ESPOO 2004 VTT PUBLICATIONS 519

Daniel Pakkala

Lightweight distributed service platform for adaptive mobile services

VTT PUBLICATIONS 519

Lightweight distributed service platform for adaptive mobile services

Daniel Pakkala

VTT Electronics

ISBN 951–38–6269–0 (soft back ed.)
ISSN 1235–0621 (soft back ed.)

ISBN 951–38–6270–4 (URL: http://www.vtt.fi/inf/pdf/)
ISSN 1455–0849 (URL: http://www.vtt.fi/inf/pdf/)

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 5, PL 2000, 02044 VTT
puh. vaihde (09) 4561, faksi (09) 456 4374

VTT, Bergsmansvägen 5, PB 2000, 02044 VTT
tel. växel (09) 4561, fax (09) 456 4374

VTT Technical Research Centre of Finland, Vuorimiehentie 5, P.O.Box 2000, FIN–02044 VTT, Finland
phone internat. + 358 9 4561, fax + 358 9 456 4374

VTT Elektroniikka, Kaitoväylä 1, PL 1100, 90571 OULU
puh. vaihde (08) 551 2111, faksi (08) 551 2320

VTT Elektronik, Kaitoväylä 1, PB 1100, 90571 ULEÅBORG
tel. växel (08) 551 2111, fax (08) 551 2320

VTT Electronics, Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland
phone internat. + 358 8 551 2111, fax + 358 8 551 2320

Technical editing Marja Kettunen

Otamedia Oy, Espoo 2004

Pakkala, Daniel. Lightweight distributed service platform for adaptive mobile services [Kevytrakenteinen hajautettu palvelualusta mukautuville liikkuville palveluille]. Espoo 2004. VTT Publications 519. 145 p. + app. 13 p.

Keywords adaptation, middleware services, Generic Service Elements (GSE), context-awareness, service personalization, pervasive computing architectures

Abstract

Distributed computing environments are becoming more heterogeneous due to the integration of different wireless and fixed networks and variety of terminal devices that can be used to access content and services on the Internet. Also, increasing user expectations of personalizable, adaptive, and context-aware mobile services bring complexity to the development of future mobile services. The distributed and heterogeneous computing environment, together with increasing user expectations, set requirements for future mobile services that are difficult to meet without service platform support alleviating service development.

In this work, a lightweight distributed service platform that has been designed with a practical design approach to support the adaptation of mobile services, and partly also personalization and context-aware functionalities of the services, is presented. A requirement analysis for future mobile services is carried out by an extensive literature review. The mobile service adaptation support functionality of the service platform is designed based on the identified requirements. Further architecture based on the concept of Generic Service Elements is designed for the service platform. The validation of the architecture is achieved by a prototype implementation. The validation was successfully achieved by the prototype implementation. It also proved the service platform's applicability to resource-constrained distributed mobile computing environments as the prototype resulted in an overall size of less than 360 KB, of which approximately one third is allocated to the mobile terminal domain.

Pakkala, Daniel. Lightweight distributed service platform for adaptive mobile services [Kevytrakenteinen hajautettu palvelualusta mukautuville liikkuville palveluille]. Espoo 2004. VTT Publications 519. 145 p. + app. 13 p.

Avainsanat adaptation, middleware services, Generic Service Elements (GSE), context-awareness, service personalization, pervasive computing architectures

Tiivistelmä

Langattomien ja kiinteiden verkkojen yhdentymisen sekä moninaisten Internetin sisällön ja palveluiden käyttöön tarkoitettujen päätelaitteiden ansiosta hajautetut tietokoneympäristöt ovat epäyhtenäistymässä. Lisäksi käyttäjien lisääntyvät odotukset koskien räätälöitäviä, mukautuvia ja tilannetietoisia liikkuvia palveluita tuovat monimutkaisuutta tulevaisuuden liikkuvien palveluiden kehitykseen. Tulevaisuuden lisääntyvät käyttäjien odotukset sekä epäyhtenäiset hajautetut tietokoneympäristöt asettavat vaatimuksia liikkuville palveluille, jotka niiden on vaikea täyttää ilman palvelun kehitystä helpottavia palvelualustoja.

Tässä työssä esitellään kevytrakenteinen hajautettu palvelualusta, joka on suunniteltu käyttäen käytäntöön perustuvaa lähestymistapaa. Palvelualusta on suunniteltu tukemaan liikkuvien palveluiden mukautuvuutta, sekä osittain myös palveluiden räätälöitävyyttä sekä tilannetietoisuutta. Laajassa kirjallisuuskatsauksessa on tehty vaatimusmäärittely tulevaisuuden liikkuville palveluille. Palvelualustan liikkuvien palveluiden mukautuvuutta tukeva toiminnallisuus on suunniteltu kirjallisuuskatsauksessa tunnistettujen vaatimusten pohjalta. Palvelualustan arkkitehtuuri on suunniteltu pohjautuen ideaan yleisistä palveluelementeistä. Suunniteltu arkkitehtuuri on vahvistettu toimivaksi tekemällä palvelualustasta prototyyppi, joka toimi odotetusti. Lisäksi palvelualustan prototyyppi toteutus osoitti alustan soveltuvan resursseiltaan rajoitettuihin hajautettuihin liikkuviin tietokoneympäristöihin, koska sen lopullinen koko oli alle 360 Kt, josta noin kolmasosa sijoittuu liikkuvaan päätelaitteeseen.

Preface

The research work that has lead to this thesis was carried out at the VTT Technical Research Centre of Finland in the Embedded Software Design and Testing group of the Embedded Software research field, which was one of the operating units of VTT Electronics during the years 2001–2003. Contribution to this thesis emerges from a few different research projects, related contributing research was completed in MIDAS in 2001–2002 within the PLA program and OPERA in 2002–2003; both being internal strategic projects of the Embedded Software research field. Additionally, the prototype implementation was achieved in ITEA/Mobilizing Internet in 2003. Although the research that has lead to this thesis has been carried out in many different projects, the concept has developed along the way and valuable experiences have been gained from various research cases.

Many people have provided their support during the years that have lead to this thesis. I would especially like to thank Mr. Juhani Latvakoski for providing valuable reviews, discussions, guidance and support during the work. I would also like to express my gratitude to Research Professor Eila Niemelä for comments provided during the writing of this thesis. Additionally, thanks to all of my colleagues at VTT Electronics that I have had many interesting and constructive discussions with. I would also like to acknowledge the CCC Group Inc., ITEA/AMBIENCE project, and the Telecommunications Systems operating unit of VTT Electronics for providing drivers and hardware for the SoapBox needed in the prototype implementation. Finally, my supervisor at the university, Professor Tapio Seppänen, and the work's 2nd reviewer, Professor Jukka Riekki receive my appreciation for the comments and guidance that they have provided during the writing process.

Besides the work itself, there has been a lot going on in my life during my studies, both ups and downs. Therefore I would like to express my deepest gratitude to my family and closest friends for the support in bad times and sharing of joy in good times. Also thanks to all of the people that have enriched my life by providing things other than work and studies to be occupied with.

Finally, I thank you Anni, for understanding, inspiration, and the encouragement that you have provided me during these years.

Oulu, December 23, 2003

Daniel Pakkala

Contents

Abbreviations

API	Application Programming Interface, interface to existing application
CC/PP	Composite Capability/Preference Profile, profiling technology
CDC	Connected Device Configuration, configuration of Java 2 micro edition
CLDC	Connected Limited Device Configuration, configuration of Java 2 micro edition
CORBA	Common Object Request Broker Architecture, middleware technology
CPU	Central Processing Unit, processing unit of a computing device
CTI	Computer Telephone Integration, term describing the convergence of computing and telecommunication systems
GSE	Generic Service Element, architectural concept for middleware services
GSM	Global System for Mobile Communications, telecommunications technology
GUI	Graphical User Interface, graphical interface for the user to interact with a computing system
HTTP	Hypertext Transfer Protocol, application layer protocol
IDL	Interface Description Language, language for describing software interfaces
JAR	Java Archive, compressed file type of Java programming language

JMF	Java Media Framework, Java-based media processing technology
JVM	Java Virtual Machine, platform for executing Java applications
ID	Identification, unique identifier of an entity
IP	Internet Protocol, network protocol
MIDP	Mobile Information Device Profile, profile of Java 2 micro edition connected limited device configuration.
NA	Networked Appliance, appliance connected to a network
OMG	Object Management Group, standardization organization for object - based technologies
ORB	Object Request Broker, part of the common object request broker architecture
OS	Operating System, software controlling hardware resources in a computing device
OSI	Open Systems Interconnection, standard for representing network protocols
OSGi	Open Services Gateway Initiative, organization for developing open service gateway technology
PAN	Personal Area Network, small-scale network surrounding a person
PDA	Personal Digital Assistant, small size portable computer
PDU	Protocol Data Unit, basic transferable data unit of a protocol
RMI	Remote Method Invocation, Java middleware technology

RPC	Remote Procedure Call, middleware technology for procedural programming languages
SP	Service Platform, software providing commonly needed services
SDO	Super Distributed Object, object technology
SMF	Service Management Framework, OSGi implementation
SOAP	Simple Object Access Protocol, application layer protocol
TCP	Transport Control Protocol, session-based protocol
UI	User Interface, interface for the user to interact with a computing system
UML	Unified Modeling Language, object-based modeling technology
W3C	World Wide Web Consortium, standardization organization for web technologies
WAN	Wide Area Network, spatially large network
WG	Working Group, small organizational unit consolidated with common interests
WLAN	Wireless Local Area Network, wireless network technology
WWRF	Wireless World Research Forum, forum seeking a consensus on the visions of the future wireless world
WWW	World Wide Web, application layer protocol widely used for the Internet
XML	Extensible Markup Language, information representation technology

1. Introduction

1.1 Motivation and Background

The motivation for this work derives from the next generation of mobile telecommunication systems and beyond, and the ongoing process of computer telephone integration (CTI). Mobile devices are not used just for traditional telephoning any more. In addition, access to services and applications can be provided to users via mobile devices, just like they are provided to a user's Personal Computer (PC). Also, the evolution of home automation systems towards Internet Protocol (IP) capable networked appliances (NA) enables the remote computer-based controlling of NAs in a home environment. The trends are moving towards systems based on utilizing different networks and the Internet in service provisioning.

This ongoing process of IP convergence enables lots of new application scenarios, but at the same time it introduces problems that were not present in the legacy systems. Considering the computing environment, wireless access networks and the diversity of end-user terminals, combined with the increasing usability and added value service requirements, raises many essential research issues to be solved before the introduction of fluent and user friendly mobile services. These research issues include the personalization of mobile services along user preferences, fluent adaptation of services to changing conditions in the distributed and heterogeneous computing environment, and context-awareness of services [1].

The latest trends in the telecommunication world are moving towards Internet protocol-based services and application frameworks and platforms [2]. Service and application platforms are also the answer to the requirement for faster service development and deployment. All of the problems introduced by the heterogeneous computing environments and IP convergence cannot be solved with new IP-based protocols, as middleware is also needed. The role of the middleware and service frameworks and platforms is growing all the time as services and applications are becoming more sophisticated and requirements for adaptive, personalizable and context-aware services are introduced [2].

While the amount of mobile services is growing all the time, users will face a new problem of service management. Without service management, services will be scattered over different terminals and systems resulting in an unfriendly situation from the user's point of view, where the user has multiple service access points and services are bound to different terminals. The growing amount of content, services, applications and NAs that need to be controlled and monitored by the user suggest that service gateway-based systems are one possible solution towards user-centric and smoothly manageable systems. These kinds of systems have only one service access point for the user, and therefore services are easily manageable and accessible. This approach has been taken in this work and addressed in [3].

Service gateway-based systems with a presence of home automation services have been identified to be a good research and validation framework for service platforms [4]. This is because the setup introduces new challenges when compared to traditional mobile services by providing an interaction-based approach to mobile systems construction [5]. Previous research has been carried out regarding the remote controlling of NAs in a service gateway-based system [6]. This research shows that service gateways can be used as intermediaries when controlling IP incapable devices from IP-based networks.

One of the research issues to be solved regarding future computing systems is the adaptation of mobile services to user context and preferences, current terminal capabilities and dynamic network characteristics. Adaptive mobile services provide added value for the end user by adapting to the user's current situation, environmental conditions, preferences and motivations to use the services. Fundamentally, the research problem is how to provide support for mobile services to meet all of the requirements, including adaptability, that are set for them, and thereby alleviate the mobile service development.

1.2 Scope and Objectives

The focus of this work is the adaptation support issues of mobile services. Because service gateways have been identified as one possible solution for future service management problems for the end user [3], this work recognizes this and tries to clarify the adaptation issues in the context of service

architectures based on the functional division between service gateways and variety of mobile terminals. The work will present distributed service platform architecture supporting the adaptation of mobile services in the service gateway-based service architecture model where the two main domains are the service gateway domain and the mobile terminal domain.

The fundamental research problem that this work is focused on is: how to support future mobile services to meet the manifold requirements set for them? These requirements are set by raised user expectations of personalized and context-aware services and by the distributed heterogeneous computing environments with characteristics such as variable quality wireless communication channels. This work studies how the adaptation of mobile services can be supported by a service platform alleviating service development to meet the requirements set for them. The objectives set for the work to be presented are manifold.

Firstly, a goal is to identify the basic requirements and perform a requirement analysis for future mobile services and service provisioning in distributed and heterogeneous computing environments. In particular, the goal is to identify the requirements that future mobile computing environments and users pose regarding the adaptation of mobile services. The only way to identify the central requirements and issues concerning adaptation support for future mobile services is a wide literature review of the state of the art research and technologies regarding adaptation support, future service provisioning trends, technologies, and closest related software architectures.

Secondly, a goal is to develop a distributed service platform providing adaptation support and alleviating the design and implementation of adaptive mobile services that is based on the requirements gathered from the literature review. A practical design approach for the design of the service platform is taken to result in small yet efficient implementations that can be introduced to resource-constrained mobile computing environments.

Thirdly, a goal is to identify and present an adaptation process that is to be implemented by the service platform to be developed and widely utilizable by different kinds of adaptive mobile services.

Finally, aside from the actual service platform functionality, an important goal is to come up with a small size extendable architecture that is based on the latest architectural concepts, applicable in resource-constrained mobile computing environments and would contribute to the state of the art regarding architectures supporting adaptation functionality. The latest architectural concepts are also to be derived from the wide literature review.

1.3 Structure of the Work

The domain of the work, related technologies and research including a couple of architecture solutions from the field of pervasive computing, are presented in the literature review in chapter 2. The adaptation of mobile services is discussed and the requirements for a distributed service platform supporting adaptation are raised and summarized in chapter 3.

Once the requirements for a distributed service platform supporting the adaptation of mobile distributed services are gained, an architecture and the components for such a platform are designed and presented in chapter 4. The presented architecture is validated, tested and evaluated in a prototype implementation, which is described in chapter 5.

In chapter 6, the outcome of this work is discussed and reviewed with existing architecture solutions of pervasive computing, and the middleware structure is compared to ideas raised in the Working Group (WG) 2 at the Wireless World Research Forum (WWRF) [6]. Targets for development are also identified and presented. Finally, conclusions from the work are presented in chapter 7.

This work has been partially published previously. Firstly, regarding the service platform that will be presented in this work, it is to be published in [1]. Secondly, the service management issues that have a strong role in the research framework selection of this work, and are utilized in the service platform design, have been previously published in [3]. Previous research that is not closely related, but outlined the service platform scenarios regarding NA controlling, has been carried out in [6].

2. Related Research and Technologies

2.1 Introduction of the Domain

The research carried out in this work includes elements from many different fields of computing. There are many terms used in the context of a distributed computing system dealing with NAs, PDAs, wireless networks, sensors, mobile services, middleware, context-awareness, and network servers similar to this work. A short introduction to these terms is provided.

The concept and term of ubiquitous computing [8, 9] was introduced and defined by M. Weiser in the early 90's. Weiser stated that the most efficient technologies are essentially invisible to the user. His goal was to make computing as commonplace and ubiquitous as electricity. At first, the research focused on small special purpose devices, network protocols and new styles of applications. Since then the field has evolved and includes a wide variety of research topics, a good overview of ubiquitous computing is provided in [10]. R. Bagrodia et al., in the mid 90's introduced the term nomadic computing [11]. Nomadic computing is defined to exploit advanced wireless communications technologies, the Internet, positioning systems and distributed computing to provide anyplace and anytime access for the user. There is also a third closely related term - pervasive computing, it has many common interests with ubiquitous and nomadic computing. Pervasive computing is a term widely used and has a variety of research interests [12]. A lot of research in pervasive and ubiquitous computing has focused on active or smart spaces [13, 14], and the services needed in this kind of environment. The term mobile computing refers to a distributed computing environment where the nodes of execution are not static, but connected via wireless networks to enable mobility of the nodes. The challenges of mobile computing are due to resource poor elements, variation of connection performance and reliability, security, and reliance on finite energy sources [15].

