2010 15th IEEE International Conference on Engineering of Complex Computer Systems

Service-based Development of Context-aware Automotive Telematics Systems

Minh H. Tran, Alan Colman and Jun Han
Centre for Complex Software Systems and Services (CS3)
Swinburne University of Technology
PO Box 218, Hawthorn, VIC, Australia
(mtran, acolman, jhan)@swin.edu.au

***Abstract*—Automotive software has increasingly become context-aware and adaptive to deal with dynamically changing environments. This paper presents our novel service-based approach to support the structural and behavioral adaptation of automotive telematics. We adopt services to (1) provide physical context facts and (2) facilitate context-aware interactions between entities of automotive telematics systems. In this paper, we introduce a layered architecture of our approach and demonstrate how the approach is applied to develop context-aware automotive telematics systems that support V2X interactions. The empirical evaluations show that our service-based approach is scalable to supporting run-time adaptation of automotive telematics.**

Keywords-SOA; context-awareness; adaptation; automotive software; context modeling; service-oriented software engineering

I. INTRODUCTION

Automotive software has increasingly become a decisive factor for competitive advantages in the automotive industry, being one of the major enabling technologies that drive vehicle innovation. Moreover, automotive software is now used more broadly. It has grown beyond the closed boundary of the controller area network (CANbus) where components are strictly controlled by electronic control units (ECUs). Telematics systems are emerging automotive technologies that combine advanced communication and vehicle technologies to improve vehicle control and safety, make vehicles more environment-friendly, and enhance driver experience [1-4]. In general, automotive telematics systems (referred to as *telematics* for short) could be classified into three groups based on its communication ranges (see Figure 1). In-vehicle (inV) telematics support seamless interaction between users' portable devices (e.g., phones, mp3 players, etc.) and vehicle built-in infotainment systems. This allows occupants in a vehicle to be entertained and/or to carry on their work. Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) systems allow vehicles to "talk" to each other and to roadside infrastructure to share safety warnings, detect hazards and proactively avoid collisions. Vehicle-to-service provider (V2SP) systems allow the driver to access to value-added services provided by third party providers (e.g., news, infotainment, location-based services, etc.). This range of V2X telematics illustrates not only significant potentials and applications of the systems, but also the open, complex and dynamically changing environments in which they operate.

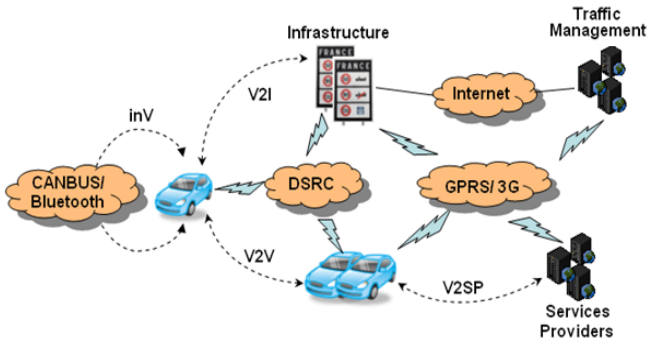


Figure 1. V2X classification of automotive telematics systems.

To operate seamlessly in such dynamic environments, telematics needs to be context aware and adapt in response to changing context [1, 5]. The context includes *physical context* and *social context* [6]. The former consists of information facts about the systems and environments, such as information about vehicle speed, road condition, traffic situation, and so on. On the other hand, the latter includes constructed relationships and interaction constraints between entities that interact with each other in the system and between the system and its environments. Intuitively, social context refers to the rules and norms that traffic users and their vehicles need to follow. For example, all vehicles must obey the speed limit set by the traffic management, or vehicles should keep a 2-second time-headway rule with a preceding vehicle for a safe breaking.

There has been much research effort on supporting context-aware telematics, especially in intelligent vehicle systems. Context-aware telematics is characterised by the need to integrate context from a diverse number of control systems and information providers that use various communication technologies. These systems/providers are both inside and outside the vehicle. External providers may continually change as the vehicle travels. Some telematics applications also need to provide human-in-the-loop interfaces that minimize distraction for the driver. While previous research (e.g. in pervasive computing) has made significant contributions to aspects of this problem domain, it is this unique combination of characteristics that makes context-aware telematics such a challenge. Existing

978-0-7695-4015-3/10 $26.00 © 2010 IEEE
DOI 10.1109/ICECCS.2010.15

solutions are application-specific and not scalable to the development of other forms of telematics such as non-safety critical V2V or V2I and value-added V2SP. More specifically, conventional telematics have been developed using the top-down functional decomposition approach where vehicle software components are embedded in hardware (i.e., ECUs). This approach has shown its value in producing stable performance systems, but it becomes less applicable and less effective in designing context-aware software that needs to deal with changes and interacts with the broader vehicle environment. Furthermore, telematics have been strongly proprietary and manufacturer specific resulting in inflexible upgrade options and high cost development. The recent shift from hardware to software provides new opportunities for telematics systems to arise. To maximize benefits from this opportunity, it is imperative that proprietary approaches give way to standardized software architecture.

To address the challenge of developing interoperable, context-aware telematics, this paper presents our novel service-oriented approach to developing context-aware telematics. The essence of our approach is to utilize service-oriented architecture (SOA) to convey and manage situation context of telematics. The approach clearly separates different software engineering concerns such as context acquisition, context management, adaptation and context-aware telematics. Moreover, the service-oriented approach uses standard well-defined interfaces to support interaction between loosely-coupled entities (services). Although SOA has increasingly been seen as a well-recognized model for software development, applications of SOA in developing context-aware automotive software is under-researched. In this paper, we present our service-based development of context-aware automotive software and demonstrate its implementation prototype.

The remainder of the paper is structured as follows. Section 2 shows motivating scenarios of context-aware telematics and analyses the research challenge of dealing with changing context. In Section 3, after discussing the advantages of a service-based approach to automotive software development, we introduce our architecture for context-aware telematics. Section 4 reports the experimental study that we carried out to evaluate our approach. Section 5 discusses related research, and section 6 concludes the paper and highlights future work

II. MOTIVATING SCENARIOS AND PROBLEM ANALYSIS

A. Motivating Scenarios

The emergence of V2X technologies leads to a broad range of telematics that can be developed to support interaction between a vehicle and other entities (e.g., surrounding vehicles, infrastructure, service providers, etc.). Vehicles not only interact to avoid crashes but also assist other travel activities. For example, the followings are some scenarios where two rent vehicles form a *cooperative convoy* to travel together to the same destination, and each vehicle could interact with service providers such as a car rental company if required. In the cooperative convoy, one vehicle is the leading vehicle whilst the other is the following vehicle. Two vehicles follow the same route chosen by the leading vehicle, and the conditions mutually agreed by the drivers of the vehicles. For instance, to make the convoy safe and convenient, the following vehicle needs to send a distance update to the leading vehicle every 10 seconds, and if either vehicle has mechanical problems (e.g., flat tire, engine issue, etc.) it needs to notify the other vehicle. As the convoy proceeds, these conditions could be adjusted by the drivers, and new conditions could also be added.

