Sindh Univ. Res. Jour. (Sci. Ser.) Vol.43 (1-A) 69-78 (2011)

SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

Modeling Feature Interactions among Web Services based on Feature Models

I. Abbasi and R. Cooper

School of Computing Science (SoCS), University of Glasgow, Scotland (UK)
imran@dcs.gla.ac.uk, rich@dcs.gla.ac.uk

cor

rec

Abstract: The online community has been growing steadily year on year and with this escalation the number of services provided has increased in an attempt to meet the demands of a computer literate audience. This has led to a progression from a human-oriented use of the web to an application-oriented structure through the use of web services composed in various ways. The web service composition process potentially involves a large number of interactions among the features of the services involved in the process. This demands a flexible modeling approach that should be capable of modeling and analyzing the interactions among such features. The specified model should be verified to guarantee that the designed service composition process yields feature interaction free services. However none of the traditional approaches (BPEL, WSCDL, OWL-S, and WSMO etc.) offers any direct support for verification of service composition at design time to evaluate its correctness. In this paper we conducted a substantial survey of current approaches used for the analysis of feature interaction in web services and an integrated approach is proposed to handle this issue in a more flexible way, by using feature modelling and model checking techniques. We suggest that the proposed approach is more scalable as compared to the traditional approaches surveyed.

Keywords: Composite web services, Feature Interactions, Extended feature Models, Model Checking, and Software Product Lines.

INTRODUCTION

Service Oriented Architecture is an emerging paradigm of software architecture, providing a platform for decomposition of software into value-added components called Web Services. Web Services provide a well-defined piece of functionality while hiding implementation details behind an interface. By adopting the standard SOA approach, programmers can integrate multiple distributed services each providing a different functionality and written in a different programming language. The integration of services yields more value-added service called " Composite Services". A composite service captures all the operations and functions provided by the sub-services/component services and allows their invocation in a defined structure (workflow structure).

Web Service Composition is an hierarchical stepwise process, to compose multiple distributed services in to composite service(s), which in turn can be composed into even bigger / value-added composite services. Composite Web Services achieve some high level functionality and satisfy predefined business goals (Pistore, *et. al.,* 2005). The Business goals describe the required behaviors of web services based on the requirements specified by the service clients or requesters.

However due to the rapid growth of services over the internet and a considerable amount of features incorporated in such services, highly demanded by the users, the traditional functional-oriented architecture of web services is changing into a more flexible feature-oriented architecture. The feature-oriented description of Web Services specifies the combination of features (implicit or explicit) associated to services. A feature reflects a stakeholder's requirement, provides a configuration option and is an increment in functionality of a service (Apel, *et. al.,* 2008). The feature-oriented Web services are a way of packaging and publishing functionality to the network for use by other applications and can aggregate other web services from a web of services to provide a higher-level set of features (Weiss, *et. al.,* 2005). Generation of an executable implementation of a composite web service

depends on the efficient description of the properties or features (Functional and Non-functional) of the component web services. This implementation satisfies the composition requirements like QoS by invoking in a suitable way the set of existing web services (component services).

The feature-oriented nature of web services demands a flexible and adaptive architecture that can accommodate the changes and enhancements to the features supported by such Services. However, due to the increase of new services paired with the dynamic nature of business environment leads to some undesirable interactions that cause a negative impact on Service Quality and User Satisfaction. Such undesirable interactions are called Feature Interactions, a significant area of research highly explored by the worldwide research community of Telecommunication systems, where features (additional units of functionality) would interfere with each other and cause some unpredictable behavior.

MATERIAL AND METHODS

Research Motivation

In a formal way, a feature can be defined as "a logical unit of behavior that is specified by a set of functional and quality requirements" (Bosch, *et. al.,* 2000). According to (Riebisch, *et. al.,*2000) in a more precise way "A feature represents an aspect valuable to the customer". The feature interaction problem is generally associated with conflicting features causing undesirable effects. A feature must be able to adjust itself to work with other features or services - a highly relevant problem called feature interaction (Pang , *et. al.,*2003). The problem of feature interaction was initially introduced in the field of telecom by Bellcore, mainly includes interactions that occur due to the incompatibility of the requirements of multiple features and interactions that occur when a specific feature behaves differently in the presence of other features.

