VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

The Title

Alexandre Sawczuk da Silva

Supervisors: Dr. Hui Ma, Prof. Mengjie Zhang

December 31, 2014

Submitted in partial fulfilment of the requirements for
PhD.

Abstract

A short description of the project goes here.

Contents

Figures

Chapter 1

Introduction

1.1 Problem Statement

As applications increasingly interact with the Web, the concept of Service-Oriented Architecture (SOA) [14] emerges as a popular solution. The key components of SOA are often Web services, which are functionality modules that provide operations accessible over the network via a standard communication protocol [6]. One of the greatest strengths of Web services is their modularity, because it allows the reuse of independent services that provide a desired operation as opposed to having to re-implement that functionality. The combination of multiple modular Web services to achieve a single, more complex task is known as *Web service composition*. At a basic level, services are combined according to the functionality they provide, i.e. the inputs required by their operations and the outputs produced after execution. Several services may be connected to each other, with the outputs of a service satisfying the inputs of the next, starting from the composition's given overall input and finally leading to the composition's required output.

An example of an application of Web service composition is ENVISION, an EU-funded project whose aim is to provide a framework for discovering and composing Web services that perform geospatial analysis on data, thus enabling environmental information to be more easily processed for research and decision-making purposes [11]. ENVISION is meant to be used by scientists who are experienced with geographic models but do not have a technical computing background, therefore the motivation of this work was to create a solution that is as simple to use as possible. The system offers a way to search for (discover) services that provide environmental data as well as processing services by using a word-based and coordinate-based search. Users create compositions by manually selecting and assembling a Business Process Model (BPM), which is later transformed into an engine-runnable representation. The environmental data and processing applications are packaged as services, meaning that they are easily available for reuse and thus contribute to the faster identification of important environmental trends.

Despite the usefulness of Web service composition, conducting such a process manually is fraught with difficulties. In order to illustrate these problems, an experiment was performed in which students with a good knowledge of programming and of Web services were asked to manually create Web service compositions to address a range of well defined real-world problems [9]. Results show that they faced a number of difficulties at different composition stages, from the discovery of potential services to their combination. During the discovery phase, the most popular search tools used by students were Web service portals and generic search engines. Authors highlighted that not a single discovery tool from the literature was used, because no single solution offers a large variety of services. This suggests that larger standardised service portals should be created, provided that the quality of the services offered is maintained. During the combination phase, students faced discrepancies between the concepts used by the interfaces of services: the terms used in the interfaces of any two services are often different from each other, even though those services may handle the exact same domain. Another problem is that the services used in a composition may produce too much data, or incur too much latency, meaning that the final composition is slowed down as a result. Standardising and automating the service composition process would eliminate the need for manually handling these difficulties, and thus developing a system capable of creating compositions in a fully automated manner is one of the holy grails of the field [12].

Automated Web service composition is complex, and in fact it is considered an *NP-hard problem*, meaning that the solution to large tasks is not likely to be found with reasonable computation times [13]. Due to this complexity, a variety of strategies have been investigated in the literature, focusing on two fundamental composition approaches: *workflow-based approaches*, where the central idea is that the user provides an abstract business process which is to be completed using concrete services, and *AI planning-based approaches*, where this abstract process is not typically required and instead the focus is on discovering the connections between services that lead from the task's provided output to its desired [13]. In workflow-based approaches, the abstract functionality steps necessary to accomplish the task requested by the user have typically already been provided, so the objective is to select the concrete services with the best possible quality to fulfil each workflow step. The advantage of such approaches is that the selection of services may easily be translated into an optimisation problem where the objective is to achieve the best overall composition quality. This optimisation is often performed using *evolutionary computation* techniques. However, the disadvantage is that a workflow must have been already defined, which likely means that it has to be manually designed. Planning-based approaches, on the other hand, have the advantage of both determining the workflow to be used in the composition and selecting services to be used at each step, abiding by user constraints. The disadvantage of such approaches is that they are typically unable to also perform quality-based optimisation on the selected services.