All of the terms presented have a slightly different approach, but are very much overlapping and the research interests in these fields are nowadays somewhat the same dealing with latest technologies and heterogeneous distributed computing environments.

2.1.1 Research Framework

The research framework for this work is derived from recent concepts presented for the interaction of the user with mobile services. One of these is the I-centric communication model [16, 17], where the user is placed at the central position of the communication system and interacts with the services within his individual communication space. In the I-centric communication model, the user may have several individual communication spaces depending on the services present in his surroundings, and these services are adapted to these communication spaces. In another concept, called the personal service environment [18], the user is provided with a computing environment where user preferences towards the services remain the same even when roaming in different networks, and services are personalized and adapted to user preferences. The personal service environment is defined to be an environment that assists a user in finding, adapting, and using mobile services that fulfill the user's needs given his personal profile, mobility, and context [18]. The personal service environment concept includes functionality for discovering services, using services, managing user profiles, and sensing context.

Figure 1 presents a scenario that summarizes the research framework used in this work at a conceptual level. In the scenario, the user has access to any service within his individual communication space using any terminal, over any network and at any time. The research framework tries to realize this scenario in service gateway-based service architecture.

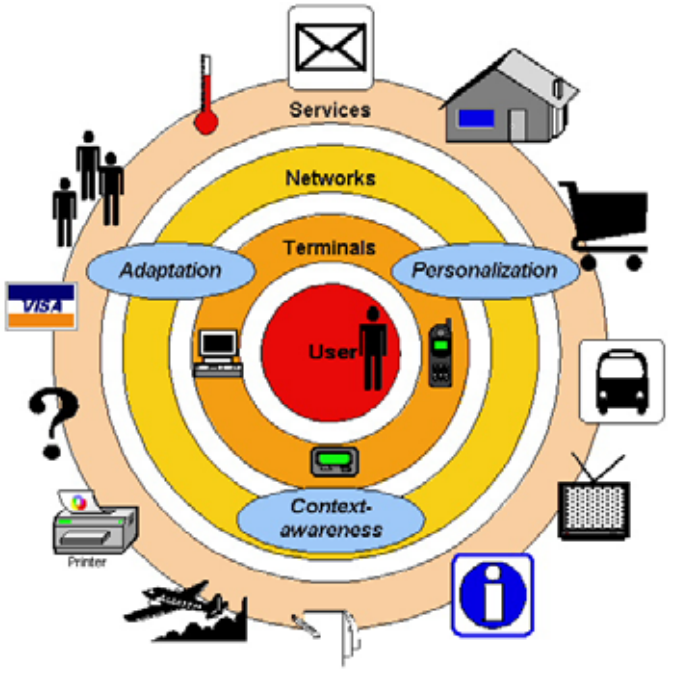


Figure 1. The scenario of the research framework.

As illustrated in Figure 1, and derived from the related research, there are three research issues and requirements regarding future mobile services in a heterogeneous distributed computing environment. These are the requirements of adaptation, personalization, and context-aware functionality of mobile services [2]. Additionally, aside from the individual services, service management is an important research issue in the kind of scenario presented. This is because the amount of services available to the user is growing all the time and the service configuration may be dynamically changing depending, for example, on the user's location.

This work studies the adaptation of mobile services in a distributed and heterogeneous computing environment and especially in service gateway-based service architecture. The goal is to develop a distributed service platform to support the adaptation of mobile services in service gateway-based service architecture. The requirements for the service platform are partly set by the scenario of Figure 1 and partly by the service architecture that is defined as service gateway-based. Literature related to this work can be found from the

fields of ubiquitous, pervasive, nomadic, and mobile computing. A conceptual framework for research on the adaptation of mobile services carried out in this work is presented in Figure 2. The research framework presented in Figure 2 is conceptual and does not define any specific technologies to be used on any of the framework components. The spectrum of technologies for implementing service gateway-based systems is very wide and the same technologies can be utilized in many different components of the research framework. There are technologies that can be utilized in any of the components of the research framework. These technologies are usually related to a way of representing information in a structured format, middleware technologies for hiding computing environment heterogeneity, and application protocols independent of transport layer protocols and different network technologies.

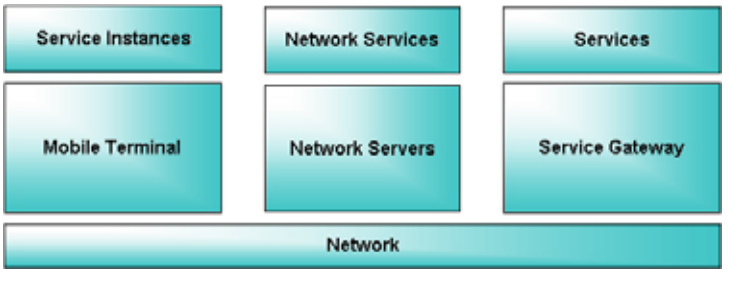


Figure 2. Research framework.

As seen from the research framework, the services of interest are distributed over different computational nodes of the system connected via a network. When mobile services are distributed, there are at least two different parts to a service; the service part, which can contain heavy calculation processes and implements the service logic and is executed at the service gateway. Service gateways are usually network servers having a good deal of processing capability and resources, therefore the program size is not an issue in this domain. The other part, referred as a service instance in this work, is the part that is executed in the mobile terminal domain. Mobile terminals are usually quite resource-constrained wirelessly connected devices having limited processing capabilities; therefore the program size and workload allocation can become a bottleneck in the mobile terminal domain. The distribution of mobile services to services and service instances as described enables the possibility for a service that is too big to be executed by the mobile terminal to be divided into two parts

that work together to implement the service. The familiar example of this kind of division is the division between the service logic and the service Graphical User Interface (GUI) also known as the thin-client approach. However, to emphasize the fact that the mobile resident part of the service can and will be much more than just a GUI in future mobile services, the term ***service instance*** is introduced and used in this work. A service instance can contain functionality related to service adaptation, off-line functionality, and much more in addition to the GUI. The configuration of the service instance may also change dynamically as a result of service adaptation. In addition, there may be a number of network servers hosting network services that can include services like network file systems or media gateways, for example.

2.1.2 Service Management

Earlier research and experiences from service management issues in service gateway-based systems [3] encouraged using the same research framework for the adaptation of mobile services. In previous research [3], it was noticed that service gateway-based systems are one possible solution to service management problems of future computing environments related to interaction with mobile services. Service gateway-based systems can also be designed in a user-centric fashion, where the user is at a central position in relation to all of his services. Figure 3, from [3] presents the problem scenario and possible solution from a user-centric approach. In the figure, the services are the remote controlling of different kinds of NAs from a mobile terminal.

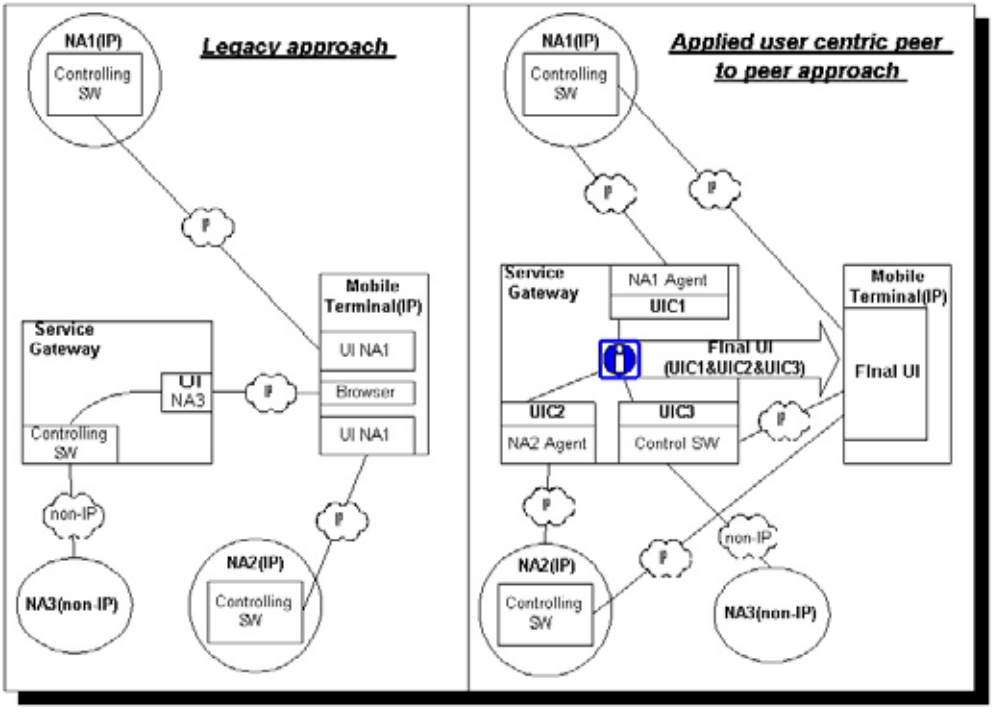


Figure 3. Approaches for service provisioning.

In Figure 3, on the left, the legacy approach for service provisioning is illustrated. In this approach, each service has its own independent UI. The services accessible directly via IP networks each have their own UI in the mobile terminal. The services not accessible directly via IP networks can be accessed using the service gateway as a gateway to IP incapable NAs. These services are usually accessed, for example, via a browser. The legacy approach is functional, but becomes cumbersome from the user's point of view when the number of services is increased. The problem is that in the legacy approach, service management is left to the responsibility of the user and the user will have an equal number of UI applications to IP-based services. Thereby the user will have multiple service access points in the legacy approach. A proposed solution to the service management problem is the user-centric approach presented on the right of Figure 3. In the user-centric approach, the service gateway is used as a central arbitrator of all services accessible to the user. As can be seen from the picture, all services are represented in the service gateway by their agents, regardless of where the service is actually located. In this way, the responsibility of service

management is transferred to the service gateway and can be automatic. In the user-centric approach, the user has only one service access point and the number of services does not affect the workload or number of UI applications needed in the mobile terminal. The user can fluently access all of his services via one UI provided by the service gateway management software. Using this UI, the user selects the service he wants to use, and the final UI that is sent to the mobile terminal is created from the UI components that service agents provide to the service gateway. When the UI is received by the mobile terminal it automatically establishes the needed control connections to the location where the service in question is located and when done it appears to the user.

As we are able to see from Figure 3 and its description, the service gateway-based systems have clear advantages when it comes to service management. This is why in this work, service gateway-based systems are chosen as the research framework for research on the adaptation of mobile services.

2.1.3 Adaptation and Adaptability

Adaptation and adaptability is recognized to be an important research topic in many different fields of computing [2, 12, 13, 14, 15, 19]. The requirement for adaptation is present on many different layers of a computing system, and should therefore be acknowledged in the overall architecture design of computing systems. When using the words *adaptation* or *adaptability,* attention should always be paid to questions like: Why something is adapted and what are the attributes creating the need for adaptation? Because of the many different system levels where the need for adaptation is present, many definitions are also available for adaptation. For example, in his article [12] Satyanarayanan defines adaptation in the context of pervasive computing:

"Adaptation is necessary when there is a significant mismatch between the supply and demand of a resource. The resource in question may be wireless network bandwidth, energy, computing cycles, memory and so on."

In the WWRF WG 2 [2], adaptation is defined a little bit differently and closer to the definition found in this work:

"By adaptation we mean the ability of services and applications to change their behavior when the circumstances in the execution environment change"

Further, in the context of mobile computing the need for adaptation is identified in [15]:

"...as the circumstances of a mobile client change, it must react and dynamically reassign the responsibilities of client and server. In other words, mobile clients must be adaptive."

The definitions presented for adaptation and adaptability in this work are generic and are simply close to what can be found in any English dictionary. The definitions are provided in the following:

Adaptation: Modifying an entity to best fit the current situation regarding the attributes creating the need for adaptation.

Adaptability: An entity's capability to accommodate the attributes creating a need for adaptation.

There are many strategies regarding the adaptation of applications, two extreme adaptation strategies and many collaborated ones can be identified [15]. Figure 4 from [15] illustrates the taxonomy of the adaptation strategies.

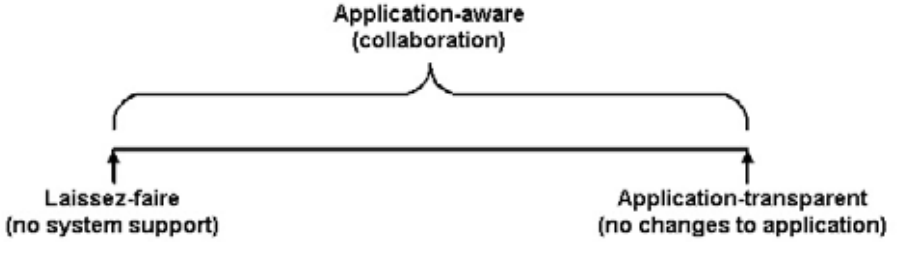


Figure 4. Range of adaptation strategies.

In the Laissez-faire approach, the responsibility of adaptation is left to individual applications and no system support is provided for the adaptation. However, this approach lacks the central arbitrator to resolve the incompatible resource demands of different applications and to enforce limits on resource usage. Even

though the system support for adaptation can be avoided in this approach, the applications become more difficult to implement and the size of the applications increases because each application needs to implement its own adaptation functionality individually.

The other extreme to adaptation strategies is the application-transparent approach, where no changes to applications are needed in order to enable adaptation, but the adaptation is left fully at the responsibility of the system. Even providing backward compatibility with existing applications, this approach has drawbacks. There may be situations where the adaptation automatically performed by the system is inadequate or even harmful.

Between these two extremes of adaptation strategies lies a number of brew solutions that are collectively referred to as application-aware adaptation. This approach emphasizes the collaborative partnership of applications and the system in the adaptation functionality. This approach permits applications to determine the best adaptation behavior for the situation, but preserves the ability of the system to monitor resources and enforce allocation decisions. Application-aware adaptation also decreases the application size compared to Laissez-faire adaptation, because part of the adaptation functionality is provided by the system and each application does not have to have an embedded adaptation functionality.

The requirements for the adaptation of mobile services are due to user requirements of added value services [2] and the characteristics of the mobile computing environment [15]. Figure 5 illustrates the issues creating a need for the adaptability of future mobile services.

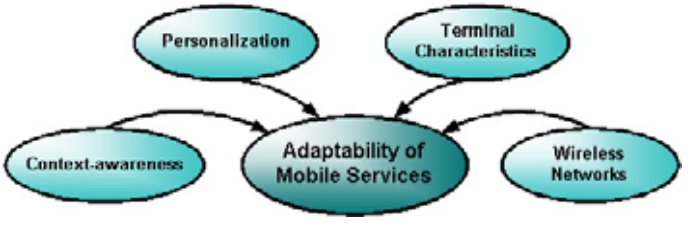


Figure 5. Issues showing requirements for the adaptability of mobile services.

The required features of future mobile services, especially context-awareness and personalization, put requirements on adaptation functionality. In context-aware computing, entities modify their behavior based on context information, in other words, the entities adapt to context information. Also, in personalization the entities are tailored or modified along user preferences, characteristics, and skills, in other words, entities are adapted to this information. On the other hand, the varying terminal characteristics like screen size, processing capability, input method etc. create a need for the adaptability of mobile services as well. Additionally, the changing quality of wireless network connections pose requirements for adaptability. In summary, adaptation functionality will be an integral requirement for future mobile services.

2.1.4 Personalization

There is no inclusive definition for personalization in the academic world. The term is widely used in many different contexts. In [20], personalization is defined as a technique aiming to increase the acceptance of information or an Internet portal by letting the user manage his own individual range of interests. The WWRF WG2 also provides a definition for personalization in their white paper [20]:

"Personalization means the appliances of actor's needs or interests to the problem of offered sets of information and control." [20 p. 4]

This definition does not provide a very clear and inclusive picture of personalization. Thereby to provide more understanding on personalization, it can simply be thought of as a technique used to personalize or tailor entities like products, services, or the information content best accepted by the consumer of the entities.

