A Flexible Approach for Modelling and Analysis of Feature Interactions in Service-Oriented Product Lines

Muhammad Imran Abbasi*, Lewis M. Mackenzie
School of Computing Science, University of Glasgow, Glasgow, G12 8QQ, UK.

* Corresponding author: Tel: 0141 572 0422; email: m.abbasi.1@research.gla.ac.uk
Manuscript submitted February 28, 2017; accepted March 28, 2017.
doi: 10.17706/jsw.12.10.823-830

Abstract: Web services technology provides interoperability between various software applications running on different platforms, allows an organisation or individual to develop more advanced services, and publishing them on the web. Service Composition potentially involves a large number of interactions among services features involved in the process. Generally, most of the interactions are desirable, however, in some scenarios such interactions may lead to undesirable interactions between components that can compromise customer preferences for QoS features such as privacy, security, and personalisation etc. Such interactions are known as *feature interactions*, and can adversely affect the overall quality of any composite service. A survey of traditional approaches such as (*BPEL, WSCDL, OWL-S*, and *WSMO* etc.) shows that none of them offers any direct support for verification of service composition at design time for evaluating its correctness. This demands a flexible approach, capable of specification and analysis of interactions among services features, and guarantee that service composition process yields feature interaction free services. In this paper, a flexible approach is proposed for handling feature interactions problem proactively at domain engineering stage by integrating the Service Orientation and Software Product Lines approaches. With the help of a motivational case study, it has been hypothesised that proposed approach allows a service engineer to model and reason about feature interactions in Service Oriented Product Lines (SOPLs).

Key words: Web services, service oriented product lines, feature modelling, feature interactions.

1. Introduction

Web services aim to provide a sophisticated framework for building complex distributed systems, focusing on interoperability, support for efficient integration of distributed processes, and uniform applications representation [1]. Web services also provide a flexible mechanism for packaging services features, making them visible and approachable to other business environments, as a distributed (loosely coupled) software components.

Web Service Composition (WSC) provides highly customised services known as Composite Web Services, composing different component services, available on the internet. Typically, web services provided by multiple organizations, perform basic activities, and can be combined in a suitable way to form complex business processes. Moreover, web services support interactions among different partners by providing a model of synchronous or asynchronous exchange of messages. Such exchange of messages can be composed into longer interactions by defining protocols, constraining the behaviours of all partners.

Due to new and more sophisticated requirements of customers, modern software systems are becoming more and more complex with every passing day. However, the complexity of such systems can be managed by modelling web services interactions at domain engineering and design stage, by integrating Service Orientation and Software Product Lines approaches. A *Software Product Line (SPL)* provides a systematic software reuse approach by handling different types of flexible software components, creating a common platform for developing a set of concrete products [2].

2. Service Oriented Product Lines (SOPLs)

Service Orientation provides a promising mechanism for supporting continuously changing customers' needs and expectations, as more sophisticated software systems are connected to the Internet. The services evolve due to dynamic addition and integration of the various services available.

Service Oriented Product Lines (SOPL) combines Service Orientation with *Software Product Lines Engineering (SPLE)* to achieve the development of more flexible and customised web services [3]. SOPLs basically introduce the concept of *service variation* that makes the service composition process more flexible by specifying variability in service components combined in different combinations or patterns. Feature based service modelling allows service engineers to make service composition process more scalable by providing the options to select more customised or best fitting services [4]. The fusion of these two popular modelling paradigms (Service Orientation and Software Product Lines) provides a great potential to develop service based solutions known as SOPLs that can tackle various challenges in development and infrastructure management of service oriented systems.

3. Feature Interactions in Services Oriented Product Lines

Interactions among services may occur at any point during a Service Composition Process. This shows that, with an increase in the number of features and services, there is a combinatorial explosion in number of scenarios with potential for an interaction. However, in some scenarios such as dynamic business environments, rapid changes in services can lead to some undesirable results due to unexpected interactions among the components in a composition process. Such undesirable interactions among web services are known as *feature interactions*, and affect the quality aspects of a composed service(s).

Feature interactions among web services of a SOPL can be divided in to two main categories; *functional and non-functional* [5]. Functional feature interactions are caused by composition of functional aspects of services or features. These include race conditions, resource contention, violation of assumptions and invocation order. On other hand, non-functional feature interactions affect quality-of-service (QoS) properties, such as *security, privacy, and availability*.