The car rental company provides a roadside service to their customers. When a vehicle experiences a mechanical issue, its telematics automatically executes a process to search for the nearest mechanic and requests a tow truck if the vehicle is no longer drivable. Context information about the vehicle location and mechanical issue is also sent to those services. The logic of the road-side assistance service process is defined by the car rental company. Depending on customers' insurance policies, different levels of support are provided.

B. Problem Analysis: Dealing with Changing Context

As illustrated in the motivating scenarios, telematics systems need to acquire various types of context and adapt their behavior in response. Context information may continually change as additional information/services could be available. There may also be considerable uncertainty as to the quality of the context information acquired (accuracy, up-to-dateness, etc.) . The context also involves various aspects that are related to people, vehicle and environment [7]. To develop context-aware telematics, two prominent research issues need to be addressed: *context modeling* and *enabling architecture* for acquiring, managing and consuming context.

First, a number of context modeling techniques has been developed, including graphical approach, mark-up scheme, object-oriented, logic-based, and ontology-based models (as reviewed in [8]). Those modeling techniques are useful in representing physical context. To present social context, we have developed a novel modeling framework that represents social context as an organized composition of interrelated functional roles whose interdependencies are expressed through contracts [6, 9]. The contracts express both functional operations and non-functional requirements that obligate the contracted roles.

Second, to reduce the complexity of engineering context-aware automotive software, research efforts have been devoted to developing a system architecture to manage the acquisition, dissemination and use of context information [10, 11]. The main objective of such an architecture is to separate software engineering concerns by pushing as much as possible the acquisition, management and dissemination of context information into a context infrastructure. While there have been significant studies on context awareness in the field of pervasive computing, there is limited research into the development of context-aware telematics. Automotive software has its own combination of characteristics such as safety critical requirements, high mobility, complex context involving intra- and inter-vehicle interaction, and human-in-the-loop control. It is important that solutions to context-aware automotive software be standards-based and open in order to accommodate business agility, increase software interoperability and flexibility, and reduce production cost.

In general, software adaptation could be classified into two approaches: *parameter adaptation* and *compositional adaptation* [12]. On the one hand, parameter adaptation is realized by modifying system variables related to behavior of the systems (e.g., operational parameters that affect various levels of QoS). On the other hand, compositional adaptation involves changing the component-based structure of the system. Dynamic re-composition of the system at run-time is achieved by adding new components, and intercepting and redirecting interactions among system components. McKinley et al. [12] identify three core concepts—separation of concerns, computational reflection and component-based design—that enable parameter and compositional adaptation. Separation of concerns is an important principle of software engineering by separating the developments of functional behavior, management aspects and crosscutting concerns (e.g., QoS requirements). Computational reflection refers to the systems' ability to observe and modify its own structure as well as behavior during execution. Component-based design is a software engineering methodology that advocates the separate, independent development and integration of system components. Moreover, it is important that software components could be dynamically composed at run-time.

III. SERVICE-BASED DEVELOPMENT OF CONTEXT-AWARE TELEMATICS: PRINCIPLES AND ARCHITECTURE

A. Design Principles

Given the challenges that context-aware telematics need to deal with, this section discusses the applicability of service-based development in designing and deploying context-aware telematics. The discussion is based on four key principles: design philosophy, dynamic binding, abstraction and granularity, and delivery mechanisms [13].

1) Design Philosophy

To accommodate changes, automotive software should be designed following two key concepts. First, software could be quickly and reliably composed from prefabricated and standardized software components. Second, an increasingly large number of software components need to be made available to and provided by both general and domain specific developers. Conventional telematics are component-based but have been designed using a rigid top-down approach. Such an approach to developing automotive software is limited and inflexible in coping with changes (e.g., *evolving requirements* and *dynamic context*). Software components are tightly integrated at design time providing limited support for customization and adaptation. Though component-based development adopts interfaces to manage interactions between components, the interaction is tightly-coupled. For instance, RMI and COBRA require a tight coupling between a client and the server (i.e., a stub and the corresponding skeleton share the same interface). The service-oriented approach allows a loosely-coupled interaction paradigm between service providers and end-user applications (i.e., service consumer) via publication and discovery mechanisms [14]. Those services could be provided by software components within a vehicle (e.g., information about vehicle dynamics) or external service providers (e.g., information about traffic condition). Moreover, the services could be either known at design time or dynamically discovered at run-time.

2) Dynamic Binding

The service-oriented approach provides a significant characteristic of dynamic binding at run-time that bridges a separation between design time and run-time adaptation. Currently, automotive telematics are designed and configured at design-time, to the best of our knowledge no support for run-time adaptation in response to changing context. The service-based development could support context awareness in two distinctive ways. First, context-aware telematics could be composed from a number of services. For each service there could be a number of potential providers who offer the similar function but may have different QoS or service different locations. Depending on the location/desired QoS, different providers could be selected to perform tasks. For example, the context about traffic patterns is provided by different traffic management systems as vehicles move from one city to another. A dynamic selection of providers allows automotive software to change service providers at run-time. Second, dynamic binding facilitates dynamic service composition, whereby a new service is composed from a set of existing services to perform a certain task. This composition is completely transparent to a consumer.

3) Abstraction and Granularity

The principles of abstraction and separation of concerns have a great influence on how context-aware telematics are able to deal with changing context, in particularly the granularity of changes. Context (also known as resolution [15]) can be fine-grained or coarse-grained. For example, if a service only provides GPS coordinates of a vehicle, it is considered as fine-grained context. But if a service provides all the location-based contexts (e.g., weather condition, traffic control, etc.) for a vehicle, it then is considered as coarse-grained. The service-based development allows software functionality and context to be delivered (e.g. published and discovered) using arbitrarily complex XML documents at a coarse-grained level of abstraction that corresponds to real-world concerns (e.g., road condition, engine status, traffic patterns, etc.). Services can be used to convey context information that is related to distinct concepts, and can be consumed by different context-aware automotive applications.

