Pervasive Services Engineering for SOAs

Dhaminda Abeywickrama and Sita Ramakrishnan

Clayton School of Information Technology, Monash University, VIC, Australia
{dhaminda.abeywickrama,sita.ramakrishnan}@infotech.monash.edu.au

Abstract. With the proliferation of ubiquitous computing devices and mobile internet, it is envisaged that future pervasive services will be increasingly large-scale and operate at an inter-organizational level. Designing and implementing pervasive services will therefore become a more complex and challenging task. Significant interest exists within the pervasive computing community for representing pervasive services at different stages of the software life cycle. While most of these efforts have focused on the detailed design or implementation of pervasive services little work has been done at the software architectural level. In this research, we propose a novel approach based on behavioral modeling and analysis techniques for representing pervasive software services and their compositions and verifying the process behavior of these models against specified system properties. This systematic, architecture-centric approach combines the benefits of principles such as UML, model transformation of Model Driven Architecture, and formal behavioral modeling and analysis techniques through model-checking, for engineering pervasive software services. In order to illustrate the validity and practical feasibility of the proposed approach we use an existing case study in transport and logistics. The approach will be evaluated to demonstrate the effectiveness of model-checking as a technique for verifying pervasive software services.

1 Introduction: Motivation and Research Objectives

Mark Weiser's vision [1] of ubiquitous computing has been the impetus behind the introduction of a new service-oriented computing paradigm known as pervasive services. A pervasive service is a special type of service that uses context information to provide relevant information or services to users, where relevance depends on the users' tasks. Ubiquitous environments facilitate the collection of information from various data sources in order to aggregate the context of entities, such as users, places or objects. The context obtained from these sources can be used to automatically adapt a service's behavior or the content it processes to the context of one or several parameters of a target entity in a transparent way, resulting in pervasive services [2]. With the advancement of ubiquitous computing devices and mobile internet, it is envisaged that future pervasive services will be large-scale and operate at an inter-organizational level, with an increasing number of actors and constraints involved [3]. The real-time requirements, highly dynamic nature, quality of context information and automation further contribute to making pervasive services complex and challenging compared to

conventional services. Thus, the design and implementation of pervasive services will be a more complex task. Significant interest exists within the pervasive computing community for representing pervasive services at different stages of the software life cycle. However, so far most of the research work has been focused on the detailed design or implementation stages [4, 5] of the software life cycle while little attention has been given to the initial phase of design such as architecture design, providing the motivation for this research.

The use of models has been a popular approach for engineers when constructing complex systems. Behavior modeling and analysis have been successfully used by software engineers to uncover errors of concurrent and distributed systems at design time. In this research, we propose a novel approach based on behavioral modeling and analysis techniques for modeling pervasive software services and their compositions and verifying the process behavior of these models against specified system properties. This systematic, architecture-centric approach combines the benefits of principles such as UML, model transformation techniques of Model Driven Architecture (MDA), and formal behavioral modeling and analysis techniques using model-checking, for engineering pervasive services (research objectives: Fig. 1a). Context-handling information is considered to be tightly coupling or crosscutting the core functionality of a service at service interface level [6]. In this research, the crosscutting context-dependent functionality of the interacting pervasive services is modeled as aspect-oriented models in UML. In order to facilitate formal behavioral analysis, these UML models are automatically translated to state machine based behavioral representations using transformation authoring of MDA. The behavioral modeling and analysis approach used in this research is particularly based on formal verification, validation and simulation techniques provided by the model-checker, the Labeled Transition System Analyzer (LTSA) [7] and its process calculus Finite State Processes (FSP). We use an existing case study in transport and logistics to explore the approach and to illustrate its validity and practical feasibility. The approach will be evaluated to demonstrate the effectiveness of model-checking as a technique for detecting design defects in pervasive service specifications.

Preliminary results of this research have been published in [8, 9]. While this paper discusses the overall research in general it particularly reports on the model transformation tool created using IBM Rational Software Architect 7.0 [10], and further research directions established on formal verification and validation of the research. Section 2 provides a brief analysis of the related work. In Section 3, the overall research methodology of the study is discussed, and Sections 3.1 and 3.2 present the model transformation tool created, and model verification steps proposed as future work, respectively. Finally, Section 4 concludes the paper.

2 Related Work

Previous approaches to the development of pervasive services have largely been at the detailed design or implementation stages [4, 5] of the software life cycle. Luo et al. [4] establish a framework that enables context-aware composition of

Web services taking into account both the user's and the service's context when composing services. In [5], the authors propose an approach to include context in the composition of Web services. However, the approach taken in this research is clearly distinctive from the above approaches as it is based at the software architectural level which is higher level and abstract design. A similar approach to this, where concurrent behavior between base programs and aspects has been modeled using FSP semantics is provided in [11]. However, our work differs significantly as it is based on pervasive services. In [6], the authors present an approach on model driven development of pervasive services using UML and models crosscutting pervasive concerns as aspects. However, they have taken no account of concurrency and distributed notions in their design or any formal verification aspects through techniques such as model-checking as proposed in this research.