As mentioned earlier, that the phenomena of feature interaction has extensively researched in telecommunication systems but it can play a great role in investigating the feature detection, interaction, composition, analysis and verification of distributed software systems such as Web Services- an implementation of Services Oriented Architectures (SOA). However due to an open, flexible and platform independent nature, the problem of feature interactions in web services domain is quite different from Telecom systems. This difference exists on the basis of various aspects such as Feature abstract problem, the distributed nature of web services (causing issues like security, manageability, privacy etc) and complex nature of composite services (due to increase in component services as compared to telecom services) (Zhang , *et. al.,*2007).

Web Services posses a considerable amount of features that can be used to describe the Quality parameters such as Security, Privacy, Scalability, Time, Availability etc, associated to such services in an implicit or explicit manner. A web service user or consumer is mostly interested to make a better trade-off among such features by their proper selection to obtain maximum desired quality. The selected features are used to design a composition process showing the composition of component services along with supported features to create a Composite Web Service that is capable to provide the quality desired by the service consumers.

The composition of web services has largely concentrated on the critical impact of compatibility - i.e. are the outputs of one service adequate to provide the inputs of its successor. However, there are other issues which are important in considering the acceptability of the composite service - these being the aspects of quality the composite service can be expected to support given those of the component services. However it is important to describe the Web Services not only with respect to their functional features but on the basis of their non-functional features as well. The non- functional features of web services represent the quality of service characteristics or business goals to be achieved by the services and are of significant importance due to their firm attachment to quality of service constraints such as level of security attained by a service, privacy preservation, availability on request and reliability etc. Chung (Chung, 1991) defines the non-functional requirements as constraints over the functionality of a target system. The non-functional features are system concerns affected by functional features, and impose constraints how the functionality is provided (Weiss, *et. al.,*2005). According to (Pulvermueller, *et. al.,*2007) features can be considered as the functional units of a system and the non-functional features as its non-functional properties. Therefore on the basis of this classification we cay say that a feature is an identifiable unit of a system and a property can help us to decide about the nature of the feature.

Interaction is at the very basis of the Web Service concept, because Web Services need to interact for the emergence of useful composite services from the interaction of various highly specialized services (Weiss, *et. al.,*2005). So an efficient semantic- rich approach is required to describe the composition of a

composite service by selecting and composing the features of its component services. This approach should be able to show the interaction among the features during the process of composition and an analysis of possible impact on the overall quality of Composite service due to the interaction and composition of these features.

Our comprehensive research literature review on feature interactions in Web Services shows that there are only a few research papers that highlight the issue of feature interactions in Web Services such as Micheal Weiss (Weiss, *et. al.*,2005), (Weiss, *et. al.*,2007), Stephen Reiff (Gorton, *et. al.*, 2007) and Zhang Jian-yin (Zhang , *et. al.*,2006). However, these research efforts provides the basic idea about modeling the feature interactions in Web Services domain, but their approach lakes a flexible and formalized architecture for specification, interactions, composition and verification of such features possessed by highly quality oriented service oriented architectures (Liang , *et. al.*,2007).

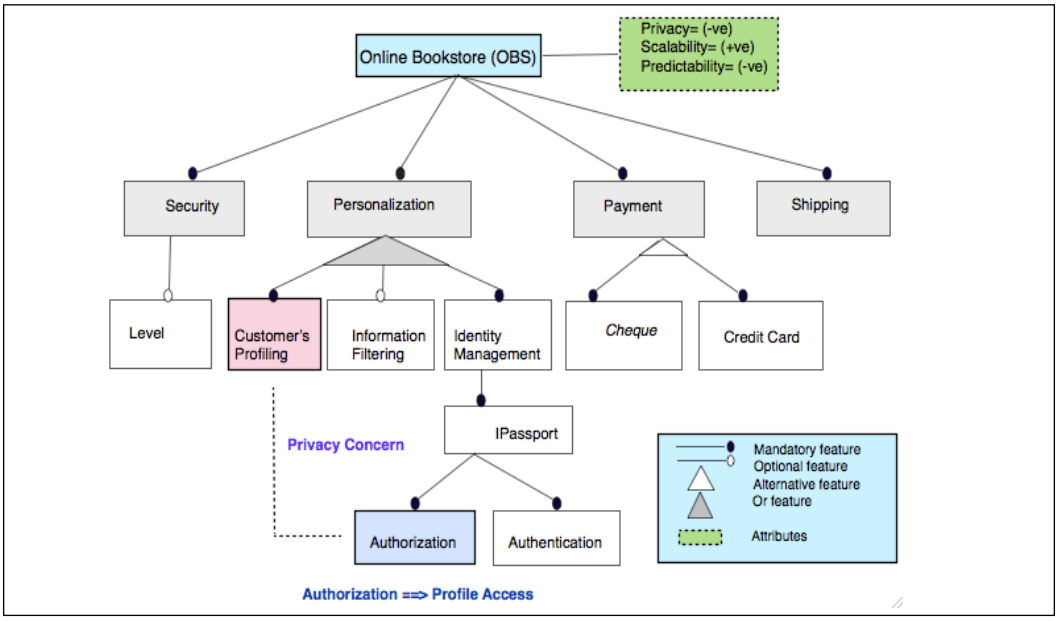
Therefore, to address the issue of feature interaction and composition in Web Services we propose an integrated approach by combining the Feature Modeling (Kang, *et. al.*,1990) (mostly used in SPLE (Krueger, *et. al.*,2007) and Model Checking (SPIN Model Checker) (Holzmann, 2003). The major purpose is to combine the capabilities of these research lines to develop an efficient mechanism to overcome the problems of specification, interactions, composition and verification of features in web services.

