Towards Service Continuity for Generic Mobile Services

Ivar Jørstad[1], Do van Thanh[2] and Schahram Dustdar[3]

[1] Norwegian University of Science and Technology, Dept. of Telematics, O.S. Bragstads plass 2E, N-7491 Trondheim, Norway
ivar@ongx.org
[2] Telenor R&D, Snarøyveien 30 N-1331 Fornebu, Norway
thanh-van.do@telenor.com
http://www.item.ntnu.no/~thanhvan
[3] Vienna University of Technology, Distributed Systems Group (DSG), Information Systems Institute A-1040 Wien, Argentinierstrasse 8/184-1, Austria
dustdar@infosys.tuwien.ac.at http://www.infosys.tuwien.ac.at/Staff/sd/

Abstract. This paper discusses models of generic mobile services. The goal is to gain understanding of the challenges in designing, developing and deploying advanced mobile data services. First, a composition model describing the components of a generic mobile service and the components relationships is given. Second, distribution models describing the distributions of the components in the former model across hosts, networks and domains are presented. After presenting these generic models, a brief mobility analysis is carried out, followed by a discussion of mobility and service continuity dependency on the service distribution. The functions necessary to provide service continuity are identified and incorporated in a service continuity layer.

1 Introduction

Till now, the focus in mobile communications has been on providing service continuity of communication services when a mobile terminal is roaming between networks, i.e. avoiding abruption of access. The underlying mechanism for achieving service continuity is called handover or handoff. With the increasing number of devices that users have at their disposition, it is quite relevant to provide service continuity across heterogeneous devices. For example, a user, when arriving at the office, may want to transfer the conversation session from the mobile phone to the multimedia PC acting as an IP phone. The goal of this paper is to examine how service continuity can be provided. The paper adopts a formal and analytical approach. It starts with an analysis of current services and derives the functions and capabilities that are necessary to achieve service continuity.

2 Modeling Mobile Services

It is possible to model mobile services according to their *composition* or their *distribution*. Whereas the composition model is concerned with the division of a service into discrete components according to their nature and role, the distribution model is concerned with the distribution of components of a service across devices, networks and domains.

2.1 Composition Model

Fig. 1 displays a UML (Unified Modeling Language [1]) Class Diagram showing the composition of a mobile service. *Service logic* is the program code that constitutes the dynamic behavior and provides the functions of a service. Usually, this does not only consist of one autonomous unit, but in this model the service logic represents the collection of program code units for a given service.

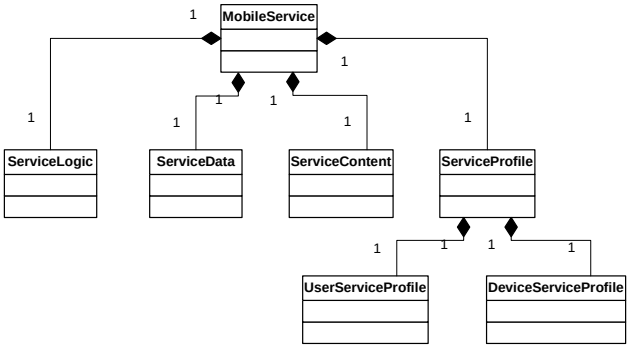

Fig. 1. Composition model of MobileService

Service state contains data used in the execution of the service logic and reflects the state of it. They are for example variable values, temporal parameters, register values, stack values, counting parameters, etc. In order to provide service continuity and personalisation we propose to introduce two additional service components; *Service content* and *Service profile*.

Service content refers to data that are the product of service usage. For example it can be a document written in a word processor or the entries in a calendar. Service content can be produced or consumed by the user.

A *Service profile* contains the service settings that are related to the user or/and the accessing device. A service profile can further be divided into a *User Service Profile* and a *Device Service Profile*.

All of the components of a mobile service as defined above can be subject to various distributions, as in other distributed systems [2].

