AN EVALUATION FRAMEWORK FOR VALIDATING ASPECTUAL PERVASIVE SOFTWARE SERVICES

Dhaminda B. Abeywickrama and Sita Ramakrishnan
Faculty of Information Technology, Monash University, Clayton Campus, Melbourne, Australia

Keywords: Pervasive services, Model-driven development, Model checking, Aspect-oriented modeling.

Abstract: Context-dependent information has several qualities that make *pervasive services* challenging compared to conventional Web services. Therefore, *sound software engineering practices* are needed during their development, execution and *validation*. This paper establishes a framework to evaluate pervasive service-oriented software architectures. The method of evaluation is based on key features comparison. The framework consists of two views: vertical and horizontal. The vertical evaluation compares several research tools to the `Aspectual FSP Generation tool` developed in this research. The tools are compared across the platform-independent and platform-specific levels of the model-driven architecture. The horizontal evaluation view is designed to validate several desired key features that are mainly required at the platform-specific level of the service specification. These criteria mainly cover two aspects: formal methods and tools employed, and the context and adaptation dimensions of the customization approach used in the services. The vertical evaluation has demonstrated that the `Aspectual FSP Generation tool` has unique features in context-dependent behavioral modeling and code generation. The horizontal evaluation has shown that the formal methods and tools employed, and the customization approach used in the services are effective towards the overall objectives of this research. The approach is explored using a real-world case study in intelligent transport.

1 INTRODUCTION

A *pervasive service* is a special type of service that adapts its behavior or the content it processes to the context of one or several parameters of a target entity in a transparent way (e.g. restaurant finder services, attractions and activities recommendation services) (Hegering et al., 2003). With the proliferation of ubiquitous computing devices and Internet, pervasive services continue to evolve from simple proof of concept implementations created in the laboratory to large and complex real-world services developed in industry. Context-awareness capabilities in service interfaces introduce additional challenges to the software engineer. Context information is characterized by several qualities that make pervasive services challenging compared to conventional Web services, such as a highly dynamic nature, real-time requirements, quality of context information and automation. The additional complexities associated with these special services necessitate the use of *solid software engineering methodologies* during their development, execution and *validation*. Most state-of-the-art approaches to pervasive services relate to the detailed design or implementation stages (Mandato et al., 2002; Mostefaoui and Hirsbrunner, 2004) of the software lifecycle, such as pervasive Web services. Little work focuses on the early phase of design such as architecture design, which is of a higher level and abstract in design.

This systematic, architecture-centric approach (Abeywickrama, 2010; Abeywickrama and Ramakrishnan, 2010; Abeywickrama and Ramakrishnan, 2008b; Abeywickrama and Ramakrishnan, 2008a) for modeling and verifying pervasive services integrates the benefits of sound software engineering principles such as model-driven architecture, aspect-oriented modeling, and formal model checking. It adopts model-driven development to represent complex crosscutting context-dependent functionality in service interfaces in a modular manner, and to automate the generation of state machine-based adaptive behavior. The crosscutting context-dependent information of the interacting pervasive services is modeled as aspect-oriented models in UML. Using model transformations (`Aspectual FSP Generation` tool), we

B. Abeywickrama D. and Ramakrishnan S..
AN EVALUATION FRAMEWORK FOR VALIDATING ASPECTUAL PERVASIVE SOFTWARE SERVICES.
DOI: 10.5220/0003461200800091
In *Proceedings of the 6th International Conference on Evaluation of Novel Approaches to Software Engineering* (ENASE-2011), pages 80-91
ISBN: 978-989-8425-57-7

ensure the correct separation of concerns of the cross-cutting context-dependent functionality at both semi-informal UML modeling and formal behavioral specification levels. The generated context-dependent adaptive behavior and the core service behavior for the pervasive services are rigorously verified using formal model checking against specified system properties.

This paper establishes a framework to evaluate pervasive service-oriented software architectures. The method of evaluation is based on key features comparison. The evaluation framework consists of two dimensions or views: vertical and horizontal. The vertical evaluation compares several research tools to the `Aspectual FSP Generation tool` developed in the current research. The tools are compared across the platform-independent model (PIM) and platform-specific model (PSM) levels of the model-driven architecture (MDA). The horizontal evaluation view is designed to validate several desired key features that are mainly required at the PSM level of the aspectual pervasive software services specification. These criteria mainly cover two aspects: formal methods and tools employed, and the context and adaptation dimensions of the customization approach used in the services. The vertical evaluation has demonstrated that the `Aspectual FSP Generation tool` has unique features in context-dependent behavioral modeling and code generation. The horizontal evaluation of the approach has shown that the formal methods and tools employed, and the customization approach used in the services are indeed effective towards the overall objectives of this research. The approach is explored using a real-world case study in intelligent tagging for transport.

The rest of the paper is organized as follows. Section 2 provides background information on this study. An overview of the evaluation framework established here is provided in Section 3. In Section 4 vertical evaluation of the research is discussed while Section 5 addresses the horizontal evaluation. Section 6 concludes this paper with a brief analysis of the evaluation results.

2 PRELIMINARIES

This section provides background information on (1) the overall pervasive services engineering process, (2) the case study, and (3) the context-dependent adaptive behavior generation process applied in the research.

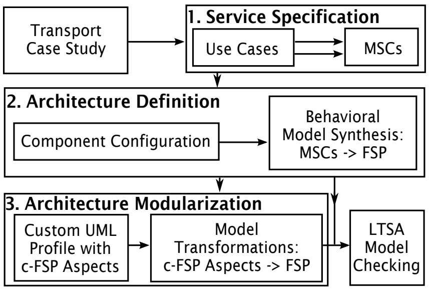
Transport Case Study
1. Service Specification
Use Cases
MSCs
2. Architecture Definition
Component Configuration
Behavioral Model Synthesis: MSCs -> FSP
3. Architecture Modularization
Custom UML Profile with c-FSP Aspects
Model Transformations: c-FSP Aspects -> FSP
LTSA Model Checking

Figure 1: Pervasive services engineering.