The overall goal of this thesis is to propose a Web service composition approach that hybridises elements of AI planning-based approaches and evolutionary computation workflow-based approaches, enabling the construction of a workflow according to user constraints as well as the optimisation of that workflow according to the quality of its composing services.

1.2 Motivations

The intricacy of Web service composition lies in the number of distinct dimensions it must simultaneously account for. On the first dimension, services must be combined so that their operation inputs and outputs are properly linked, i.e. the output produced by a given service is usable as input by the next services in the composition, eventually leading up to the desired overall output. On the second dimension, the composition must meet any specified user constraint or preference. A constraint is defined as a user restriction that must be met in order for a composition solution to be considered valid, and this mainly concerns execution flow features (e.g. the composition must have multiple execution options – branches – according to a given condition) [18, 16, 7]. A preference, on the other hand, is a user restriction that would be desirable to observe, even though a composition solution is still considered valid if this restriction is not met (e.g. between two services with similar functionality, a user would always prioritise the use of one service over the other in a composition) [18]. It must be noted that constraints relating to Quality of Service (QoS) values are not included in this dimension. On the third dimension, the resulting composition must achieve the best possible overall Quality of Service (QoS) with regards to attributes such as the time required to execute the composite services, the financial cost of utilising the service modules, and the reliability of those modules.

User constraints and preferences are a common requirement when performing Web service composition. For example, consider the case of a user who wishes to book transportation and accommodation for a trip, including transportation from the airport to the hotel, a room at the hotel for a given number of days, and transportation back to the airport [3]. In this scenario, the user is likely to have constraints on the attributes of a given service (e.g. the hotel booking service must either be that of a two-star or that of a three-star hotel), as well as conditional constraints (e.g. if the hotel booking service is for a two-star hotel, then the transportation to and from the aiport should be arranged using a taxi service, otherwise a shuttle service should be used). Different techniques have been explored to achieve compositions that consider such constraints, including the use of AI planning with rules encoding user constraints [2] and the representation of the composition as a constraint satisfaction problem to be processed with a solver engine [7]. However, this territory has not been widely explored by applying evolutionary computation techniques (a GA approach for composition that considers inter-service relationships has been proposed [21], even though no actual examples of these relationships were presented). Due to the flexibility and efficiency of these techniques, it would be interesting to focus on the investigation of ways in which to extend them to apply these constraints.

Several techniques have been proposed to address the composition problem, such as variations of AI planning [4], Evolutionary Computation (EC) techniques [17], and hybrid optimisation algorithms [15]. These approaches produce promising results, however the great majority of them only account for two composition dimensions at once. For example, AI planning techniques for composition focus on guaranteeing functional correctness (first dimension) and fulfilment of constraints (second dimension), while EC techniques such as Genetic Algorithms (GA) and Genetic Programming (GP) focus on QoS (third dimension) in addition to functional correctness (first dimension) but do not include conditional branches (second dimension).

1.3 Research Goals

Thus, the objective of this work is to propose a Web service composition approach that simultaneously considers elements from all the three dimensions described above when generating solutions. This approach employs Genetic Programming (GP) for evolving a population of near-optimised solutions, at the same time restricting the structure of candidates according to functional correctness and user constraints. Specifically, each dimension is addressed as follows:

- **First dimension (functional correctness):** The solutions are represented as trees where the way in which services are linked to each other is restricted to preserve functionality (more details in Subsection **??**).

- **Second dimension (user constraints):** A branching conditional constraint is included as one of the possible nodes in the tree representation of solutions, thus also enabling the enforcement of user constraints (see Subsection **??**).

- **Third dimension (Quality of Service):** A fitness function is used to optimise candidate solutions with regards to their overall QoS attributes (see Subsection **??**).

The research questions to answer:

i) How to create a representation that allows for compositions with branches to be optimised, assuming that users have provided the branching conditions?

ii) How to extend this representation so that users do not have to provide branching conditions, only multiple outputs?

iii) Can these representation be used in conjunction with multi-objective optimisation techniques?