4) Delivery Mechanisms

Features of conventional telematics (i.e., functions and QoS) are generally pre-installed by manufacturers. Software upgrade is difficult, and customization is time consuming and costly. Customized options (e.g., satellite navigation, Bluetooth, WiFi) could be added to vehicles but require vehicles to be brought to a garage for a mechanical procedure. The service-oriented approach could provide the revolutionary concept of "automotive software as a service" where functionalities could be dynamically identified and composed, and QoS negotiated. These technical advantages make possible new business models that have the great potential to deal with business agility through the ability to evolve as customer requirements and market demands change.

B. *Service-based Architecture for Context-aware Telematics*

We have developed a layered architecture for context-aware and adaptive telematics that embodies the abovementioned four principles, as shown in Figure 2. In particular, the architecture supports a clear separation between the use of context (application level) and the management of context (social context and physical context). Services are used to convey context information to telematics. The systems could also acquire context using dynamic binding to context service providers. This binding is contracted based on whether the context is relevant to the application and whether services are available to deliver the required QoS.

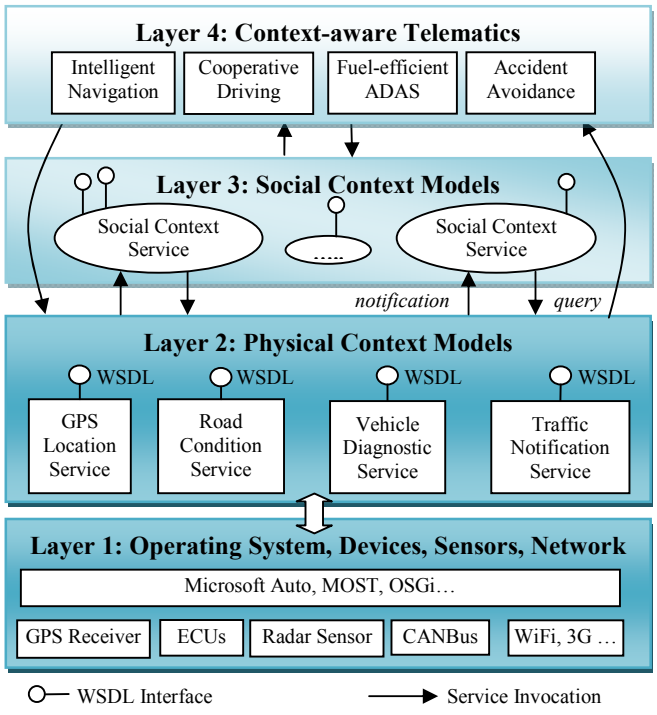


Figure 2. Architecture of context-aware automotive telematics

Layer 1 supports the operation and integration between hardware devices and sensors (e.g., in-vehicle ECUs, radar sensors, GPS receiver, 3G modem, etc.). Communication between this heterogeneous set of devices is supported by a number of networks such as CANbus, Bluetooth, dedicated short range communication (DSRC), and so on. There have been a number of embedded operating systems and vehicle platforms such as Microsoft Auto 4.0, Media Oriented Systems Transport (MOST), Automotive Open System Architecture (AUTOSAR), Open Service Gateway Initiative (OSGi), and so on, developed to support integration of components in Layer 1.

Using components and applications made available by Layer 1, *Layer 2* wraps those functions as physical context services that expose standard WSDL interfaces allowing upper layers to consume the services. Layer 2 also manages the discovery and binding to external services that are provided by third-party providers. Services in this layer are often coarse-grained services that could be reused by multiple applications (i.e., not specific to end-user applications).

The management of context and adaptation is handled by *Layer 3*. This layer includes social context models that represent relationships and interaction constraints between entities as well as how interactions are affected by physical context facts. The social context models could be implemented as services [6, 16]. The social context-based services invoke other services to acquire physical context, and at the same time they could be invoked by telematics.

Layer 4 includes context-aware telematics that use social context and physical context services to "sense" the context and adapt in response to changes of the context. It should be noted that the telematics applications at Layer 4 may access directly physical contexts at Layer 2 without going through a social context at Layer 3.

In the layered architecture, our unique contribution is to present social context models at Layer 3 that *explicitly* model context-aware interactions and adaptation of telematics. Therefore, the remainder of this section focuses on discussing this layer in more detail. In [6], we have presented our approach to modeling social context as a ROAD (Role-Oriented Adaptive Design) composite [17, 18]. For context-aware entities (e.g., telematics), social context is a *subjective* representation the relationships and interaction constraints that the system has with other entities in respect to a specific goal. That is, social contexts are considered as proxies of interaction between entities. A system could have multiple social contexts. Each social context is modeled as a ROAD composite consisting of functional *roles*, interaction constraints expressed in *contracts* as Event-Condition-Action (ECA) rules, and an *organizer* role. Telematics systems and other entities(e.g., infrastructure, external services) interact through the ROAD composite structures by playing roles in them. The organizer of the composite can dynamically create/delete roles, write rules into contracts that mediate the interactions between those roles, and dynamically bind services/systems to play those roles in the composite. While other implementations are possible, a social context model can be implemented as a service that exposes a set of WSDL interfaces, corresponding to the number of its functional roles.

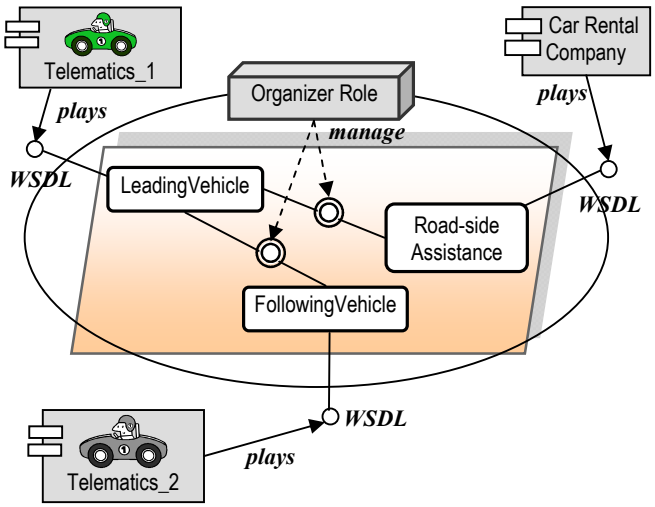


Figure 3. Cooperative convoy social context modeled as a ROAD composite.