2.1 Pervasive Services Engineering

The overall pervasive service-oriented development process is divided into three stages (Figure 1) (Abeywickrama and Ramakrishnan, 2008b; Abeywickrama and Ramakrishnan, 2008a). First, using the case study we extract use cases and define a service specification for the system under consideration using message sequence charts. Second, the architecture for the system is defined using a component configuration and an architecture model in Finite State Processes (FSP) using the LTSA-MSC tool. Third, the architecture model synthesized from the previous step is modularized with aspect-oriented models in UML called the `contextual-FSP aspects` (`c-FSP aspects`), and automatically transformed into FSP before applying model checking using the Labeled Transition System Analyzer tool (LTSA).

2.2 Case Study

The research approach is explored using a real-world case study in intelligent tagging for transport known as the ParcelCall project. ParcelCall (Davie, 2002) 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 (RFID), sensor networks, and services for transport and logistics. This case study is particularly appealing to the current research as it provides several scenarios for representing software services that interoperate in a pervasive, mobile and distributed environment. 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. The reference scenario used here describes an `awareness monitoring and notification pervasive service`, which alerts with regards to any exceptional situations that may arise on transport items, primarily to the vehicle driver of the transport unit. The threshold values for environment status (e.g., temperature, pressure, acceleration) of transport items and route (location) for the vehicle are set by the carrier organization in advance. The service alerts if items' environment status exceeds acceptable levels or if an item is lost or stolen during transport. The primary context parameters modeled in the study include item identity, location, temperature, pressure and acceleration.

2.3 Adaptive Behavior Generation

The *notion of context* used in this research is based on a definition provided by ()analyti for context in information modeling. They describe context as a set of objects, each of which is associated with a set of names and another context called its reference. Furthermore, they enhance the definition for context by stating that each object of a context is either a simple object or a link object (attribute, instance-of, ISA) and each object can be related to other objects through attribute, instance-of or ISA links. (Analyti et al., 2007) use traditional object-oriented abstraction mechanisms of attribution, classification, generalization and encapsulation to structure the contents of a context.

The model transformation tool created in our study is called the `Aspectual FSP Generation tool`. The transformations have been applied to the reference scenario in intelligent transport. We use model transformations to automate the application of design patterns and generate infrastructure code for the `c-FSP aspects` using FSP semantics. The current study explores the strengths of both semi-informal UML meta-level extensions and formal finite state machines for representing the context-dependent behavior of software services, and model transformation techniques are applied as a bridge to enforce correct separation of concerns between these two design abstractions. 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 for the pervasive services engineer.

This approach focuses on the application of model-driven development for engineering pervasive services at finite state machine level. An aspect in FSP can be identified as an independent finite state machine that executes concurrently and synchronizes with its base state machine. In general, an aspect in FSP needs to contain synchronization events (transitions) to coordinate with its base state machine and other aspects. Also, each aspect type (e.g. `context`, `trigger`, and `recovery`) contains its unique constructs which can be generated automatically using model transformation techniques. For example, a `trigger aspect` requires constructs to alert and send notifications while a `recovery aspect` needs constructs to recover from exception-handling situations. On the other hand, a `context aspect` has attribution, instance-of, ISA, and reference constructs from the notion of context applied here. In Figure 2, the models and activities of the development process are represented as ellipses and square boxes respectively. The development process is structured into three main flows of activities. `Flow 1` and `Flow 2` extensively apply model transformations where `Flow 1` uses a model-to-text JET transformation and `Flow 2` applies an effective pipeline of model-to-model and model-to-text JET transformations. Both `Flow 1` and `Flow 2` originate from the `c-FSP-UML profile`. `Flow 3` represents activities involved for rigorously verifying the context-dependent adaptive behavior and the core service behavior of the pervasive software services using formal model checking.

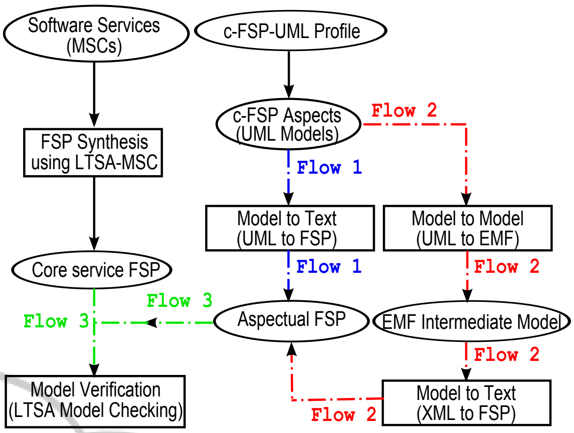
Software Services (MSCs)
c-FSP-UML Profile
FSP Synthesis using LTSA-MSC
c-FSP Aspects (UML Models)
Flow 2
Flow 1
Model to Text (UML to FSP)
Model to Model (UML to EMF)
Flow 1
Flow 2
Core service FSP
Flow 3
Flow 3
Aspectual FSP
EMF Intermediate Model
Flow 2
Model Verification (LTSA Model Checking)
Model to Text (XML to FSP)
Flow 2

Figure 2: Adaptive behavior generation process (source: (Abeywickrama and Ramakrishnan, 2010)).