2.2 Distribution Model

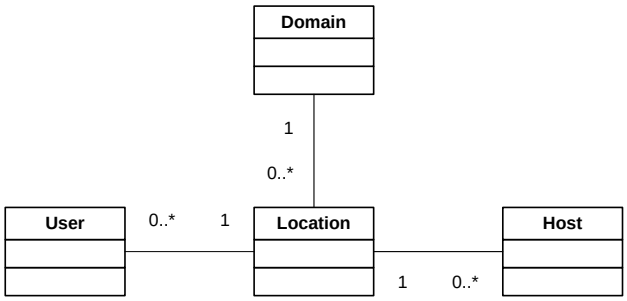

Fig. 2. Relationship between notions used in the distribution models

According to the Distributed Computing paradigm, the distribution of a service/application should be hidden and the mechanisms to support distribution are incorporated in the Distributed Computing middleware such that the developer can concentrate on the core functions of the application [3][4]. However, for mobile services, distribution plays a crucial role that must be considered at service design. Indeed, when the user is moving and is accessing services from different places, the location of a service and its components relative to the user's location will have great influence on its availability, quality, service continuity and personalization offerings.

In order to model the distribution, we introduce the notions that are depicted in Fig 2. A user can be at one location at a time but one location may be visited by zero or more users. A location may have zero or more hosts. One or more locations belong to a domain. A domain is defined as a set of locations containing a number of hosts that are controlled by an actor. The access to and from the domain is controlled by this actor, e.g. enterprise, home user, telecom operator, etc.

2.2.1 Monolithic Services

The first distribution model is a system where all components of the mobile service, i.e ServiceLogic, ServiceData, ServiceProfile and ServiceContent are installed in the same host which is located at the same location as the end-user. Such services can be called *monolithic*, as they constitute a single, autonomous unit. Examples of this service type are word processors, spreadsheets, stand-alone games, calculator, etc. With such a service, if the user is at the same location as the host containing all the service components, he will have access to the service.

2.2.2 Thin-client/Server Services

The previous model is very restricted. This is partly remedied by the second model (Fig. 3) which splits the service logic into two parts; one generic part (GenericServiceLogic) and one specialized part (ServiceLogic). Service content, data and profiles are all co-located with the specialized part. The generic part is a thin-client presentation layer. Typical examples of the GenericServiceLogic are Telnet or rlogin.

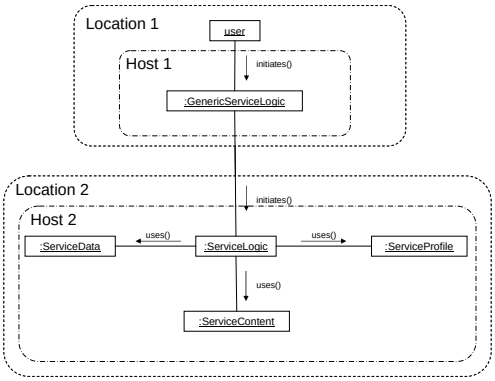

Fig. 3. Distribution model for thin-client/server mobile services

2.2.3 Client/Server Services

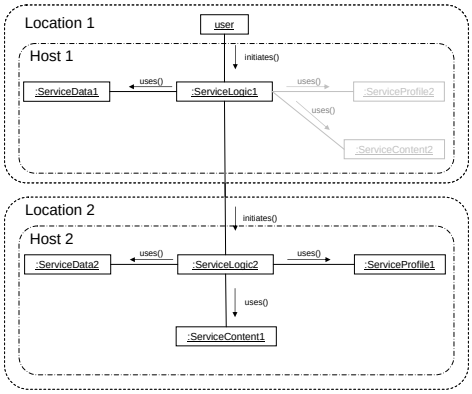

Fig. 4. Distribution model for client-server mobile services

In the third model (Fig. 4), a model similar to the previous is defined. The difference is that while in the previous model, the client application (GenericServiceLogic) was generic and used for a lot of different services. In the client/server model, Service-Logic1 is a client application specialized for a particular service.

Both components have their own service data, and ServiceLogic2 has access to a service profile and the service content as well. As an option, ServiceLogic1 can also keep its own service profile and service content. A refinement of this model is the multi-component service. In addition to employing a client/server distribution, the server is then further divided into two or more components.

It is worth emphasizing that a service type is characterized by both the service composition and the service distribution.

2.3 Mobility Analysis

All the presented models are static and represent common views of distributed systems that do not regard user mobility as an issue. It is thus important to add and discuss the notion of movement of the user between different Locations, and to consider what effects these movements have on services of various types. For clarity, we introduce the following axiom:

Axiom 1: *"Mobility does not have any impact on service availability and continuity as long as the user moves together with all discrete components of the service."*

From this axiom, it follows that: *"Only changes in relative Location between user and parts of, or all discrete components of a service have impact on service continuity in that particular service."*

Movements of the user will have different impact on each service type due to their distribution. Whereas concepts like personal mobility and device mobility is usually concerned with the communication service at network layer (OSI layer 3), service continuity is a concept that supports generic, data based services. The service continuity concept can be broken into two types; seamless service continuity and non-seamless service continuity.

