Improving the Quality of Distributed Composite Service Applications

Dionysios Efstathiou[1], Peter McBurney[1], Noël Plouzeau[2], and Steffen Zschaler[1]

1 **Department of Informatics, King's College London**
{dionysios.efstathiou, peter.mcburney, steffen.zschaler}@kcl.ac.uk
2 **IRISA, University of Rennes 1**
noel.plouzeau@irisa.fr

Abstract

Dynamic service composition promotes the on-the-fly creation of value-added applications by combining services. Large scale, dynamic distributed applications, like those in the pervasive computing domain, pose many obstacles to service composition such as mobility, and resource availability. In such environments, a huge number of possible composition configurations may provide the same functionality, but only some of those may exhibit the desirable non-functional qualities (e.g. low battery consumption and response time) or satisfy users' preferences and constraints. The goal of a service composition optimiser is to scan the possible composition plans to detect these that are optimal in some sense (e.g. maximise availability or minimise data latency) with acceptable performance (e.g. relatively fast for the application domain). However, the majority of the proposed optimisation approaches for finding optimal composition plans, examine only the Quality of Service of each participated service in isolation without studying how the services are composed together within the composition. We argue that the consideration of multiple factors when searching for the optimal composition plans, such as which services are selected to participate in the composition, how these services are coordinated, communicate and interact within a composition, may improve the end-to-end quality of composite applications.

1998 ACM Subject Classification C.0 [General]: System architectures

Keywords and phrases Service Composition, Optimisation, Dynamism, Evolution

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.49

1 Introduction

Service-orientation promotes the creation of new value-added applications by composing pre-existing services regardless their location and platform technology [16]. *Service composition* [8, 9] is not only limited to functional composition, but also takes into account *non-functional* issues. Indeed, an application that does not obey the user's Quality of Service (QoS) constraints might be as useless as a service not providing the desired functionality.

Service composition in distributed environments faces four challenges: (a) a large space of possible deployment architectures are possible, (b) satisfaction of multiple users' (possibly conflicting) QoS preferences and constraints, (c) continuous system evolution and high dynamism since the components and their relationships are not known at design-time and may change at run-time, and (d) absence of an entity with global knowledge. These challenges demand a dynamic self-adaptive distributed solution [11, 17] for finding the composition architectures of high-quality that achieve the users' functional and non-functional goals.

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 49–55

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We argue that to achieve high-quality compositions, the composer should take into account not only the quality of each service in isolation, but also how services are coordinated, how they communicate and how they interact within the composition. Despite the vast research on the field of service composition, little is known about how these factors combined affect the aggregation of optimal compositions. The contention of this research is that the combined consideration of these factors may improve the overall quality of the composed applications.

The remainder of this paper is organised as follows. In Section 2 we present a motivating scenario for our research. Section 3 discusses some representative alternative approaches for optimising the process of service composition. Section 4 outlines our preliminary work for enabling the efficient on-the-fly creation of high-quality composite services. Finally, Section 5 concludes by outlining the plans of future work towards achieving our research goals.

2 Motivating Scenario

Our case study is a time-critical continuously evolving application: a fire-fighter tactical information and decision support system. The system's goal is to achieve improved knowledge of the emergency situation for better decision making via efficient sharing of real-time information between people in various hierarchical levels. For example, a commanding officer in an fire-fighting situation wants real-time information about on-field conditions, the availability of resources, and others, to make better decisions for risk identification, resource allocation, and others.

