2009 35th Euromicro Conference on Software Engineering and Advanced Applications

Modeling Service Oriented Architectures of Mobile Applications by Extending SoaML with Ambients

Nour Ali, Muhammad Ali Babar
Lero, University of Limerick, Limerick-Ireland
{Nour.Ali, Muhammad.AliBabar}@lero.ie

Abstract

Mobile applications need to dynamically adapt to requirements of new environments (or locations) as users and their devices continuously move. Service Oriented Architecture (SOA) is a recent approach for designing and developing open and distributed systems. However, SOA has to be extended in order to fully accommodate the requirements of mobile services. This paper presents an approach called AmbientSoaML, which introduces ambients in Service oriented architecture Modeling Language (SoaML) [8] proposed by the OMG in order to allow its models to include mobility primitives. Ambients are considered to be the service providers and the service consumers for providing/consuming mobility services. They also represent the boundaries that services have to cross when moving from one location to another. This paper demonstrates the use of SoaML for modeling SOA of a mobile application in order to motivate the problem our research purports to address.
Keywords: SOA, SoaML, ambients, mobility.

1. Introduction

Next generation of mobile technologies are characterized with autonomous, dynamically adaptable, and heterogeneous components collaborating in open intra-organizational environments to provide solutions. Mobile applications are one of the kinds of systems that need to dynamically adapt to requirements of new environments (or locations) as users and their devices continuously move. Hence, it is becoming increasingly important to provide software engineering techniques that support the development of loosely coupled, platform independent, flexible and dynamic software.

Service Oriented Computing (SOC) principles [1] and Service Oriented Architecture (SOA) [2] are recent approaches for designing and developing open and distributed systems. One of the principles of designing service-based systems is that boundaries are explicit. These can be at different levels such as processes, network, or geographical boundaries. Crossing boundaries is an expensive activity as it requires reconfiguration or security [3].

Previous work reports that SOA is appropriate for Mobile Services but it needs to be extended with several conceptualizations such as providing models for describing mobile services, as well as providing a mobility controller for coordinating mobile services [4].

Additionally, we assert that Mobile SOA should be extended with notions for representing the environment (or locations) boundaries where services should be adapted when mobility occurs. This would facilitate the understanding of how mobile services need to adapt to their new environments, policies and constraints.

Ambient Calculus [5], [6] provides an abstraction for modeling mobility. It introduces the concept of ambient, which represents a bounded place where computation occurs. Ambients can model the location hierarchy encountered in distributed systems and model the mobility as the crossing boundaries by entering and exiting capabilities.

Previously, ambients have been introduced for Component Based Software Architectures [7]. This work presents an approach called AmbientSoaML which incorporates ambients in SOA. Concretely ambients are introduced to the Service oriented architecture Modeling Language (SoaML) [8] proposed by the OMG in order to allow its models to include mobility primitives. Ambients are considered to be service providers and service consumers for providing/consuming mobility services. They also represent the boundaries that services have to cross when moving from one location to another. We motivate the need of extending SoaML for mobile services by building a SOA model of a mobile application using SoaML, and later extending it with ambients for representing movements across boundaries.

The structure of the paper is the following: Section 2 gives an overview of SoaML, and Ambient Calculus. In Section 3 we describe an example of GSM in Mobile Networks. Then, SoaML is used to model a SOA of the example and highlight the limitations of SoaML for modelling the mobility characteristics of the GSM. Section 4 presents Ambient SoaML and shows how Ambient SoaML is used to model the SOA of the GSM in the Mobile Networks with mobility characteristics. Section 5 presents the related works and Section 6 concludes the paper.

978-0-7695-3784-9/09 $26.00 © 2009 IEEE
DOI 10.1109/SEAA.2009.25

2. Background

In the following, SoaML metamodel and Ambient Calculus are presented. These approaches have inspired our work for defining Ambient SoaML.