4. Problem Identification and Research Motivation

To illustrate feature interaction problem in web services, we consider a case study of a typical online bookstore (OBS) web service as shown in Fig. 1, where an abstract feature model for such a service is developed. The OBS service provides an online book shopping facility based on customer's preferences and profile information. The composite service (OBS) is composed of services (features), such as *Personalization control, Security Control, Online Payment and Shipping services* which are associated with further sub-services. We consider only the *Personalization service (feature)* to describe the feature interactions.

The Personalisation service (feature) is composed of three sub-services (features); *Customer's Profiling, Information Filtering, and Identity Management*. The Customer's Profiling service collects and stores a customer's information (address & preferences) in a profile. Similarly, the Information Filtering service stores more relevant results, matching to a customer's profile, and the Identity Management service provides a unique identity for customers, with which they can be identified by service providers.

The identity management feature (service) is implemented by ‘*personalisation feature*’, in different ways, using third party services available on the web. In the OBS feature model service providers use *iPassport Web Service* [5] to authenticate customers. But a keen observation of such an arrangement shows that, the iPassport feature facilitates other third party services (involved in the composition process), to access customers profile. The iPassport feature (service) is composed of two sub-features (services); *Authorisation and Authentication* of customers.

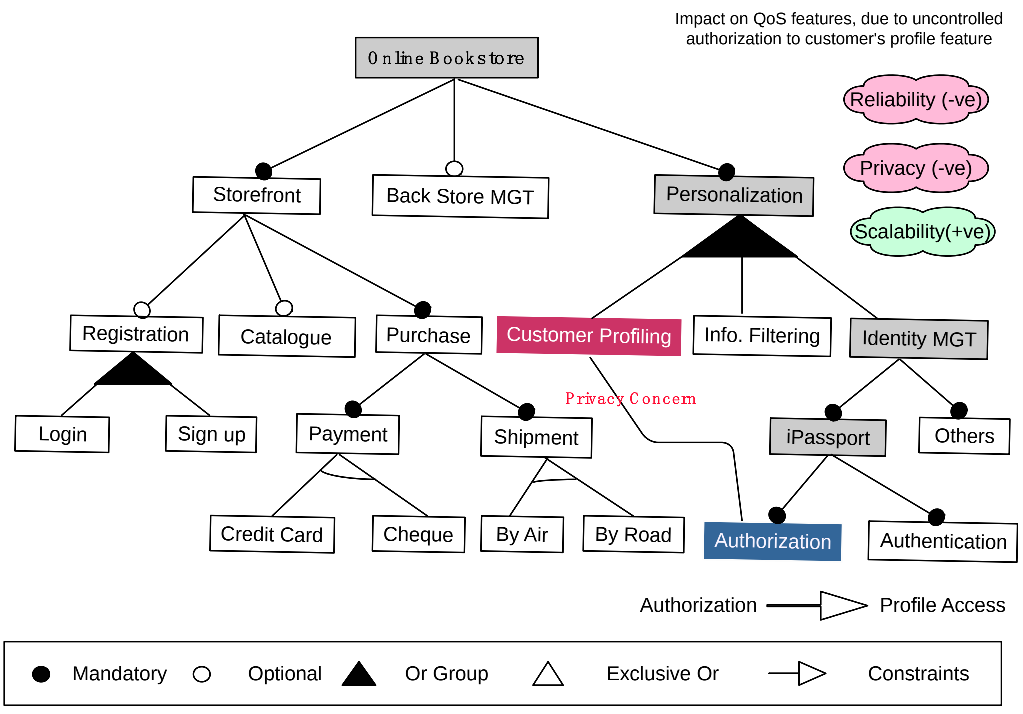

Fig. 1. Feature Model of a composite online bookstore web service.

The *authorisation feature* enables service providers, to access a customer’s profile (personal information). However, such profile information can effectively be shared, among different service providers (trusted or untrusted) on the web, for any purpose, without the customer’s knowledge. If a customer is only interested in sharing his/her profile information with trusted service providers such as the OBS service itself, and doesn’t want to share with other untrusted (third party) providers such as sub-contractors of OBS, the service feature model (shown in Fig. 1) is not suitable.