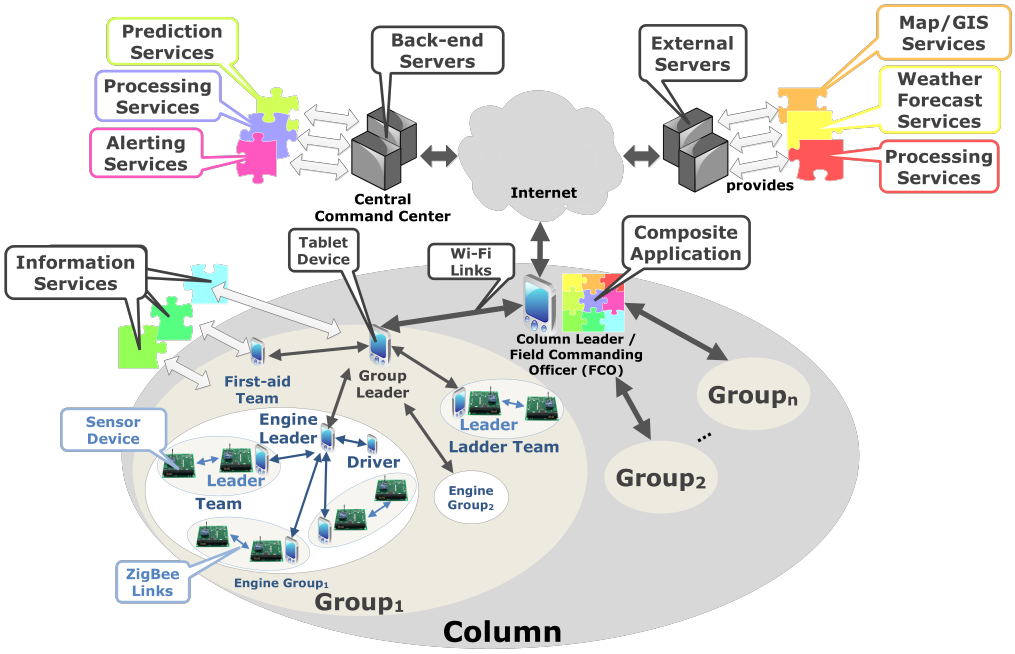

Figure 1 High-level architecture of the fire-fighting application scenario.

Figure 1 presents the groups emerging during an emergency operation which are (in order of hierarchy): team, engine group, group, and column. Each group has a leader, and each leader is equipped with a tablet. Each fire-fighter carries a number of special sensors (e.g. temperature, air quality, GPS module), and a number of sensors are deployed in the field to provide real-time data. The Field Commanding Officer (FCO), who is the highest ranked

person in the field (in our example the Column Leader), combines real-time information about the condition of the fire, local geography, fire evolution prediction services, available resources (e.g. personnel, water pumps), and others, to achieve better decision making. This application is composed of spatially distributed service components hosted on back-end servers and on-field mobile devices that cooperate to achieve a global goal.

Frequent changes of the composition architecture are necessary to deal with system dynamism (e.g. disconnections, battery depletion). Also, the system must continuously adapt to hierarchical and team changes, for example, due to initial danger underestimation there is a need for reinforcements which must be followed by corresponding adaptations (e.g. hierarchical changes, arrival of new services, etc.). Furthermore, the system must take into account that users of different hierarchical levels have different non-functional requirements. For example, the FCO demands to get information from the field in a fast way, while the Group Leader may require to have reliable messaging with all the members of his group.

For a given composite application, there may be many architectures that provide the same functionality, but with different levels of QoS. However, the parameters that influence the quality are not known before system execution. As a result, continuous reconfiguration is necessary to maintain the the high quality of the application during run-time.

3 Related Work

The related work of our research span several areas of the literature: (a) service composition models; (b) QoS-aware service composition; (c) self-adaptation; and (d) optimisation. We summarise the key lessons learned from our literature review that we will use to design our solution approach to the studied problem.

Service Composition Models. In inherently distributed environments, decentralised coordination of services may alleviate the problems (performance bottleneck, single-point-of-failure) [3] of traditional *service orchestration* [9], by enabling the distributed formation of compositions based on dynamic conditions (e.g. battery level, current bandwidth, etc.).

As services can be combined in unpredicted ways, there are many possible configurations to decentralise the coordination of a composite service which raises the question of what is the best way to achieve a decentralised coordination which satisfies a set of non-functional goals. Many researchers studied the problem of how to distribute service orchestration [2, 3, 4, 7, 15]. The inherent distributed and dynamic nature of service-oriented systems indicates the need for an adaptive and automated technique which can dynamically switch between possible orchestration approaches based on run-time conditions.