3 EVALUATION FRAMEWORK

This evaluation framework (Abeywickrama, 2010) mainly validates the main contributions or deliverables of this study against several key evaluation criteria. The main tools used in this study include the `Aspectual FSP Generation tool` created in this research, the LTSA model checker and the LTSA-MSC tool. First, this research has developed a custom tool (`Aspectual FSP Generation tool`) applying an effective pipeline of model-to-model and model-to-text JET transformations using the IBM Rational Software Architect to automate the application of de-

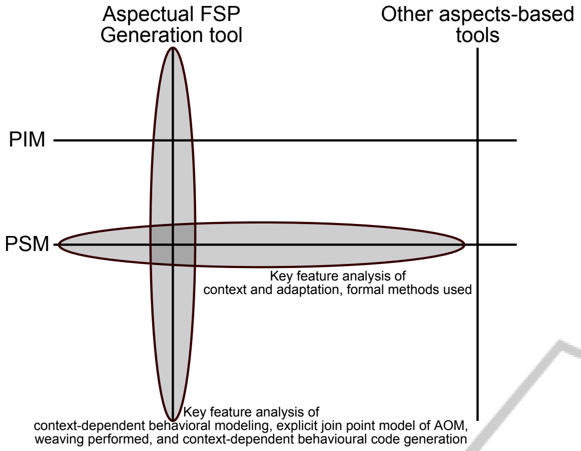
Aspectual FSP Generation tool
Other aspects-based tools
PIM
PSM
Key feature analysis of context and adaptation, formal methods used
Key feature analysis of context-dependent behavioral modeling, explicit join point model of AOM, weaving performed, and context-dependent behavioural code generation

Figure 3: Evaluation framework: vertical, horizontal views.

sign patterns and generate infrastructure code for the *aspects*. Second, this research has performed rigorous specification and verification of concurrent models of pervasive software services and their compositions using the LTSA model checker to assure the correctness and quality of the pervasive services design. Third, in this study the interaction patterns defining the pervasive software services of the specification have been modeled using the LTSA-MSC tool, from which a core service model was extracted. The evaluation framework established in this paper intends to validate the aforementioned three main deliverables against key evaluation criteria.

The method of evaluation used here is based on key features comparison. The evaluation framework developed here does not produce additions to the research methodology but instead validates the methods and tools used in the research as a whole. The evaluation framework consists of two views: *vertical* and *horizontal*. The prototype or proof of concept nature of the methods and tools employed in this research, however, makes it challenging to compare their benefits with the more advanced industry-based tools. Figure 3 illustrates the two views of the overall evaluation approach.

4 VERTICAL EVALUATION

This section discusses the vertical view of the evaluation framework (Abeywickrama, 2010). In the vertical view of the evaluation framework, several tools on aspect-oriented modeling (AOM) are compared with the `Aspectual FSP Generation tool` as a whole at the PIM and PSM levels of model-driven development. As the `Aspectual FSP Generation tool` covers several modeling layers of the MDA stack such as PIM and PSM levels, the evaluation process is essentially vertical in nature. The vertical evaluation essentially provides an analysis of several features of the `Aspectual FSP Generation tool` against several aspect-oriented modeling-based tools. This evaluation is based on the following criteria: context-dependent behavioral modeling at the PIM level, explicit joinpoint model of AOM at the PIM level, weaving performed at the PIM level or PSM level, and context-dependent behavioral code generation from the PIM level to PSM level. The results of this evaluation are presented in Tables 1- 4.

4.1 Aspectual FSP Generation Tool

The `Aspectual FSP Generation tool` created in this research (Abeywickrama, 2010; Abeywickrama and Ramakrishnan, 2010; Abeywickrama and Ramakrishnan, 2008b; VIDE, 2009) using IBM Rational Software Architect provides for context-dependent behavioral modeling at the PIM level, and context-dependent behavioral code generation from the PIM level to the PSM level of model-driven development. This tool effectively applies model-driven development in pervasive services engineering at the state machine level. Context-dependent behavior at the service interface level has been modeled using a custom UML profile called the `c-FSP-UML profile`, and aspect-oriented UML class models called the `c-FSP aspects`. The profile supports an explicit joinpoint model of AOM at the PIM level, and towards this the profile defines several stereotypes such as `Aspect`, `ContextAspect`, `TriggerAspect`, `RecoveryAspect`, `Advice` and `Pointcut`. The `Aspectual FSP Generation tool` can be employed to generate PSMs in formal behavioral specification level in FSP using an effective chain of model transformations. Model transformations are employed here to automate the application of design patterns and generate infrastructure code for the `c-FSP aspects` using FSP semantics. Also, using transformations the correct separation of concerns both at UML modeling and FSP behavioral specification levels is ensured. 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. Weaving between an aspect and a base state machine is performed using an explicit weaving mechanism at the executable state machine level in FSP. The context modeling and transformations features of the `Aspectual FSP Generation tool` have been explored using the reference scenario on intelligent transport.

4.2 Groher and Schulze's Approach

(Groher and Schulze, 2003) present an approach for specifying crosscutting concerns using aspect-oriented modeling and discuss the seamless integration of those models to implementation. In their approach, UML has been customized for supporting aspect-oriented modeling using UML's standard extension mechanisms, such as stereotypes, tagged values and constraints. Their design notation for aspect-oriented modeling provides a base package for modeling the business logic, an aspect package for modeling the crosscutting concerns, and a connector (weaving rules) for linking the aspect and the base elements. The connector includes program execution points (pointcuts), and actions to be executed at those points (advices). The weaving in their approach is essentially performed at the PIM level and not at the PSM level as in the current study. The authors have implemented an AspectJ code generator using the CASE tool *Together* from Borland. In their tool aspect-oriented validation and code generation in AspectJ have been implemented as modules. The authors' work (Groher and Schulze, 2003) is not based on pervasive services, which is a key difference to the current research. Also, the code generation is at the implementation level with AspectJ whereas in the current research it is at the architectural level with FSP.