Personalization functionality in a context-aware computing system can be based on utilizing the context of the user, which may contain needs, preferences, history, behavior of the user, and location related aspects. In [20], personalization is divided into two different types, implicit and explicit personalization. Implicit personalization is defined as taking place without the knowledge of the actor (user) within the system, and the actor has no influence

on the personalization consciously. An example of implicit personalization is the adjustment of the navigation paths on web sites by learning the user's behavior. Explicit personalization extends its range outside the system boundaries and is controlled or steered by the user. In explicit personalization, the user can affect the appearance of objects. For example, a user can select so-called "skins" and "themes" for his applications. In explicit personalization, the user consciously affects personalization.

The personalization of services is adapting a service according to a user's individual needs, characteristics, and desires. Profiling is a widely used technique used when implementing personalization functionality to different systems [20, 21].

2.1.5 Context-awareness

Context-awareness, in short, is a feature of a computing system utilizing the relevant information about current situations in the execution environment, to provide added value in the form of behaving acceptably and according to the situation. The added value can come from many different operations of a context-aware system, for example, distraction elimination in certain situations, automatic controlling of NAs based on current situation, or targeted information presentation according to the situation.

There are many coexisting definitions available for context-awareness. In the field of ubiquitous computing, context-awareness is amongst other definitions defined in [10], where the following text can be found:

*"Ubicomp applications need to be context-aware, **adapting** their behavior based on information sensed from the physical and computational environment." [10 p. 30]*

Context-awareness is also defined in the field of pervasive computing [12], where we can see that context-awareness is also a key requirement for pervasive computing systems:

"A pervasive computing system that strives to be minimally intrusive has to be context-aware. In other words, it must be cognizant of its user's state and surroundings, and ***must modify its behavior*** *based on this information." [12 p. 15]*

There are many other definitions for context-awareness not presented here that have slight differences in their meanings. From the two definitions presented here it can be seen that adaptation functionality is an integral part of a context-aware system.

There are many coexisting definitions for the context available, all of these definitions are not presented here but extensively discussed in [10]. The definition of context made in [22] is general enough to be used widely, when some of the other definitions are more or less application-specific. In [22], context is defined as follows:

"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." [22 p. 3]

This is the definition adopted and used in this work for context. When conceptualized further, context can also be thought of as a snapshot of the state of the environmental conditions surrounding an entity in a certain situation. Respectively, context information is simply information describing the environmental conditions surrounding an object. In context-aware computing, entities utilize context information to modify their behavior to be best suited to the current context. The main contribution of context-awareness to a computing system is to provide added value services to the user by utilizing the relevant information about the user's surroundings and the user himself.

Until recently, context information has been unreachable for computing systems, but with developments in sensor technology, smaller sensors can be embedded, for example, in clothing [23], terminals [24], physical objects and spaces, and some context information can be retrieved using the sensors available. The unprocessed measurement data from the sensors themselves is not very usable within the computing system, but with semantics given to the measurement data

it becomes useful for a computing system as context information. For example, if the location of a user is known, and it is known that the location refers to a meeting room, it can be assumed that the user is currently in a meeting and should not be disturbed with issues irrelevant to him while he is in a meeting.

There are many different ways of presenting and utilizing the context information. In [25], the context is divided into five different categories: physical context (i.e. location, time), environmental context (i.e. weather, lighting, sound), personal context (health, mood, schedule), social context (group activity, presence), and application context (i.e. visited web sites, sent/received e-mails). In addition, many different classifications of context information can be found that are not in coherence with each other. For example, [26] divides context into four different categories: computing context (i.e. connectivity, communication costs, bandwidth, resources), user context (i.e. a user's profile, location, presence of other people, social situation), physical context (lighting, noise levels, traffic conditions, temperature), and time context (time of the day, week, month, season of the year).

As we can see, the different classifications of context information are not in coherence and different terms are used quite freely. However, the classification of context information at a conceptual level is not a very fruitful approach to context-aware system design. The needed context information in a context-aware computing system is always dependent on the application. For example, a health monitoring application does not need to know about the sent and received e-mails of a user. Therefore, the context information presentation mechanisms are often designed to be extendable by utilizing basic ontology and semantics for context information presentation.

When divided roughly, there are two abstraction levels of context information, low- and high-level context information. Low-level context information can be collected using the sensors available in the system, and by giving pre-defined value limits to the measuring results. As an example of low-level context information, the body temperature of a user might be 38 degrees Celsius and a priori definition states that with this temperature value the user has a fever. High-level context information, like the users social situation, can be derived from the lower level context information (i.e. surrounding sounds). The accumulation of low-level context information using sensors is a rather simple

task when compared to deriving high-level context information from low-level context information. Recent research for deriving high-level context information relies on the heavy usage of Artificial Intelligence (AI) methods. For example, in [27], Naive-Bayes networks and classifier are used for deriving the high-level context information from low-level context information available to the system.

Context-aware systems are found in many different applications. Context-awareness has been utilized, for example, in tour guides [28, 29], active spaces [13, 14], home environments [30], and communication [25].

Context-awareness has been identified as one of the requirements for future mobile services and computing systems [2]. By designing future computing systems to be context-aware, the systems provide added value to their users by observing the users current situation and adapting their behavior to it.

When considering context-aware systems in general, there are some basic issues and components found in almost every system. These include context sensing, context information presentation and processing, adaptation, and on top of all, the context-aware applications utilizing the context information.

The behavior of a context-aware system can be reactive or proactive, or at its best both of these. Reactive systems are only able to sense the current context and behave according to it, but when proactivity is added to the context-aware system it can make assumptions of the near future and behave more sophistically than purely reactive systems. Even though proactivity is a useful characteristic of a context-aware system, it is very hard to implement. However, at least one example of proactive systems already implemented can be found, as the Aura architecture is capable of proactive behavior [13].

Context-aware systems are still under heavy research, most of the research done in the field of context-aware computing is focused on certain application areas and therefore the solutions found are not interoperable as they should be in commercial products. This is one challenge for future research on context-aware computing, because if service vendors have a common tool or method to model context in a unified way, they would be able to provide the users with context-sensitive, personalized services and products, independently of the runtime technology [31].

2.1.6 Functionality Distribution

A basic issue to be solved when designing any kind of distributed system is the distribution of resources, functionality and workload between the separate computational nodes of the system [32]. In the kind of environment presented in Figure 2, the workload allocation becomes an important factor because the mobile terminals are resource-constrained devices unable to execute big workloads. The available resources in mobile computing environment may vary a lot depending on the terminal used and the network connection. This indicates that the workload allocation of a mobile system should be dynamic and adaptive, instead of static.

The term *client thickness* refers to the workload allocated for a client of the client server communications model. The client thickness of a mobile node in a pervasive computing environment is an issue when there is no one size fits all solution. This issue is covered in [12], where a good definition for a minimum acceptable thickness for the client is provided:

"For a given application, the minimum acceptable thickness of a client is determined by the worst-case environmental conditions under which the application must run satisfactorily." [12 p. 15]

This approach keeps the client small enough to be executed in constrained environments like in mobile nodes with small CPU and memory capacity, but still able to cope with the challenges set by the environmental conditions of the mobile computing environment.

The requirements set for the future mobile services like adaptability, personalization and context-awareness will affect the client thickness of a mobile node, as more and more functionality in the mobile terminal domain is needed to support these requirements, the workload allocated to the mobile terminal keeps increasing. So the thin-client approach will be inadequate for future mobile services to tackle problems like sudden disconnection, variable network bandwidth, and to answer to the need for adaptation, personalization and context-awareness. Instead, a distributed service platform that provides middleware support in both the mobile terminal and network server domains is needed.

2.1.7 Middleware

No commonly accepted definition for middleware is available in the academic world that would cover all of the areas where the term is used. Middleware was first introduced in the context of distributed enterprise systems. In enterprise systems, middleware is all about the integration of different systems [33]. Recently, with the development in software and hardware technology, middleware has also been introduced to the field of mobile [19, 34] and pervasive computing [13, 14], and embedded systems [33]. The common factors for mobile and pervasive computing are that often they are both distributed and heterogeneous computing environments including a variety of different hardware, operating systems, and networks. In embedded systems, middleware is needed to provide transparency from heterogeneous hardware platforms.

The clear role of middleware is to provide transparency from the distributed heterogeneous computing environment underneath it for the application and service domain. Figure 6 illustrates the role and structure of middleware in a computing system as a transparency provider for the applications and services from different operating systems, networks and network protocols, different types of terminal devices, various types of hardware platforms, and software from various vendors.

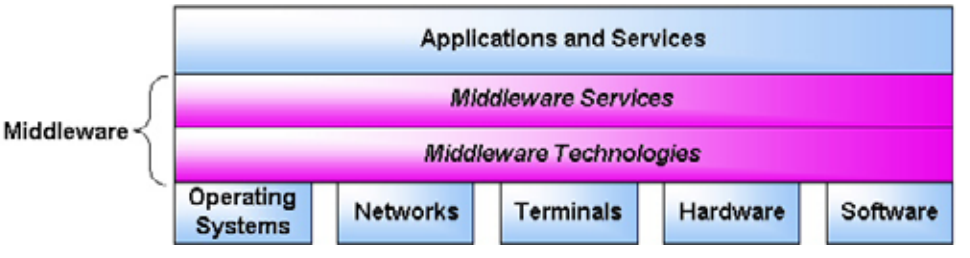


Figure 6. Role of middleware in a computing system.

Middleware is software that resides over the operating system and hardware and below the applications and services. It can be divided into middleware technologies and middleware services that utilize the middleware technologies to provide services for applications and services. These middleware technologies include technologies like CORBA, Java RMI, and RPC or can be based on customized solutions for providing transparency from heterogeneous execution

environment in a distributed computing environment. One role assigned to middleware, in addition to integration, is to provide commonly needed services that can be utilized by the applications. These upper middleware layer entities are called middleware services. Middleware services are one approach to alleviate the development of future mobile services to meet the requirements of adaptive, personalizable, and context-aware functionality. As the functionality that is widely and commonly needed by the applications and services is provided from a middleware service, the workload and size of the applications and services decreases. This is because the commonly provided functionality is not implemented by the services and applications individually. One type of middleware services are Generic Service Elements (GSE), that fall within the main focus of this work and which are introduced and discussed in [2, 34], and in the following chapter.

2.1.8 Generic Service Elements

The Wireless World Research Forum (WWRF) [7] defines Generic Service Elements (GSE) to be upper middleware layer services that provide functionality needed by the services and applications [2]. Working Group 2 of WWRF has identified eight generic service elements supporting adaptive applications [2]: environment monitoring, event notification, distributed application framework, perception service, modeling services, mobile distributed information base, ontology service and semantic matching engine. These GSEs are presented as the corner stones for the success of systems beyond 3rd generation telecommunication systems. The eight GSEs identified in [2] are shortly described in the following:

Environment Monitoring: An environment monitoring component allows software to observe its changing conditions in its surroundings using rules that have been defined a priori. The observed changes in the environment are expressed to other software components that have an interest in them. For notifying these changes, an event notification service may be utilized. The observed changes may be used for creating various models of the environment, the middleware system, and the user.

Event Notification: Event notification service is responsible for transporting events that have been previously observed by the environment monitoring service. It has to be able to solve the correct set of recipients for a given event, and deliver the events to the recipients using a proper representation format and medium. Decoupling of event producers and consumers is required from the event notification service.

Distributed Application Framework: One of the main requirements for the distributed application framework is to allow the creation of new services from existing ones. Further requirements for the distributed application framework are to support dynamic auto-configuration of services and service discovery.

Perception Service: The perception service is responsible for collecting and storing values of perceptions within a certain time interval. Perceptions in this context are observable entities or anything that can be used for learning purposes.

Modeling Services: The primary mission of modeling services is to collect data and build models of the phenomenon they are trying to learn. The models are not used by the modeling services themselves, but are provided as support services for intelligent applications learning services. The modeler does not have to understand the semantics of the data it is learning, just learn dumbly following its learning algorithm. The client of the modeler, however, is aware of the semantics of the data used in the learning process.

Mobile Distributed Information Base: The mobile distributed information base should be especially suited for storing XML documents, because it is a format widely used for representing information. The essential characteristics of the information base include high availability, consistency, support for weakly connected and disconnected operations, and intelligent data synchronization.

Ontology Service: The ontology service is required to support representation, manipulation and storage of ontologies of varying detail level. The ontologies should be extendable to incorporate new concepts and sub-concepts. The service needs to support reasoning over ontological information and share it with third parties. It also needs to be able to cope with problems of fragmented knowledge

caused by heterogeneous environments by supporting partial mappings and other techniques between ontologies.

Semantic Matching Engine: Semantic matching uses ontological information provided by the ontology service to match and reason over instances of ontological knowledge, such as profiles. It has to be able to determine whether a given document of semantic information conforms to ontology and supports manipulation of semantic information, for example, combining two or more overlapping semantic information sets.

From the GSE definitions presented, it can be seen that definitions emphasize the role of AI methods, ontologies, semantics, models, and learning as supporting technologies to support adaptive applications. However, these technologies are still under heavy research and are not mature enough to be used widely in the commercial solutions of today. In addition, the mentioned adaptation support technologies typically require a lot of processing capability, which can become a bottleneck in very resource-constrained mobile computing environments. The research related to GSEs is at its starting point and therefore the set of the GSEs identified in [2] may not be final but changing. This can be seen for example in [34], where service discovery and auto-configuration are identified to be GSEs, but [2] states that these two are not GSEs, but included in the distributed application framework. However, it should be noted that GSEs are defined only at the conceptual level and therefore based on the definitions of [2], it is difficult to say how they will be realized.

2.1.9 Profiles and Profiling

Profiles in general describe information about an object. This information can be anything seen as relevant in the application the profile will be used for. For example, user profiles can describe characteristics of a user, such as name, age, and occupation. In profiles, this information can be presented in a specified format understandable to both humans and computing systems. However, it is not required that all profiles should be understandable to the user of the system. This is the case, for example, regarding a terminal profile describing terminal capabilities within the computing system. Profiling in general is presenting and processing the profiles within the computing system with a set of tools for

processing the profiles. To avoid misunderstandings, the definitions for a profile and profiling as used in this work are provided in the following:

Profile: Structured set of information describing an object.

Profiling: Presenting and processing of information using profiles for information presentation and set of tools for processing the profiles.

Because some of the profiles have to be understandable to both humans and computing systems, self-descriptive technologies for structured information presentation like XML, for example, are often used in profiling. By using standardized XML-based technologies in profiling an interoperable, profiles between the systems of different vendors are achieved. However, if the profiling is used for internal data presenting and processing of a system and is not visible outside the system, customized proprietary solutions for profiling can be used as well. User profiles are used in the context of personalization of telecommunications and Internet services [35]. Profiling can also be used in context-aware computing to present and deliver context information within a context-aware system.

2.2 Technologies

Some relevant and promising technologies that can be utilized within the research framework of this work are presented in this chapter. Firstly, middleware technologies like CORBA, Java RMI, and RPC are needed to hide the distributed and heterogeneous nature of the underlying computing platform. When the computing environment diversity is hidden by the middleware technologies, middleware services can be built for utilizing the middleware technologies. The middleware services can provide support for the mobile services in adaptation, personalization and context-aware functionality. The technologies that seem promising in this context include technologies for structured information representation like XML, for information exchange like SOAP, and for profiling like CC/PP. The same technologies that are useful for middleware services can be utilized in the service domain as well. When moving upwards, application or service framework technologies like OSGi will be needed to execute the services and the middleware services. From a

programming language point of view, the Java 2 Platform, spanning the whole spectrum of computing from embedded to enterprise systems, forms a promising platform for pervasive and mobile computing. The field of technologies is evolving all the time and new technologies like Super Distributed Objects (SDO) that extend the object model to represent real world entities as interworking autonomous and cooperating objects in a distributed computing environment are being standardized. The technologies mentioned and to be presented are just examples and many other technologies useful within the research framework selected can be found.

2.2.1 Common Object Request Broker Architecture

Common Object Request Broker Architecture (CORBA) [36] is an open distributed object computing infrastructure being standardized by the Object Management Group (OMG) [37]. CORBA specifies a system, which provides interoperability between objects in a heterogeneous distributed environment and provides transparency from the network, different operating systems, and different programming languages in a distributed computing environment. CORBA does not specify an implementation of a component but the interfaces of it, this way interoperability between components can be achieved. Component interfaces are described using the OMG Interface Description Language (IDL), which has a technology independent syntax for describing object encapsulations. IDL specified interfaces can be written and invoked from any programming language that provides bindings to CORBA; bindings to CORBA are available in most popular programming languages like Java, C, and C++. CORBA is a widely used middleware technology in distributed enterprise environments, because it provides clear advantages by automating many common object oriented distributed programming tasks and provides a common framework for interoperability of the different parts of a system. Figure 7 provides an overview of CORBA ORB architecture with an example function call flow.