3 A Methodology for Pervasive Services Engineering

In this section, we provide an overview of the research methodology applied for engineering pervasive software services. Next we present the model transformation tool implemented using IBM Rational Software Architect, and further research directions established on model verification and validation, as part of the research methodology. The research approach is explored using a real-world case study in intelligent tagging for transport and logistics called the ParcelCall project [12], which is a European Union project within the Information Society Technologies program. The case study describes a scalable, real-time, intelligent, end-to-end tracking and tracing system using radio frequency identification, sensor networks, and services for transport and logistics. A significant subset of the ParcelCall case study is exception handling that needs to be enforced when a transport item's context information violates acceptable threshold values. This research is focussed on this subset.

The overall pervasive service-oriented development process is divided into three main stages (Fig. 1b). First, using the case study subset, a service specification for the system under consideration is defined. For this purpose, the relevant use cases for the case study subset are determined and the services that realize the identified use cases are specified. The identified use cases and their relationships are expressed using an UML use case diagram. As defined by Kruger et al. [13], we identify a software service as the interaction pattern or the interplay of several components collaborating to complete a desired task. Furthermore, we identify services as first-class modeling elements as opposed to first-class implementation elements, such as Web services. Message sequence charts (MSCs) provided by the LTSA-MSC tool [14], which is an extension to the LTSA tool, are used to describe the interaction patterns defining the services for the case study subset.

Second, the architecture for the system under consideration is defined. To this end, first a component configuration that implements the extracted services is defined using an UML deployment diagram. The component configuration of

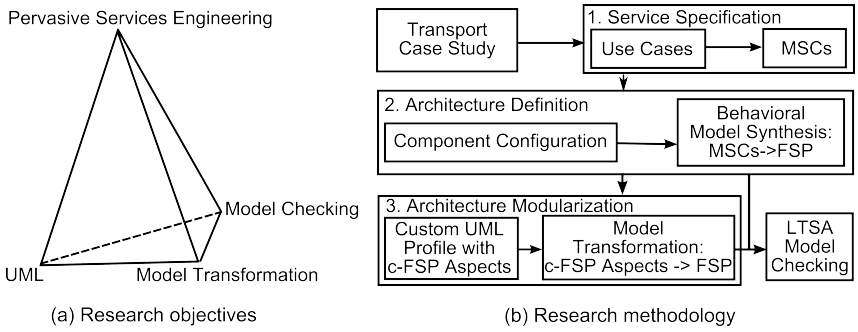

Fig. 1. Research objectives and methodology.

the approach is based on a distributed version of the Observer pattern called the Event-Control-Action architecture pattern [15]. Second, a behavioral representation of the service specification in the form of FSPs (architecture model) is generated automatically using the LTSA-MSC tool's FSP synthesis feature. The architecture model provides the basis for modeling and reasoning about the system design where the components are modeled as labeled transition systems.

Third, the architecture model synthesized in the previous step is modularized by applying separation of concerns. Context-handling information is considered to be crosscutting the core functionality of a service at service interface level, which results in a complex design that is hard to implement and maintain. Therefore, a custom UML profile is proposed from which a UML model template and aspect-oriented models in UML (contextual-FSP or c-FSP aspects) [9] are derived for the case study subset. A custom prototype tool has been implemented to automate the translation of the c-FSP aspects to formal FSP to facilitate rigorous verification by the LTSA. Details of this tool implementation and the proposed model verification process using the LTSA are discussed next.

3.1 Model Transformation

The custom prototype tool has been built using the Java Emitter Templates (JET) transformation authoring feature of IBM Rational Software Architect (7.0) [10]. JET is an open source technology developed by IBM and is part of several IBM Rational modeling platform 7.0 products. The transformation is used to automate the application of design patterns and generate infrastructure code for the c-FSP aspects using FSP semantics (aspectual FSP generation). An added benefit of applying JET transformation is that it allows the end-user modification of the code generator using templates. The application of model driven development in pervasive services engineering at state machine level is novel. The main benefits of this approach are improving the quality and productivity of service development, easing system maintenance and evolution, and increasing the portability of the service design. Two variations of the model transformation tool have been built. Initially, a model-to-text transformation was implemented with

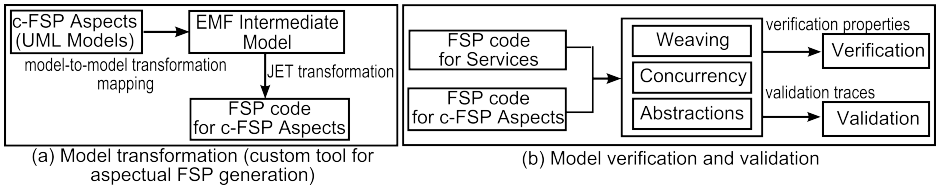

Fig. 2. Model transformation, model verification and validation.