4.3 Whittle and Jayaraman's Approach

(Whittle and Jayaraman, 2008) present a UML-based aspect-oriented modeling tool called MATA that applies graph transformations for specifying and composing aspect models. Their work is different to most other approaches on aspect-oriented modeling in three respects. First, there is no support for explicit joinpoints and composition is considered as a special form of model transformations. Second, the use of graph transformations for aspect composition, and third, support for statically analyzing aspect interactions using critical pair analysis, make their approach different to other approaches. In their approach (Whittle and Jayaraman, 2008), the composition of a base model and an aspect model (weaving) is specified using a graph rule. A main difference in the authors' approach is that graph rules have been defined over the concrete syntax of the modeling language and not at the meta-level as in most known approaches on model transformations. The authors use a cell phone example to demonstrate the validity of their method and tool. Tool support for MATA has been built using IBM Rational Software Modeler. The tool uses graph transformation as its underlying theory for aspect composition. MATA uses AGG as its graph rule execution tool as it supports critical pair analysis. Similar to (Groher and Schulze, 2003)'s approach, MATA is not based on pervasive services. The authors' work (Whittle and Jayaraman, 2008) does not support explicit joinpoints as provided in the current research. Also, no code generation of the models has been provided in their work.

4.4 Cottenier *et al.* Approach

()cottenierMotorolaWEAVRAspect present Motorola WEAVR, which provides aspect-oriented weaving for UML state charts that include action semantics. Motorola WEAVR is an industrial-strength aspect weaver for UML 2.0, which is implemented as an add-in to the Telelogic TAU G2. The tool essentially performs four main functions. First, the tool's profile allows engineers to define aspects in UML 2.0. Second, it presents a joinpoint visualization engine for visualizing and validating the effects of the aspects. Third, the tool provides full aspect weaving at the modeling level. Finally, the tool's simulation engine allows the simulation of the aspect models without breaking their modular structure. The authors have provided several UML stereotype classes for identifying various constructs of an aspect at the PIM level. Motorola WEAVR introduces two fundamental language constructs to support aspect-oriented modeling. First, the authors provide constructs to specify the locations or joinpoints in the models where crosscutting behavior emerges. In the approach, action joinpoints capture call expressions, timer set actions or constructor calls, whereas transition joinpoints capture sets of execution paths within a state machine. Second, the authors provide constructs to specify the actual behavior of the crosscutting concerns using the connector stereotype. Weaving process consists of two phases: advice instantiation and advice instance binding. As weaving is performed at the PIM level with aspects woven into executable UML models, it allows PSMs and source code to be generated automatically. Source code, which can be for example in C or C++, is generated using an optimizing code generator. Several examples on exception handling and recovery have been developed by the authors to demonstrate the validity of their approach. However, the authors' work (Cottenier et al., 2007) is not in the pervasive computing domain as in the current research. Also, their PSMs are not at the formal behavioral specification level as in our study.

4.5 Fuentes *et al.* Approach

()fuentes present an aspect-oriented executable modeling UML 2.0 profile called AOEM for designing pervasive applications. Using the AOEM profile, (Fuentes et al., 2008) demonstrate how aspect-oriented executable design models for context-aware pervasive applications can be constructed and executed. These models are run and tested using *Populo*, which is an eclipse plug-in created by the authors for interpreting executable UML models. The AOEM profile aims at addressing two challenges in pervasive applications. They are the crosscutting nature of context-awareness, and the complexity associated with reasoning about the correctness of the design model. In their approach, weaving of aspects to core components is performed at the PIM level. Special composition rules expressed as pointcuts in the AOEM profile describe how aspects are applied to the core components. An aspect-oriented model weaver, which is a type of compiler or preprocessor, has been developed for weaving. The weaver essentially creates the design model by injecting the crosscutting behavior of the aspects into the core modules that the aspects crosscut. The authors illustrate their approach using a location-aware intelligent transportation system. Although (Fuentes et al., 2008) provide for modeling of adaptive behavior at the PIM level, they do not provide any model transformations to generate PSMs. Also, in general, their context models are at the context-aware application and middleware levels and not at the service interface level (state machine level) as in the current study.

Tables 1- 4 show that the `Aspectual FSP Generation tool` satisfies all the criteria as opposed to the other tools.

Table 1: Comparison matrix: PIM level support for context-dependent behavioral modeling.

Evaluation Criteria (PIM level)	4.1	4.2	4.3	4.4	4.5
Context-dependent behavioral modeling	■				▨

Table 2: Comparison matrix: PIM level support for explicit joinpoint model of aspect-oriented modeling.

Evaluation Criteria (PIM level)	4.1	4.2	4.3	4.4	4.5
Explicit joinpoint model of AOM	■	■		■	■

Table 3: Comparison matrix: PIM level or PSM level support for weaving.

Evaluation Criteria (PIM level or PSM level)	4.1	4.2	4.3	4.4	4.5
Weaving	■	■	■	■	■

Table 4: Comparison matrix: PIM level and PSM level support for context-dependent behavioral code generation.

Evaluation Criteria (PIM level & PSM level)	4.1	4.2	4.3	4.4	4.5
Context-dependent behavioral code generation	■				

Complete	■
Partial	▨
No cover	

5 HORIZONTAL EVALUATION

This section describes the horizontal dimension of the framework (Abeywickrama, 2010). In contrast to vertical evaluation discussed above, horizontal evaluation validates only particular features at a specific level of abstraction of the MDA stack. The horizontal evaluation view is designed to validate several desired key features that are mainly required at the PSM level of the aspectual pervasive software services specification. These evaluation criteria cover two aspects of the study. They are the formal methods and tools employed in the study, and the context and adaptation dimensions of the customization approach used in the services.