This is because the *identity management feature* associated with iPassport service, compromises or violates customer’s preferences for some features, such as *privacy, reliability, predictability* etc. Such features are known as *non-functional or QoS features*, and are implicitly associated with OBS web service.

The feature model shown in Fig. 1, allows an enhancement of some features such as the *scalability* of the OBS Web Service, as more results (services) are provided to the customer in response to his/her query by accessing services from various third-party services providers, however, this arrangement compromises other features such as *privacy and predictability*. Therefore, in such a situation, a customer’s wishes concerning non-functional features should be taken in consideration, as to whether features such as scalability and privacy are important or not. Thus, a problem of *feature interactions* emerges under such circumstances, and needs to be addressed well before deployment or implementation of real web services.

5. Proposed Approach for Modelling and Analysis of FI in SOPLs

Interactions among web services are mandatory to obtain the highly customised services known as

Composite Web Services, satisfying customers' needs. Generally, most of the service composition scenarios show that service interactions are desirable, however, there are some scenarios in which such interactions may lead to undesired *service interactions* or *feature interactions*.

Therefore, an efficient approach is required to model *non-functional interactions* among web services early at design stage, in such a way that all the potential (desired and undesired) feature interactions can be identified and managed at the domain engineering (services design) stage, and suitable solutions can be developed to control or avoid such interactions. Such an approach can make service development process more efficient, and quality-oriented services can be obtained as a result.

To achieve these objectives, an integrated approach is proposed for modelling and analysis of feature interactions in web services at the design stage by integrating three well-known modelling paradigms; *Goal Modelling* [6], *Feature Modelling* [7], and the *light weight formal modelling language Alloy* [8]. A stepwise view of the proposed approach is presented in Fig. 2.

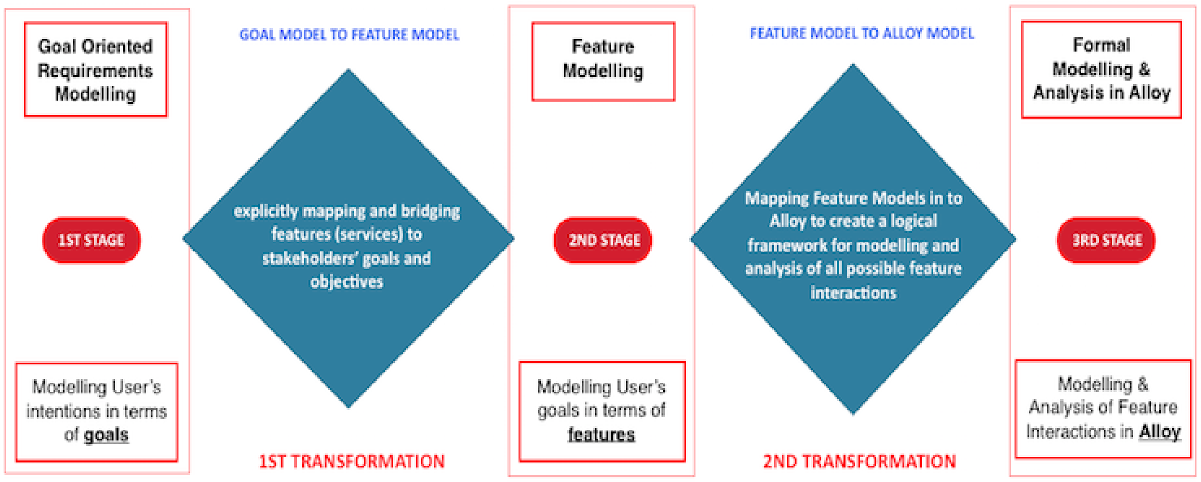
Goal Oriented Requirements Modelling
1ST STAGE
Modelling User's intentions in terms of goals
GOAL MODEL TO FEATURE MODEL
explicitly mapping and bridging features (services) to stakeholders' goals and objectives
1ST TRANSFORMATION
Feature Modelling
2ND STAGE
Modelling User's goals in terms of features
FEATURE MODEL TO ALLOY MODEL
Mapping Feature Models in to Alloy to create a logical framework for modelling and analysis of all possible feature interactions
2ND TRANSFORMATION
Formal Modelling & Analysis in Alloy
3RD STAGE
Modelling & Analysis of Feature Interactions in Alloy

Fig. 2. Stepwise view of proposed approach.