A Motivating Example

To illustrate the feature interaction problem in web services and applicability of our proposed approach, let us consider an example of a typical On-line Bookstore (OBS)- a web service providing the online book shopping facility by using a customer's preferences and profile information. The composite service (OBS) provides some features such as Personalization control, Security Control, Online Payment and Shipping services. These features contain further sub-features presented in **(Fig. 1)**. We only consider the Personalization feature to analyze the problem of feature interactions.

The Personalization feature is composed of three sub- features named as Customer's Profiling, Information filtering and Identity management. Customer's Profiling collects and stores customer's information (address and preferences) in a profile, while information filtering service stores more relevant results, matching to the customer's profile. The identity management service provides a unique identity to the customers, with which the service providers can identify them.

The personalization feature can implement the identity management sub-feature in different ways by using third party services. We consider Microsoft's IPassport web service (as mentioned in (Weiss, *et. al.*,2005). The IPassport service authenticates service customers to service providers and provides access to the customers profile. So the IPassport service provides two main features i.e. Authorization and Authentication of customers.



Fig, 1. Feature Model of an On-line Bookstore Web Service

The authorization feature of the IPassport service provides access to the customer's profile (personal information) to service providers no matter who are they. It means that the customer's profile information can effectively be shared among different service providers (trusted or urn-trusted) for any purpose, without the customer's knowledge. If the customer is only prepared to share his profile information with the trusted provider (OBS service) and doesn't want to share with untrusted (third party) service providers e.g. sub-contractors of the OBS web service. In such a situation, the presented model doesn't fit with this requirement because the customer's profile information can be accessed by any third party service provider through the IPassport's Authorization feature, and then can be used for any ill-legal purpose. So customer's privacy and security is compromised in such architecture. In other words we can say that the identity management feature using IPassport service affects/violates some of the customer's concerns such as privacy, reliability, predictability etc. These concerns are called the non-functional requirement / features implicitly associated with the OBS web service. However the model increases the scalability of the OBS service by providing more results to the customer's query but the privacy and predictability features are affected negatively due to such arrangement of authorization feature. Therefore in such situation the customer's decision are taken in to account, whether the features such as scalability or privacy are more important. This is the problem of feature interactions.

An integrated approach is proposed in this paper that provides an effective approach to control the problem of feature interaction in composite web services by exploiting feature modeling and model checking techniques. The customer's preferences are described as constraints on features (to be selected and composed) in a feature model. The specified feature model is checked for valid combinations of features and possible conflicts by using model checking techniques. If any feature conflicts arises during the composition of features than such conflicts are resolved by refactoring the feature models according to specified algebraic laws for such models (R. Gheyi, *et. al.,* 2008). So all the possible conflicts among the features are detected at design stage and a feature-interaction free web service is guaranteed for the composition process.