The object is a CORBA entity consisting of identity, interface, and implementation. Objects are also referred to as Servants. A Servant is an implementation of operations that support the CORBA IDL interface. The client is the program entity that invokes the function call from an object implementation. The fact that the client might be invoking a remote function call

should be transparent to the client and the function call should be as simple as the calling method of an object.

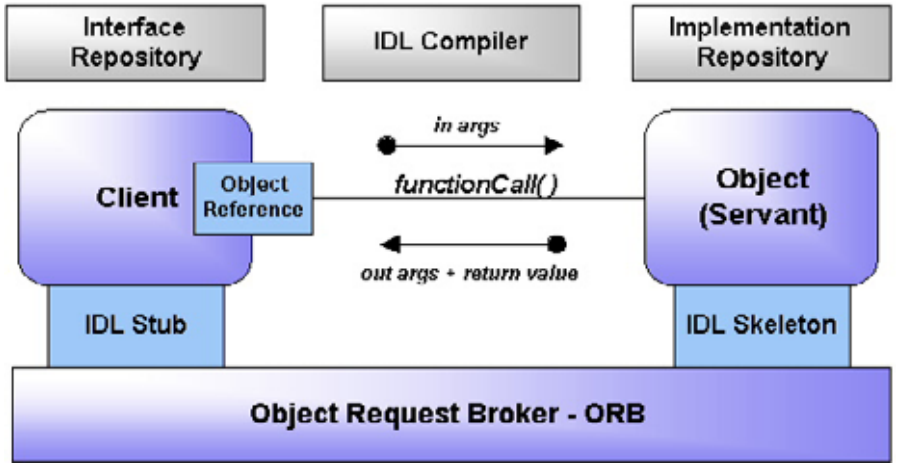


Figure 7. Function call in CORBA ORB architecture.

The ORB provides a mechanism for transparently communicating the client requests to the target object implementation. This simplifies distributed programming because the function calls made to remote objects appear to be local. ORB is also responsible for finding the object implementation, transparently activating it if necessary, delivering the request from the client to the object, and returning any response to the client. The IDL Stub of the client and the IDL Skeleton of an object serve as connectors between the client and server applications and the ORB. The role of the IDL compiler is to perform automatic transformation between the IDL definitions and the target programming language.

In summary, when the logical function call takes place between the client and the object, the actual function call request is passed through the IDL interfaces and ORB in the CORBA.

2.2.2 Remote Procedure Calls

Remote Procedure Calls (RPC) is a technique used in distributed systems and procedural programming for enabling the calling of a remote process to appear as a local process call to the calling party [32]. Usually the remote process is executed on a different machine than the calling process and the call is performed over the network. RPC enables such an arrangement in procedural programming languages, so that the client of the client-server communication model can call processes on the server, as if they were local processes. Figure 8, applied from [32], illustrates phases and needed functionalities of a RPC implementation.

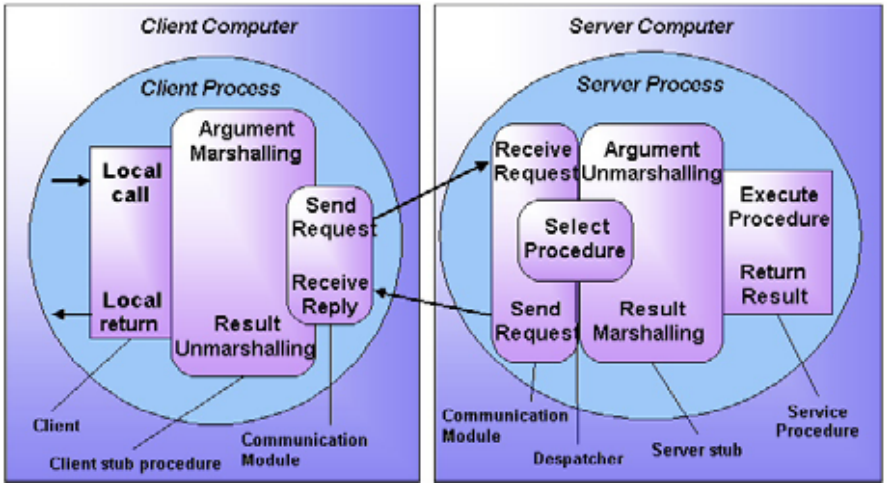


Figure 8. RPC implementation structure.

2.2.3 Java Remote Method Invocation

The Java Remote Method Invocation (RMI) [38] is an object-oriented version of the RPC. The RMI enables objects to call the methods of a remote object executed on another computer. The RMI provides transparency from the object distribution by providing a mechanism that enables the method calling of remote objects, residing on different virtual machines and hosts, in the same way that local methods are called. Java-based programs can call methods of remote objects after obtaining a reference to the remote object. The reference can be

obtained from the naming service provided by Java RMI or receiving the reference as a return value. Java RMI uses object serialization to marshal and unmarshal parameters.

The basic Java RMI is not designed for wireless environments and is not suited for mobile computing as such [19]. This is because of Java RMI's heavy and wasteful usage of TCP connections. For example, getting a single object reference using GSM data connection takes 8 to 20 seconds, depending on the Java virtual machine used. However, because Java RMI is becoming more and more popular in distributed computing as a middleware technology, some attempts to modify Java RMI to wireless environments has been made [19, 39].

2.2.4 Extensible Markup Language

Extensible Markup Language (XML) [40] is a simple and flexible text format that was originally designed for large-scale electronic publishing. Nowadays, XML is widely used for exchanging data on the web and elsewhere, and its role is emphasizing all the time. The clear advantage that XML provides is that it is a platform, software, and hardware independent way of describing and exchanging information. The purpose of XML is to describe data; it provides a structured format for storing and sending data and is designed to be self-descriptive. In Figure 9 an example message stored in XML format.

```
<message>
<to>John</to>
<from>Michael</from>
<heading>Invite</heading>
<body>Welcome to play football this friday!</body>
</message>
```

Figure 9. Message stored in XML format.

The example in Figure 9 contains a message from Michael to John, where Michael invites John to play football on Friday. As one can see from the example, XML is a self-descriptive way of presenting information. The tags,

marked <tag>, are not specified by XML, but specified by the author of the XML description. The example demonstrates the main advantage of XML, flexibility as a format for representing structural information.

2.2.5 Simple Object Access Protocol

Simple Object Access Protocol (SOAP) [41] is a lightweight XML-based protocol for exchanging information in a decentralized, distributed computing environment. SOAP provides a message framework, by defining the structure of the message and processing mechanism for the messages. SOAP does not provide transport capabilities between the communicating peers, but can be used in conjunction with some Internet transport technology like HTTP for example.

SOAP consists of three parts: an envelope, a set of encoding rules and conventions for RPC usage. The envelope defines a framework for describing what is in the message and how to process it, and the encoding rules are for expressing instances of application-specific data types. The conventions for RPC usage include conventions for remote procedure calls and responses.

SOAP messages consist of three main elements: SOAP envelope, SOAP header, and SOAP body. The envelope and the body are mandatory elements, while the header is optional. The header and the body are encapsulated inside the envelope.

2.2.6 Composite Capability/Preference Profiles

Composite Capability/Preference Profile (CC/PP) [42] is a profile that describes device capabilities and user preferences, and can be used in the adaptation of content that is to be presented to the device provided by the CC/PP Profile. The CC/PP is based on Resource Description Framework (RDF) [43], which has been designed as a general-purpose metadata description language by the W3C [44]. The RDF provides a framework with the basic tools for both vocabulary extensibility and interoperability. RDF is based on XML and is designed to describe metadata or machine understandable properties of the web.

A CC/PP profile contains a number of CC/PP attribute names and association values for the attribute names. A server uses these values to determine the most appropriate form of a resource to be delivered to the client, who provides the CC/PP profile and is requesting the resource. The CC/PP attribute names and permissible values with their associated meanings constitute a CC/PP vocabulary.

The CC/PP profiles might include some sensitive and private data of the user. The CC/PP itself does not provide adequate security or privacy mechanism, but it can still be used in conjunction with a system providing these features.

2.2.7 Open Services Gateway Initiative

The Open Service Gateway Initiative (OSGi) [45] is an alliance forum of over forty companies formed for the development of open specifications for the delivery of multiple services over wide-area networks to local networks and devices. The OSGi Alliance provides OSGi Framework and a service platform specification for an open service gateway. The specification consists of standardized OSGi APIs that define interfaces between the inter service interaction, services and OSGi Framework, device access, service management system and the OSGi Framework. Client interface to services is not specified in the OSGi service specification. Figure 10 presents the structure and standardized APIs of the OSGi service platform specification.

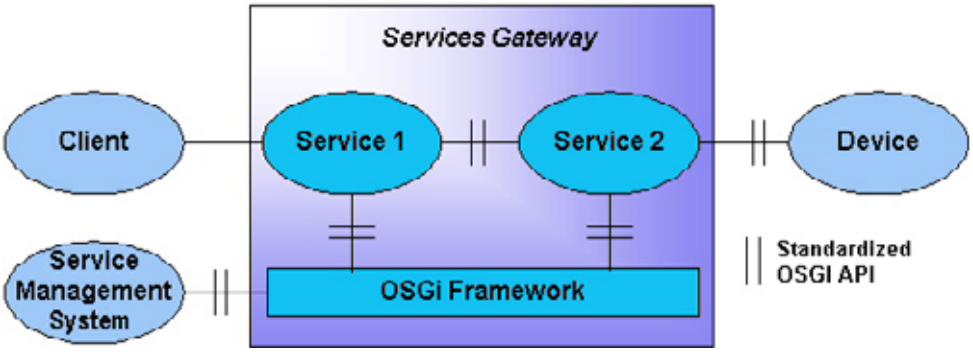


Figure 10. OSGi specification structure and standardized APIs.

The OSGi Service Platform specification [46] is currently in release 3. In addition to the framework specification, it specifies services like: message logging, service tracking, XML parsing, saving preferences, registering resources and servlets to HTTP and administrating services. The OSGi specification also specifies service cradle-to-grave life cycle management, inter-service dependencies, data management, device management, client access, resource management and security.

The OSGi compliant services, also called bundles, are JAR packets that include the control interface, service code and possible service interfaces of the service. If a bundle provides some service for other services it registers its service interface, which is separated from the implementation of the service within the OSGi Framework. If a bundle uses a service of another service it fetches the service interface within the OSGi Framework where it has been registered. The OSGi specification also includes a device access specification, which defines how different devices are connected to the OSGi service platform. The OSGi bundles can be downloaded to a service gateway (set-top box, cable or DSL modem, PC, Web phone, automotive, multimedia gateway or dedicated residential gateway) over the network on demand, while the gateway manages the installation, versioning and configuration of these services.

2.2.8 Java 2 Platform

The Java 2 platform [47] is based on the efficient utilizing of networks and the idea that the same software should run on many different kinds of computers, consumer gadgets, and other devices. The Java 2 platform consists of editions, optional packages, profiles, and configurations and, at the lowest level, APIs. There are three different editions in Java 2 Platform: Enterprise, Standard and Micro edition. Figure 11 from [47], gives a good overview of the Java 2 Platform organization.

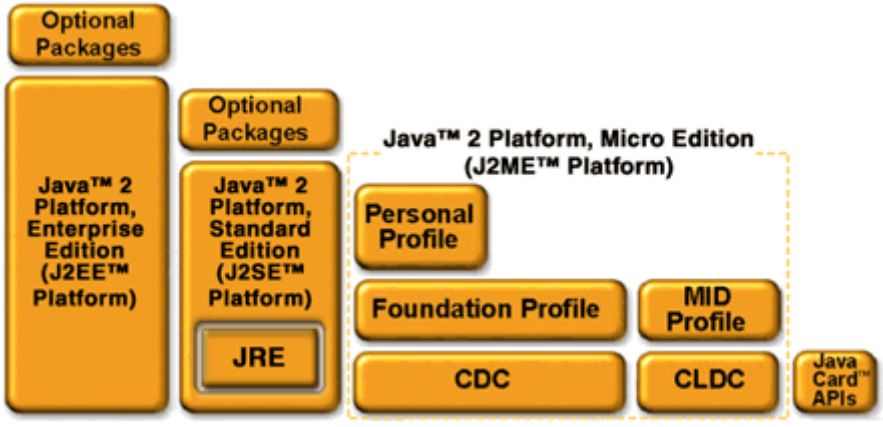


Figure 11. Java 2 Platform organization.

The Java 2 Enterprise Edition (J2EE) is targeted at developing and running multi-tier enterprise applications for enterprise systems. The usual environment for J2EE is an enterprise server with a high processing capability. The virtual machine used in J2EE is the same as in J2SE. J2EE simplifies enterprise applications by defining a standardized, modular and re-usable component structure called Enterprise JavaBeans (EJB), and providing a set of services for these components. The services provided by the J2EE include database access, CORBA technology, a security model, Java servlets API, JavaServer Pages and XML technology. J2EE technology can be used for integrating enterprise applications from different vendors with middleware.

The Java 2 Standard Edition (J2SE) is targeted at developing and running client applications to enterprise and web systems and for developing normal applications for PCs. The usual environment for J2SE is a normal desktop PC. J2SE includes libraries for networking, graphical UIs, security, Remote Method Invocation (RMI) and much more in optional packages.

The Java 2 Micro Edition (J2ME) is targeted at developing and running applications in constrained environments. The constrained environment can be, for example, a mobile phone, smart card, PDA or a set-top-box. J2ME consists of two different configurations; Connected Device Configuration (CDC) that is targeted at embedded devices like PDAs and set-top-boxes or devices that share similar characteristics of network connectivity and memory footprints. Another

configuration, Connected Limited Device Configuration (CLDC), is targeted at smaller and more constrained embedded devices like mobile phones, NAs and pagers. Perhaps the most known setup of J2ME is the CLDC with Mobile Information Device Profile (MIDP), this setup can be found in most new mobile phones sold today. The MIDP is currently in its version 2 and downloadable Java applications targeted at J2ME MIDP 2.0 are already available commercially.

The Java 2 platform, as we can see, spans the while spectrum of computing from enterprise to embedded systems. Therefore, it forms an important enabler technology for pervasive [48] and mobile [49] computing.

2.2.9 Super Distributed Objects

The Super Distributed Objects (SDO) is a standardization effort of the super distributed objects special interest group in OMG [37]. The idea of SDOs is to provide a standard computing infrastructure to enable the modeling of real world entities like devices, software components, and services as objects and deploy them in a highly distributed environment [50, 51]. The SDOs are also defined to seamlessly interwork in a platform independent manner with each other, and to ubiquitously aid users in accomplishing their tasks. The super distribution refers to the massive number of objects beyond centralized control therefore autonomously or cooperatively performing their tasks. The autonomous behavior of the SDOs enable them to join and leave a network, discover other SDOs, provide their functions, form an application service without the intervention of users or other SDOs. The SDOs are also able to establish and maintain dynamic relationships to represent their acquaintances, discover other SDOs, communicate with other SDOs, or form groups of interworking SDOs. There is no fixed infrastructure for SDOs to communicate with each other within a system or application service, therefore each SDO has the role of access point or gateway and the communication path between SDOs may continuously change. Therefore, SDOs communicate in an ad-hoc and peer-to-peer manner. SDOs and application services formed out of single SDOs are defined to adapt and customize their structure and behavior according to the current environmental conditions and the user's individual situation to be able to provide self-adapting and context-aware services.

Figure 12, redrawn from [50], presents the functional components of the reference architecture presented for SDOs. The SDOs themselves are logical representations of a hardware or software entity of the environment providing well-known functionality in a standard way. In addition to SDOs, there are two other functional components in the reference architecture, user interface and service logic. These components utilize SDOs in their execution. The user interface is responsible for human-machine-interaction and adapts to a user's preferences and terminals, as well as to different application services provided by the SDOs. The service logic component is responsible for controlling the behavior of the SDOs by utilizing profiles. These profiles can contain user preferences and service parameters.