As shown in Fig. 2, the proposed solution consists of following steps.

5.1. Constructing Service Goal Models

At first stage an abstract Service Goal Model of a composite web service (composed of component services) is constructed by exploiting goal modelling techniques. Service goal models specify possible combinations to achieve desired goals of customers. Service goal model construction is based on three main concepts of the goal modelling paradigm known as *goals, soft-goals,* and *plans or tasks*.

Goal models are used to specify stakeholders' objectives or intentions in terms of goals, and variation points among goals as well. The goals are used as a reference for eliciting, elaborating, structuring, specifying, analysing, negotiating, and modifying, requirements of a customer, with respect to services [9]. The component services are modelled as goals and actions (depending on stakeholders needs), and non-functional (quality of service) aspects are modelled as *soft-goals*, using constraints specified for the interaction or composition of services.

5.2. Creating Service Feature Models

Service Feature Models also known as *Service Oriented Product Lines (SOPLs)* are constructed by mapping service goal models to corresponding components of a feature model based on a set of *transformation rules*. Typically, a SOPL specifies the interacting component services, combined for a composition process to obtain a composite service. In this context, a service composition is realised as a configuration of a SOPL and variation points describe different possible combinations of component services, as shown in Fig. 3. The concept of variability in SOPLs makes a service composition process more flexible by providing a facility to have different composition alternatives for a composite service, and an opportunity to select the best fitting combinations of services, according to a customer's requirements or preferences.

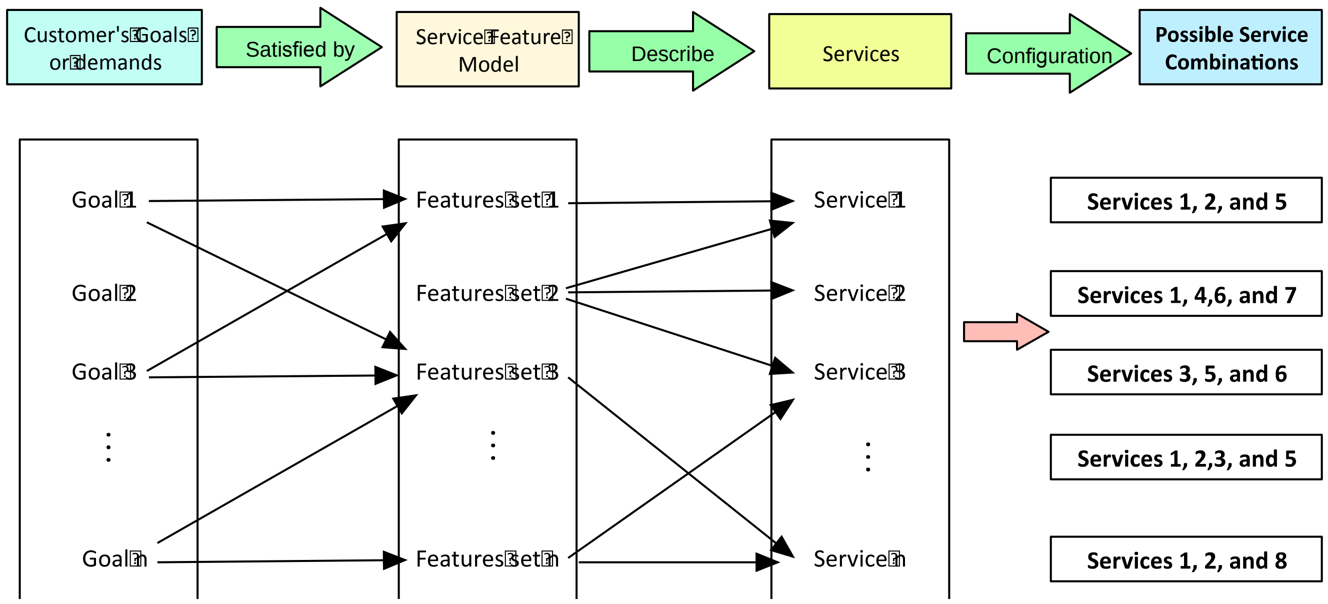

Fig. 3. From service goal models to service feature models and creation of service compositions.