5.1 Formal Methods and Tools Used

In this subsection, the formal methods and tools used in the current study at the PSM level of abstraction are evaluated. (Clarke et al., 1996) provide several criteria that formal methods-based approaches and tools need to support. According to ()clarkeFormal, although some of these criteria are ideals, it is still considered good to aim for them. As provided in Section 2.1, the research methodology of the current study contains three stages: service specification, architecture definition and architecture modularization. In the present study, formal methods and tools have been applied during the service specification and architecture definition stages of the research methodology, and finally for model checking the aspectual pervasive software services specification. The *LTSA-MSC tool* has been used for specifying the software services of the service specification and generating a behavioral model in FSP. Using this generated behavioral model, a core service model was extracted. The formal model checker, the *LTSA*, has been used to verify the aspectual pervasive software service specification against specified system requirements. This subsection evaluates the application of the aforementioned formal methods and tools in the current research against the criteria provided by (Clarke et al., 1996).

- *Early Payback.* Early payback is one of the key benefits of the current study. This study is focused on the architectural level of the software life-cycle. This architecture-centric approach builds models of pervasive software services and their compositions and verifies their behavior against specified system properties. Building architectural models of pervasive software services allows the software engineers to validate the actual correctness of the services before the services are implemented later in the software life-cycle. Thus, it provides early payback or feedback to the service engineer on the validity of the services.
- *Incremental Gain for Incremental Effort.* In the study, the PSMs of the aspectual pervasive software services specification have been derived following the three incremental stages of the research methodology: service specification, architecture definition and architecture modularization. Each of these stages has its own deliverables such as an message sequence chart specification for software services, architecture model for the software services in FSP, and a modularized architecture with aspect-oriented models to represent context-dependent behavior at the service interface level. This demonstrates that the service engineer can receive the benefits of the methodology in an incremental manner.
- *Multiple Use.* The pervasive services engineering methodology established in this research in general covers the requirements and architecture design stages of the software life-cycle. Therefore, the benefits of this methodology can be seen in multiple stages of the software life-cycle. This design methodology effectively facilitates the transition from requirements-oriented scenario descriptions of pervasive software services to architecture-centric behavioral models of aspectual pervasive software services.
- *Integrated Use.* The formal methods and tools used in this research are widely known in both academia and industry. First, this research has modeled the pervasive software services using message sequence charts provided by the LTSA-MSC tool. Message sequence charts are one of the most widely used sequence chart notations for describing system behavior. Second, the model checking tool employed in this study (LTSA tool) is widely used for behavior modeling and analysis and is well supported with documentation (Magee and Kramer, 2006).
- *Ease of Use.* This research applies three automated tools in the pervasive services engineering process: the LTSA-MSC tool to generate the architecture model in FSP which is later used to extract the core service model, the `Aspectual FSP Generation tool` to generate context-dependent behavior in FSP, and the LTSA tool for simulating, animating and model checking pervasive services. The use of automated tools such as the LTSA makes the engineering process much easier to understand and adopt for the service engineer.
- *Efficiency.* One of the limitations of the current study is efficiency in terms of time and space. This is mainly attributed to the formal model checking technique used for verifying the pervasive services specification. One of the main challenges associated with model checking technique is the state space explosion problem. Nevertheless, by using action hiding and minimization features of the LTSA model checker the present study has proven sufficiently efficient in modeling and verifying the case study subset of this research.
- *Ease of Learning.* The graphical interfaces provided in the LTSA-MSC tool and the `Aspectual FSP Generation tool` and the automated nature of these tools effectively reduce the need to know formal FSP to start realizing the benefits of this research. The use of basic and high-level message sequence charts in the service specification stage of the methodology are widely used and understood in both academia and industry. Also, the sequence chart notion and semantics applied in the LTSA-MSC tool are restricted to basic features. It does not include complex constructs such as message queues, gates, parametric messages, or dynamic creation and termination of instances.
- *Orientation Toward Error Detection.* The approach presented in this paper is oriented towards detecting errors in the aspectual pervasive software services specification using the formal model checking technique. Aspects can introduce an additional correctness problem in software specifications because of their crosscutting

effect and obliviousness nature. Therefore, tool support if possible automatic, is highly desirable to ensure the correctness of the specification. This research employs formal model checking to find errors concerning safety, progress, and fluent linear temporal logic assertions in the service specification, which can be hidden behind the individual components and aspects, or in the woven model. In the study, two techniques - counterexamples and witness executions - have been employed to point out any errors in the specification, which can be used by the service engineer to improve the state models or the system properties for the aspectual pervasive software services.

- *Focused Analysis.* This research is explored using a modified subset of a real-world case study called the ParcelCall project. The case study subset focuses on exception handling that needs to be enforced when a transport item's context information violates acceptable threshold values. The reference scenario used in the research describes an `awareness monitoring and notification pervasive service`, which alerts with regards to any exceptional situations that may arise on transport items primarily to the vehicle driver of the transport unit. This is an example where the research is focused on analyzing only a particular aspect of the system and not the entire system. Also, at the PSM level only temperature and pressure context properties have been modeled and verified. Similarly, other context properties such as item identity, location and acceleration can also be supported.

- *Evolutionary Development.* The incremental and iterative nature of the activities performed in each stage of the methodology essentially facilitates evolutionary system development. This can be demonstrated by the fact that the engineering process, which is initiated as a scenario-based specification expressed as message sequence charts, has evolved into a modularized service architecture where complex context-dependent information has been separated from the core service behavior as aspect-oriented models.

5.2 Context and Adaptation of the Customization Approach

This subsection evaluates the customization approach used in the pervasive services of the current study. This evaluation focuses mainly on the PSM level of abstraction. ()kappel and ()schwinger present a comprehensive and uniform evaluation framework, which can be used to compare customization capabilities of approaches originating from the *mobile computing* and the *personalization* domains. The notion of customization refers to the adaptation of an application's services towards the current context. Their framework has two orthogonal dimensions, which are context and adaptation, and the mapping between context and adaptation represented by the notion of customization. (Kappel et al., 2003) and (Schwinger et al., 2005) provide detailed criteria for both the context and adaptation dimensions of the framework. Their evaluation framework aims at providing three main benefits. First, it provides a structured and uniform view of the various aspects of customization. Second, the framework can be effectively used as a conceptual framework for evaluating existing customization approaches. Finally, the framework can be effective for developing any future customization approaches. The criteria for context and adaptation from (Kappel et al., 2003; Schwinger et al., 2005) are outlined next before applying them to validate the customization approach used in the pervasive software services of the current research.