2.1. Soa Modelling Language (SoaML)

The SoaML metamodel extends the Unified Modeling Language 2.0 (UML) in order to support SOA. This section only presents the concepts of SoaML that are extended in section 3. The concepts used are based on the revised UPMS submission presented in [8]. In the following, the concepts are the presented (see white classes in **Error! Reference source not found.**):

- *ServiceInterfaces* describe the operations used between a service provider and a service consumer from the perspective of the provider. *ServiceInterfaces* are used as a type of a *ServicePoint* or a *RequestPoint*. A *ServiceInterface* can imply the realization and usage of one or more UML interface.
- *ServiceContracts* define the terms, conditions, interfaces and choreography that interacting participants must agree. They specify how services are provided and consumed based on interactions and behaviours involving the participants.
- *Participants* allow defining the service providers and consumers. When a *Participant* is a provider it contains at least a *ServicePoint*. A *ServicePoint* defines a capability offered by one entity to others. When a *Participant* is a consumer it contains at least a *RequestPoint*. A *Requestpoint* defines the connection point through which a *Participant* makes requests or consumes services. A participant can be a consumer, a provider or both.
- *ParticpantArchitectures* are the high level view of a SOA that defines how a set of participants work together for providing and using services. *ServiceContract* instances can be included in an architecture and when they are related to a *Participant* it implies that the *Participant* fulfils the contact.
- *ServiceChannels* provide a communication path between consumer Requests (ports) and provider services (ports).

2.2. Ambients in Ambient Calculus

Ambient Calculus (AC) [5] is a process algebra for modelling both mobile computation (logical mobility) and mobile computing (physical mobility) in a uniform way. The authors of AC argue that the main difficulty with mobility is the handling of administrative domains. They state that networks are not flat, instead they are partitioned in levels of administrative domains, and that mobility involves the authorization to enter or exit certain domains. As a result, AC aims to be a model that captures the notion of locations of mobility and authorization to move.

AC introduces a concept called ambient. An ambient is a place that is limited by a boundary and where computation happens. The main characteristics of an ambient are explained [5], [6]:

- An ambient is a bounded place where computation happens. A boundary determines what is inside or outside an ambient. As a result, the boundary of an ambient determines what can move. Examples of ambients are a webpage, a laptop, or a mobile phone.
- Ambients can be nested within other ambients, forming a tree structure. Administrative domains are organized hierarchically. In this way, mobility is represented as navigation across a hierarchy of ambients.
- Each ambient has a collection of local running processes.
- Each ambient moves as a whole with all its subcomponents. When an ambient moves, it moves its local processes and its sub-ambients. For example, when a laptop moves all its threads, address spaces, and file systems within it move.

The moving capabilities of ambients are exiting and entering ambients. An exit capability allows an ambient to exit its parent ambient. An enter capability allows an ambient to enter into a sibling ambient (see Figure 1).

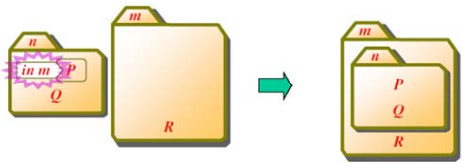


Figure 1. Applying the enter capability to the ambient *n* taken from [9]

3. Example: GSM in Mobile Networks

This section presents the example used to explain our work presented in this paper.

A mobile network offers two services to mobile devices: the voice telephony service and the short message service (sms). A mobile device can be a mobile phone or a PDA that has a Subscriber Identity Module (SIM) card which is issued from a mobile network. In GSM, mobile devices can enter to mobile networks which have not issued their SIM card.

A mobile device can only enter a new mobile network, if the new mobile network accepts the mobile device. The mobile network has to check that the mobile phone is not blacklisted before accepting it. Additionally, every network only accepts mobile devices that fulfil specific power requirements. As a result, a mobile phone

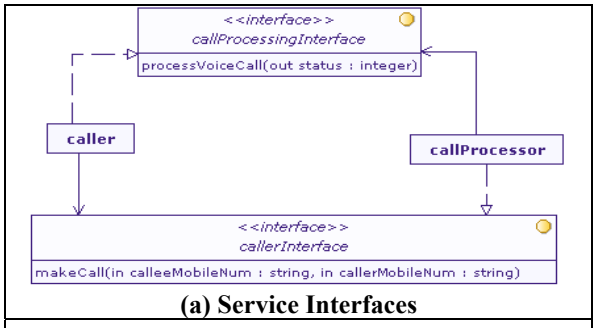
has to send its mobileId, simId, and power level for entering to a mobile network.

When a mobile device is accepted in a mobile network, the receiving network checks whether the SIM card is issued from it or not. If it is not, it checks the original mobile network of the SIM card and it communicates with the original mobile network. The new mobile network registers the mobile device as a visitor and offers it the telephony service. However, the sms service is offered by the original mobile network.