Set of objectives to find answers to these research goals:

1)

1.4 Organisation of Proposal

Chapter 2

Literature Review

A workflow-based Web service composition solution can be decomposed into a series of steps [13], as shown in Figure 2.1 and discussed below:

1. *Goal specification:* The user's goal and preferences for the composition are specified. An abstract workflow is generated, showing details about data flow and functionality, and the QoS requirements are determined based on the user's information.

2. *Service discovery:* Candidate concrete services that are functionally and non-functionally suitable to fill the slots in the abstract workflow are discovered in a service repository. At this stage, candidates have varying quality levels.

3. *Service selection:* A technique is employed to find which discovered services best fulfil each slot in the abstract workflow specified earlier, and a concrete Web service composition is generated.

4. *Service execution:* An instance of the concrete composition generated above is made executed.

5. *Service maintenance and monitoring:* The created instance is constantly monitored for failures and/or changes to the composing atomic services.

It is important to draw a distinction between semi-automated and fully automated composition approaches. Fully automated is ..., while semi-automated is ... [A survey of automated web service composition methods]. This work focuses on performing fully automated composition, but not really on service discovery, maintenance and monitoring.

The following types of constraints and user preferences have been presented in literature:

- Conditional branch structures that reflect user preferences [18]. The paper covers two types of user preferences that require branching. The first type is when the user prefers one service instead of another according to a condition (i.e. the if-else construct – note that this construct eventually closes into a diamond, and only one output is produced). The second type is when the user specifies a list of services with similar functonalities ranked according to personal preferences. If the service with the highest priority fails, then the service with the second-highest priority will be executed (e.g. PayWithCard service with higher priority, PayInCash service with lower priority).

Figure 2.1: Typical steps in a workflow-based automated Web service composition solution.

- Preferences specified using a logic language (based on linear temporal logic) [16]. For example, specify that you do not want to book a Hilton hotel, but you want a 3-star hotel paid using a credit card (this language requires relatively detailed semantic information about services). Another example: prefers not book air ticket until the hotel has been booked (order of service execution).
- Hard constraints on services: service properties (i.e. service must have a specific property with a specific value) and coreography details (essentially if-else constraints that close into a diamond) [3].
- Preferences specified using the Knowledge Interchange Format (KIF) language, as it provides a well-defined syntax and semantic that can be applied to constraints [5]. The types of preferences that can be defined are the same as those in [16].
- Temporal and causality constraints: Constraints on the flow structure of the composition (e.g. if service 1 executes, then service 2 must also execute) [7].
- Logical constraints: any logical expressions on integer or string values, e.g. Hotel should be in Barcelona and the cost should be under $450 a day [7].
- If/then constraints: if a constraint X holds, then another constant Y must hold as well. E.g. If a hotel has less than 4 stars, then its cost a day must be under 100 dollars [7].
- Preferences specified using PDDL3 (Planning Domain Definition Language). In PDDL3, preferences are described using logical formulae (i.e. logical statements encoding constraints) and temporal preferences (i.e. how often a constraint must hold as the planning steps through its states) [8].
- Data flow constraints of the composition specified using a visual language. For example, if a room is available book it, otherwise cancel the entire transaction. The basic idea is to check if the data net of constraints is satisfied by the plan presented as the solution, with some heavy formalisation [10].
- Use of a representation that employs domain objects, allowing the specification of control flow requirements and also of logical constraints [1].
- Global constraints based on attributes of a single service (e.g. only one service that costs 50 may be invoked). Solved using ILP [20].
- Constraints on the resources consumed while creating the composition (note they are not part of the solution itself) [19].

Bibliography

[1] Bertoli, P., Kazhamiakin, R., Paolucci, M., Pistore, M., Raik, H., and Wagner, M. Control flow requirements for automated service composition. In *Web Services, 2009. ICWS 2009. IEEE International Conference on* (2009), IEEE, pp. 17–24.