Integrating Feature Modeling and Model Checking Techniques

A. Feature Modeling

Feature Modelling is an efficient and flexible modeling approach, basically used for identifying commonalities and differences among all possible potential products of an SPL. The output of feature modeling is a compact representation of all potential products of an SPL, called the **"**Feature Model". A feature model represents different ways in which a software system can be composed in terms its associated features and relations among them. A valid composition of the features is called a configuration. In feature modeling, a feature can be considered as a distinctive characteristic of a software product (Benavides, *et. al.,* 2005). A Feature Diagram is a common way to depict a feature model in a graphical tree-like structure. The root of the tree is called the parent feature; represents the actual software system/SPL and sub-nodes are called the child features of this node. A basic feature model/feature diagram depicts four types of relations among features, i.e. Mandatory, Optional, alternative and or- relation. In a Mandatory relation a child feature must always be present in the SPL's products when its parent feature is present, while in an Optional relation the child feature may or may not be present in a product when its parent feature is present. In an Alternative relation a child feature may be present in a product when its parent feature is included, but only one feature of the set of child features may be present. Similarly, in an or-relation the child feature may be present in a product when its parent feature is included but at-least one child feature is included (Benavides, *et. al.,* 2005).

The basic feature models as presented by (Kang, *et. al.,* 1990) only deal with modeling the functional characteristics provided by a Software system called ***"functional features***", and do not support the modelling of non-functional/quality of service characteristics. If these characteristics could model properly, this would increase the number of potential products of a software system such as an SPL. However to overcome these shortcomings of feature models, (Benavides, *et. al.,* 2005) modified the basic feature models by adding the notion of *"extra-functional/QoS"* features. The enhanced feature models are known as ***"Extended Feature Models"***. The extended feature models support the modeling of extra-functional features as well. The modelling of such QoS/ non-functional features allows QoS-driven service selection and composition. The extended feature models include some extra concepts as compared to basic feature models such as a prominent characteristic of a product, Attribute, any characteristic of a feature that can be measured e.g. availability and cost of a service, Attribute domain, the space of possible values where the attribute takes its values, and Extra-functional feature a relation between one or more attributes of a feature e.g. availability <70% etc.

Another variation of feature models provides the modeling of propositional formulas as well, to describe the ***crosstree constraints*** on features of a product showing the dependency or priorities among them. A sample extended feature model has been shown in figure 1, where a dotted-line rectangular box describes the extra-functional properties along with their domain values.

B. Exploiting Model Checking for automatic analysis of Feature Models

A feature model of a composite web service defines the associated services by a unique combination of features possessed by such services. The defined combinations may contain some undesired interactions among features, called conflicts, which affect the quality of composed service, expected by a user/requester. To analyze such feature conflicts/interactions, a feature model can be mapped to a propositional formula (Hemakumar, *et. al.*, 2008), where each feature is considered as a distinct Boolean variable and the constraints on feature combinations are encoded in formula as well. A variable in the formula has value ***True*** if its corresponding feature is present in a product or it is ***False*** otherwise. The rules for composition and interactions among the features are de- scribed in the form of propositional formulas, accompanied by a set of algebraic laws for feature models based on feature algebra (Apel, *et. al.*, 2008), and Hofner, *et. al.*, 2009).

The conflicts/interactions among the features of services can be found by using model checking techniques such as SPIN (Holzmann, *et. al.*, 2003). SPIN is an automaton-based verification system used for the models of distributed software systems (Zhang , *et. al.*, 2007). The Spin can be used to check the feature models for possible contradictions among features (Hemakumar, *et. al.*, 2008). The Spin is based on PROMELA (a modeling language for concurrent processes), designed for verification. To analyze the feature interactions among features, the variables and constraints on the specified features of a feature model are represented as propositional formulas, A Promela translator can be used to convert these formulas into a Promela program, that states the properties (composition & interaction) of features. The stated properties are verified by the Spin by using exhaustive exploration of the system state-space (Hemakumar, *et. al.*, 2008).

Proposed Technique for Analysis of Feature Interactions in Composite Web Services

Our research work is partially inspired by the proposals of Adithya Hemakumar (Hemakumar, *et. al.*, 2008) and Rohit Gheyi et.al (Gheyi, *et. al.*, 2008). The authors in (Hemakumar, *et. al.*, 2008) proposed an approach for verifying the contradictions in feature models by translating the propositional semantics of a feature model into the Promela language and model checked by the Spin model checker. We have adopted this idea to implement it in the domain of feature- rich web services to analyze the feature interactions in such services by using model checking techniques. In this way the feature interaction problem can be handled at the design level to provide basis for development of feature-interaction free services for the user, as desired. In (Gheyi, *et. al.*, 2008), authors have proposed a number of algebraic laws for refactoring and reasoning about feature models. We use these laws to resolve the feature interactions identified by the model-checking phase, by suitably refactoring the feature models.