3.1 Using SoaML for Modelling GSM in Mobile Network example

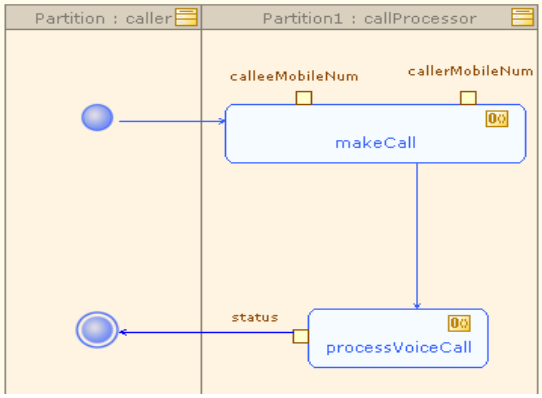
In this section, SoaML metamodel is used to model the GSM example. We have used the Objecteering SOA modeling tool [11] to model our example.

The telephony service is modelled in Figure 2. There are two Service Interfaces: *caller* and *callProcessor* (see Figure 2 (a)). The *caller* service interface uses the *callerInterface* and realizes the *callProcessorInterface* interfaces. The *caller* service interface represents a role that uses the *makeCall* operation to send the information needed for making a call through the *calleeMobileNum*, and the *callerMobileNum* parameters. The *caller* role will receive the *status* of the call (e.g., busy, connected, unavailable, etc) through the *processVoiceCall* operation of the *callProcessingInterface*. The *callProcessor* service interface represents the role that will receive the information needed for making the call and will use the *callProcessingInterface* for sending to the caller the status of the call. Figure 2 (b) shows the *VoiceTelephony* service contract represented in a collaboration diagram with the *caller* and the *callProcessor* roles. The choreography of the contract (see Figure 2 (c)) is represented in an activity diagram which shows the order of the service invocations. The *caller* initiates the voice telephony service by making a call request which the *callProcessor* receives with the needed information, and then it processes the voiceCall, and sends back the status of the call to the caller.



(a) Service Interfaces

(b) Service Contract



(c) Service Contract Choreography

Figure 2. VoiceTelephony Service

Figure 3 shows the SoaML specification of the sms service contract. It can be noticed that the sms service is a composed service[1]. The *sms* service is composed of two simpler services: the *SendingSms* and the *receivingSms*. In this scenario, the *SmsProcessor* plays two roles: the *Processor* of the *sendingSms* service, and the deliverer for the *deliveringSms* service. There are two other roles are the *SmsSender* and the *SmsReciever*.

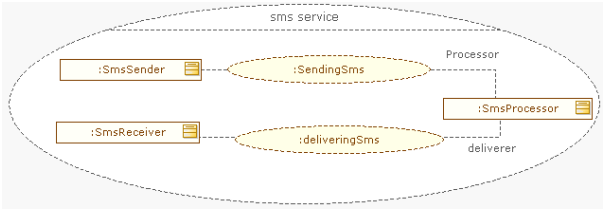


Figure 3. sms Service Contract

Figure 4 shows the architecture of the example. It shows that there are three participants: *MobileDevice*, *smsProcessor*, and *voiceTelephonyProcessor*. The *MobileDevice* participant has two *RequestPoints*: one for sending sms, and one for making calls. It also has a *ServicePoint* for receiving sms. The *smsProcessor* and *voiceTelephony* participants are the service providers of a *MobileDevice*. The *MobileDevice* is also a service provider to the *smsProcessor* when it plays the role of a sms receiver. There are three service channels in the architecture connecting the ports of the participants.

[1] The voicetelephony service can also be specified as a composite service. However, for simplicity we will assume it is a simple service.

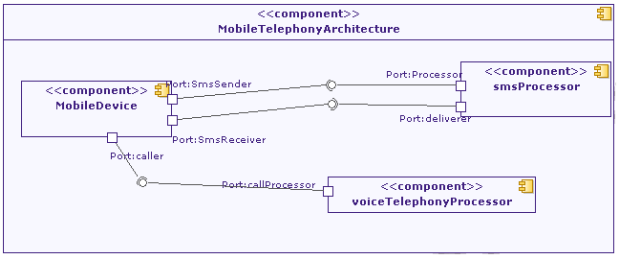


Figure 4. SoaML architecture of example

3.2 Discussion

It can be noticed from the above discussions and architectural models, that the full requirements of the example cannot be modelled using only SoaML. These requirements are the ones concerned with mobility. The business logic is defined using SoaML, e.g., the sms service. However, SoaML is not capable of modelling the possible movements of a MobileDevice between different networks. The Mobile Device has to exit one network and enter to a new one after which the new network accepts it. In this case, a notion for boundaries is needed in order to provide capabilities of these boundaries to accept new mobile devices. Hence, it is important to have an approach that can provide the notion of boundaries for modelling the movements of a mobile device across different boundaries.

