Facilities planning*. John Wiley & Sons. 60, 123

[170] Tsalgatidou, A. and Pilioura, T. (2002). An overview of standards and related technology in web services. *Distributed and Parallel Databases*, 12(2-3):135–162. 24

[171] Urbieta, A., Barrutieta, G., Parra, J., and Uribarren, A. (2008). A survey of dynamic service composition approaches for ambient systems. In *Proceedings of the 2008 Ambi-Sys workshop on Software Organisation and MonIToring of Ambient Systems*, SOMITAS '08, pages 1:1–1:8, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 97, 189

[172] Van Der Aalst, W. M., Ter Hofstede, A. H., and Weske, M. (2003). Business process management: A survey. In *Business process management*, pages 1–12. Springer. 21

[173] Vandermeer, D., Inst, G., and Thomas, H. (2005). FUSION : A System Allowing Dynamic Web Service Composition and Automatic Execution College of Computing. 77, 78

[174] Verma, K., Gomadam, K., Sheth, A. P., Miller, J. A., and Wu, Z. (2005). The meteor-s approach for configuring and executing dynamic web processes. Technical report, University of Georgia. 79

[175] Vipul Kashyap, Christoph Bussler, M. M. (2008). *The Semantic Web: semantics for data and services on the Web*. Springer ISBN 978-3-540-76452-6. 25, 26

[176] Waldvogel, M. and Bauer, D. (2007). A Flexible Middleware for Multimedia Communication : Design , Implementation , and Experience. 17(1999):1–18. 18, 57

[177] Wang, H. (2004). Web services: problems and future directions. *Web Semantics Science Services and Agents on the World Wide Web*, 1(3):309–320. 7, 8

[178] Weerawarana, S., Curbera, F., Leymann, F., Storey, T., and Ferguson, D. F. (2005). *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more*. Prentice Hall PTR. 23, 27, 39

[179] Wei, Y. and Blake, M. B. (2010). Service-oriented computing and cloud computing: Challenges and opportunities. *IEEE Internet Computing*, 14(6). 26, 27, 123

[180] Wellman, B. and Haythornthwaite, C. (2008). *The Internet in everyday life*. John Wiley & Sons. 17

[181] Wu, D., Sirin, E., Hendler, J., Nau, D., and Parsia, B. (2003). Automatic web services composition using shop2. In *Workshop on Planning for Web Services*. 28, 29

[182] Xiong, P., Fan, Y., and Zhou, M. (2010). A petri net approach to analysis and composition of web services. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 40(2):376–387. 26

[183] Yang, Q. (1997). *Intelligent Planning: A Decomposition and Abstraction Based Approach*. Springer-Verlag. 8

[184] Yue, P., Di, L., Yang, W., Yu, G., and Zhao, P. (2007). Semantics-based automatic composition of geospatial web service chains. *Computers & Geosciences*, 33(5):649–665. 60

[185] Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q. Z., Votano, J., Parham, M., and Hall, L. (2003). Quality driven web services composition. *Proceedings of the twelfth international conference on World Wide Web WWW 03*, 1(11):411. 5

[186] Zhang, J. F. (2009). PhD thesis. Distributed Graph-Based Multi-Agent Planning. 37, 48, 51

[187] Zhang, L. and Manocha, D. (2008). An Efficient Retraction-based RRT Planner. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3743–3750. 23

[188] Ziaka, E. and Bassiliades, N. (2008). Translating web services composition plans to owl-s descriptions. Technical report, University of Thessaloniki. 6, 12, 28, 49

[189] Zou, G., Chen, Y., Xiang, Y., Huang, R., and Xu, Y. (2010). Ai planning and combinatorial optimization for web service composition in cloud computing. *Proceedings of the International Conference on Cloud Computing Virtualization 2010 CCV 2010*, pages 28–35. 36

[190] Zozaya-Gorostiza, C. (2012). *Knowledge-based process planning for construction and manufacturing*. Elsevier. 44