XPath expressions to navigate the UML models for c-FSP aspects and extract model information dynamically to the transformation. However, JET's support for UML models has two main limitations. First, using JET it is not possible to access stereotype information of the UML models although custom XPath functions can be written to perform this, and second JET authoring finds the complexity of the UML meta-model hard to manage. Therefore, a more effective solution was implemented by first creating a model-to-model mapping transformation which extracts relevant stereotype information from the UML models, and builds a code-generator specific intermediate Eclipse Modeling Framework model which can then be consumed by the JET transformation. Thus, a multi-stage transformation (Fig. 2a) has been used to transform the UML models to formal FSP semantics.

3.2 Model Verification and Validation

This section presents a discussion on the proposed future work of this research. For model verification (Fig. 2b), first the generated FSP for the c-FSP aspects need to be woven with their base state machines using an explicit weaving mechanism. This can be modeled as the parallel composition of the aspectual and base state machines in FSP. Synchronization events can be introduced to control the coordination and weaving order between the base program and aspects. Concurrency and distributed notions will be added to the interacting pervasive software services and their compositions, such as shared objects, message passing and concurrent architectures. In addition, proper abstraction mechanisms need to be enforced on the models as part of preparation for verification. Formal model-checking techniques provided by the LTSA, such as safety, progress and absence of deadlocks, will be used to verify the process behavior of the services against system properties specified to extensively cover the system requirements. The LTSA checks for property violations and if any violations are found it produces a trace to the violation known as a counterexample, which can be used to improve the state models or the system properties for the pervasive services. Validation of the models will be performed using the simulation and animation features of the LTSA. The research approach will be evaluated to demonstrate the effectiveness of model-checking as a technique for verifying pervasive software service specifications. Potential defects can be injected to the service specification and results can be compared with the defect free service specification.

4 Conclusion and Future Work

To summarize, the primary research objective of this research is a systematic, architecture-centric approach for engineering pervasive software services based on the principles of UML, model transformation techniques, and formal model-checking. Future work of this research involves model verification and validation as discussed in the paper.

References

1. Weiser, M.: The Computer for the 21st Century. Scientific American, 265(3), 94–104 (1991)
2. Hegering, H. G., Kupper, A., Linnhoff-Popien, C., Reiser, H.: Management Challenges of Context-Aware Services in Ubiquitous Environments. In: Goos, G., Hartmanis, J., van Leeuwen, J. (eds.) DSOM 2003. LNCS, vol. 2867, pp. 246–259. Springer, Heidelberg (2003)
3. Buchholz, T., Kupper, A., Schiffers, M.: Quality of Context: What It Is And Why We Need It. In: 10th Workshop of the HP OpenView University Association (HPOVUA'03). (2003)
4. Luo, N., Yan, J., Liu, M., Yang, S.: Towards Context-Aware Composition of Web Services. In: Fifth International Conference on Grid and Cooperative Computing. pp. 494–499. (2006)
5. Mostefaoui, S. K., Hirsbrunner, B.: Context-Aware Service Provisioning. In: IEEE/ACS International Conference on Pervasive Services. pp. 71–80. (2004)
6. Sheng, Q. Z., Benatallah, B.: ContextUML: a UML-based Modeling Language for Model-Driven Development of Context-Aware Web Services. In: International Conference on Mobile Business (ICMB'05). pp. 206–212. (2005)
7. Magee, J., Kramer, J.: Concurrency: State Models and Java Programs, 2nd Edition. John Wiley and Sons, Worldwide Series in Computer Science (2006)
8. Abeywickrama, D., Ramakrishnan, S.: A Model-Based Approach for Engineering Pervasive Services in SOAs. In: 5th ACM International Conference on Pervasive Services (ICPS'08). pp. 57–60. (2008)
9. Abeywickrama, D., Ramakrishnan, S.: Towards Engineering Models of Aspectual Pervasive Software Services. In: 3rd ACM Workshop on Software Engineering for Pervasive Services (SEPS'08). pp. 3–8. (2008)
10. IBM Rational Software Architect Transformation Authoring, `http://publib.boulder.ibm.com/infocenter/rsdhelp/v7r0m0`
11. Douence, R., Le Botlan, D., Noye, J., Sudholt, M.: Concurrent Aspects. In: 5th International Conference on Generative Programming and Component Engineering (GPCE'06). pp. 79–88. (2006)
12. Davie, A.: Intelligent Tagging for Transport and Logistics: The ParcelCall Approach. Electronics and Communication Engineering Journal 14(3), 122–128 (2002)
13. Kruger, I. H., Mathew, R., Meisinger, M.: Efficient Exploration of Service-Oriented Architectures using Aspects. In: 28th International Conference on Software Engineering. pp. 62–71. (2006)
14. Uchitel, S., Kramer, J., Magee, J.: Synthesis of Behavioral Models from Scenarios. IEEE Transactions on Software Engineering 29(2), 99–115 (2003)
15. Costa, P. D., Pires, L. F., van Sinderen, M.: Architectural Patterns for Context-Aware Services Platforms. In: 2nd International Workshop on Ubiquitous Computing. pp. 3–19. (2005)