Moreover, the abovementioned model does not include a mechanism of modelling the situation when a MobileDevice (with the SIM card) is in a different Mobile Network than the original one, the *voiceTelephonyProcessor* of the new network should provide the *voiceTelephony* service, and the *smsProcessor* of the original network will serve the sms service to the Mobile Device. This logic is related with mobility and implies that the Mobile Networks (the boundaries) have agreed (contracts) to allow these kinds of changes in the service compositions. As a result, mobility implies crossing boundaries and adapting service channels of a SOA.

Hence, it is clear that there should be an approach for modelling mobility concerns separately from the business logic. In this way, a separation of concerns is also achieved and can be traced from architecture design to the later stages of software development lifecycle.

4. Ambients in SOA

In order to deal with the issues discussed in the previous section and address the limitations of SoaML, we propose to extend SoaML [8] with Ambient Calculus for supporting the design of SOA based mobile systems in a technology independent way. In the following, we describe how our approach extends the SoaML metamodel in order to incorporate Ambients. We call this approach AmbientSoaML. Then, we remodel the GSM Mobile Network architecture using the AmbientSoaML.

4.1. AmbientSoaML- Extending SoaML with Ambients

In this section, the SoaML metamodel is extended for modeling a SOA architecture with mobility. Specifically, the metamodel is enriched with the following concepts (see **Error! Reference source not found.**):

- An *Ambient* is a bounded place where participants can be located and where they can request and provide services. Ambients represent the service environments and can control what can be part of the environment they represent. An ambient has been introduced as a special kind of participant because it can also offer and request services. Since an ambient can also have participants in its boundary which can request and provide services, it has an *AmbientArchitecture*.
- An *AmbientArchitecture* defines the boundary of the environment which the ambient represents. These boundaries can be geographical, physical, etc. The boundary of an ambient can include other ambients and participants. These participants can request services from their parent ambient and from their sibling participants (which can be ambients or not). Services can be requested/provided from the exterior. However, these requests/provisions imply a crossing of boundaries which the parent ambient needs to control e.g. for ensuring security. This means that there can be service channels among siblings and between parent ambients and their children. However, there cannot be Service Channels between participants of different ambients. To represent the crossing of boundaries between distributed services, service channels among ambients of distributed services have to be created.
- A *MobilityServiceInterface* is an interface that describes the operations needed for moving.
- A *MobilityService* is a capability offered by an ambient for providing mobility. Currently, we have included the basic Ambient Calculus capabilities of *entering* and *exiting* ambients. An ambient must provide at least these two services. The *entering* service allows an external ambient to become the child of an ambient. The *exiting* service allows a child of an ambient to become its sibling. These capabilities are the primitives that cause changes needed for adaptability in mobility. In the future, the model can be extended by adding more mobility capabilities. A *MobilityService* is represented by a Port (Service Point) on an Ambient, through which the service is delivered. A *MobilityService* meets the needs of a consumers Request as defined in a