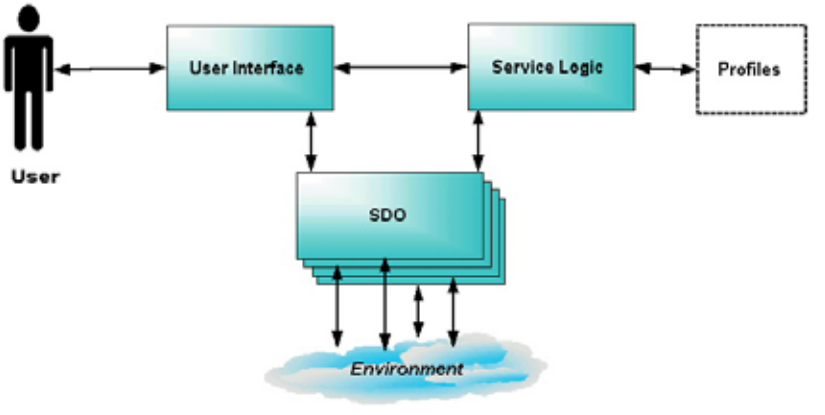


Figure 12. Functional components in the SDO reference architecture.

A single SDO is proposed to have two different types of interfaces, operational interfaces that enable access to SDO's services and management interfaces that enable the announcement and discovery of the SDO services, service control, monitoring and configuration of its resource data [51]. The SDOs are also proposed as a solution to model and capsulate services and other real world entities in the I-centric communication model at the middleware layer of a computing system [51].

2.3 Existing Architecture Solutions

There are many interesting projects in the field of pervasive and ubiquitous computing that have studied adaptation and adaptability [13, 14]. A couple of these projects are presented in this chapter to provide an insight to existing architecture solutions related to this work to some extent. Both of the architectures to be presented are focused on active spaces [14], which are also referred to as smart spaces [13], and integrated environments in the literature.

Although the presented architectures have a strong focus on pervasive and ubiquitous computing and active spaces, there are some interesting issues common to the architecture to be presented in this work. In particular, the services provided by the systems presented and the solutions related to adaptation and context representation and utilization are of special interest. It should be noted that both of the systems to be presented rely on an extensive amount of computing resources to be available in order to be utilized.

2.3.1 Project Aura

Project Aura [13] is a research project focusing on distraction free pervasive computing, especially in the context of active spaces. The project aims to reduce user distraction and thereby increase the effectiveness of the system by exploiting the plentiful computing resources. The fundamental principle of project Aura is captured in the following [13]:

"The most precious resource in a computing system is no longer its processor, memory, disk or network, but rather human attention. Aura aims to minimize distractions on user's attention, creating an environment that adapts to the user's context and needs." [13 p. 22]

Aura is specifically targeted at integrated pervasive computing environments involving wireless communication, wearable and handheld computers and smart spaces. The research carried out in Aura spans every system level from the hardware to applications and end users. To accomplish its goal, Aura applies two broad concepts. The first concept is proactivity, which is defined as a system layer's ability to anticipate requests from higher layers. The second concept is

self-tuning, which is defined as a system layer's ability to adapt by observing the demands made on them and adjusting their performance and resource usage characteristics accordingly. Both of these techniques are used for lowering the demand for human attention towards the computing system. Figure 13, from [13], presents the Aura architecture including its main components and their roles.

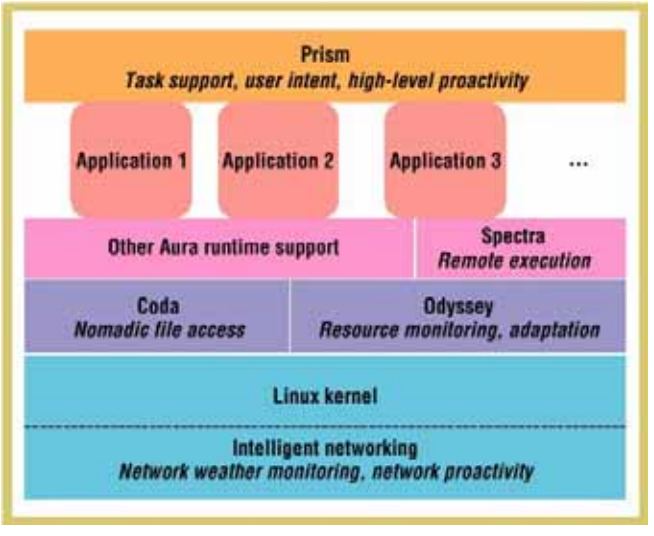


Figure 13. Aura architecture.

The Aura architecture is divided into different layers. The lowest layer is Intelligent Networking that supports network weather monitoring and proactivity. Intelligent Networking is based on two components periodically gathering information from wireless access points and performing monitoring and predictions on the network state based on the gathered information. Over the networking layer is the OS layer, which is based on a Linux kernel in Aura. Over the Linux kernel, Aura has modified versions of the Coda [52] file system for nomadic, disconnectable, bandwidth-adaptive file access and Odyssey [53] supporting application-aware adaptation and resource monitoring. On the next layer of the architecture is a component called Spectra, which is a remote execution mechanism that uses context to decide how to best execute the remote call. The other Aura runtime support is also found on this layer. Over the presented layers reside the Aura applications that utilize the service provided by the lower layers. Traditionally, the application layer is the topmost layer of a

computing system architecture, but the Aura architecture introduces a new layer residing over the application layer – a task layer, also called Prism.

Prism supports user task monitoring, user intent monitoring and high-level proactivity. The key ingredients of the prism architecture are: explicit representation of tasks as coalitions of abstract services, task configuration appropriate to the environment by context observation and environment management infrastructure that assists with resource monitoring and adaptation. The capabilities of Prism are encapsulated in components: Task Manager, Context Observer, and Environment Manager. In addition, a set of components called Service Suppliers provide services to support a user's task. The Prism's infrastructure supports interactions between the components and is built on existing middleware such as remote procedure call or CORBA.

2.3.2 Gaia Metaoperating System

The Gaia metaoperating system [13] has been designed to support the development and execution of portable applications for active spaces. Gaia has been built as a distributed middleware infrastructure that coordinates software entities and heterogeneous NAs contained in active spaces. Gaia is based on CORBA technology. Gaia provides services to query, access, and use existing resources and context of active spaces. It also provides a framework to develop user-centric, resource-aware, multidevice, context-sensitive, and mobile applications. By extending the concepts of traditional operating systems to ubiquitous computing spaces, Gaia simplifies space management and application development for active spaces and provides a single programmable entity instead of scattered resources. The services provided by Gaia are comparable with services provided by traditional operating systems including program execution, I/O operations, file-system manipulation, communications, error detection and resource allocation. Figure 14, from [14], presents the Gaia architecture and its main components.

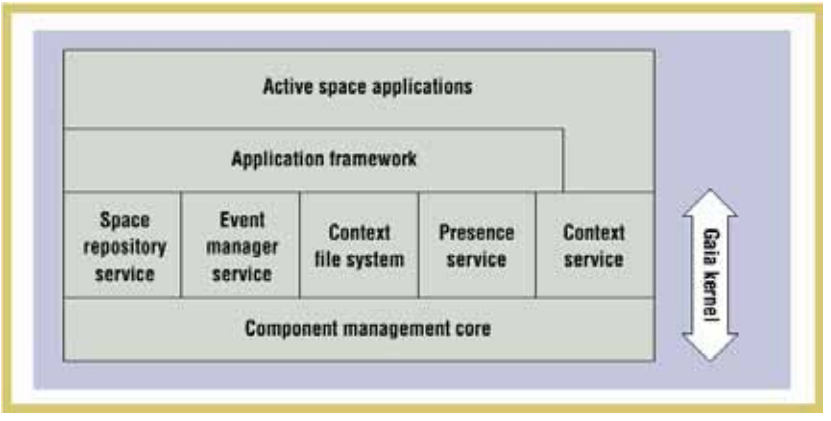


Figure 14. The Gaia architecture.

Gaia architecture has three main components: the kernel, application framework and the applications. The Gaia kernel is further divided into the component management core and a set of basic services used by all Gaia applications. These services are Event Manager, Context Service, Presence Service, Space Repository, and Context File System.

The Event Manager is based on a decoupled communication model and utilizes event channels to distribute events in the active space. Context Service lets applications query and register for context information, which helps applications to adapt to their environment. Presence service maintains information about resources in active space. Space Repository provides XML-based information storing active space entities. Context File System enables the use of context to distinguish meaningful information from the irrelevant.

The infrastructure of the Context Service of the Gaia kernel is based on components called context providers, which offer information about the current context. The context providers include sensors that track people's location, room conditions like temperature and sound, weather and current stock prices. Other components infer higher-level contexts on the basis of sensed information. The context model is based on first-order logic and Boolean algebra, which enables simple rule writing to describe context information.

2.4 State of the Art Summary

The latest trends regarding service provisioning are clearly moving towards individually provided services instead of mass-produced. This can be seen from the concepts of I-centric communication model [16, 17] and Personal Service Environment [18].

The basic requirements identified for future mobile services are adaptability, personalization, and context-awareness. These features are partially overlapping resulting in a situation where the features of personalization and context-awareness can be seen as features utilizing and therefore setting requirements for adaptation functionality.

As the amount of available services is growing all the time, service management becomes an important issue in future computing systems. Service gateway-based systems have been identified as one possible solution regarding service management [3] and are therefore selected as the research framework in this work.

As the mobile terminals that can be used to access the services via the service gateway are often quite resource-constrained, and the design approach in the work is practical, the functionality distribution becomes an issue to be taken into account in the service platform design. The system workload allocated to the mobile terminal should be as small as possible, without jeopardizing the operation of the system in mobile computing environments.

Profiling, as a technique, provides a useful way of representing processing structured information in a way that it can be understandable to both humans and computing systems. Therefore, it is possible to describe user preferences towards a computing system. Profiling is also a technique that can be implemented quite easily within the computing system with small resources and processing capability requirements.

Middleware services, more specifically the GSE concept, provide a good starting point for the architecture design of distributed service platform supporting the adaptation and alleviating of the development of adaptive mobile services. The GSEs currently defined only at the conceptual level rely on a strong utilization

of AI methods that usually require a lot of processing capacity. A practical design approach targeting lightweight service platforms taken in this work provides an alternative and possibly complementary approach to the GSE concept. Additionally, the existing architectures from the field of pervasive and ubiquitous computing, like Aura and Gaia, supporting adaptation and context-aware functionality, provide a good reference for the architecture design.

Middleware technologies such as CORBA, Java RMI, and RPC have an important role in providing transparency from the underlying heterogeneous and distributed computing environment and for being the enablers of middleware services. Other technologies, like OSGi and SDO, are needed as frameworks for executing the service platform and services, and modeling real life entities as accessible interworking objects within a computing system.

2.4.1 Term Definitions Summary

For the convenience of the reader, the definitions of the central terms made in chapter 2 are summarized and presented as used in this work in Table 1.

Table 1. Definitions of the central terms used in the work

Adaptability	An entity's capability to accommodate according to the attributes creating a need for adaptation.
Adaptation	Modifying an entity to the attributes creating a need for adaptation.
Context	Any information that can be used to characterize the situation of an entity. Can also be thought of as a snapshot of the state of the environmental conditions surrounding an entity.

Dynamic Adaptation	Modifying an entity to attributes that are dynamic and continuous in their nature within a short time frame. For example, modifying a service instance to the available network bandwidth.
Environmental conditions	Any information related to the user and user interaction within a computing system. (E.g. terminal capabilities, temperature, time, location, bandwidth, user preferences, etc.)
Generic Service Element	Generic Service Element is a distributed middleware entity performing a well-defined task and providing support for other middleware entities and services. The distributed structure is transparent to client entities of a generic service element.
I-centric communication	I-centric communication describes the communication relations surrounding the user himself and is driven by the user and his needs. In I-centric computing, the human with his natural characteristics is placed at the central position of the computing system.
Mobile Service	A service that is available and accessible independently of the user's location, access network and terminal.
Profile	Structured set of information describing an object.
Profiling	Presenting and processing of information using profiles for information presentation and a set of tools for processing the profiles.

Service	Any entity providing additional value or functionality to the user or the system. The term service in this work also includes applications.
Service Instance	A service as the user sees it. The same service can have multiple service instances each having different outlooks from the user's point of view. The service instance can be seen as a user interface to the service from where it was instantiated. Service Instances can include much more functionality than just a GUI.
Static Adaptation	Modifying an entity to attributes that are static and discrete in their nature within a short time frame. For example, modifying a service instance to terminal capabilities.
User Context	Any information that can be used to characterize the situation of the user. Can be imagined as a snapshot of the user in a current situation with current environmental conditions.

3. Design of Adaptation Support

The requirements for adaptive mobile services are set by the nature of future computing environments and added value service requirements of the user. The heterogeneous user terminals, unreliable and variable quality of wireless links together with raised user experience expectations present a difficult challenge for future mobile services. One method to alleviate the service development to meet the requirements set for the services is to provide commonly needed functionality such as adaptation support from middleware services. The adaptation of mobile services has been researched in many interesting projects [13, 14]. However, this work focuses on the adaptation of mobile services in the service gateway-based distributed service architecture where the services are distributed as well. The work especially tries to identify the vital functions that are needed in the form of GSEs to provide a service platform capable of supporting the execution of adaptive mobile distributed services. Thereby the adaptation strategy selected for the service platform is application-aware adaptation where the services of the system have the final word in adaptation behavior, but the service platform provides the tools and framework for the services to decide the best adaptation behavior.

Adaptation of mobile services in general is seen as one of the basic requirements for future mobile computing and telecommunications systems along with personalization and context-awareness [2]. These three features of future mobile services complement and partly utilize the same functionalities in their execution. These functionalities can be provided to services by a distributed service platform providing the needed functionalities as middleware services.

The goal is to provide an open API to service platform as a "toolbox" to be used in the creation of adaptive mobile services. This open API will be the Service Platform API that should provide all of the needed data types, service interfaces and classes for services to use in order to have access to platform services, and also to be executable on the service platform. Adaptation will span many different system layers in future computing systems, therefore when we talk about adaptation, it is always important to define what we are adapting and why. In this work, the mobile distributed services are adapted to user context, and adaptation is studied as a part of a small-scale context-aware system. This kind

of adaptation includes the adaptation of mobile services to different terminal capabilities, user environmental conditions, and user preferences.

In the following subchapters, the adaptation of mobile distributed services is discussed. Requirements for a service platform supporting the execution of adaptive mobile distributed services also are presented and summarized.

3.1 User Context Information

A service platform supporting the adaptation of mobile services should be aware of the user context in order to be able to provide the attributes creating the need for adaptation to services utilizing the service platform. The service platform should also provide the means for internal representation of the context information.

User context information contains the attributes creating the need for the adaptation of mobile services. This context information includes all of the environmental conditions surrounding the user that are relevant in the interaction between the user and a mobile service from a service point of view. The user context information can be divided to include two kinds of information: information that is static and discrete and information that is dynamic and continuous in its nature within a short time frame. For example, user terminal capabilities and user age are static context information, as the available network bandwidth and lighting conditions are dynamic context information. The division of context data into these two classes implicates that two type of adaptation is needed as well: static and dynamic adaptation.

There are a few central issues related to user context information that should be taken into account when designing a service platform providing user context information as attributes for the adaptation of mobile services. These issues include internal presentation of context information, processing of context information, and privacy of the user context information.

3.1.1 Internal Presentation and Processing

The processing and representation of user context information can be carried out in many different ways. One possibility would be to rely on heavy usage of methods of artificial intelligence like the use of ontologies, semantics, modeling, and learning in the context representation and processing. However, this kind of approach evidently increases the system's capability requirements for the execution environment and system size.

The processing and representation of the context information can also be carried out by utilizing profiling techniques and keeping the representation and processing of the context information simple but efficient. This approach has been taken in this work and it keeps the system size acceptable for resource-constrained mobile computing environments, while providing added value for the system utilizing the context information.

In the service platform to be designed, profiling techniques will be used for implementing the adaptation support functionality of the service platform. First, the user context information is managed and delivered by profiling. In the profiling of user context information, four different profiles will be needed. First, a user characteristics profile that describes the characteristics of the user, second, a user preferences profile that describes the user's preferences towards the service domain, third, a terminal profile describing the user terminal characteristics, and finally, the environment profile that describes the dynamically changing environmental conditions around the user. The first three profiles contain static context information and the last profile contains dynamic context information.

In addition to the profiling of user context information, profiling will be used for describing the a priori context states in the interest of the adaptive services utilizing the service platform to be developed. These profiles are called service profiles and the processing of these profile services, profiling. Figure 15 illustrates the roles of both the service profile and the environment profile to be utilized in the dynamic adaptation functionality of the service platform. The environment profile consists of dynamic user context information and the service profile consists of a priori context definitions that the services are adaptive to. The role of the dynamic adaptation middleware is shown in the

middle, comparing these two profiles using simple first order logic and Boolean algebra and notifying the service if a service profile matching the current user context is found.