QoS-Aware Service Composition. In the current literature, optimal service composition is synonymous with the process of optimal QoS-aware service selection [5, 10, 12, 18]. In this problem, the goal is to find the combination of services that offers the required functionality, respects user's QoS requirements, and optimises the overall QoS of the composition. The main limitation of this family of approaches is that they only take into account the QoS of each service in isolation without studying how services are composed together. For instance, consider two services of very good quality (e.g. high availability, low response time). However, when it comes to composing them, the overall QoS of the composite application may be very low due to various reasons (e.g. slow/faulty communication link between interacting services or with the orchestrator that coordinates them).

Self-Adaptation. To realise a self-adaptive composition system, the following questions, as presented by Salehie and Tahvildari [14], need to be answered: (a) *where* the adaptation should occur; (b) *when* the adaptation action(s) should be applied; (c) *what* elements should

be affected; (d) *why* should adaptation be performed (adaptation goals); (e) *who* should be involved in the adaptation process; and (f) *how* the adaptation actions should be realised. The work of Cardellini *et al.* [6] is a first step in the right direction for answering the above questions in the context of dynamic service composition. We motivate the need for an autonomic controller with the goal to adjust the system's configuration on-the-fly based on the current conditions, because the configuration of the composition system is highly sensitive to the dynamic nature of the service-based environment.

Optimisation. Dynamic service composition can be seen as a dynamic multi-objective optimisation problem, where, given a set of services and a set of composition plans, the goal of the optimiser is to find the trade-off configurations that optimise the overall composition based on dynamic functional requirements and multiple, conflicting non-functional objectives. The optimisation algorithms used to solve this problem can be divided into two main categories: exact [12, 19] and approximative [5, 13], based on their precision (how close to optimal) and computational (how fast) complexity. Exact algorithms are able to produce solutions with guaranteed optimality but at exponential cost. On the other hand, approximative algorithms are able to produce sub-optimal solutions of "good" quality at polynomial time complexity. Some of the techniques applied to solve the studied problem are the following: linear and integer programming methods [12, 19], local search techniques [13], evolutionary metaheuristics [5], and others.

Composition Optimisation Frameworks Ardagna and Mirandola in [1] proposed a broker-based framework for run-time QoS-aware composition optimisation. However, their approach focuses only on the optimisation problem of QoS-aware service selection, ignoring the important issues of run-time adaptation, dynamic conditions, and distributed orchestration. Cardellini *et al.* [6] proposed a framework for run-time QoS-driven adaptation of SOA systems. However, their solution ignores the dynamic networking conditions for achieving the optimal composition. At the same time, they ignore the interaction and communication patterns for optimising the exchange of data between the various participating services. Finally, they only consider centralised orchestration, neglecting the fact that coordination of decentralised services may increase the overall composition performance.

4 Towards our Research Goal

Our goal is to supply a decision maker[1] with a set of trade-off composition configurations, and based on some problem-specific knowledge (e.g. QoS preferences, application context, etc.), to choose the optimal one.

As depicted in Figure 2, the composition system receives as input a set of abstract composition plans which describe abstractly the required functional tasks, a set of dynamic user functional and non-functional requirements, and a set of already deployed concrete services. Users insert requests into the composition engine, the front-end of the system. The composition engine identifies which concrete services implement the abstract tasks required by the service composition. Then, the optimisation engine tries to produce optimal composition configurations to realise user's goals. These configurations are provided to the decision maker agent who is responsible for optimising users' profit based on their dynamic preferences and constraints.

After a composite application is deployed, the monitoring component supervises the performance of the participating services and checks periodically whether it is necessary to

[1] An automatic agent whose goal is to maximise the user's profit.