5.3. Formalisation of Service Feature Models

Service Feature Models (SFMs) are formalised to the light weight modelling language *Alloy* [10], for further analysis of service interactions. A SFM is connected to a corresponding *Feature-Solution Graph (FS-Graph)* to specify the conflicting combinations of services (*feature conflicts*). A FS-Graph integrates quality features (attributes) with configurable service feature models providing a flexible way to specify trade-offs between quality features such as *security, privacy, and reliability* etc [11]. The FS-Graphs are encoded in Alloy logic for analysis of feature interactions, providing possible solutions to resolve such interactions.

5.4. Analysis of Service Interactions in Alloy Analyzer

Alloy Analyzer is used to analyse service feature models and corresponding FS-Graphs encoded in Alloy. Formal properties for composition and interaction of features (services) are specified at this stage and checked by *Alloy Analyzer* for potential feature interactions in a service composition process. Alloy Analyzer performs an exhaustive analyses of service interactions based on a *set of constraints* by checking the specified properties for interacting services (features) involved in a service composition process.

Alloy Analyzer produces an *error free validated output model*, if all the services interactions adhere to specified constraints types specified in a FS-Graph, however, if a service feature model (SOPL) fails to conform to specified constraints or an undesired combination of services is detected, Alloy Analyzer produces a *counter-example* showing an erroneous situation or *feature interaction problem*. The counter-examples produced by the Alloy Analyzer play a major role in rectifying the issues detected in a service feature model. To resolve or avoid the detected service conflicts, a SOPL model is redesigned or refactored by exploiting *feature model re-factoring* techniques [12]. This mechanism facilitates a *service engineer* to refactor a composite service (features) model according to desired goals or requirements.

6. Related Work

A survey of existing work on *feature interactions* in *Service Oriented Systems* domain, shows only limited research efforts to date, providing an abstract introduction but no scalable or effective solutions to identify, avoid, control or resolve such interactions in SOPLs. Some of these efforts are briefly presented in this section.

Michael Weiss *et al.* [13] first introduced the idea of feature interactions among web services and classified them in to two major categories; *functional and non-functional features*. They used *goal-oriented analysis (Goal Models)* and *scenario modelling (Use Case Maps)* to specify and detect feature interactions among functional and non-functional features of web services. The approach presented lakes the direct mapping of goals to features and verification of feature interactions in combination to the goal models specified.

Authors in [14] have exploited model checking techniques (SPIN) for detection and verification of contradictions among features of a feature model by encoding the propositional semantics of the feature models in *Promela*. Promela specifications are checked by the *SPIN* Model Checker by considering the specified properties or assertions. However, the approach does not show its suitability for composite or large scale feature models, and for handling of the *state explosion problem* caused by an increase in number of features.

Both Goal and Feature modelling approaches have reasonable support from the formal methods research community. Various formalisations have been proposed and used for the specification and verification of feature models such as model checking [15], and theorem proving [16] etc. The results and experiences of these research lines can effectively be used to model and analyse the feature interaction problem in SOPLs.

In [17] S. Apel *et. al*, proposed a *feature oriented design paradigm* based on *Feature Oriented Software Development* concepts, and created a new version of *Alloy* supporting the concept of *feature* known as *FeatureAlloy*. Although, the suitability of *FeatureAlloy* has been shown by considering case studies from three different domains, the concept has not been applied in the web services domain, which is quite different from traditional software systems.

Authors in [18] have used feature models for developing a feature-oriented approach to web service customisation. The developed approach is capable of addressing some important challenges such as reducing complexity, automated validation of feature models, and dynamic deployment of services. Feature models are used as service description artifacts, facilitating the service customisation process. The proposed approach is based on a Model-Driven Development (MDD) framework to automate large parts of its operation.

7. Conclusions and Future Work

A more flexible, integrated approach has been developed to model and analyse the feature interactions problem in SOPLs by integrating three popular modelling paradigms; *goal modelling*, *feature modelling*, and lightweight formal modelling language *Alloy*. To show the suitability of the proposed approach, a motivational case study is developed, where the emergence of potential feature interactions problem among the services of an online bookstore web service is shown.

Based on a substantial survey of related research work, it has been hypothesised that the proposed approach provides a flexible mechanism for specification, identification, and management of potential feature interactions among web services, at an earlier design stage (requirements engineering).