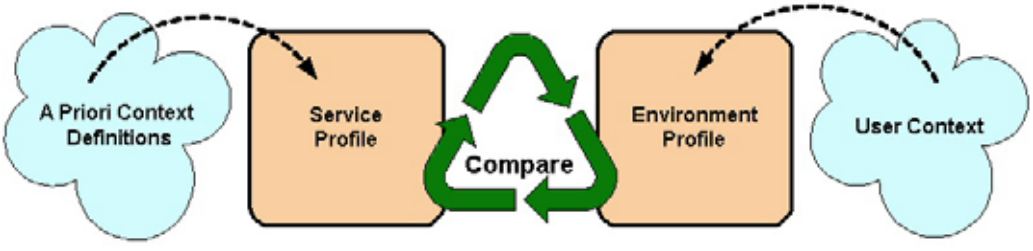


Figure 15. Profiles used in the dynamic adaptation.

As we can see, profiling will be an important and heavily used technique in the service platform to be developed. All of the profiles of the service platform except service profiles will use the same profile structure and processing methods for profiles. Therefore, the service platform should include a middleware service supporting profiling, a Generic Profiling GSE. The services of this GSE will also be useful for mobile services utilizing profiling in their operation. Other GSEs whose responsibility covers the hosting of profiles, utilize the services of the Generic Profiling GSE in profiling. The user preferences profile, user characteristics profile and terminal profile will be hosted by the Access Control GSE, and the environment profile by Environment Monitoring GSE. More details about these GSEs and their functionality will be provided later in chapter 4.

The service profiling differs a great deal from the other profiling carried out in the service platform in that a service profile and tools for processing the profiles have to be developed separately and embedded into the service platform adaptation middleware services. Figure 16 summarizes all of the needed profiles, their information content, and hosting entities in the service platform to be developed.

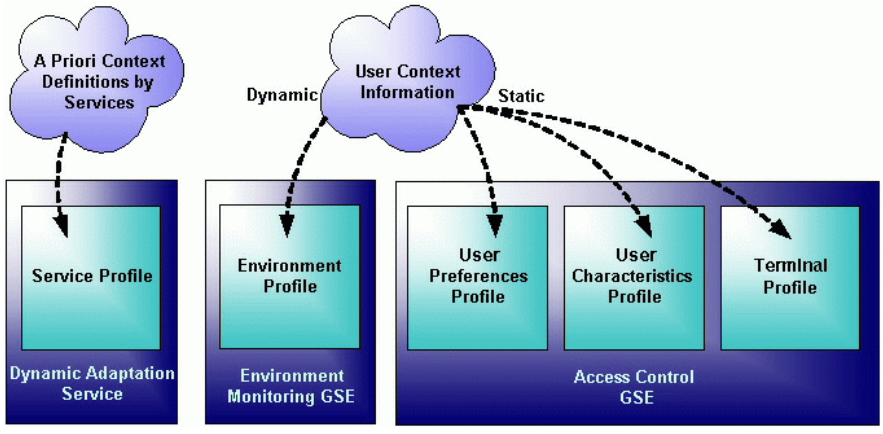


Figure 16. Profiles, their content and hosting entities.

All of the profiles to be included in the service platform will be described in more detail later in chapter 3.

3.1.2 Privacy

A big concern related to context-aware computing in general, is the privacy of the user. Privacy risks arise from the data used and needed in context-aware computing: the user context information, which can be very private in its nature. Unless special attention is not paid to the privacy issues, the misuse of user context information is possible and the privacy of the user will suffer.

For example, in a study of location-aware mobile services from the user point of view, users were worried about their privacy and the "big brother" phenomenon when considering services enabling people to be located [28]. Although security issues are not in the main scope of this work, special attention should be paid to guarantee the privacy of user context information sensed and processed in the service platform to be developed.

The user should be able to control the privacy level of his context information. The user should be able to set the context information that she/he provides in the profiles to be guarded, resulting in a situation where only trusted services and the service platform itself has access to this information. Not-trusted services

should only have access to abstract context information, and to information that is defined not to be guarded by the user. In this way, not-trusted services executing at the service platform cannot have access to private data and possibly misuse it. The trusted services that need to have access to guarded context information can gain their trusted status via the service authorization process, which should be provided by the service platform. The authorization process can be based on a simple user query. In this process the service requesting the trusted status is described to the user in detail. The description should include information about the service provider, contact information of the service provider, the description of the service, explanation why access to private context information is needed and how this information will be utilized and protected. If the user accepts the service's trusted status query, the service gains the trusted status and is authorized to access private context information. The service authorization process that will be a part of the service platform to be developed is illustrated in an event trace diagram in Appendix 2 and the event trace is described in detail in Appendix 3.

The basic adaptation support functionality, provided by the service platform, should not require services to have trusted status. This is because the adaptation process used in the service platform should be based on utilizing the abstracted context data and to be open for use by all services to gain the benefits from application-aware adaptation approach. In case the service includes its own adaptation process independent of the adaptation mechanism provided by the service platform, the service needs to gain trusted status in order to get access to the raw context data. However, the adaptation functionality provided by the service platform should be designed universally applicable, so that the majority of services do not have to rely on the Laissez-faire adaptation approach, but utilize the adaptation support provided by the service platform.

3.1.3 User Context Sensing and Delivery

The service platform supporting the adaptation of mobile services has to be context-aware. Therefore it must have the functionality to sense the user context information and be able to deliver it to the services consuming the context information.

The sensing of user context should take place near the user, therefore it would be natural that the context data be gathered inside a Personal Area Network (PAN) and delivered to a system using the end user's terminal as a communication gateway to the Wide Area Network (WAN). Another option is to sense the user context via embedded terminal's sensors. Context sensors are becoming more popular, for example, in mobile phones, where proximity sensors can be found even today. However, if the context sensing takes place via the terminal, the context is actually terminal context instead of user context. Therefore, if the actual user context is needed, the sensing of context should take place in the user's PAN where the sensors can be embedded, for example, in clothing and short range communication technologies, like Bluetooth [54], for example, can be used for context information delivery inside the PAN. Figure 17 illustrates the data flows of user interaction and context data flow from the context producing user's PAN to the context consuming context-aware service in execution at the service gateway.

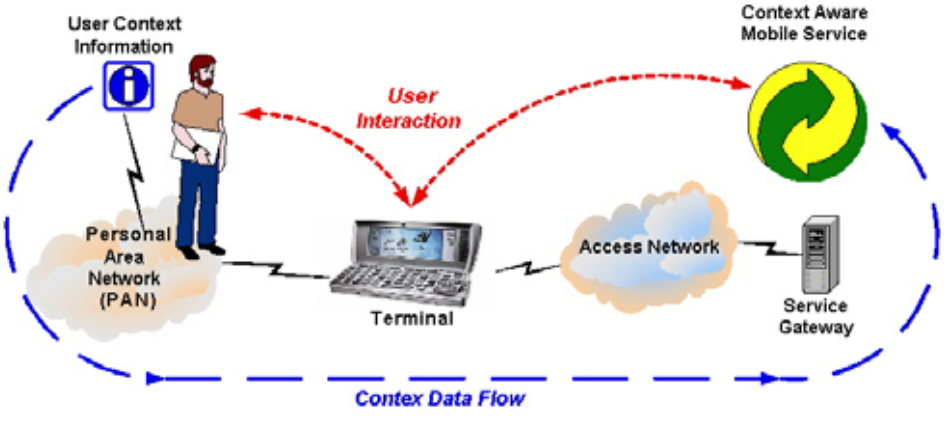


Figure 17. User interaction and context data flow.

In Figure 17, user interaction with a context-aware mobile service is illustrated. As can be seen, the communication channels for user interaction data and user context data are different. The user's interaction with the service takes place between the terminal and the service gateway, while the context data flow originates from the user's PAN, which is connected to the user's terminal. The delivery of user context data should be as automatic and transparent a process as possible from the user's perspective. This approach for context information delivery is derived from the recent development of sensor technology and

wearable computing [23, 24]. The approach in which user context data is derived from the PAN has a major advantage when compared to the approach in which the context information is gathered using the terminal. Users prefer to use many different terminals to access the same services. When context information is derived from the PAN, which is always present when the user is near a terminal, the amount of sensors and computing at the terminal device can be reduced, freeing more resources for the services to execute. Additionally, the context information derived from the PAN also describes the actual user context instead of just the terminal context.

The user context information that is provided by the distributed service platform should be available in both the service gateway and mobile terminal domains to allow continuous operation and adaptation support for distributed services utilizing the service platform even in dynamically changing environmental conditions. For example, if the connection between the mobile terminal and the service gateway is not available, the context information should still be available for service instances executing in the mobile terminal domain. In this situation, the user context delivery to services executing at the service gateway is impossible.

3.2 Static Adaptation

Static adaptation is adaptation that utilizes the static context information and can also be seen as a more familiar function of personalization. Personalization, in the context of a computing system, means filtering and adapting information sources and services to user needs, preferences, skills and environmental conditions [20]. Personalization is also referred to as tailoring [31], with this definition personalization can be thought of as the tailoring of services for individual users so that users get the feeling that the service is specifically tailored for his needs and preferences. Thereby a tailored service has more value to the user and the overall user experience is better than in the mass produced services.

The requirement of personalization supporting functionality of the service platform architecture is set by the user's added value requirements. In the future when there will be lots of content and mobile services available for the users to

select from, the user experience affected by the personalization of mobile services will play a major role in a service's success on the market. As there will be more and more content and mobile services available in future computing systems, an average user is not able or willing to adopt these new services if special attention is not paid to the personalization of the services.

The personalization of mobile services is very closely related to the adaptation of mobile services. Personalization can be seen as a part of the overall adaptation process and specifically part of static adaptation. Personalization of a service takes place at the beginning of the static adaptation process, when the user requests to use the service. At this point, the service is instantiated and the service instance is personalized to user preferences before it is sent to the user. The personalization support functionality of the service platform of this work is to be implemented by the static adaptation service residing on the middleware layer of the service gateway domain.

In this work, the focus is in the personalization of mobile services to user preferences, characteristics, and terminal equipment. In other words, personalization is seen as part of the overall adaptation process of mobile services by utilizing the user context data available. Other aspects of personalization, like filtering of information content, are neither in the focus or covered in this work.

3.2.1 The User Preferences Profile

User preferences towards a computing system and services it provides can be very manifold. This is especially true in the context of services of pervasive computing system that can have the nature of being always "on" or active. For example, a user could prefer a computing system that realizes via his mobile terminal, not to disturb him, when he is in a meeting or other situation where disturbance is not welcome. A user could also prefer that when he is preoccupied with some other task like driving a car, for example, that the computing system would not require much interaction with the user. Also, willingness of the user to use a computing system may vary a great deal depending on the situation the user is in. For example, if the user is at sleep, he probably is not interested in the latest offers from his service provider, however, if the user is not preoccupied in

any way and is awake, he might well be interested in this kind of information. As we can see, user preferences towards a computing system can be very difficult to be taken account of from a computing system point of view.

It is very important to identify the common *preference variables* between a user in any situation and the computing system including any kind of services. This is achieved in this work and a set of preference variables for describing the user preferences towards the service domain is proposed. The preference variables are identified to be *willingness*, *interaction*, and *disturbance*. These variables are managed and presented by means of profiling and used by the services to adapt their behavior to the preference variables.

The user preferences profile should provide and store a user's semantic context by naming the preferences profiles and also present user relation and attitude towards the service domain. The user preferences towards the service domain are derived from the preferences profile information by the static adaptation middleware. The role of the user preferences profile and identified preference variables between the user and the service domain are illustrated in Figure 18.

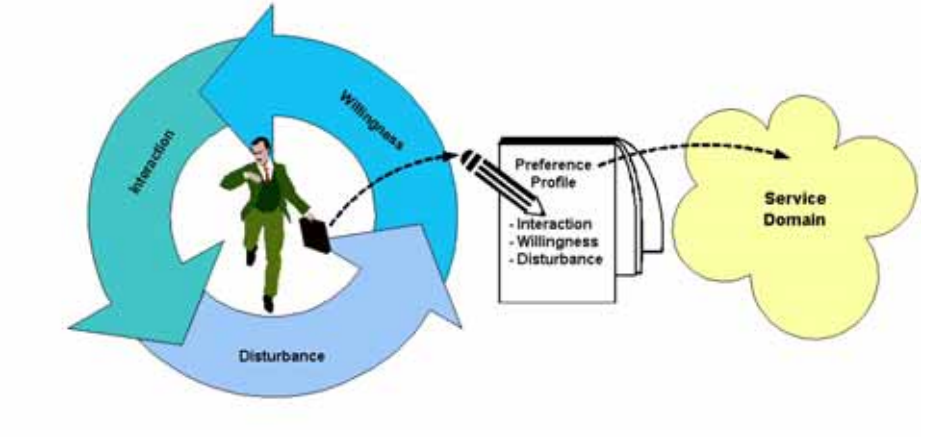


Figure 18. User preferences regarding the service domain.

Each preferences profile should include values for the three preference variables that describe the user's preferences towards any type of service present in a computing system. The identified preference variables disturbance, willingness and interaction can have numerical values from 1 to 4 describing the level of

disturbance allowed, willingness to use the services, and interaction limits for the services. The semantics for all of the preference variables values are following: 1 = NOT, 2 = LOW, 3 = NORMAL and 4 = HIGH. The descriptions for the identified preference variables are provided in the following.

Disturbance: This preference variable describes how much the system or a service is allowed to disturb the user in a way that the service's actions will have an immediate effect on the user's environment or are perceived by the user. An example of this could be if the user is in meeting his disturbance preference variable would be set to NOT, meaning that the system or any service is not supposed to disturb him with any kinds of perceivable notification sounds or vibrations of the terminal. If the disturbance variable is set to LOW, the user is not so against disturbances originated from the system and the services, however, unnecessary disturbances should be avoided when this value is set. If the disturbance variable is set to NORMAL, the user does not see the alarms and notifications as a disturbance, rather just normal computing system interaction methods. In this case, the system can use the alarms and notifications normally. If the variable is set to HIGH, the system and the services are encouraged to produce as much push type information and notifications for the user as possible. This could for example be the triggering value for advertising services.

Willingness: This preference variable describes the user's willingness to use the system or any of its services. So, the variable actually describes the user's attitude and interest towards the system and the services it is providing. If the willingness preference variable is set to NOT, this means that the user is not at all interested in the system or any service. This could be the situation if the user is asleep or wants the system to sleep. If the variable is set to LOW, the user is not very interested in the system. However, in this case, the user observes the system at some degree and if something very interesting happens, the user possibly will use the service of interest. If the variable is set to NORMAL, the user is normally willing to use the system and its services and is fully aware of the system. If the variable is set to HIGH, the user is highly aware of the system and very willing to use any service.

Interaction: This preference variable describes the user's capability to interact with the system. For example, if the user is asleep, the variable would be set to NOT, meaning that the user is not capable of any kind of I/O interaction with the

system and its services. If the variable is set to LOW, the user is capable of simple interaction with the system; for example, the user can answer a query but not give text-based input or control anything. If the variable is set to NORMAL, the user is available for normal interaction with the system. This can be the case, for example, if the user is at his PC using the keyboard and mouse. If the variable is set to HIGH, the user is available for very rich interaction with the system and the services and the services are encouraged to interact with the user.

Figure 19 illustrates and describes the relations between the example set of preferences profiles and the values of the preference variables in these profiles.

	Normal	Outdoors	Meeting	Exercising	Sleep	Entertain	In Car	At Duty	User Defined
Disturbance	3	3	1	2	1	4	3	2	1-4
Willingness	3	3	2	2	1	4	2	2	1-4
Interaction	3	2	2	2	1	4	2	3	1-4

Figure 19. Example preference profiles and their preference variables.

Even though there may be a default set of preference profiles, the user should be able to create and use his own preference profiles by defining the three preference variable values for each preference profile. The new preference profiles created by the user should be automatically saved by the system.

3.2.2 The User Characteristics Profile

The user characteristics profile describes the characteristics of the user using the service platform and its services. The characteristics profile of the user is to be managed by the Access Control GSE. The user characteristics can be used for service personalization if the user allows the publication of the characteristics he provides for the characteristics profile.

The user characteristics profile contains information entities describing user characteristics. These information entities can be, for example, last and first

name, favorite and disliked colors, age, address, e-mail address, phone number, fax number, occupation, and the hobbies of the user. The information provided in the characteristics profile can be used for personalization of services and service outlook. An example of utilizing the characteristics information of the user would be filtering of advertises to user interests, or modifying the outlook of a UI to a user's favorite colors.