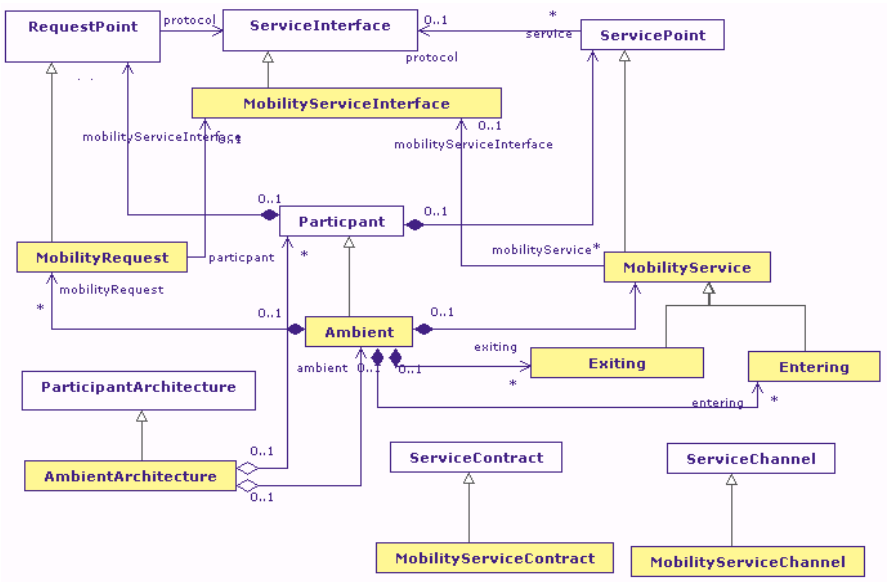


Figure 5. AmbientSoaML metamodel

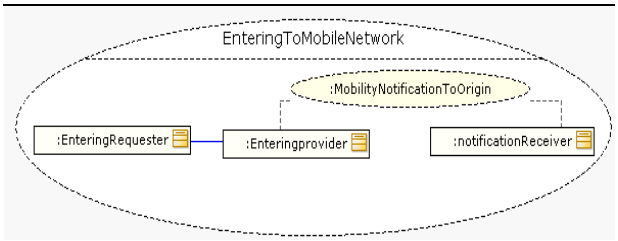
MobilityServiceInterface and can also designate a role in a *MobilityServiceContract*.

- *MobilityServiceContract* is the specification of the agreement between the consumer (an ambient) of a mobility service and the provider of a mobility service (another ambient).

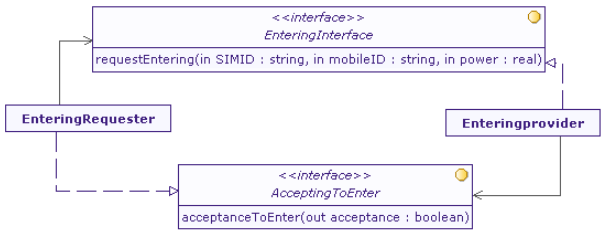
4.2. Modelling Ambients in Ambient SoaML

In the following, we are going to model the needed concepts for defining ambients of the GSM Mobile Network using AmbientSoaML. It has to be taken into account that since AmbientSoaML includes new primitives for mobility basing on SoaML, the designer can use SoaML to model the functionality without mobility and then model the mobility characteristics. We have already demonstrated how to use SoaML for modelling the functionality of the GSM Mobile Network in section 3.1.

Entering Service - The entering service can be modelled differently depending on each domain. In the case of our example, the definition of the Entering service depends on three roles (which are MobilityService Interfaces): the *EnteringRequester*, the *Enteringprovider* and the *notificationReceiver* (see Figure 6 (a)). It can be noticed that the Entering service is a composite service because its roles use the *MobilityNotificationToOrigin* MobilityService contract. This contract is needed in order to notify the original mobile network of a mobile device that it has moved. Figure 6 (b) shows the definition of the *EnteringRequester, and Enteringprovider* roles. The *EnteringRequester* realizes the *EnteringInterface* and uses the *AcceptingToEnterInterface*. The *EnteringInterface* has an operation called *requestEntering* which has three parameters: *SIMID*, *mobileID*, and *power*. The *AcceptingToEnter* interface has the *acceptanceToEnter* operation which has an *out* parameter called *acceptance* for returning the answer if the requester is accepted to enter or not. The ambient provider has to provide the *requestEntering* and invokes the *acceptanceToEnter* from the consumer. The *EnteringRequester* is the *MobilityServiceInterface* corresponding to the role of the mobile ambient.



(a) *EnteringToMobileNetwork* MobilityContract



(a) Mobility Contract *EnteringToMobileNetwork*

Figure 6. The *EnteringService*

Distributed Sms service - When a mobile device enters a new ambient, the sms service becomes distributed, i.e., the mobile device would be in an ambient different from the original ambient. As a result, a distributed sms service is defined in see Figure 7. In this way, two new roles are defined for providing a distributed sms service: smsRecepAmb and smsOrigAmb.