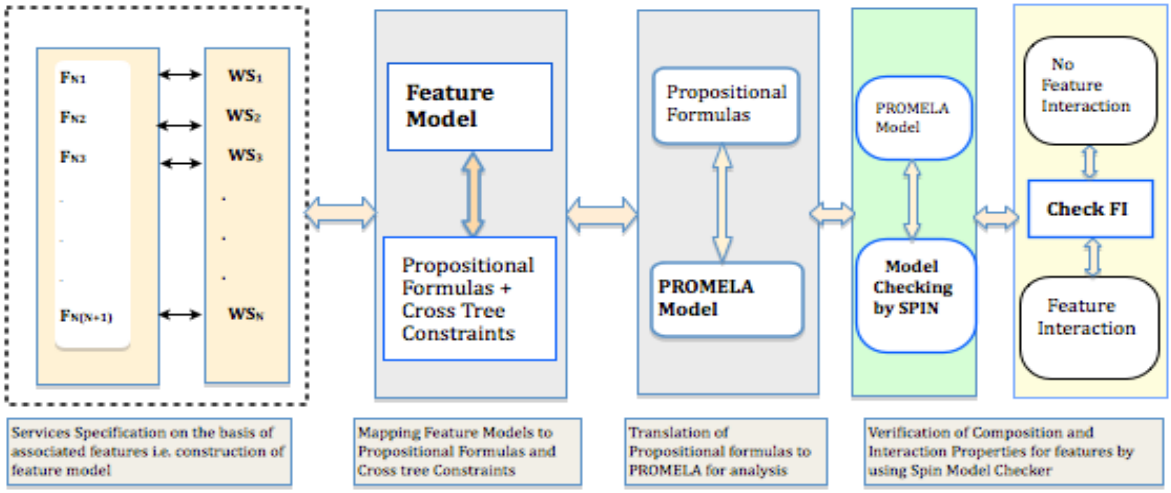


Fig. 2. Proposed Architecture for specification and analysis of feature interactions in web services

The proposed framework mainly deals with the problem of features specification and interactions in composite web services. We propose an efficient framework for analysis of feature interactions among the features of composite web services by using the feature modeling and model checking techniques. The feature model of composite service can be constructed by merging/composing the feature models of individual services (Segura, *et. al.,* 2007), (Acher, *et. al.,* 2009) and can be analyzed for checking the interactions among the specified features, whether they are composed according to the desired/specified rules, by using model checking. Hence by using the proposed approach a service developer can detect and resolve the possible undesired interactions in the feature model of composite service, at design time and a quality-oriented feature interaction free composite web service can be guaranteed for implementation.

The proposed framework consists of four phases as described in **(Fig. 2)**.

1) Construction of a feature model of a composite service.
2) Mapping the feature model to propositional formulas and constraints.
3) Translation of propositional formulas & constraints to create a Promela Model.
4) Analysis of feature interactions (described in the Promela) by using the Spin model checker.

A. Construction of feature model of composite service

The feature model of a composite service is constructed at this stage, modeling the individual component services and associated features in a tree-like structure called feature diagram. It can be considered as a domain model, specifying the features of services in a relational form, as desired by the user. It is assumed that the constructed feature model represents the composite service that is expected to provide all the required features.

B. Mapping feature model to propositional formulas and constraints

The feature model of a composite web service specifies the structure of a service and its component services and doesn't provide the semantic meanings of the modelled service, such as the relations among the services and combinations among the features of such services. To analyze the behavior of services and associated features, the feature model (domain model) is transformed into a set of propositional formulas and crosstree constraints among features. The semantics of a feature model are described in the form of propositional formulas, augmented with a set of rules to compose the features in a specific order. The Boolean Constraints Propagation (BCP) Algorithm is used to propagate the constraints related to the feature selection criteria. At this stage (BCP propagation) the possible conflicts/interactions among the features (selected) are exposed showing whether the constraints for feature selection are violated or satisfied. This transformation provides a sound basis for the further formal analysis of the feature model by using model checking techniques, described in the next phase. Algebra for the composition of features (Apel, *et. al.,* 2008) is described and different combinations of features are formed according to the algebraic laws for feature models.