The user characteristics profile includes information that a user might consider private and therefore he may not be willing to expose all of the information of the characteristics profile to all services. This is why the profiling mechanism used in the user characteristics profile should provide the possibility to guard and protect this private information from not-trusted services. The user should also control the level of publishing of this information. The characteristics profile should be editable by the user via the profile editor service that can be provided as a system service.

The characteristics profile can be extended to include almost any characteristic of a user, because the implementation of the characteristics profile is based on the profiling mechanism to be provided by the Generic Profiling GSE. The characteristics provided by the user should also be saved permanently.

3.2.3 The Terminal Profile

The terminal profile is needed for describing the user's current terminal capabilities. Having this information, the services can adapt to a user's current terminal. For example, a PDA and a laptop differ very much in their characteristics and capabilities as an execution environment for the service instances. The same service can have a different outlook and behavior depending on the characteristics of the terminal. For example, if the user is using a PDA with a touchscreen and no keyboard, services should be provided, for example, from dropdown menus instead of textual input where applicable.

The terminal profile is hosted by the Access Control GSE and utilizes the profiling services provided by the Generic Profiling GSE. The terminal profile should contain at least the information about screen size, screen coloring, processing capability, input method, and energy source of the user terminal. By

providing this information, as context information to services, the service platform supports adaptation of services to the terminal capabilities and characteristics.

3.3 Dynamic Adaptation

Dynamic adaptation utilizes the dynamic context information and is a continuous process that takes place while a service is in use and active. The requirements for dynamic adaptation are set by the continuously changing environmental conditions in a user's surroundings. One good example of an attribute creating a need for dynamic adaptation is the network bandwidth that can vary a lot in a short time frame in distributed computing systems that includes wireless communication links.

In the field of context-aware computing, the role of dynamic adaptation pronounces. This is because context-aware services must sense the context and adapt their behavior to the sensed context continuously.

In this work, the dynamic adaptation support will be provided as a middleware service of the distributed service platform developed in this work. The dynamic adaptation functionality will be based on the profiling of dynamic context information to environment profiles, constructing the current context from the information in environment profiles, and comparing the current context with the a priori context definitions in the service profiles.

3.3.1 Environment Profile

The Environment Monitoring GSE will be responsible for managing and hosting the environment profile that describes the user context information that is dynamic in its nature. The environment profile should be retrievable from the Environment Monitoring GSE via its service interface for the middleware layer components. When a service is in use and the dynamic adaptation process is active, the adaptation middleware retrieves the environment profile with certain frequency and compares the user context to the available service profiles containing the a priori context definitions made by the services.

The environment profile contains up to date information about users dynamic context. It is the only profile used for defining the user dynamic context and is necessary from the viewpoint of implementing the support for dynamic adaptation of services. It contains three kinds of information: spatial-, connectivity- and sensor-information. This information can be utilized in service instantiation and adaptation process.

Not-trusted services are adapted based on the service profiles and cannot have direct access to raw sensor measurements because of the privacy reasons. Only trusted services, for example, services related to health care and managed by a trusted party can have direct access to actual measurement data like heart rate, blood pressure, respiration rate, and body temperature. This is a security issue that must not be discarded but taken into account at the architecture level. The raw measurement data will be accessible to trusted services via the platform interface.

The environment profile to be managed by the Environment Monitoring GSE consists of three kinds of context information. These information types are categorized as follows: connectivity information, spatial information, and sensor information, and are described in the following:

The connectivity information includes information about the active connection between the service gateway and the user. This information includes retrievable and relative values of bandwidth and latency information.

The spatial information consists of the spatial relations of the user. The platform supports the delivery of any type of location objects. The location object can contain information like: latitude- and longitude resolution, latitude, longitude, altitude resolution, measurement unit for altitude, orientation of the user in relation to the terminal and distance of the user in relation to the terminal. Retrieval of this kind of information is not a trivial task, but these kinds of research results have already been presented [24, 29]. Because it is not likely that there will be only one valid presentation for location information, the Environment Monitoring GSE accepts delivery of any kind of location objects. The Environment Monitoring GSE will provide a tag field for the delivery of location objects, so that the receiving peer can identify the location object currently in use. Because the focus of this work does not cover location-based

systems, no location information or objects are used, but the delivery mechanism will be tested in the prototype implementation.

The sensor information includes any information that can be derived from the user or the user's surroundings using sensors. There are many different things that can be measured from the user's surroundings with sensors, but the Environment Monitoring GSE takes a practical approach and by default it supports only sensor information that is retrievable with current or near future technology with reasonable costs. This sensor information is following: heart rate of the user, blood pressure of the user, respiration rate of the user, noise level in the user's surroundings, body temperature of the user, humidity in the user's surroundings, environment temperature in the user's surroundings, lighting conditions, and proximity of the user from the terminal. In addition to these default types of sensor data, the Environment Monitoring GSE will accept delivery of any sensor data by providing a tag field for unknown sensor data delivery. Based on the value of this tag field the receiving peer of the sensor data flow knows which sensor is broadcasting the data and thereby can also receive sensor data not supported by default by the Environment Monitoring GSE.

3.3.2 Service Profiling

Service profiling is a technique used for implementing the adaptation process. Each service can have one or more service profiles, each describing different contexts. The adaptation middleware monitors these service profiles and compares them to the current context, and if new service profile conditions are found to match the current context, the service is requested to adapt to conditions defined in its matching service profile.

The service profiling solution of this work will not be based on the profiling mechanism provided by the Generic Profiling GSE, because it differs so much from other profiling carried out on the service platform. The service profiling is based on a different profiling mechanism that uses simple first order logic and Boolean algebra for defining a priori context definitions. The service profile implemented in this work will support four different a priori context definitions and the AND and OR operations between them. The number of a priori context definitions made in the service profile is usually smaller or equal to four. This is

because if more than four a priori context definitions are made in the same service profile, context conditions of the service profile become so rare that they are almost never met.

On the other hand, if the number of the service profiles for one service is big, this can be a sign of too much reactivity of a service. Although the number of service profiles for one service is not limited, because the amount of service profiles is dependent on the service functionality. For example, a health monitoring service can have a great number of service profiles without being too reactive. To provide a better insight into the operation of the service profiling, an example pseudo code algorithm of a service profile containing two a priori context definitions is provided in the following:

if*(user near desktop PC)*

AND

if*(lighting at the PC is not sufficient)*

THEN

send*(adaptation request)*

So the basic logic of the adaptation based on service profiling is to define a priori context conditions in the service profiles and bind the actions related to these conditions to the adaptation request. The data types and classes needed for creating service profiles and using the adaptation functionality of the platform are to be provided by the Service Platform API. The UML structure diagram of the service profile designed for the service platform is shown in Figure 20. In the figure, one can see the structure of the service profile and the data types, utilities, and interfaces related to it.

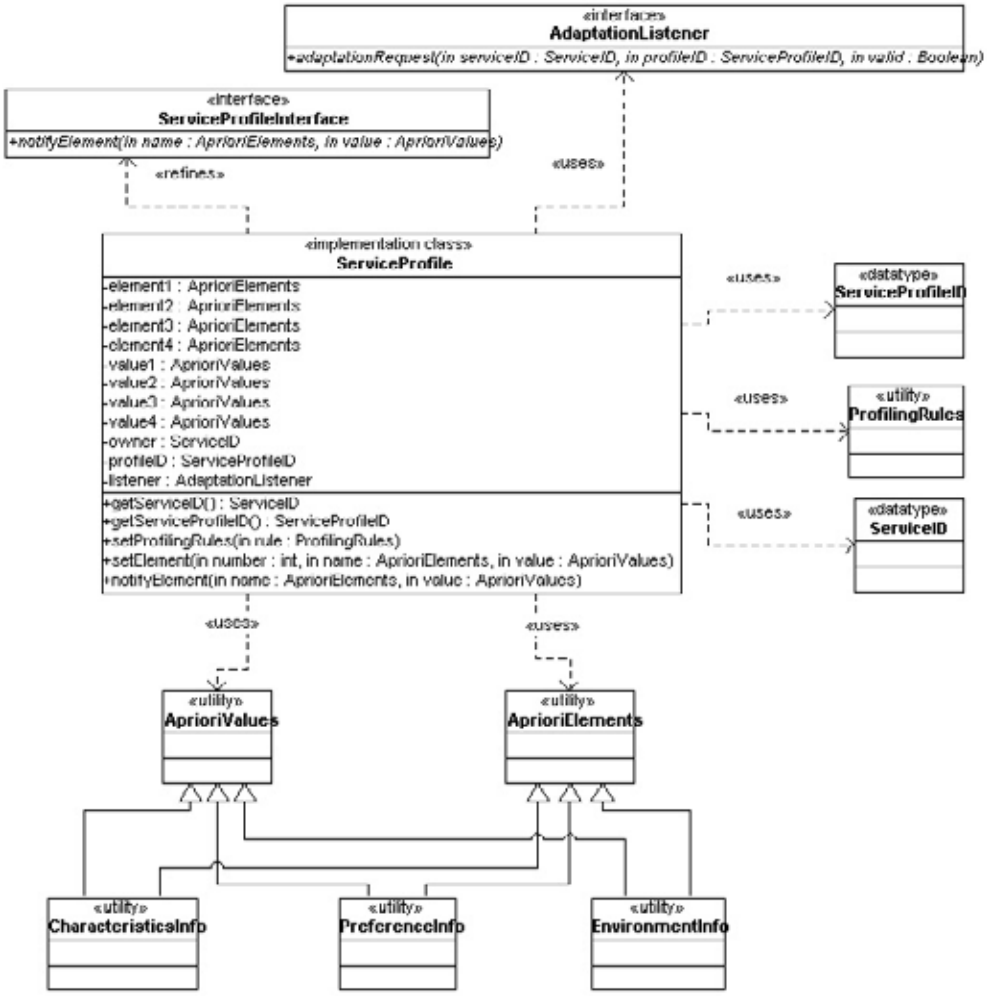


Figure 20. UML structure diagram of the service profile.

As can be seen from Figure 20, the service profile implements a ServiceProfileInterface, which defines operations to be used in the matching process with dynamic and static context information. The entities calling this interface will be the static and dynamic adaptation middleware. The service profile also utilizes an AdaptationListener interface, which is an interface to notify the service profile owner about matching or unmatching service profiles with the current context information. Each service profile contains its owner's ServiceID and ServiceProfileID given to the profile by the owning service. The service profile uses this identification information when sending adaptation

requests to services to be able to target the request to the right service and inform about the right service profile. This information is needed because each service can have multiple service profiles in use at the same time.

The service profile contains operations to set one to four a priori context elements with defined context values under monitoring. CharacteristicsInfo, PreferencesInfo, and EnvironmentInfo utilities that refer to respective profiles define the a priori context elements and the context values. When generalized, these utilities define the a priori context elements and the possible values of these elements. For example, lighting in the user's surroundings is a context element, and dark is a possible context value for this context element. The service profile also utilizes the ProfilingRules utility, which contains tags for all possible AND and OR logical combinations of the four context elements.

3.4 Adaptation Process

This chapter describes the adaptation process designed for the service platform to be developed. The adaptation process is the end-to-end process including both static and dynamic adaptations presented earlier. It is based on the application-aware adaptation approach [15], where some of the adaptation functionality is provided by the system, but final adaptation decision is left to the services. Figure 21 presents the adaptation process as a sequentially advancing block diagram and it is used for going through the process, step by step.

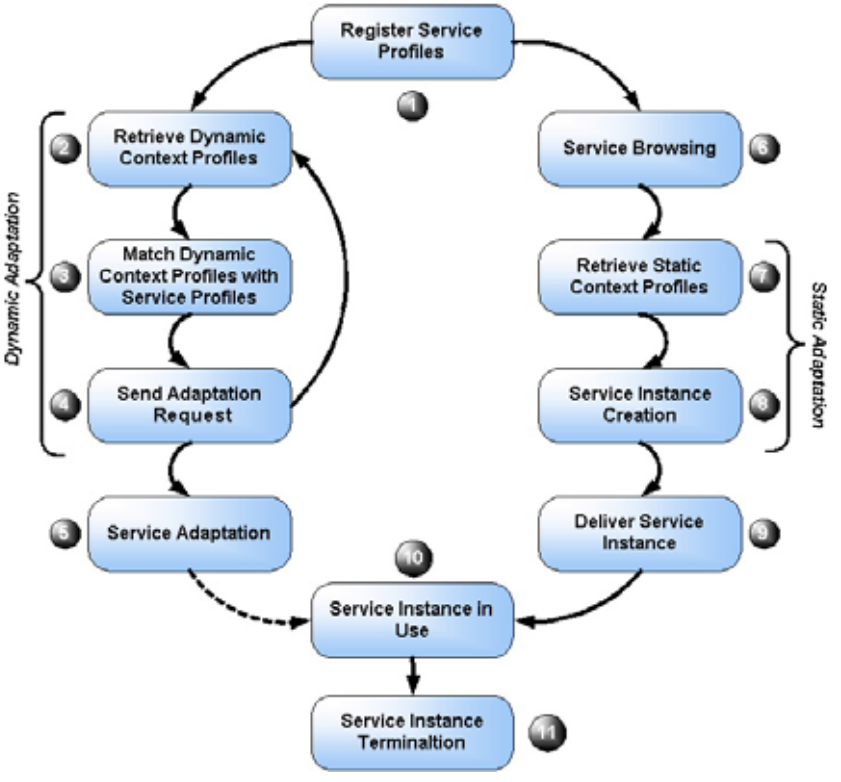


Figure 21. The adaptation process.

The following description assumes that the service is installed on the service platform and the first thing the service does after installation is register its service profiles with the service platform, using the platform interface.

1. The service has been installed and it registers its service profiles with the service platform using the services provided by the platform interface.

2. The dynamic adaptation process takes place immediately. The dynamic adaptation process starts by retrieving the profiles that contain dynamic context information. On the service platform, there is only one profile containing dynamic context information - the environment profile.

3. The dynamic adaptation process continues by matching the dynamic context information in the profiles with the a priori context definitions made by the service profiles.

4. The third step of the dynamic adaptation process is to send an adaptation request to services owning the service profiles depending on if the service profile matches the current dynamic context. If the service profile matches the current context, an adaptation request with TRUE as its parameter is sent to the service owning the service profile. If profile does not match, the request will be sent but with the parameter FALSE. After the adaptation request has been sent, the responsibility of the service platform ends and the final decision about adaptation behavior is left to the service itself. The dynamic adaptation process returns to step 2 and continuously executes steps 2 to 4.

5. When the service receives an adaptation request from one of its service profiles, it can freely decide on its actions. Possibilities are manifold, if the service instance exists, the service can adapt the service instance to fit in with the new context. The service can also change its logical behavior, adjust resource usage, perform a control action or even discard the adaptation request.

6. Now the service profiles have been installed and the dynamic adaptation process is active. Next the user browses the services to select the one he wishes to use and selects it.

7. At this point, the profiles containing the static context data are retrieved. In the service platform, these profiles are terminal profile, user characteristics profile, and user preferences profile.

8. The static context information in profiles is taken into account in the process of service instance creation. The service instance is created to best fit with the static context.

9. When the service instance has been created, it will be delivered to the user's mobile terminal for execution.

10. Now the service instance has been adapted to best fit with the current static context and the dynamic adaptation process of the service is active. At this point, the user is able to see the UI of the service and is able to use the service. If the service receives an adaptation request as a result of the dynamic adaptation process, it can adapt the service instance in use dynamically.

11. Finally, when the user doesn't want to use the service any more, the user terminates the service instance. The dynamic adaptation process is, however, still active until the service profiles are unregistered by the services having registered them.

As one can see from Figure 21, the dynamic adaptation process is continuous in its nature. This is advantageous for services that are not targeted at interaction with the user, but, for example, for automatic controlling of NAs based on the context information. The dynamic adaptation process is active until the service has unregistered its service profiles from the service platform using the services provided in the platform interface. The static adaptation process takes place only once in conjunction with the service instance creation in the presented process in Figure 21. However, the static context information can be consulted when needed using the services provided by the platform interface.

As we were able to see from Figure 21, a service for delivering the service instances for remote execution in the mobile terminal domain is needed on the platform. This service is to be implemented by the Service Delivery GSE of the service platform. Also, a service providing the delivery and creation of events in a distributed service platform is needed to support, for example, the adaptation of service instances. This service is to be implemented by the Event Notification GSE of the service platform.