We define seamless service continuity as: *"…the ability to pick up a service at a new Location, where service disruption is bounded by the time it takes the user to move between the two Locations."*

Non-seamless service continuity is defined as: *"…the ability to pick up a service at a new Location, where service usage can proceed from the 'point'/state where it were left at the previous Location, but where additional disruption is introduced due to a required reorganization of the service composition."*

Additional disruption in the second definition can be due to necessary user interaction (e.g. installation of required client software).

2.3.1 The Notion of Movement in UML

The basis for the notation used in the following analysis is UML collaboration diagrams. However, UML does not define a notion for movement, which is one of the most critical aspects in this analysis. The notion of movement is thus introduced using the stereotype <<moves>> along with a unidirectional association defining the direction of movement.

2.3.2 Service Type Specific Analysis

The remainder of this section describes a usage scenario for a specific service type.

Based on this analysis, the next section will provide an initial framework for improved service continuity support in generic mobile services. Due to space limitation, only the client/server service type defined in Fig. 4 is considered. Mobile agents and mobile code [5] are technologies that earlier have been suggested as solutions to some of the challenges with service continuity.

Consider a scenario where UserX is accessing a service S1 in Location1 through Host1, using ServiceLogic1A. At one point, UserX moves from Location1 to Location2. The question is then how to ensure service continuity. There are two alternatives:

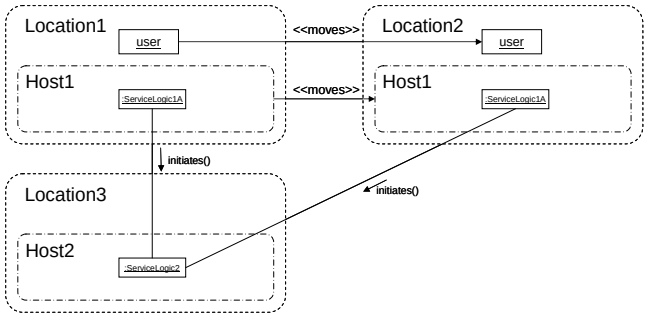

Fig. 5. User moves together with Host to a new Location

1. If Host1 moves together with the user, as depicted in Fig.5, there is no relative movement between the user and the service components that he is directly accessing. Service continuity is obtained by ensuring the continuity of communication between ServiceLogic1A and ServiceLogic2. This is the familiar case of the terminal handover that mobile communications have focused on. In these figures, other components of the service than service logic are excluded to ensure clarity and avoid cluttering. It is implicit that all the other components of the service are coexisting with the service logic as earlier described.
2. If Host1 is not moving together with the user, it will not always be possible to realize seamless service continuity, but often only non-seamless service continuity. For seamless service continuity, a copy or an instance of the ServiceLogic1A must already exist at the new Host, such that the user can continue the service. For non-seamless service continuity, this service logic must be installed at the new Host. In both cases, it must be possible to reinitiate the communication towards ServiceLogic2. Typical example of such a case is a web browser, e.g. Internet Explorer, of which two instances are installed on two PCs.

3 Support for Service Continuity: Service Continuity Layer

The service continuity layer can be seen as a management layer with support functions for realizing maximum service continuity and availability of generic services due to user movement relative to service components. Service continuity should be

supported by appropriate middleware. Others have argued that the application layer is suitable for providing flexible solutions to handling service continuity and mobility issues [6].

A service continuity layer needs to have access to relevant information from the application layer (e.g. current state of service), but also from the lower layers (e.g. to infer decisions about network resources). The elements of the service continuity layer are 1) a monitor, 2) a handover manager, 3) an interoperability evaluator, 4) a service composition module and 5) an input/output redirector.

First, there is a need for a *monitor* which keeps a resource map and includes mechanisms and methodologies for describing surroundings of a host more detailed than only nodes in the current network (e.g. WLAN zone), which can be deduced from ARP requests on an Ethernet (i.e., neighbour discovery). There is a need for describing network boundaries, domain boundaries, restrictive elements (middleboxes [7]) etc. These must be known by the service continuity layer so that possible re-distributions of a service, to provide service continuity, can be identified.