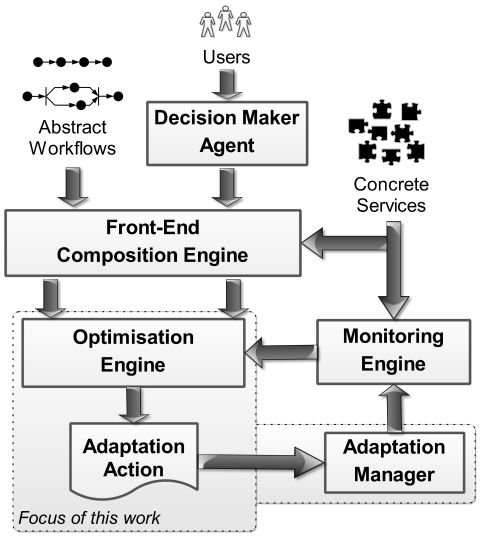

Figure 2 High-level Architecture and Focus of Our Research.

trigger a reconfiguration action by checking the quality of the composition. If an adaptation is triggered, the composition configuration will be dynamically updated according to the reconfiguration policy. The adaptation manager is responsible for implementing/applying the selected strategy. A simple adaptive behaviour of the composition system is to replace dynamically the bad performing services with better ones.

4.1 Scope of Research

The scope of our work is limited to finding, and maintaining a high-quality composition architecture based on users' dynamic preferences and real-time conditions. This means that other optimisation actions that may achieve the same goal, such as reconfiguring the amount of resources reserved for each service, replicating services, and others, are not considered in this study. Also, we do not focus on how to discover services in a distributed service environment. In our use-case scenario, this assumption can be justified by the fact that the descriptions (not the actual implementation) of the various services is known in advance.

4.2 Degrees of Freedom

The first step for optimising the composition process is to identify the most interesting degrees of freedom that enable us to change the provided quality of the composite applications without modifying their functionality. According to our literature review, we identified the following freedom variables: (a) how to decentralise the coordination of the service composition, (b) how to choose the communication patterns between interacting parties in a composition, and (c) how to select the concrete services for participating in the composition (see QoS-Aware Service Composition in Section 3).

As mentioned previously, there are many ways to decentralise the coordination of services in a distributed environment (e.g. how many distributed orchestrators to choose, where to place them, and others). Indeed, the way of realising the coordination of distributed services affects the overall quality of the composite application. Secondly, the various possible communication and interaction patterns between the participated services create opportunities for further improving the quality of the composition. For example, a Team

Leader A wants to send some data to his Group Leader. However, due to physical obstacles (e.g. in a building emergency scenario), the communication link between them is slow and faulty. Another Team Leader B passes nearby A that has a better connection with the Group Leader. Thus, it is better to forward A's data through B.

While there is a lot of work for service composition optimisation, to the best of our knowledge, there is no approach that takes into account simultaneously the above degrees of freedom for providing high-quality service compositions configurations at run-time.

4.3 Choosing an Optimisation Technique

The process of choosing the ideal technique is itself a multi-objective problem. Guided by our use-case scenario, we focus on approximate algorithms, and especially on the promising field of optimisation metaheuristics, such as Evolutionary Algorithms [20], that can provide "good enough" solutions in polynomial time, rather than solving the problem to true optimality.

5 Conclusion and Future Plans

Configuring composite applications of high quality that respect users' conflicting QoS objectives in the context of a continuously evolving distributed service-oriented environment, is a highly challenging research problem that requires deep investigation. Existing optimisation approaches focus only on the QoS of the participated services in isolation and ignore the existence of multiple factors that may affect the end-to-end quality of the composite application. In this paper, we proposed the consideration of multiple degrees of freedom when optimising the overall quality of a composite application. These degrees of freedom include the following: how services are selected to form a composition, how these distributed services are coordinated, how they communicate and interact with each other.

Further work needs to be done to establish whether the consideration of multiple factors leads to composite applications of improved quality. The next step towards achieving our goals is to design the meta-model that captures the main concepts of our service composition domain and abstracts from low-level details. The designed meta-model will enable the generation of model instances that represent possible composition plans. After generating the set of the possible plans (design space), our goal is to apply and compare various optimisation techniques (exact and approximate) to choose the more suitable approach. Finally, we aim at investigating the trade-off of executing the actual optimisation in a fully centralised way, hierarchical way, or totally distributed way among the network nodes.

References

1 Danilo Ardagna and Raffaela Mirandola. Per-Flow Optimal Service Selection for Web Services Based Processes. *Journal of Systems and Software*, 83(8):1512–1523, 2010.