C. Translation of propositional formulas and constraints to Promela Model

The semantics of a feature model, specified in the propositional logic are translated to the Promela Modeling language (Hemakumar, *et. al.,* 2008) to verify the specified properties for the composition of features. The properties for the desired combinations/interactions of features are described in LTL formulas and verified by model checker in next step. A logic for the combination of features is described here to check it whether this satisfy the expected model or not. The Promela program specifies a state-space of a feature model, showing the set of all states that can possibly occur during a computation. Using assertions in model checking checks the correctness properties for selection and composition of features.

D. Analysis of feature interactions by using Spin model checker

The feature model translated in the Promela is checked by the Spin model checker, to verify the specified correctness properties for feature interactions. The basic goal is to verify the error states corresponding to feature interactions that cannot be reached for a given feature model. In model checking, a state-space of a Promela program is generated in order to search for a counter example if one exists, to the correctness specification (Holzmann, *et. al.,* 2003). Assertions are used to check the defined properties of a model and Spin evaluates the assertions as part of its search space. If during the search it finds a computation leading to a false assertion then, either the program is incorrect or the assertion does not properly express a correctness property that holds for a model. The undesired interactions among the features can be identified in a counter-example, and can be resolved by

refactoring the feature model at propositional logic level by using the algebraic laws for refactoring the feature models (Gheyi, *et. al.,* 2008).

RESULTS AND DISCUSSION

Literature Survey and Related Research Work

The feature interaction problem was basically introduced in Telecommunication Systems by Bellcore, and considered an important obstruction for the development of new services (Zhang, *et. al.,* 2006) (Gibson, 1997). It mainly includes the interactions that occur due to incompatibility among the requirements of multiple features and a feature behaves differently in the presence of other features. A considerable number of research papers have discussed this issue in the context of telecommunication systems with respect to different perspectives. Interested readers are referred to (Keck, 1998), (Calder, 2003), (Gibson, 1997), and (Jackson, 1998). A substantial research survey on feature interactions in web services produces only a limited number of research papers providing a conceptual introduction and no formal and in-depth details about the composition and verification of such interactions among the features of services. A brief survey of research efforts to feature interactions in web services and related systems is given in following.

Michael Weiss (Weiss, *et. al.,* 2005), (Weiss, *et. al.,* 2007) first introduced the idea of feature interactions among web services by categorizing the features into two main categories i.e. Functional and Non-functional features. He presented a goal- oriented analysis and scenario modelling approach by using Goal-oriented Requirement Language (GRL) to detect feature interactions among functional and non- functional features of services. However this approach lakes a formal specification and analysis of feature interaction to verify that the specified model conforms to the desired goals.

(Reiff *et. al.,* and Gorton, *et. al.,* 2007) discuss feature interactions in business processes by integrating features (as policies) and service-targeted business processes as base systems and policies as feature mechanisms for specifying user-centric requirements and system variability. They discuss the different ways in which features (policies) interact with business processes and analyze such interactions in relation to other classical approaches such as POTS or telecommunication features. Authors in (Zhang , *et. al.,* 2007) have classified the feature interaction problem in web services in four major categories i.e. Deadlock, Loop-situation, Invocation error, Race condition and Resource contention. They translated the web service description into PROMELA model and verified the specific properties (specified in LTL) by using the SPIN model checker. However their work investigates the feature interaction problem at message interactions level in composed services. Similarly (Blair , *et. al.,* 2001) first introduced the problem of feature interactions at design time and runtime in middleware systems. They focus on techniques for specification of individual components / services integrated with their composition.

In (Classen, *et. al.,* 2007) authors proposed a problem-oriented feature interactions detection approach for software product lines, that is based on feature diagrams for capturing variability, problem diagrams for system's description in its context and event calculus for automated reasoning. Xuanzhe Liu (Liu, *et. al.,* 2005) explored the existence of feature interaction problem in middleware software systems by detecting ultimate causes of interactions in such systems at requirements level, in the context of feature interaction techniques in Telecommunication systems.