- *Context.* Context characteristics are important for the management of context data. These characteristics can be categorized as *scope* of context, *representation*, *acquisition*, and the access *mechanism* used.
  - *Scope.* The scope of context comprises several characteristics as outlined next. First, the scope includes different context properties *(C.P.)* supported by the system, such as location, time, device, network and user. Second, it describes the ability to extend *(C.E.)* the properties in order to handle any unforseen requirements. Finally, for each of the context properties the time dimension needs to be considered with the chronology *(C.C.)*, validity *(C.V.)*, and availability *(C.Av.)* of context data.
  - *Representation.* The representation of context is characterized by two issues: reusability *(C.R.)* and abstraction *(C.Ab.)*. First, the representation of context needs to provide mechanisms to reuse already defined context from various context sources, such as geographic information systems or device profiles. Second, the level of abstraction used for representing context can be physical, such as sensed context data, or logical, which is derived context.
  - *Acquisition.* Acquisition is characterized by two key issues: automation *(C.Au.)* and dynamicity *(C.D.)*. First, the degree of automation identifies who is responsible for acquiring context, which can be either a human (manual)

or the system (automatic) or a combination of both (semi-automatic). Second, the degree of dynamicity indicates when the context is acquired, which can be statically at system start-up or dynamically at run-time.

– *Mechanism.* This characteristic identifies the mechanism *(C.M.)* used for acquiring the context information and made accessible to services, which can be pull-based or push-based. Pull-based approaches request for context information while push-based approaches provide context information when context changes.

- *Adaptation.* The second dimension of the evaluation framework is provided by the notion of adaptation. Adaptation can be categorized as the *kind* of adaptation, the *subject* of adaptation, and the *process* of adaptation.

– *Kind.* The kind of adaptation can be a built-in adaptation operation *(A.O.)* such as filter content, add links, or an extension mechanism *(A.Ex.)*, which introduces a new user-defined adaptation operation. The adaptation can affect *(A.Ef.)* the system when certain parts of the system are added, removed or transformed. Complex adaptations *(A.C.)* can be formed by combining a series of adaptation operations.

– *Subject.* The subject of adaptation is characterized by several issues as discussed next. First, the level *(A.L.)* of the Web application that is affected by the adaptation can be content level, hyperbase level, and presentation level. Second, these layers contain several application elements *(A.El.)* that can be adapted, such as links, pages, access structures and input fields. Third, the granularity of adaptation *(A.G.)* can be micro or macro adaptation depending on the number of application elements affected in the adaptation.

– *Process.* The process of adaption comprises several characteristics such as task *(A.T.)*, automation *(A.A.)*, dynamicity *(A.D.)* and incrementality *(A.I.)*. The process of adaptation can contain several tasks such as initiation, proposal, selection, production, presentation and reversion of the adaptation. This separation effectively allows fine-grained control about the automation and dynamicity of the tasks. As in the context dimension, automation identifies who is responsible for performing the tasks which can be performed automatically, manually or semi-automatically. Dynamicity of the adaptation indicates when the adaptation tasks are performed, which can be at design time (static) or at run-time (dynamic). Incrementality signifies whether the adaptation is performed from scratch or incrementally. Performing the adaptation incrementally means that the subsequent adaptations are done based on the results of the pervious adaptations.

Having presented the evaluation criteria provided in (Kappel et al., 2003; Schwinger et al., 2005), next the *context* and *adaptation* dimensions of the customization approach used in the services are evaluated using those criteria. The results of this evaluation are summarized in two tables respectively: Table 5 and Table 6. This evaluation is part of the horizontal evaluation dimension of the framework, and mainly covers the PSM level of the MDA stack. However, several examples on the PIM level are also provided. This evaluation is presented in three logical sections as suggested in (Kappel et al., 2003; Schwinger et al., 2005). First, general details of the customization approach are provided covering issues on *origin*, *major focus*, *technology*, *architecture* and *implementation* (if applicable) of the customization approach. Second, the context dimension of the approach is described, and third, the adaptation dimension of the approach is discussed.

The current research *originates* from the pervasive computing domain in general, and specifically from the pervasive services domain. This research is at the software architectural level, and no *implementation* of services such as pervasive Web services, has been considered in the study. The *main focus* of the `awareness monitoring and notification pervasive service` is to alert on any exceptional situations that may arise on transport items primarily to the vehicle driver of the transport unit. The component configuration of the *architecture* defined for the pervasive software services specification is based on an event-control-action architecture pattern. The research approach has been applied to a modified subset of a real-world case study in intelligent transport called the ParcelCall project. As stated previously in this paper, three automated *tools* have been used in the research: LTSA-MSC tool, `Aspectual FSP Generation tool`, and the LTSA model checker. The customization of the architecture can be considered internal as the system is not aware of the customization in terms of knowing about context or adaptation.

The *context dimension* of the customization approach is described next (Table 5). The primary context parameters modeled at the PIM level of abstraction comprise location, temperature and pressure. However, at the PSM level only temperature and pressure primary context properties have been

Table 5: Current study's context characteristics.