2 Adam Barker, Jon B. Weissman, and Jano I. van Hemert. Eliminating The Middleman: Peer-to-Peer Dataflow. In *Proceedings of the 17th International Symposium on High Performance Distributed Computing*, pages 55–64, 2008.

3 Boualem Benatallah, Marlon Dumas, and Quan Z. Sheng. Facilitating the Rapid Development and Scalable Orchestration of Composite Web Services. *Distributed and Parallel Databases*, 17(1):5–37, 2005.

4 Walter Binder, Ion Constantinescu, and Boi Faltings. Decentralized Orchestration of Composite Web Services. In *International Conference on Web Services*, pages 869–876, 2006.

5 Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. An Approach for QoS-Aware Service Composition Based on Genetic Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1069–1075. ACM, 2005.

6 Valeria Cardellini, Emiliano Casalicchio, Vincenzo Grassi, Stefano Iannucci, Francesco Lo Presti, and Raffaela Mirandola. MOSES: A Framework for QoS Driven Runtime Adaptation of Service-Oriented Systems. *IEEE Transactions on Software Engineering*, 99, 2011.

7 Girish Chafle, Sunil Chandra, Vijay Mann, and Mangala Gowri Nanda. Orchestrating Composite Web Services Under Data Flow Constraints. In *IEEE International Conference on Web Services*, pages 211–218, 2005.

8 Anis Charfi and Mira Mezini. Hybrid Web Service Composition: Business Processes Meet Business Rules. In *Proceedings of the 2nd international conference on Service oriented computing*, ICSOC '04, pages 30–38. ACM, 2004.

9 Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

10 Michael C. Jaeger, Gero Mühl, and Sebastian Golze. QoS-Aware Composition of Web Services: An Evaluation of Selection Algorithms. In *OTM Conferences (1)*, pages 646–661, 2005.

11 Massimiliano Di Penta, Raffaele Esposito, Maria Luisa Villani, Roberto Codato, Massimiliano Colombo, and Elisabetta Di Nitto. WS Binder: a Framework to Enable Dynamic Binding of Composite Web Services. In *International Workshop on Service-oriented Software Engineering*, pages 74–80, 2006.

12 F. Rosenberg, P. Celikovic, A. Michlmayr, P. Leitner, and S. Dustdar. An End-to-End Approach for QoS-Aware Service Composition. In *IEEE International Enterprise Distributed Object Computing Conference*, pages 151–160, 2009.

13 Florian Rosenberg, Max Benjamin Müller, Philipp Leitner, Anton Michlmayr, Athman Bouguettaya, and Schahram Dustdar. Metaheuristic Optimization of Large-Scale QoS-aware Service Compositions. In *IEEE International Conference on Services Computing*, pages 97–104, 2010.

14 Mazeiar Salehie and Ladan Tahvildari. Self-Adaptive Software: Landscape and Research Challenges. *TAAS*, 4(2), 2009.

15 Quan Z. Sheng, Boualem Benatallah, Marlon Dumas, and Eileen Oi-Yan Mak. SELF-SERV: A Platform for Rapid Composition of Web Services in a Peer-to-Peer Environment. In *VLDB*, pages 1051–1054, 2002.

16 Latha Srinivasan and Jem Treadwell. An Overview of Service-Oriented Architecture, Web Services and Grid Computing. *HP Software Global Business Unit*, 2005.

17 Danny Weyns, Sam Malek, and Jesper Andersson. On Decentralized Self-Adaptation: Lessons from the Trenches and Challenges for the Future. In *Proceedings of the ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 84–93, 2010.

18 Tao Yu, Yue Zhang, and Kwei-Jay Lin. Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints. *ACM Transactions on the Web*, 1, 2007.

19 Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng. Quality Driven Web Services Composition. In *Proceedings of the 12th International Conference on World Wide Web*, pages 411–421, 2003.

20 Eckart Zitzler, Marco Laumanns, and Stefan Bleuler. A Tutorial on Evolutionary Multiobjective Optimization. In *Metaheuristics for Multiobjective Optimisation*, pages 3–38, 2003.