Currently, more sophisticated case studies from the SOPLs domain are being explored to check the scalability and effectiveness of proposed approach. It is argued that approach presented in this paper, opens the doors to the research community to explore the feature interaction problem in SOPLs by integrating

Software Product Lines Engineering (SPLE) and information systems concepts, to develop quality oriented services.

References

[1] Alonso G., Casati F., Kuno H., & Machiraju V., (2004) *Web Services: Concepts, Architectures and Applications*.

[2] Apel S., Batory D., Kästner C., & Saake G. (2013). *Feature-Oriented Software Product Lines*.

[3] Gunther S., & Berger T. (2008). Service-oriented product lines : Towards a development process and feature management model for web services. *Proceedings of 12th International Software Product Line Conference*.

[4] Medeiros F. M., De, A. E. S., & De, L. M. S. R. (2010). Designing a set of service-oriented systems as a software product line. *Proceedings of 4th Brazilian Symposium on Software Components, Architectures and Reuse*.

[5] Weiss M., Esfandiari B., & Luo, Y. Towards a classification of web service feature interactions. *Processing*.

[6] Saidani, O., Kaabi, R. S., Kraiem, N., & Baghdadi, Y. (2013). A framework for goal-oriented methods for services.

[7] Sanchez, L. E., Diaz-pace, J. A., Zunino, A., Moisan, S., & Rigault, J. (2015). An approach based on feature models and quality criteria for adapting component-based systems.

[8] Torlak, E., Taghdiri, M., Dennis, G., & Near, J. P. (2013). Applications and extensions of Alloy: Past, present and future. *Math. Struct. Comput. Sci*.

[9] Oster, Z., Ali, S., Santhanam, G., Basu, S., & Roop, P. (2012). A service composition framework based on goal-oriented requirements engineering, model checking, and qualitative preference analysis. *Serv. Comput.* 7636.

[10] Sloane, A. M. (2009). *Software Abstractions: Logic, Language, and Analysis by Daniel Jackson, The MIT* Press.

[11] Chavarriaga, J., Noguera C., Casallas R., & Jonckers V. (2015). Managing trade-offs among architectural tactics using feature models and feature-solution graphs. *Proceedings of the 2015 10th Colombian Computing Conference*.

[12] Alves, V., Gheyi, R., Massoni, T., & Kulesza, U., Borba, P., & Lucena, C. (2006). Refactoring product lines. *Proceedings of the 5th Int. Conf. Gener. Program. Compon*.

[13] Weiss, M., & Esfandiari, B. (2005). On feature interactions among web services 1 INTRODUCTION. *Int. J*.

[14] Hemakumar, A. (2008). Finding contradictions in feature models. *Dept . Electr. Comput. Eng. Univ. Texas Austin*.

[15] Cordy, M., Classen, A., Schobbens, P. Y., Heymans P., & Legay, A. (2012). Managing evolution in software product lines: A model-checking perspective. *ACM Int. Conf. Proceeding Ser*.

[16] Gheyi, R., Massoni, T., & Borba, P. (2011). Automatically checking feature model refactorings. *J. Univers. Comput. Sci*.

[17] Apel, S., Scholz, W., Lengauer, C., & Christian, K. (2010). Detecting dependences and interactions in feature-oriented design.

[18] Nguyen, T., & Colman, A. (2010). A feature-oriented approach for Web service customization. *Proceedings of the 2010 IEEE 8th International Conference on Web Services*.

Muhammad Imran Abbasi is working as a researcher at School of Computing Science, University of Glasgow, Scotland UK. His research interests include service oriented systems, feature-oriented design, feature modelling, service oriented product lines, and process mining. He is particularly interested in integrating popular software modelling paradigms such as (feature modelling, goal modelling), and formal techniques such as (software abstractions), and their useful fusion to model and analyse services oriented systems.

Lewis M. Mackenzie is a senior lecturer in School of Computing Science, University of Glasgow, UK. His research interests are in mobile ad hoc, vehicular and wireless sensor networks, advanced computer architectures and simulation. He is also interested in physics of computing, quantum computing and philosophy of science and has co-authored a book covering topics in these areas. He holds a BSc in mathematics and natural philosophy and a PhD in physics.