For example, Figure 3 shows a ROAD composite representing a cooperative convoy social context between two vehicles that is modeled from the leading vehicle's perspective. The social context includes three functional roles (*LeadingVehicle*, *FollowingVehicle* and *Road-side Assistance*). The social context model could be implemented as a Web service that exposes three WSDL interfaces. The interfaces allow three actors, *Telematics_1* of the leading vehicle, *Telematics_2* of the following vehicle, and *Car Rental Company*, to bind to the roles statically or dynamically. After binding to the roles, the actors could interact with one another by invoking methods provided by the service. The reader is referred to [6] for detail discussion of social context models. As an example, the following is a WSDL interface of the following vehicle role that includes operations to notify the distance between two vehicles and mechanical problems:

```
<wsdl:types>
  <s:schema elementFormDefault="qualified"
targetNamespace="URI">
  <s:element name="ReturnDistance">
    <s:complexType />
  </s:element>
  <s:element name="DistanceResponse">
    <s:complexType>
      <s:sequence>
        <s:element minOccurs="1" maxOccurs="1"
name="DistanceResult" type="s:int" />
      </s:sequence>
    </s:complexType>
  </s:element>
  </s:schema>
        ............
</wsdl:types>
<wsdl:message name="DistanceSoapIn">
  <wsdl:part name="parameters"
element="tns:ReturnDistance" />
</wsdl:message>
<wsdl:message name="DistanceSoapOut">
    <wsdl:part name="parameters"
element="tns:DistanceResponse" />
</wsdl:message>
        ............
<wsdl:portType name="ConvoyServiceSoap">
  <wsdl:operation name="NotifyDistance">
    <wsdl:input message="tns:DistanceSoapIn" />
    <wsdl:output message="tns:DistanceSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="NotifyMechanicIssue">
    <wsdl:input message="tns:IssueSoapIn" />
    <wsdl:output message="tns:IssueSoapOut" />
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="ConvoyService"
type="tns:ConvoyServiceSoap">
  <soap:binding
transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="NotifyDistance">
    <soap:operation soapAction="URI:NotifyDistance"
style="document" />
    .....
  </wsdl:operation>
</wsdl:binding>

        ............
```

IV. EVALUATION

To validate our service-based development, we have implemented and evaluated a prototype of Context-Aware Telematics (CAT) that supports V2X interactions, particularly the cooperative convoy and road side assistance described in the motivating scenarios. CAT adapts its structure and behavior in response to changing contexts of road condition, traffic condition and vehicle dynamics (e.g., speed, mechanical issues).

A. Context-Aware Telematics Implementation

CAT is implemented based on the layered architecture discussed in the previous section. In particular, the architecture of CAT includes four layers as illustrated in Figure 4.

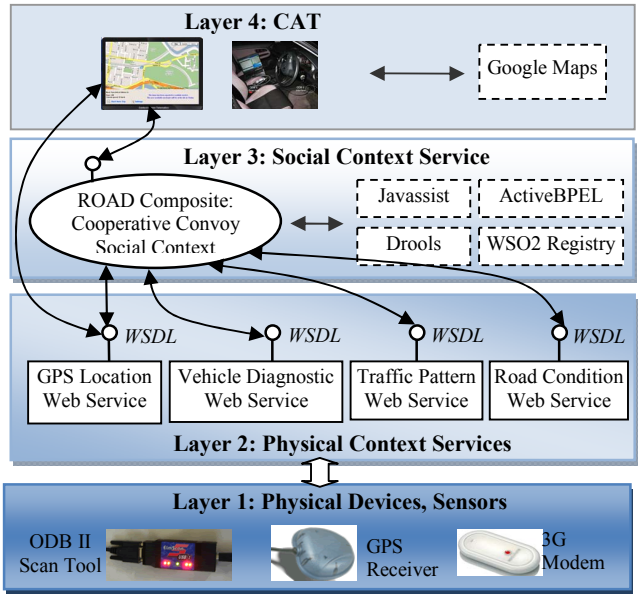


Figure 4. System architecture of CAT

Layer 1 consists of devices used to collect physical context information, including a GPS receiver, ODB II scan tool and 3G modem. Four Web services are implemented at Layer 2 to provide physical context information about vehicle location, vehicle status, road condition and traffic condition. As an example, Figure 5 shows a WSDL interface of the vehicle location Web service that wraps conventional GPS software.

Layer 3 includes a social context model that represents the cooperative convoy of two vehicles. This social context is implemented as a run-time ROAD composite that supports both structural and behavioral adaptation. The *structural* adaptation is supported by the reflection mechanism. We use the Javassist library [19] to dynamically adapt the structure of the ROAD composite. At run-time Javassist allows generating new classes for new functional roles of the ROAD composite, and modifying existing binary classes to cope with changes of requirements. For example, new functional roles, including *Infrastructure* and *TravelGuide*, can be added dynamically to the run-time convoy composite to accommodate context-aware interactions between the vehicle and a traffic management system, and a travel guide service provider. As illustrated in

Figure 6, one of the adaptation mechanisms provided by CAT uses a generic *IAccess* interface that provides the *set* and *get* methods to a target class. For each of the run-time generated classes (i.e., *Infrastrucre* and *TravelGuide*), there is a corresponding adapter (i.e., *InfrastructureAdapter* and *TravelGuideAdapter*, respectively) that implements the *IAccess* interface.

Another aspect of structural adaptation is dynamic binding at run-time. More specifically, CAT allows dynamically discovering and binding external services at run-time. We used WSO2 Registry [20] for dynamically looking up service endpoints.

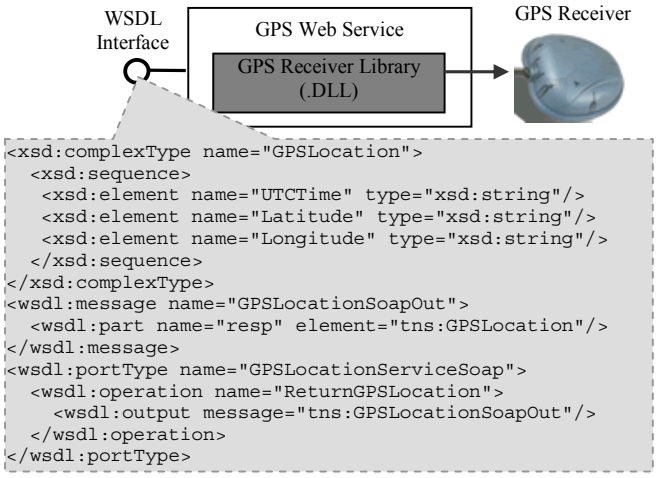


Figure 5. WSDL interface of GPS Web service.