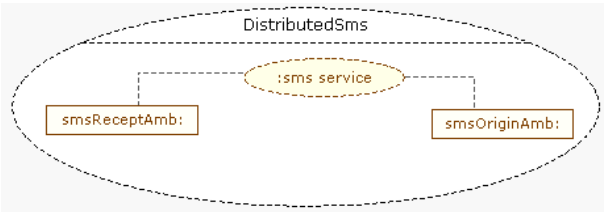
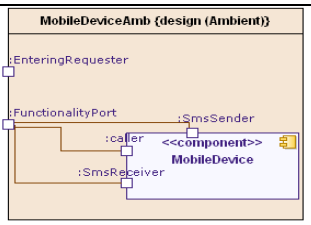


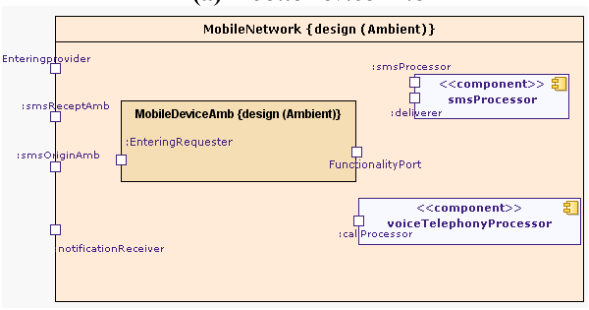
Figure 7. The *Distributed Sms service*

Ambients - In the example, we can distinguish two types of ambients (see Figure 8):

- Mobile Device Ambients: Their boundary is determined by the physical device. In section 3.1, the Mobile Device has been modelled as a component. This component would contain the functional parts. Since the mobile device is mobile, an ambient should contain the functional parts. As a result, an ambient called *MobileDeviceAmb* exists. The *MobileDeviceAmb* has a MobilityRequest port of type *EnteringRequester*. In addition, the *MobileDeviceAmb* has a port which is composite of the functional ports of the *MobileDevice* participant.
- Mobile Network Ambients: Their boundary is determined by the radio network of the Base Transceiver Stations. This kind of ambient is not mobile and is a provider of the entering capability. As a result, it has the MobilityService port of type *Enteringprovider*. A Mobile Network can only locate ambients of kind MobileDeviceAmb. This restriction is important because a Mobile Network can locate other Mobile Networks. In addition, the Mobile Network includes two partcipicants (components): the *smsProcessor* and the *voiceTelephonyProcessor*.



(a) *MobileDeviceAmb*



(b) *MobileNetwork*

Figure 8. Ambients

4.3 Mobile SOA configurations of the GSM example

When different kinds of ambients are defined, instances can be created in order to build configurations of mobile SOA. This section describes the different configurations that occur when a mobile device moves according to the models presented in the previous sections.

Figure 9 shows a configuration of the architecture with ambients. It shows that there is an instance of a *MobileDeviceAmb* ambient called *MP1*. It is currently configured in a MobileNetwork ambient instance called *MovistarSpain*. *MovistarSpain* has a sibling ambient instance called *VodaphoneIreland*. Since *MP1* is located in *MovistarSpain* (its original MobileNetwork), it has service channels that connect it to the participant instances of *smsPMov*, and to the *vTMov*.