Scope of Context												Representation of Context		Acquisition of Context					Access of Context	
Property (C.P.)								Extensibility (C.E.)	Chronology (C.C.)		Validity (C.V.)	Reusability (C.R.)	Abstraction (C.Ab.)	Automation (C.Au.)			Dynamicity (C.D.)		Mechanism (C.M.)	
location	temperature	pressure	time	device	network	user	application		history	future				manual	semi-automatic	automatic	static	dynamic	push	pull
■	■	■	▨	▨	▨	▨	▨	■	▨	▨	▨	■	■	▨	▨	■	▨	■	■	▨

Explicitly supported	■
Not explicitly supported	▨
Not applicable	

Table 6: Current study's adaptation characteristics.

Kind of Adaptation							Subject of Adaptation												Process of Adaptation						
Operation (A.O.)	Extensibility (A.Ex.)	Effect (A.Ef.)			Complexity (A.C.)		Level (A.L.)				Element (A.El.)						Granularity (A.G.)		Tasks (A.T.)	Automation (A.A.)			Dynamicity (A.D.)		Incrementality (A.I.)
		add	remove	transform	simple	complex	content	hyperbase	presentation	others	text	audio	image	video	link	others	micro	macro		automatic	semi-automatic	manual	static	dynamic	
■	■	■	▨	■	▨	■				■						■	■	▨	■	■	▨	▨	▨	■	■

Explicitly supported	■
Not explicitly supported	▨
Not applicable	

modeled *(C.P.)*. Nevertheless, at the PSM level other context properties such as item identity, location and acceleration can be supported as well *(C.E.)*. These context parameters as a whole constitute the physical context of the study. Context modeling at the PIM level is provided by the `c-FSP-UML profile` and the `c-FSP aspects`. The profile essentially provides several stereotypes to represent the core service behavior, the context-dependent behavior, and the dependencies between the core service behavior and the context-dependent behavior at the service interface level. The use of stereotypes essentially supports the notion of extensibility *(C.E.)*. The object-oriented notions used in the profile such as generalization further support the notion of extensibility. At the PSM level, the notions of attribution, classification, generalization and encapsulation from the context definition have been modeled to structure and link the objects defined in the aspects *(C.E.)*. In the study, validity period *(C.V.)*, chronology *(C.C.)* or availability *(C.Av)* of context are not supported as the time dimension of the context properties have not been considered. The explicit support provided for attribution, instance-of and ISA notions at the PSM level, facilitates reusability of context *(C.R.)*. The approach provides a high-level inference mechanism to automatically derive higher-level logical context *(C.Ab)*. Both primary and logical context are modularized into aspects as atomic context aspects and composite context aspects respectively. Thus, the study supports an explicit separation between physical and logical context *(C.Ab)*. For example, low-level temperature readings from the RFID tags are inferred as low temperature or high temperature during the refinement step of the pervasive service. At the PIM level, the profile is maintained manually by the service engineer *(C.Au.)*. At the PSM level, both physical and logical context information are acquired automatically *(C.Au.)* and at run-time *(C.D.)*. The pervasive service engineer using the LTSA animator can select values for the temperature or pressure readings from a range of values at run-time. The mechanism *(C.M.)* used to acquire context and made accessible to the pervasive service can be considered push-based as context readings from the RFID tags are provided based on context changes and not on requests. The second dimension of the customization approach is provided by the notion of *adaptation* (Table 6). In this study, there are two types of adaptation operations: triggering and recovery operations *(A.O.)*. At the PSM level, both these operations have been supported by the Trigger and Recovery c-FSP aspects *(A.O.)*. Trigger aspects, for example Trigger Aspect Adverse Environment Status, effectively send notifications or alert messages to the vehicle driver when a transport item's context information violates acceptable levels. Recovery aspects, for example Recovery Aspect Adverse Environment Status, model any recovery actions that need to be enforced after an exception situation is raised by a Trigger aspect. Both these aspects have been modeled as independent state machines at PSM level, which synchronize with their base state machines using explicit synchronization events. The adaptation operations provided by the aspects are associated with the core service model through weaving, and the behavior of these aspects can affect the core service behavior depending on the context information *(A.Ef.)*. This can be, for example, executing or modifying a service based on context information *(A.Ef.)*. At the PIM level, adaptation operations are specified using the stereotypes: TriggerAspect and RecoveryAspect. As the adaptation process contains a series of stages, it can be considered a complex process *(A.C.)*. Also, a distinct separation can be identified between the different tasks of the adaptation process *(A.T.)*. When an item's context information is violated, first, the pervasive service alerts the vehicle driver by sending an SMS. Second, the service can perform any appropriate recovery actions to remedy the situation, such as control the refrigerator's temperature *(A.C.)*. Third, the service adaptation can be extended as follows *(A.Ex.)*. If the environment status of items is critical the service can alert the goods tracing server and eventually the customer being affected through the goods information server *(A.Ex.)*. In the current study, context information is represented at the service interface level *(A.L.)* that essentially consists of operation invocations and the exchange of respective input/output parameters. The core service elements represented at the modeling level include states, transitions, processes, and services *(A.El.)*. Any adaptation operation, which can be a triggering operation or a recovery operation, is bound to the transitions of the core service model. Therefore, the adaptation level and adaptation elements of this study are the service interface level *(A.L.)* and transitions *(A.El.)* respectively. As a result, different Web application levels such as content, hyperbase and presentation are not applicable in this study nor the Web adaptation elements of text, audio, image, video and link provided in (Kappel et al., 2003; Schwinger et al., 2005). The adaptation granularity *(A.G.)* can be considered micro considering the number of elements affected by the adaptation process in the study. Also, it is performed automatically *(A.A.)* and at run-time *(A.D.)*. The adaptation process can be considered incremental *(A.I.)* as recovery operations are dependent on triggering operations at both PIM and PSM levels.