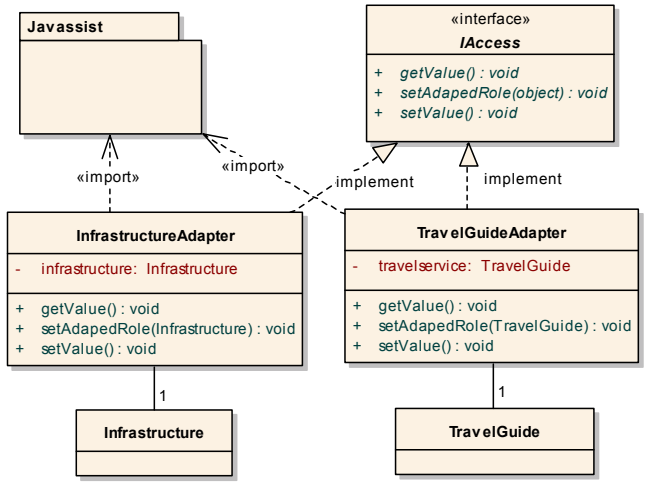


Figure 6. CAT supports dynamic class generation using Javassist library.

The *behavioral* adaptation is supported by a rule-based system. Interaction constraints are expressed in forms of Event-Condition-Action rules. We use Drools engine [21] for our prototype. In addition, to support interaction between a vehicle and the car rental company, Layer 3 also includes a service composition that orchestrates activities when a vehicle is broken down (e.g., validate a customer policy, search for the nearest, available mechanic, search for a tow truck, etc.). We model this service composition as a BPEL—Business Process Execution Language—process executed by ActiveBPEL engine [22]. As an example, Figure 7 shows a successfully executed BPEL process of a road-side service (i.e., the customer's cover policy is valid; the vehicle is not drivable; a mechanic garage is found and a tow truck is ordered).

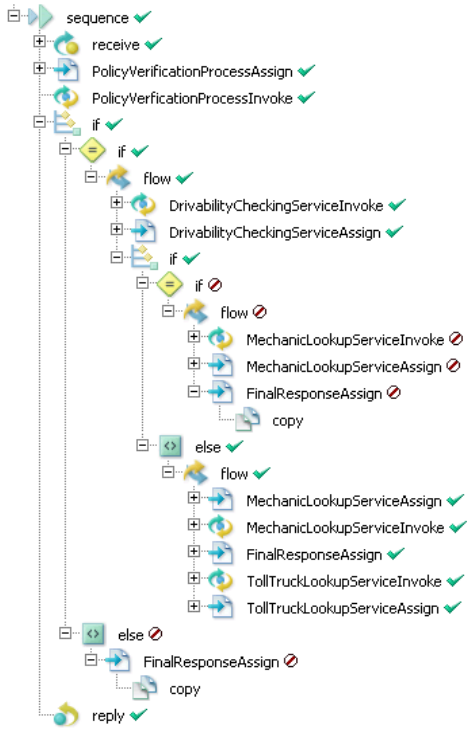
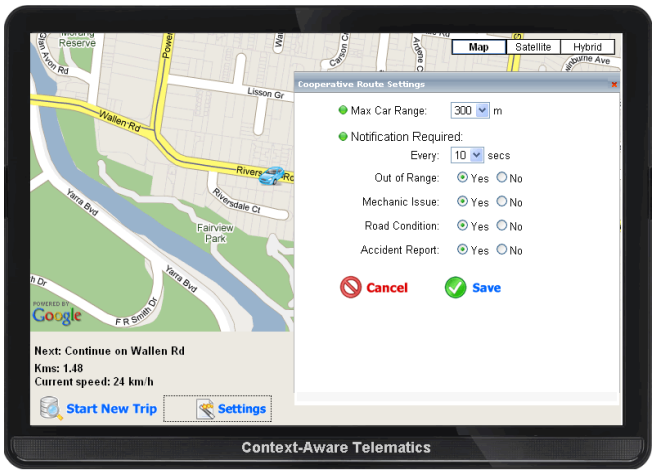
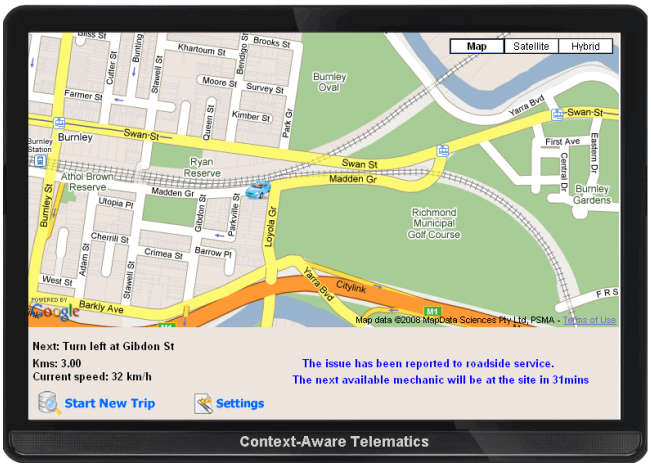


Figure 7. Executable BPEL process of road-side assistance

Layer 4 contains CAT that includes user interface (UI) allowing the driver to interact with the system. CAT uses Google Maps to display travel routes, overlay vehicle positions and information related to other location-based services. Figure 8 shows example UI screenshots of CAT.



(a) UI allows the driver to set constraints of the cooperative convoy



(b) Real-time information about the convoy is overlaid on Google map

Figure 8. User interfaces of CAT

B. *Evaluation Results*

We have conducted an empirical experiment to evaluate our approach. The main objective is to evaluate the adaptation capability of CAT. In particular, we focus on our evaluation on the feasibility of service-based approach in supporting run-time structural and behavioral adaptation. The experiment involves the following equipment:

- Test vehicle: 2006 Subaru Impreza
- Laptop located in a vehicle: Intel T2300 1.6GHz, 2GRAM, Windows XP.
- 3G Internet broadband.
- Vehicle diagnostic scan tool: ELMScan[1] that is used to read vehicle status from CANbus using On-Board Diagnostics II (ODB II) protocol.
- GPS receiver: Haicom HI-204 III USB[2].

Figure 9 shows the in-vehicle experiment setting. CAT is run on a laptop that is connected to an ELMScan tool. The scan tool connects to the vehicle CANbus via ODB II interface. A GPS receiver is connected to the laptop to collect the GPS location of the vehicle. CAT interacts with external services using a 3G broadband modem.

Figure 9. In-vehicle experimental setting of CAT.

[1] http://www.scantool.net/scan-tools/
[2] http://www.ja-gps.com.au/gps-haicom-204.html

1) *Structural Adaptation*

CAT deals with changing context by supporting structural adaptation in the form of run-time reconfiguration of a ROAD composite and dynamic service binding. Structural adaptation imposes a number of overheads, including (1) the generation of new roles (i.e., Java classes) within the ROAD composite, (2) the binding of new services to the roles, and (3) a number of processes concurrently executed by the composite. For these overheads we ran a series of tests to establish if the overheads imposed would be significant.