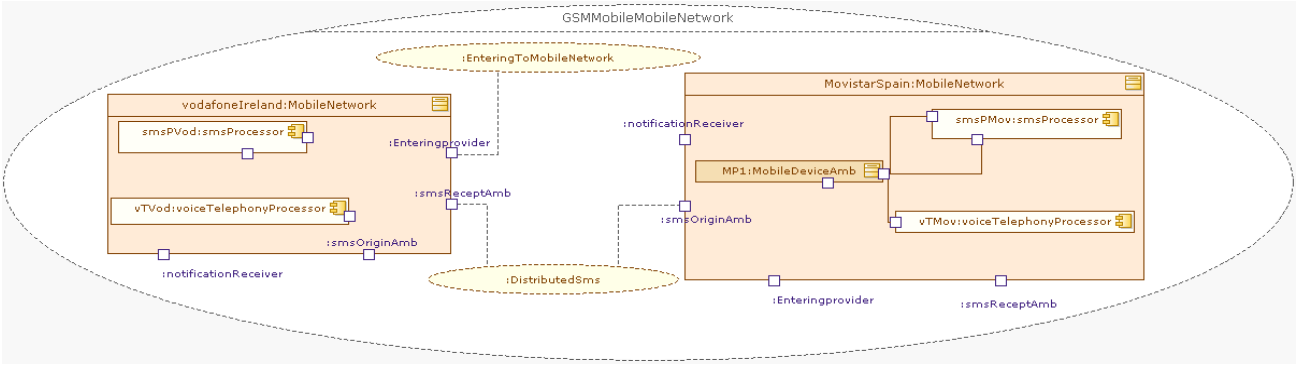


Figure 9. A Configuration of Ambients in SOA when MP1 is in MovistarSpain

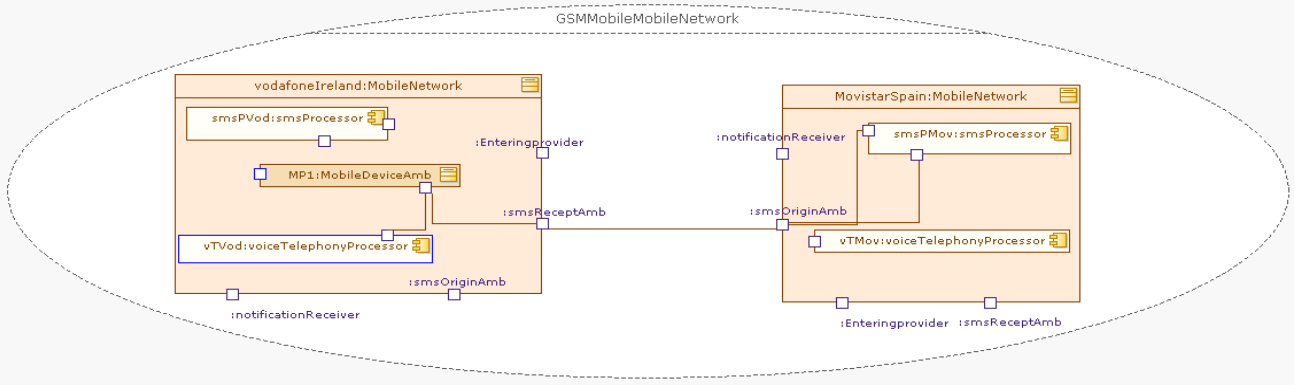


Figure 10. A Configuration of Ambients in SOA when MP1 has entered vodafoneIreland

When the *MP1* ambient instance exits the *MovistarSpain* ambient, the MP1 satisfies the mobility service contract of the *EnteringToMobileNetwork*. As a result, the *vodafoneIreland* notifies the *MovistarSpain* ambient, and *MovistarSpain* removes the service channels that MP1 had with the *vTMov* instance. Then, the *MP1* enters the *vodafoneIreland* ambient. As a result, the configuration of the SOA architecture changes to become the one shown in Figure 10.

5. Related Work

Previously, UML has been used for modeling mobile services. The work in Belaunde et al. [12] presents the usage of a UML dialtect called SPATEL for modelling SOA of mobile phones. SPATEL annotates service interfaces with non functional features required in mobile phones as well as representing different GUI frameworks. These features are important, however, AmbientSoaML provides primitives for modeling the distributed SOA as well as mobility. Other researchers, such as Maamar et al. [13] extend UML state chart diagrams to include locations for service compositions purposes. AmbientSoaML provides an explicit notion of locations independently of the diagrams used.

Ambients have also been previously used in service oriented applications. Loke et al. [14] use service domains (similar to our concept of ambient) for referring to geographical boundaries that are associated with a set of services. Each service domain stores mapping tables which stores ambient services (services in a boundary) and operators are used for computing services of a user profile. The implementation techniques used in Loke et al. could be used in AmbientSoaML in future works for calculating services of user profiles.

In our approach, however, ambients are inspired from AC. That is, ambients are more abstract and can model different kinds of boundaries not only geographical. The permissions of entering boundaries are also taken into account by the use of the mobility capabilities. In addition, in Loke et al.'s work, users can only use the services of the boundary they are in. However, in some cases (as shown in the GSM example) services that are provided in different ambients are needed.

The work of Celentano et al. [15] discusses the use of ontologies for ambient design in service based applications. Our approach is similar, since AmbientSoaML is an ontology which follows an OMG standard. AmbientSoaML in comparison to Celentano et al. focuses on modelling SOA as well as mobility.