Jian-yin Zhang (Zhang, *et. al.,* 2006) investigate the problem of feature interaction in web services by proposing a novel multilayer detection system (WSFIDS), based on application of immune principles. They describe experimentally that by applying immune principle, the system provides an effective method to detect both known and previously unknown feature interactions. The authors in (Gheyi, *et. al.,* 2008) proposed a theory for feature models refactoring in the context of theorem proving PVS. In addition they presented a complete, sound catalog of algebraic laws, making up special feature models refactoring that preserve configurability of such models. The proposed PVS theory can be useful for the researchers who wish to prove different properties about feature models. They implied a reduction strategy to reduce every possible feature model to propositional formulae by using the algebraic laws. The application of reduction strategy aims to show that the two feature models in the propositional logic are equivalent. However the presented theory only supports the refactoring of basic feature models and doesn't handle cardinality-based feature models. Mathieu Acher (Acher, *et. al.,* 2009) presented a set of composition operators (Merge and insert) for composing the feature models with respect to different concerns by preserving the overall properties of such models. Each operator states where it is applied, what features will be composed and how the composition is made. Each composition is defined by rules that formally describe the structure of the resulting feature model.

In (Hemakumar, *et. al.,* 2008) authors suggested a run-time approach to expose contradiction in feature models statically using model checking and an incremental algorithm. They presented their experimental results by finding contra- dictions in different feature models and reported that model checking could be used effectively but is slow due the large size of feature model. They developed an Incremental Consistency Algorithm (ICA), which increasingly verified stronger properties of contradiction freedom. According to their results the ICA is bit faster than the model checking approach.

CONCLUSIONS AND FUTURE WORK

The term feature refers to an aspect of a software system either functional or non-functional. Feature Interaction is basically a software engineering concept that occurs when the value of one feature would affect (positively or negatively) the behavior of the other features. Negative feature interaction has been first introduced as a problem in the Telecom systems, but a closer analysis and comparison of such systems to other software systems such as Services Oriented Architectures, shows that, feature interaction is also inevitable in these systems due to their flexible nature. We are investigating the feature interaction and composition mechanisms for web service com- position processes, because an in-depth analysis of such composition processes reveals that a variety of feature interactions (message interaction, QoS feature interaction etc.) occur among the features of composed services. We are taking the both (functional and quality of service) features of web services in to consideration and propose a flexible framework for their specification, interaction, composition and verification, by integrating the Feature Modeling and Model checking techniques.

According to our observation, the proposed architecture should be more scalable and flexible as compared to the previous developed architectures, and over- come two major issues related to these approaches. i.e. providing flexible specification and modelling sup- port for the feature of web services (Feature Modelling) and verification, for the correct interaction and composition of features in composite services. This paper is an initial step towards developing the pro- posed framework and the basic aim of writing it is to get some valuable response and suggestions from the research community. According to our best knowledge no work has done to implement the idea of using feature models for analysis of feature interactions in web services augmented with exploiting the model checking techniques.

In future, we are working towards the implementation of proposed architecture in the context of different scenarios from real world, and developing a prototype system that will be able to demonstrate the suitability of this architecture.

ACKNOWLEDGEMENT

This is the extended version of our own paper presented and published as Conference proceedings in "International conference on Compute and Emerging Technologies (ICCET 2011)" held on 22-23 April 2011 at Shah Abdul Latif University, Khairpur, Sindh, Pakistan.

REFERENCES

Classen, A. (2007) Problem-oriented feature interaction detection in software product lines. In L. du Bousquet and J.L. Richier, editors, ICFI, IOS Press, 203-206.

Hemakumar, (2008) Finding contradictions in feature models. In S. Thiel and K. Pohl, editors, SPLC (**2**): Lero Int. Science Centre, University of Limerick, Ireland. 183-190.

Benavides, D., P. T. Martın-Arroyo, and A. R. Cortes. (2005) Automated reasoning on feature models. In O. Pastor and J. F. E. Cunha, editors, CAiSE, of Lecture Notes in Computer Science, Vol. **(3520)** 491-503.

Keck D. O. and P. J. Kuhn. (1998) The feature and service interaction problem in telecommunications systems. A survey. IEEE Trans. Software Eng, 24 (**10**): 779–796.

Pulvermueller, E., A. Speck, J. O. Coplien, M. D' Hondt, W. Demeuter, and A. Hrsg. (2001) Feature interaction in composed systems. Proceedings. ECOOP 2001 workshop 08 in association with the 15th European conference on object-oriented programming, Budapest, Hungary, June 18-22, 2001. Monograph, Universitat Karlsruhe, Aug. 02 2007.

Zhang J.Y. and S. Sur. (2007)Detecting feature interactions in web services with model checking techniques. The Journal of China Universities of Posts and Telecommunications, 14 (**3**): 323-334 .