CAT adopts the Javassist library to manipulate the structure of a ROAD composite at run-time. The library allows CAT to inspect, edit, and create Java binary classes. The inspection mechanism provides similar capabilities as to what available in Java reflection APIs. The evaluation shows that the performance of Javassist is superior to that of Java reflection when modifying classes dynamically. As shown in Figure 10, using the Javassist library to modify Java classes at run-time is significantly faster than using Java reflection APIs.

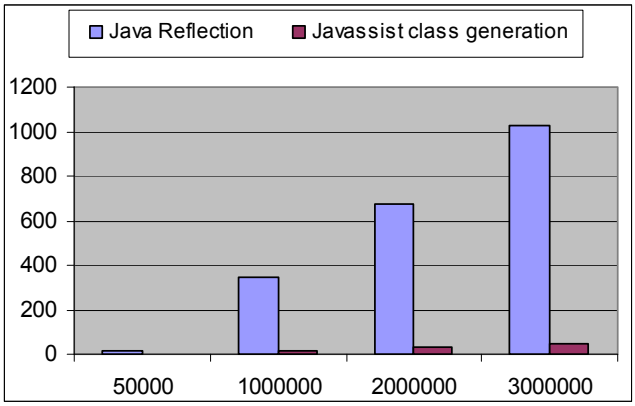


Figure 10. Comparison of dynamic class modification at run-time (X-axis: number of invocations, Y-axis: milliseconds).

Response time measure is used to evaluate the dynamic binding adaptation of CAT. In particular, we compare the response times of static binding and dynamic binding. In the case of static binding, BPEL processes are referenced to fixed endpoints, whereas in the case of dynamic binding services are dynamically discovered and bound to the processes. Figure 11 shows response time measures of a single road-side service process when static binding and when dynamic binding were used. We run 50 tests (*n=50*), on average the process with static binding took 2.48s (standard deviation = 0.58), and the process with dynamic binding took 3.04s (standard deviation = 0.79).

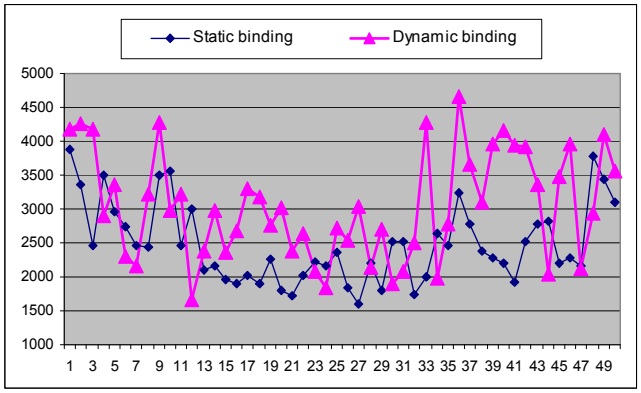


Figure 11. Response time measure (X axis: test runs, Y axis: milliseconds).

Furthermore, we tested CAT in various settings, ranging from one service composition process to multiple *concurrent* processes. The results reported below are collected from the test of 20 BPEL processes. We could have tested with a larger number of processes but we argue that 20 is a realistic number for an in-vehicle CAT system in the automotive domain. Each composition is related to one driving task carried out by the driver/vehicle at one time, thus 20 concurrently running processes is a reasonable assumption for telematics. Figure 12 (a) and (b) shows snapshots of the CPU usages in test cases when there are one process and 20 concurrent BPEL processes invoked by CAT, respectively.

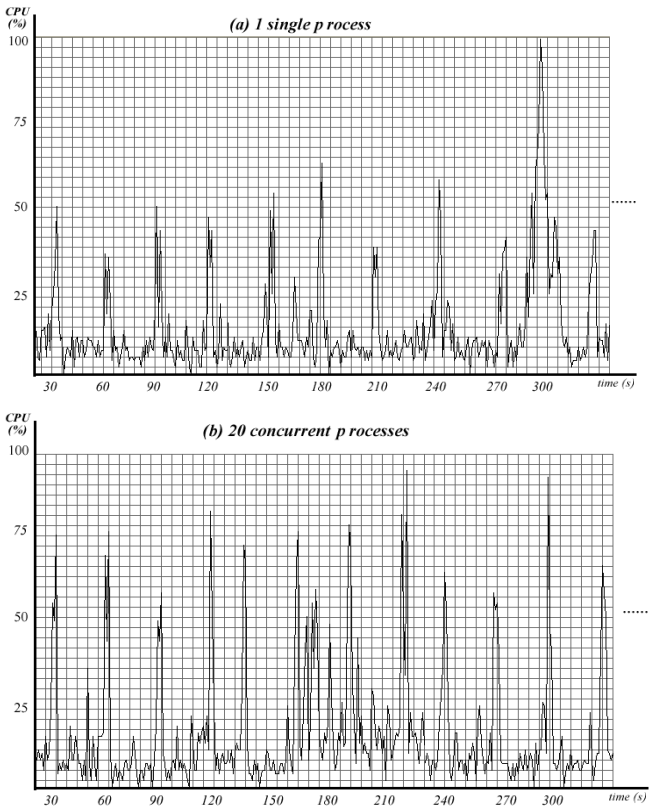
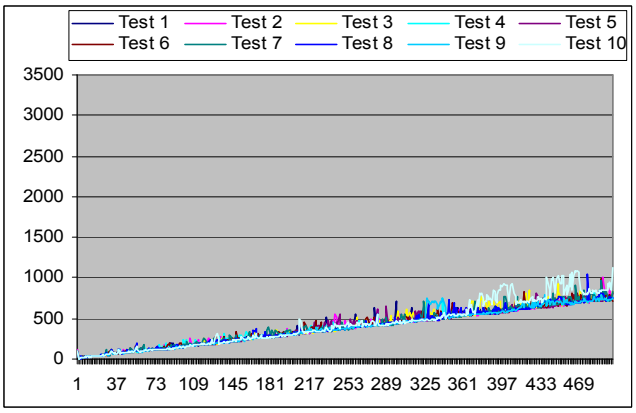


Figure 12. CPU usage measure (a) one process, (b) 20 concurrent processes.