[2] Boustil, A., Maamri, R., and Sahnoun, Z. A semantic selection approach for composite web services using OWL-DL and rules. *Service Oriented Computing and Applications 8*, 3 (2014), 221–238.

[3] Boustil, A., Sabouret, N., and Maamri, R. Web services composition handling user constraints: towards a semantic approach. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services* (2010), ACM, pp. 913–916.

[4] Chen, M., and Yan, Y. Qos-aware service composition over graphplan through graph reachability. In *Services Computing (SCC), 2014 IEEE International Conference on* (2014), IEEE, pp. 544–551.

[5] Gamha, Y., Bennacer, N., Naquet, G., Ayeb, B., and Romdhane, L. B. A framework for the semantic composition of web services handling user constraints. In *Web Services, 2008. ICWS'08. IEEE International Conference on* (2008), IEEE, pp. 228–237.

[6] Gottschalk, K., Graham, S., Kreger, H., and Snell, J. Introduction to web services architecture. *IBM systems Journal 41*, 2 (2002), 170–177.

[7] Karakoc, E., and Senkul, P. Composing semantic web services under constraints. *Expert Systems with Applications 36*, 8 (2009), 11021–11029.

[8] Lin, N., Kuter, U., and Sirin, E. *Web service composition with user preferences*. Springer, 2008.

[9] Lu, J., Yu, Y., Roy, D., and Saha, D. Web service composition: A reality check. In *Web Information Systems Engineering–WISE 2007*. Springer, 2007, pp. 523–532.

[10] Marconi, A., Pistore, M., and Traverso, P. Specifying data-flow requirements for the automated composition of web services. In *Software Engineering and Formal Methods, 2006. SEFM 2006. Fourth IEEE International Conference on* (2006), IEEE, pp. 147–156.

[11] Maué, P., and Roman, D. The envision environmental portal and services infrastructure. In *Environmental Software Systems. Frameworks of eEnvironment*. Springer, 2011, pp. 280–294.

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

[13] MOGHADDAM, M., AND DAVIS, J. G. Service selection in web service composition: A comparative review of existing approaches. In *Web Services Foundations*. Springer, 2014, pp. 321–346.

[14] PERREY, R., AND LYCETT, M. Service-oriented architecture. In *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on* (2003), IEEE, pp. 116–119.

[15] POP, C. B., CHIFU, V. R., SALOMIE, I., AND DINSOREANU, M. Immune-inspired method for selecting the optimal solution in web service composition. In *Resource Discovery*. Springer, 2010, pp. 1–17.

[16] SOHRABI, S., PROKOSHYNA, N., AND MCILRAITH, S. A. Web service composition via the customization of golog programs with user preferences. In *Conceptual Modeling: Foundations and Applications*. Springer, 2009, pp. 319–334.

[17] WANG, L., SHEN, J., AND YONG, J. A survey on bio-inspired algorithms for web service composition. In *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on* (2012), IEEE, pp. 569–574.

[18] WANG, P., DING, Z., JIANG, C., AND ZHOU, M. Automated web service composition supporting conditional branch structures. *Enterprise Information Systems 8*, 1 (2014), 121–146.

[19] XIANG, C., ZHAO, W., TIAN, C., NIE, J., AND ZHANG, J. Qos-aware, optimal and automated service composition with users' constraints. In *e-Business Engineering (ICEBE), 2011 IEEE 8th International Conference on* (2011), IEEE, pp. 223–228.

[20] YOO, J. J.-W., KUMARA, S., LEE, D., AND OH, S.-C. A web service composition framework using integer programming with non-functional objectives and constraints. *algorithms 1* (2008), 7.

[21] ZHANG, Z., ZHENG, S., LI, W., TAN, Y., WU, Z., AND TAN, W. Genetic algorithm for context-aware service composition based on context space model. In *Web Services (ICWS), 2013 IEEE 20th International Conference on* (2013), IEEE, pp. 605–606.