Holzmann. G. (2003) The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, Boston, Massachusetts, USA.

Zhang. L. J. (2007) Services computing. 349Pp.

Bosch. J. (2000) Design and Use of Software Architectures. Addison-Wesley.

Pang J. and L. Blair. (2003) Separating interaction concerns from distributed feature components. Electr. Notes Theor. Comput. Sci, 82 (**5**): 33-42 .

Zhang, J., F. Yang, and S. Su. (2006) Detecting the web services feature interactions. In K. Aberer, Z. Peng, E. A. Rundensteiner, Y. Zhang, and X. Li, editors, WISE, vol. of Lecture Notes in Computer Science, (**4255**): 169–174.

Kang, K., S. Cohen, J. Hess, W. Novak, and S. Peterson. (1990) Feature-oriented domain analysis (FODA) feasibility study. Technical Report CMU/SEI-90-TR- 21, Software Engineering Institute, Carnegie Mellon University.

Blair, L., G. S. Blair, J. Pang, and C. Efstratiou. (2001) Feature interactions outside a telecom domain. In E. Pulvermuller, A. Speck, J. Coplien, M. D'Hondt, and W. D. Meuter, editors, FICS, of Technical Report, Univ. of Karlsruhe, Institut fur Programmstrukturen und Datenorganisation,. vol. (**14**): 15-20.

Chung. L. (1991) Representation and utilization of non- functional requirements for information system design. In CAiSE, 05-30.

Acher, M., P. Collet, P. Lahire, and R. France.(2009) Composing Feature Models. In 2nd International Conference on Software Language Engineering (SLE'09), LNCS, LNCS, Oct. 20 Pp.

Calder, M., M. Kolberg, E. H. Magill, and S. Reiff-Marganiec. (2003) Feature interaction: a critical review and considered forecast.

Beek, M. H. T., A. Bucchiarone, and S. Gnesi. Formal (2008) methods for service composition.

Jackson M. and P. Zave, (1998) Distributed feature composition: A virtual architecture for telecommunications services. IEEE Transactions on Software Engineering, Special Section: Managing Feature Interactions in Telecommunications Software Systems. 24 (**10**): 831-847.

Pistore, M., P. Traverso, and P. Bertoli. (2005) Automated composition of web services by planning in asynchronous domains. In S. Biundo, K. L. Myers, and K. Rajan, editors, ICAPS, AAAI, 02–11.

Riebisch. M. (2003) Towards a More Precise Definition of Feature Models, Books On Demand Publ. Co. Norderstedt, 64-76.

Weiss M. and B. Esfandiari.(2005)On feature interactions among web services. Int. J. Web Service Res. 2 (**4**): 22-47.

Weiss, M., B. Esfandiari and Y.Luo. (2007) Towards a classification of web service feature interactions. Computer Networks, 51 (**2**): 359-381.

Gibson. P. (1997) Feature requirements models: Understanding interactions. In Feature Interactions in Telecommunication Networks, IOS Press, 46-60..

Hofner P.and B. Moller. An extension for feature algebra. In S. Apel, W. R. Cook, K. Czarnecki, C. Kastner, N. Loughran, and O.erstrasz, (2009) editors, FOSD, ACM International Conference Proceeding Series, ACM, 75-80.

Gheyi, R., T. Massoni, and P. Borba. (2008) Algebraic laws for feature models. Journal of Universal Computer Science, 14 (**21**): 3573-3591.

Apel, S., C. Lengauer, B. Moller, and C. Kastner. (2008) An algebra for features and feature composition. In J. Meseguer and G. Rosu, editors, AMAST, of Lecture Notes in Computer Science, vol. (**5140)**: 36-50.

Gorton S. and S. Reiff-Marganiec. (2007) Towards feature interactions in business processes. In L. du Bousquet and J.L. Richier, editors, ICFI, 99-113. IOS Press,.

Segura, S., D. Benavides, A. R. Corte's, and P. Trinidad. (2007) Automated merging of feature models using graph transformations. In R. Lammel, J. Visser, and J. Saraiva, editors, GTTSE, of Lecture Notes in Computer Science, vol. (**5235**): 489–505.

Liu, X., G. Huang, and H. Mei. (2005) Feature interaction problems in middleware services. In S. Reiff-Marganiec and M. Ryan, editors, FIW, 313-319. IOS Press.