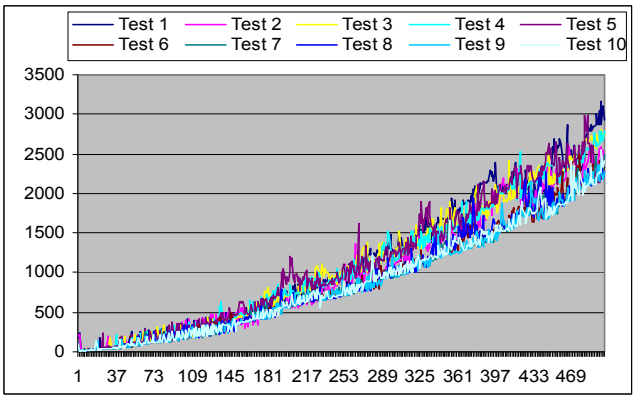
Each test was repeated 50 times ($n=50$) and with the interval between each run is 30s. On average, 50.38% of CPU was used when one process was executed with the standard deviation of 13.19%, and 65.16% of CPU was used when 20 processes were executed with the standard deviation of 11.54%. There is no significant difference between the two tests. This result indicates that our approach is scalable to support multiple processes.

2) Behavioral Adaptation

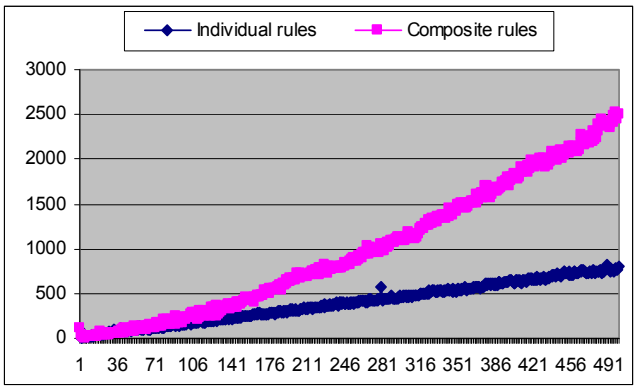
CAT adopts ECA rules to express behavioral constraints. For example, ECA rules can be used to specify contextual interaction rules such as "when rain is detected, CAT will change the following distance between two vehicles from 300m to 200m". These rules could be defined at design-time or dynamically loaded at run-time. Using the Drools engine, CAT is able to load multiple individual rules concurrently as well as loading a compound rule. Figure 13 shows the response time comparison of dynamic rule loading. In particular, Figure 13(a) presents the results of 10 test runs, each of which is tested with the maximum number of 500 individual rules. Figure 13(b) shows the results of 10 test runs, each run involves loading a compound rule that contains 1 to 500 individual rules. Figure 13(c) compares the response time of the two rule loading settings. The results indicate that given the same number of rules, loading a compound rule took much more time than loading individual rules concurrently.



(a) Test runs of loading concurrent individual rules



(b) Test runs of loading compound rules



(c) Average loading times of concurrent individual rules and compound rules

Figure 13. Response time measures of dynamic rule load (X-axis: number of rules, Y-axis: milliseconds).

The above evaluation shows the feasibility of implementing a range of context-aware telematics using a service-based approach. These applications can readily be deployed as service compositions whose service can be discovered and bound at runtime. Furthermore the performance testing shows that this approach is scalable to a requisite number of simultaneous applications. Response times for bindings showed a small performance hit for dynamic binding relative to static binding. Whether or not the absolute binding time is an issue will depend on the application. In many scenarios, such as the convoy and road-side assistance, the time taken for binding to context services will not be significant relative to the task.

V. Related Work

Previous researchers have investigated a number of aspects of service-oriented computing in the automotive domain (e.g., formal modeling, verification, intra- and inter-vehicle networked service interoperability, automatically generated UI for dynamic services, etc.). er Beek et al. [23] studied the use of formal modeling and verification techniques in the requirement analysis phase of automotive software development. The researchers adopt UML profile and UML state machines to model the system requirements, and use the on-the-fly model checker to verify the correctness of properties. Bisdikian et al. [24] developed an open telematics platform using existing Web service interfaces and the Tiered Services over Public Wireless LANs (ts-PWLAN). Whilst the research shows some progress in bringing services into vehicular networks, only simple forms of context is addressed by the research. Moreover, the proposed platform involves tightly-coupled interaction between entities. Thus, it is limited in scale to support open and dynamic environment as analysed above.

Yu et al. [25] and Zhang et al. [7] focused on developing Open Service Gateway Initiative (OSGi) as an standard-based infrastructure for context-aware telematics. They adopt OSGi to connect internal and external vehicular networks (e.g., CANbus, MOST, Bluetooth, GPRS, etc.), and an ontology to model context information and relationships between context attributes. However, the research presents limited details on how telematics systems use context information to adapt to changes in the environment systematically. The Sentient Car project [26] investigated an implementation of a vehicle-based sentient space where telematics systems are context-aware and has the ability to adapt to changing environments. However, the approach is relied on a tightly-coupled model of telematics systems, and only simple forms of context (e.g., location, air pollution sensors) are used. Wu et al. [5] developed ScudWare which is a semantic and adaptive middleware platform for smart vehicle space. ScudWare aims to support context-aware interaction and semantic integration between entities in smart vehicle space by integrating a multi-agent technique with a context-aware and adaptive management service. Whilst ScudWare is rich in functionality, its architecture model is heavily loaded due to a multi-agent platform. Also, it is unclear how a semantic virtual agent is used to support a set of related tasks. Furthermore, ScudWare uses ontology to classify properties of users, environment and vehicle, but does not take into account of the relationships between these three entities. Hildisch et al. [27] focused on implementation techniques to automatically generate UI for dynamic services in the automotive domain. As services could be composed at run-time, they are unknown at design time of the human-machine interface (HMI) system. They proposed that each service could be accommodated by a description of the service's semantics. This description describes semantic UI of the service that can be used to integrate the service with the existing HMI system.

Compared to previous research, our service-based approach emphasizes models of context-aware interactions (i.e., social context) between components of automotive telematics systems, and adopts services to convey context and support adaptation. The architecture of our approach supports the separation of concerns, different levels of abstraction (e.g., physical context and social context) and dynamic binding. Moreover, our approach is able to support both structural and behavioural adaptation. The empirical evaluations showed that, with respect to overhead at least, the implementation approach is feasible and scalable for telematics applications.

VI. Conclusions and Future Work

This paper has presented our service-based approach to developing context-aware automotive telematics systems (referred to as *telematics* for short). Our approach utilizes services as a means to acquire situation context (both physical and social contexts) and assist telematics in adapting to the contexts. In particular, we have developed a layered architecture for developing telematics and managing the contexts. The architecture enables the separate development of telematics and the management of context and adaptation. We have validated our approach in the implementation of a context-aware telematics (CAT) prototype. CAT supports cooperative driving tasks by taking into account the physical and social contexts about vehicle, driver and environment. An empirical study was carried out to evaluate various aspects related to structural and behavioral adaptation of CAT. The evaluation indicates the feasibility of our approach in terms of the overhead imposed by role generation, binding, concurrent processes, binding, and rule processing. However, much work needs to be done to provide modeling/implementation methods and tools to support developers using such an approach. As

future work, we will focus on other aspects of service-based development in context-aware automotive software, including verification of adaptation, QoS management, trust and security.