Second, a *handover manager* is needed. High-level service handover has earlier been considered in [8]. Handover between cells in mobile telecom systems (GSM) is based on measurements of the surroundings (e.g. of received signal strength level) for the system to be able to act proactively as handover becomes necessary to provide a sustained service. Although a soft handover is not necessarily required for service continuity in data based services, the idea of monitoring surroundings could be applied here also, and performed by the service continuity layer, thereby increasing the user experience of mobile services. The monitor together with the handover manager can instruct the service composition module to design and implement a new service composition of an existing service as soon as it is recognized that a handover will be necessary, thus decreasing the disruption time in service usage.

The third required element is an *interoperability evaluator*. If a service is to be re-organized based on decisions by the monitor, a new service will be composed using either a replication of or an equivalent of each of the current components of the service. To ensure sustained service access, it must be ascertained that potential components for the new composition are interoperable with, and provide a satisfactorily equivalent interface as, the current components. The task of the interoperability evaluator is thus to match compatible components based on both their *interfaces* (syntax) and their *behavior* (semantics).

A fourth element is the *service composition* element. Based on what the Service Continuity Layer knows about the systems and the restrictions, it can dynamically compose a new service by using only components that have been validated by the interoperability evaluator. In general, and as specified for XML Web Services, service composition can be either choreographed or orchestrated [9]. Orchestration seems to be the most feasible for the service continuity layer considered in this paper.

The fifth and last element is an input/output redirector. Input/Output (I/O) redirection can be used between service logic components in a mobile service to avoid moving entire components around, when requirements otherwise would suggest this as a solution. A mechanism for performing the actual redirection must exist, and in addition, a generic way of representing I/O for transport between services must be defined to simplify the interfaces between service components.

4 Conclusion

This paper initially describes the composition of generic mobile services and then provides an overview of possible distribution models for such services. The paper proceeds with a user-movement based analysis of one of the models. For the model chosen, it is suggested that service continuity can be assured if either a) a new instance of the initial service logic is available in the new location or b) a re-implementation of the initial service logic is available in the new location, and c) the communication between ServiceLogic1A and/or B and ServiceLogic2 can be reinitiated.

The analysis culminates with required functionalities included in a Service Continuity Layer which consists of a) resource map of surroundings, b) high level service handover functionality, c) interoperability evaluation and compatibility matching, d) service composition and e) generic i/o redirection between service components (service logic).

As part of future work, the user-movement based analysis will be extended and UML modeling of concepts discussed will be carried out.

References

1. Martin Fowler, UML Distilled: A brief guide to the standard object modeling language, 3rd Edition, ISBN 0-321-19368-7, 2004
2. George Coulouris et. al, Distributed Systems: Concepts and Design, 3rd Edition, ISBN 0-201-619-180, Addison-Wesley, Pearson Education, 2001
3. ITU-T X.901 | ISO/IEC 10746-{1,2,3,4}, Open Distributed Processing Reference Model Part 1,2,3 AND 4
4. Kerry Raymond, Reference Model of Open Distributed Processing (RM-ODP): Introduction - kerry@dstc.edu.au - CRC for Distributed Systems Technology -Centre for Information Technology Research - University of Queensland Brisbane 4072 Australia
5. Stefano Camapdello & Kimmo Raatikainen, Agents in Personal Mobility, Proceedings of the First International Workshop on Mobile Agents for Telecommunication Application (MATA'99). Ottawa Canada October 6-8 1999. World Scientific, pp. 359-374
6. Proceedings of the Italian Workshop "From Objects to Agents: Intelligent Systems and Pervasive Computing" (WOA'03), Italy, ISBN 88-371-1413-3, September 10-11, 2003
7. IETF, RFC 3303: Middlebox communication architecture and framework, August 2002
8. Thomas Strang, Claudia Linnhoff-Popien, Matthias Roeckl, Highlevel Service Handover through a Contextual Framework, MoMuC 2003, 8th International Workshop on Mobile Multimedia Communications, Munich/Germany, October 2003
9. Chris Peltz, Web Services Orchestration: A review of emerging technologies, tools, and standards, Hewlett Packard, Co., January 2003