6 CONCLUSIONS

To summarize this paper has discussed the *evaluation framework* developed to validate the main methods and tools employed in this study for engineering pervasive software services. The method of evaluation used here is based on key features comparison. The evaluation framework consists of two dimensions or views: vertical and horizontal. The vertical evaluation of the research compared several research tools to the Aspectual FSP Generation tool developed in the current research. The tools were compared across the PIM and PSM levels of the MDA stack. This evaluation was based on several criteria: context-dependent behavioral modeling at the PIM level, explicit joinpoint model of AOM at the PIM level, weaving performed at PIM or PSM levels, and context-dependent behavioral code generation from the PIM level to the PSM level. The horizontal evaluation view was designed to validate several desired key features required mainly at the PSM level (i.e. FSP) of the aspectual pervasive software services specification. These evaluation criteria mainly cover two aspects. They are the formal methods and tools employed in the study and the context and adaptation dimensions of the customization approach used in the pervasive services.

The results of the evaluation are assuring. The vertical evaluation has demonstrated that the `Aspectual FSP Generation tool` has unique features in context-dependent behavioral modeling and context-dependent behavioral code generation. The horizontal evaluation of the approach has shown that the formal methods and tools employed in the research, and the customization approach used in the services are indeed effective towards the overall objectives of this research. However, the prototype or proof of concept nature of the methods and tools employed in this research makes it challenging to compare their benefits with more advanced industry-based tools, which is a limitation.

REFERENCES

Abeywickrama, D. B. (2010). *Pervasive Services Engineering for SOAs*. Ph.D Thesis, Faculty of IT, Clayton Campus, Monash University, Australia.

Abeywickrama, D. B. and Ramakrishnan, S. (2008a). A Model-Based Approach for Engineering Pervasive Services in SOAs. In *Proc. 5th International Conference on Pervasive Services (ICPS'08)*, pages 57–60, Sorrento, Italy. ACM.

Abeywickrama, D. B. and Ramakrishnan, S. (2008b). Towards Engineering Models of Aspectual Pervasive Software Services. In *Proc. 3rd Workshop on Software Engineering for Pervasive Services (SEPS'08)*, pages 3–8, Sorrento, Italy. ACM.

Abeywickrama, D. B. and Ramakrishnan, S. (2010). Model-Driven Development of Aspectual Pervasive Software Services. In *Proc. 14th IEEE International Enterprise Distributed Object Computing Conference Workshops*, pages 49–59, Vitoria, Brazil. IEEE.

Analyti, A., Theodorakis, M., Spyratos, N., and Constantopoulos, P. (2007). Contextualization as an Independent Abstraction Mechanism for Conceptual Modeling. *Information Systems Journal*, 32(1):24–60. Elsevier Science Ltd., Oxford, UK.

Clarke, E. M., Wing, J. M., and Alur, R. (1996). Formal Methods: State of the Art and Future Directions. *ACM Computing Surveys*, 28(4):626–643. ACM.

Cottenier, T., van den Berg, A., and Elrad, T. (2007). Motorola WEAVR: Aspect Orientation and Model-Driven Engineering. *Journal of Object Technology*, 6(7):51–88. Chair of Software Engineering, ETH Zurich, Switzerland.

Davie, A. (2002). Intelligent Tagging for Transport and Logistics: The ParcelCall Approach. *Electronics & Communication Engineering Journal*, 14(3):122–128. Institution of Electrical Engineers, London, UK.

Fuentes, L., Gamez, N., and Sanchez, P. (2008). Aspect-Oriented Executable UML Models for Context-Aware Pervasive Applications. In *Proc. 2008 5th International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES'08)*, pages 34–43, Budapest, Hungary. IEEE.

Groher, I. and Schulze, S. (2003). Generating Aspect Code from UML Models. In *Proc. 3rd International Workshop on Aspect-Oriented Modeling co-located with 2nd International Conference on Aspect-Oriented Software Development (AOSD'03)*, Boston, USA.

Hegering, H.-G., Küpper, A., Linnhoff-Popien, C., and Reiser, H. (2003). Management Challenges of Context-Aware Services in Ubiquitous Environments. In *Self-Managing Distributed Systems*, volume 2867 of *Lecture Notes in Computer Science*, pages 321–339. Springer Berlin / Heidelberg.

Kappel, G., Proll, B., Retschitzegger, W., and Schwinger, W. (2003). Customisation for Ubiquitous Web Applications: A Comparison of Approaches. *International Journal of Web Engineering and Technology*, 1(1):79–111. Inderscience Publishers, Geneva, Switzerland.

Magee, J. and Kramer, J. (2006). *Concurrency: State Models and Java Programs*. John Wiley and Sons, second edition.

Mandato, D., Kovacs, E., Hohl, F., and Amir-Alikhani, H. (2002). CAMP: a Context-Aware Mobile Portal. *IEEE Communications Magazine*, 40(1):90–97. IEEE.

Mostefaoui, S. K. and Hirsbrunner, B. (2004). Context-Aware Service Provisioning. In *Proc. IEEE/ACS International Conference on Pervasive Services (ICPS'04)*, pages 71–80, Beirut, Lebanon. IEEE.

Schwinger, W., Grün, C., Pröll, B., Retschitzegger, W., and Schauerhuber, A. (2005). Context-Awareness in Mobile Tourism Guides - A Comprehensive Survey. Technical report, Johannes Kepler University, Linz, Austria.

VIDE (2009). VIsualize all moDel drivEn programming (VIDE), WP 11, Deliverable number D11.3 (European Commission supported Specific Targeted Research Project, Information Society Technologies). WWW page. http://www.vide-ist.eu/download/VIDE_D11.3.pdf (Last accessed on 02/04/2011).

Whittle, J. and Jayaraman, P. (2008). MATA: A Tool for Aspect-Oriented Modeling based on Graph Transformation. In *Models in Software Engineering*, volume 5002 of *Lecture Notes in Computer Science*, pages 16–27. Springer Berlin / Heidelberg.