ACKNOWLEDGMENT

This original research was supported by the Commonwealth of Australia, through the Cooperative Research Centre for Advanced Automotive Technology (AutoCRC). We would like to thank anonymous reviewers for their comments.

REFERENCES

[1] M. Broy, "Challenges in automotive software engineering," in *Proceedings of the 28th International Conference on Software Engineering*, Shanghai, China, 2006, pp. 33-42.

[2] A. Polychronopoulos, M. Tsogas, A. J. Amditis, and L. Andreone, "Sensor Fusion for Predicting Vehicles' Path for Collision Avoidance Systems," *IEEE Transactions on Intelligent Transportation Systems,* vol. 8, pp. 549-562, 2007.

[3] J. C. McCall, D. P. Wipf, M. M. Trivedi, and B. D. Rao, "Lane Change Intent Analysis Using Robust Operators and Sparse Bayesian Learning," *IEEE Transactions on Intelligent Transportation Systems,* vol. 8, pp. 431-440, 2007.

[4] M. C. McCallum, J. L. Campbell, J. B. Richman, J. L. Brown, and E. Wiese, "Speech Recognition and In-Vehicle Telematics Devices: Potential Reductions in Driver Distraction," *International Journal of Speech Technology,* vol. 7, pp. 25-33, 2004.

[5] Z. Wu, Q. Wu, H. Cheng, G. Pan, M. Zhao, and J. Sun, "ScudWare: A Semantic and Adaptive Middleware Platform for Smart Vehicle Space," *IEEE Transactions on Intelligent Transportation Systems,* vol. 8, pp. 121-132, 2007.

[6] M. H. Tran, J. Han, and A. Colman, "Social Context: Supporting Interaction Awareness in Ubiquitous Environments," in *Proceedings of International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous'09)*, Toroto, Canada, CD ISBN: 978-963-9799-59-2, 2009.

[7] D. Zhang, X. H. Wang, and K. Hackbarth, "OSGi based service infrastructure for context aware automotive telematics," in *IEEE 59th Vehicular Technology Conference*, 2004, pp. 2957- 2961.

[8] T. Strang and C. Linnhoff-Popien, "A Context Modeling Survey," in *Workshop on Advanced Context Modelling, Reasoning and Management, as part of the Sixth International Conference on Ubiquitous Computing UbiComp'2004*, Nottingham, England, 2004.

[9] A. Colman, M. H. Tran, and J. Han, "An Adaptive Architecture for Context-Aware Interaction in Pervasive Applications," in *International Workshop on Context-Aware Pervasive Communities: Infrastructures, Services and Applications (CAPC 2008) Held in Conjunction with the 6th International Conference on Pervasive Computing (PERVASIVE 2008)*, Sydney, Australia, 2008, pp. 237-244.

[10] A. K. Dey, G. D. Abowd, and D. Salber, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications," *Human-Computer Interaction,* vol. 16, pp. 97-166, 2001.

[11] J. I. Hong and J. A. Landay, "An Infrastructure Approach to Context-aware Computing," *Human-Computer Interaction,* vol. 16, pp. 287-303, 2001.

[12] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and B. H. C. Cheng, "Composing Adaptive Software," *IEEE Computer,* vol. 37, pp. 56-64, 2004.

[13] A. Elfatatry, "Dealing with Change: Components versus Services," *Communications of the ACM,* vol. 50, pp. 35-39, 2007.

[14] M. P. Papazoglou and W.-J. van den Heuvel, "Service oriented architectures: approaches, technologies and research issues," *The International Journal on Very Large DataBases (VLDB),* vol. 16, pp. 389-415, 2007.

[15] W. Wibisono, A. Zaslavsky, and S. Ling, "Towards a Service-Oriented Approach for Managing Context in Mobile Environment," in *Proceedings of the Sixth International Conference on Service-Oriented Computing (ICSOC'08)*, Sydney, Australia, 2008, pp. 210-224.

[16] J. King and A. Colman, "A Multi Faceted Management Interface for Web Services," in *Proceedings of the Australian Software Engineering Conference 2009 (ASWEC'09)*, Gold Coast, Australia, 2009, pp. 191-199.

[17] A. Colman, "Role-Oriented Adaptive Design," in *PhD Thesis, Faculty of Information and Communication Technologies* Melbourne, Australia: Swinburne University of Technology, 2006.

[18] A. Colman and J. Han, "Using Role-based Coordination to Achieve Software Adaptability," *Science of Computer Programming,* vol. 64, pp. 223-245, 2007.

[19] "Javaassist," http://www.csg.is.titech.ac.jp/~chiba/javassist/.

[20] "WSO2 Registry," http://wso2.com/.

[21] "Drools: Business Logic Integration Platform," http://www.jboss.org/drools/.

[22] "Active Endpoints," http://www.activevos.com.

[23] M. H. ter Beek, S. Gnesi, N. Koch, and F. Mazzanti, "Formal verification of an automotive scenario in service-oriented computing," in *Proceedings of the 30th International Conference on Software Engineering (ICSE' 2008)*, Leipzig, Germany, 2008, pp. 613-622.

[24] C. Bisdikian, I. Boamah, P. Castro, A. Misra, J. Rubas, N. Villoutreix, D. Yeh, V. Rasin, H. Huang, and C. Simonds, "Intelligent pervasive middleware for context-based and localized telematics services," in *Proceedings of the 2nd international workshop on Mobile commerce (WMC'2002)*, Atlanta, Georgia, 2002, pp. 15-24.

[25] Z. Yu, X. Zhou, Z. Yu, D. Zhang, and C.-Y. Chin, "An OSGI-based infrastructure for context-aware multimedia services," *IEEE Communications Magazine,* vol. 44, pp. 136-142, 2006.

[26] P. Vidales and F. Stajano, "The Sentient Car: Context-Aware Automotive Telematics," in *First European Workshop on Location Based Services (LBS)*, London, UK, 2002.

[27] A. Hildisch, J. Steurer, and R. Stolle, "HMI Generation for Plug-in Services from Semantic Descriptions " in *Proceedings of the 4th International Workshop on Software Engineering for Automotive Systems* Minneapolis, 2007.