6. Conclusions and further work

This paper presents an approach called AmbientSoaML for modelling SOA for mobile systems. AmientSoaML enriches SoaML with ambients in order to provide an explicit notion of location boundaries that mobile services are expected to cross. We have also introduced the concept of ambient for modelling mobility with entering and exiting capabilities. In this way, ambients are in charge of controlling boundary crossing procedures. This is important for adaptability in mobility, since ambients can provide a transparent way of the dynamic adaptations needed for mobility. In addition, separation of concerns is also achieved as *MobilityServiceContracts* are specialized for mobility agreements independently of the rest of the business agreements.

Currently, we are working on modelling non-functional requirements which are important in mobile systems. Ambients are used in order to include non functional requirements that are common for the boundaries they represent.

In addition, we are also working to exploit the Model-Driven Architecture (MDA) approach for modeling and implementing SOA of mobile systems. Concretely, we are working on including AmbientSoaML metamodel in the Eclipse Modelling Framework (EMF). Then, use the Graphical Modeling Framework (GMF) for building our graphical editor that will allow users to model AmbientSoaML SOAs. At the same time, we are working on implementing a lightweight middleware for

mobile devices. This middleware will map AmbientSoaML to a technological specific platform. Finally, we will implement transformations using Query/View/Transformation (QVT) in order to transform from AmbientSoaML technology independent models to the specific models, and use MOF script for generating executable code.

7. Acknowledgements

This work was supported by Science Foundation Ireland grant 03/CE2/I303_1 to Lero - the Irish Software Engineering Research Centre.

8. References

[1] Papazoglou, M.P.: Service-Oriented Computing: Concepts, Characteristics and Directions. In: Proc. of WISE 2003, Roma, Italy, December 10-12, pp. 3–12 (2003)

[2] Singh, M., P., Huhus, M. N., "Service-Oriented Computing Sematics, Processes, Agents", John Wiley & Sons, ISBN: 0-470-09148-7.

[3] MSDN library, "Service Oriented Architectures", http://msdn.microsoft.com/en-us/library/bb833022.aspx

[4] Jorstad, I, Dustdar, S., Thanh, D. V., "Service-Oriented Architectures and Mobile Services", CAiSE Workshops (2), pp. 617-631, 2005.

[5] Cardelli, L., Gordon, A. D. "Mobile Ambients", Foundations of Software Science and Computational Structures: First International Conference, FOSSACS '98, LNCS 1378, Springer, 1998, pp. 140-155.

[6] Cardelli, L. "Abstractions for Mobile Computation", In Vitek, J. and (Eds.), C. J., editors, Secure Internet Programming: Security Issues for Distributed and Mobile Objects, volume 1603 of LNCS, Springer Verlag, pp. 51-94, 1998.

[7] Ali, N., Millan, C., Ramos, C., "Developing Mobile Ambients Using an Aspect-Oriented Software Architectural Model", International Conference Proceedings of Distributed Objects and Applications (DOA) 2006, LNCS 4276, 2006.

[8] SoaML Service oriented architecture modeling language- a response to the OMG UPMS UML profile and Metamodel for Services, December, 2008. Available at: http://www.omg.org/docs/ad/08-12-17.pdf

[9] Cardelli, L., Gordon, A., "Principles of Wide Area Programming", Presentation at the 13th International School for Computer Science Researchers Lipari Island. Available at: lucacardelli.name/Slides/200107%20%20Lipari%20School%20-%2002%20Ambient%20Calculus.pdf

[10] Meta-Object Facility (MOF) 1.4 Specification. TR formal/2002-04-03. http://www.omg.org/technology/documents/formal/mof.htm

[11] Objecteering SOA Free Edition: http://www.objecteering.com/solutions_soa_solution.php

[12] Belaunde, M., and Falcarin, P., "Realizing an MDA and SOA Marriage for the Development of Mobile Services", Proceedings of the 4th European conference on Model Driven Architecture: Foundations and Applications ECMDA-FA, pp. 393-405 2008.

[13] Maamar, Z., and Lahkim, M., "A Specification Approach to Compose Mobile Web Services Using Service Chart Diagrams", CAiSE Workshops, 2003.

[14] Loke. S. W., Krishnaswamy, S., Naing, T., "Service Domains for Ambient Services: Concept and Experimentation", mobile Networks and Applications 10(4), pp. 395-404, 2005.

[15] Celentanio, A., Okroglic, A., Pittarello, F., "An ontology based Approach to Interaction Ambient Design", Mobile Data Management, 8th International Conference on Mobile Data Management (MDM 2007), Mannheim, Germany, May 7-11, 2007.