3.5 Adaptation Support Middleware

The middleware supporting adaptability in this work is divided in two different parts. The division is made along the fact that some of the functionality needed in the adaptation process is static and discrete and some dynamic and continuous in its nature. These two middleware services, supporting the execution of the overall adaptation process, are Dynamic Adaptation Service and Static Adaptation Service. Both of these services are used at the beginning of the adaptation process, but once the service has been instantiated and delivered, the static adaptation phase is over and the main responsibility of the adaptation remains with the Dynamic Adaptation Service. Requirements for both of these adaptation middleware services introduced are described in more detail in the following subchapters.

3.5.1 Dynamic Adaptation Service

The dynamic adaptation service is responsible for supporting the continuous adaptation that starts when a service registers its service profiles and ends when the service profiles are unregistered. The function of the dynamic adaptation service is based on retrieving the environment profile containing the dynamic context information from the Environment Monitoring GSE, and comparing the dynamic context information with the a priori context definitions in the service profiles provided by the services. After the comparison, an adaptation request is sent to the service owning the service profile. If the service profile matches the adaptation request, the parameter TRUE is sent, and if the service profile does not match, FALSE is sent.

The structure and the logic of this middleware service will not be very straightforward, therefore the functionality of the dynamic adaptation service and entities involved are explained in more detail in the following Figure 22 and its step-by-step description.

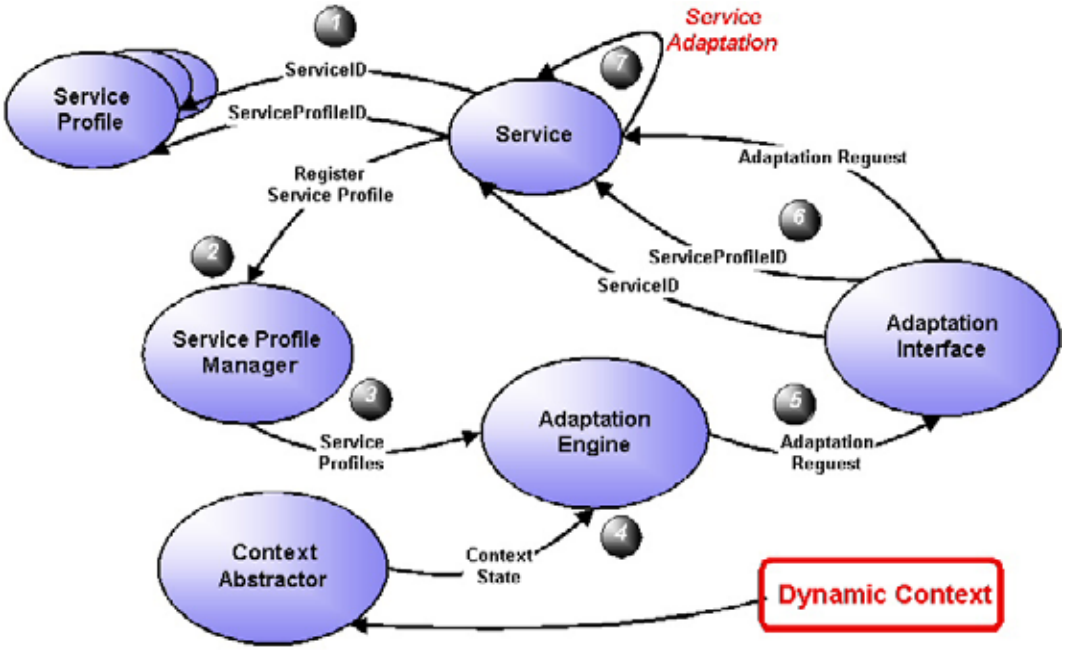


Figure 22. The logic and structure of the dynamic adaptation service.

1. First, a service creates a service profile and gives its ServiceID and a unique ServiceProfileID to the service profile. The service also provides the Adaptation Interface what is needed later for the service profile.

2. When the service profile is created, the service registers the profile to Service Profile Manager.

3. The Service Profile Manager provides the functionality for service profile registering and unregistering. It adds all of the registered service profiles to the Adaptation Engine and removes all of the unregistered profiles from the Adaptation Engine.

4. The Adaptation Engine retrieves the dynamic context information periodically and compares the current dynamic context to the a priori context definitions found in the service profiles. The Adaptation Engine retrieves the dynamic context information using a Context Abstractor, which abstracts the raw measurement data into useful context data. In the service platform, the dynamic context is found from the environment profile.

5. After comparing the dynamic context data with service profiles, adaptation requests will be sent to the owners of the service profiles using the Adaptation Interface. If the service profile matches current context the adaptation request, the parameter will be TRUE, otherwise FALSE.

6. The Adaptation Interface receives the adaptation request in the service domain. The adaptation request also contains the ServiceID and the ServiceProfileID of the target service.

7. The service receives the adaptation request and the final decision about adaptation behavior is left to the service taking the application-aware adaptation approach.

3.5.2 Static Adaptation Service

The static adaptation service is responsible for the static adaptation support functionality provided by the service platform. The basic operational principles

of the static adaptation service are the same as in the dynamic adaptation service: both are based on profiling and processing of profiles. Where the profile data processed in the dynamic adaptation service is dynamic and continuous in its nature, the data processed in the static adaptation service is static and discrete in its nature.

The profiles processed at the static adaptation middleware include terminal profile, user preference profile and user characteristics profile. In implementation, the main focus will be on adaptation and not personalization, so the implementation of the static adaptation service will be a minimum implementation that will support personalization only by providing the needed information in the form of profiles. The static adaptation service will provide services for retrieving the terminal profile, user characteristics profile, and user preferences profile. The services can retrieve these profiles whenever needed, and use the static context information contained in these profiles as applicable.

The information can be used for personalization purposes, and also to retrieve information about user preferences towards the service domain. The user preference information should always be consulted by the services before taking any kind of action that will have a direct effect on the user context. For example, any kind automatic controlling of NAs, based on the context information should not take place if the user has set his preferences to the sleep profile, where the three preference variables have value 1.

In summary, the Static Adaptation Service should provide services for the retrieval of profiles containing static context information that can be used for personalization purposes and for guiding service behavior according to user preferences.

3.6 Adaptive Distributed Mobile Services

The adaptation of distributed mobile services cannot just be the responsibility of the middleware services. Middleware services can provide important support for adaptation process, but it is quite clear that the services should also be designed to be adaptive as well. This is due to the adaptation strategy selected for the service platform, the application-aware adaptation.

The distributed mobile services and services in general have two different parts: service logic and the user interface. The part implementing the service logic and containing all of the heavy calculation processes is referred to as a service in this work. The part implementing the user interface and additional functionality related to user interface and service adaptation is referred to as a service instance in this work.

In summary, a functional adaptive distributed mobile service contains two parts: service, which is executed in the service gateway domain and the service instance, which is executed in the mobile terminal domain. The life cycle of a distributed service executed by the service platform developed in this work is presented in Figure 23.

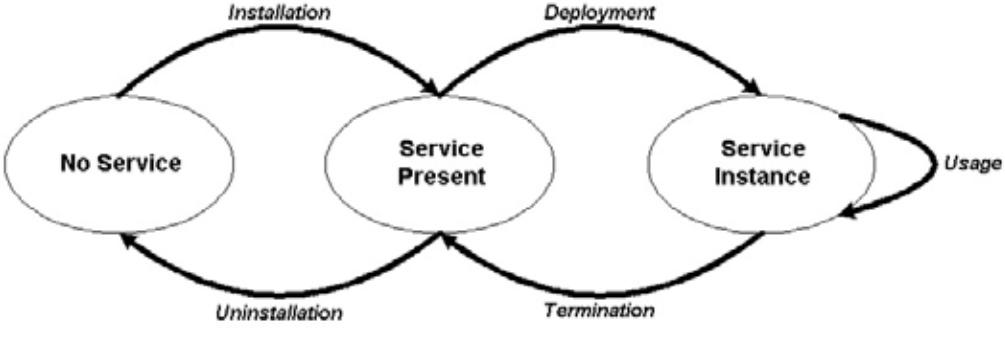


Figure 23. Life cycle of a distributed service.

As seen from Figure 23, the service is first installed to the system and registered to the service platform in the service gateway domain. When a user selects a service to be used, it is instantiated and deployed to the user's mobile terminal. The actual usage and interaction with the service takes place via the service instance in the mobile terminal. After the user stops using the service, the service instance will be terminated. The services from the service gateway domain can also be uninstalled from the system and the service platform.

In this work, the Service Platform API sets the requirements for the services to be adaptive, and provides the tools for the service's adaptation. There are some requirements for the adaptive services that have to be recognized when designing the services. These requirements are brought to attention in the following:

Reactivity: The platform provides a dynamic adaptation service that gives adaptation requests to services about the changes of user context. It should be noted that services should not be designed to be too reactive to the context changes, because this will generate overhead in the network usage and also the usability of the service will suffer if the service instance changes all the time. Basically, the level of reactivity is dependent on the service, but the usability of the service and resource optimization should be the guiding factors when making the design decisions related to the reactivity of adaptive services.

Bifunctionality: Distributed services must have two different operational modes in the service lifecycle presented in Figure 23. First an operational mode for the state service present. In this state, the user is not using the service, but the service can have background processes that require no user interaction (e.g. data gathering from the network). In this state, the service may not have any knowledge of user context or no means of interacting with the user. The second operational mode is the state in which the service has been selected and instantiated. In this mode, the service can be in interaction with the user and the user context data is available for the service. This is also the operational mode where the reactivity of the service becomes an issue.

3.7 Requirements Summary

In this chapter we have taken a look at the functionality needed to support the adaptation of mobile services. Now it is time to summarize the requirements of a service platform and its architecture supporting the adaptation of mobile services.

The first issue to be solved when designing a service platform supporting adaptability of mobile services is to decide on the distribution of the functionality of the adaptation process. Whether to leave the adaptation completely to the responsibility of individual services and only provide the context information via the service platform, or to include all of the functionality in the service platform making the adaptation fully transparent to the services. A third option is to combine the first two and provide some adaptation functionality via the service platform while retaining some of the responsibility with the service itself. The third option is an approach for the distribution of the

adaptation functionality used in this work. In this way, the adaptation functionality can be provided in such a way that the privacy of user context data can be guaranteed and the functionality is general enough to be used by different services designed to have different tasks. This approach also decreases the service size when compared to the option where each service has its own adaptation process and mechanism.

The GSEs will be the basic building blocks of the architecture to be presented for the service platform. The role of the GSEs, providing commonly needed services for the services and other middleware entities, is one of the obvious advantages of the GSE concept. GSEs have been briefly mentioned in previous chapters mainly as hosting entities for the profiles and implementing entities of needed platform functionality. The GSEs and the architecture based on them will be further discussed in chapter 4.

A clear need for context sensing and delivery functionality was recognized for a service platform supporting the adaptation of mobile services. This was due to the fact that the phenomena that the context information describes create a need for adaptation in the first place. The context sensing and delivery functionality will be implemented in the mobile terminal domain of the service platform; the implementing component will be the Context Sensing middleware service.

Two different types of context information were identified, static and dynamic context information. It was also identified that profiling will be a technique much used in internal context presentation of the service platform. Therefore, a platform component to provide a basic profiling mechanism for the platform and the services is needed. This profiling mechanism should also guarantee the privacy of user context data.

Four different profiles based on common profiling mechanism that can be provided by service platform were identified. These profiles were user characteristics profile, user preferences profile, terminal profile and the environment profile.

There was one different kind of profiling mechanism identified that does not utilize the same profiling mechanism as the others; this was the mechanism for

service profiling. The service profiling mechanism is to be implemented by the middleware services of the service platform.

In the context of user preferences and user characteristics profile, a need for saving data permanently was identified. The platform will provide a service for saving data permanently on a distributed service platform. This service will be useful for other services and platform components to save certain data permanently as well.

The need for event delivery within the distributed service platform was also identified. Therefore the service platform will provide an event creation and delivery service to be utilized by platform components and services. The event delivery service is useful, for example, in service instance adaptation or in the communication between a service and a service instance.

A need for platform service delivering service instances from the service gateway domain to be executed in the mobile terminal domain. Therefore, this kind of service delivery component will be provided as part of the service platform.

Because the service platform of this work is distributed, some kind of communication mechanism between the parts of the platform is needed. The communication mechanism will be a customized message-based solution and will be abstracted in order to gain independence of any specific communication protocol and platform portability.

Separate platform management components are also needed to manage and join the service interfaces of different middleware services to one uniform platform interface that is provided for the services. Last but not least, platform control components are needed for controlling the overall distributed service platform consisting of parts presented so far. In the next chapter, the architecture for the service platform that is based on the requirements presented, and supports adaptation of mobile services, is presented.

4. Architectural Design

In chapter 3, requirements for a service platform supporting the adaptation of mobile services were presented. This chapter presents an architecture that is based on those requirements and provides more detailed descriptions of the components of the architecture. The architecture presented in this chapter will be implemented, tested and validated in chapter 5.

The architecture for the distributed service platform supporting the adaptation of mobile services presented in this work is found in Appendix 1. The notation used in Appendix 1 is not based on any standard, but is used in [55]. Notation is called conceptual architecture structure, and the elements used in it are presented in the following:

Component: Component is a replaceable entity and the symbol for component is a quadrangle. Components have stereotypes, which define the role of the component. The stereotype of a component is marked as follows: <<*Stereotype*>>.

Service: Service is a component that provides functionality to other components.

Application: Application is a component that utilizes services to perform some specific tasks useful to the user.

Domain: Domain is a specified part of the system. The execution environment, physical boundaries or the abstraction level can define a domain. The symbol for domain is also a quadrangle, but the weight of the line is heavier than in components. Every component belongs to one or more domains.

Port: The symbol for a port is a directed triangle. A triangle pointing into a component represents an offered interface, and a triangle pointing out from a component represents a needed interface.

The entities with their roles and functions seen in the architecture in Appendix 1 are explained in this chapter. As one can see, the architecture of the service platform is vertically divided into two main domains: service gateway and mobile terminal. In addition, the architecture is horizontally divided into three

layered domains: connectivity layer, middleware layer and service layer. Notations (N) and (M), Network and Mobile respectively, in the names of the entities are used for outlining the fact that these entities may not be equal to their implementations even if they have the same names and the components reside in different computational nodes.

4.1 Architecture Overview

The focus of the work is in the adaptation of mobile distributed services. Therefore, middleware services studied and presented in this work are limited to services that provide support for adaptive mobile services. Despite this, the architecture is not bound only to systems for adaptation, but is extendable to support other functionalities as well.

As already mentioned, the architecture of the service platform is divided into five different domains. The first division of entities in the service gateway and mobile terminal domains is due to the distributed nature of the service platform. The second division of entities on different layers is due to different abstraction levels in this kind of computing system. An overview of the service platform architecture with all of the components is presented in Figure 24. For a more detailed architecture figure, please refer to Appendix 1.

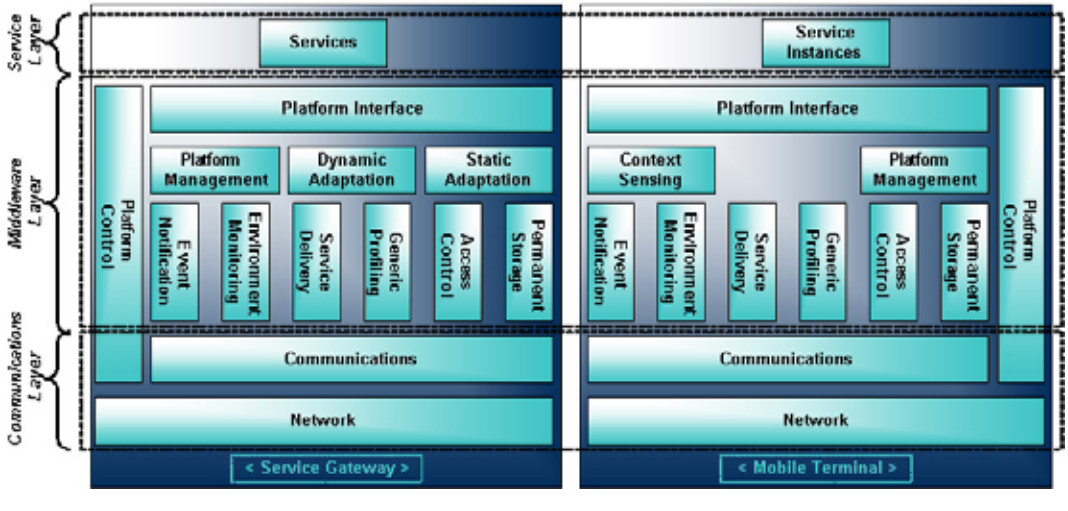


Figure 24. Overview of the service platform architecture.

The fundamental idea of this architecture is simply to provide support in adaptation functionality for services and service instances that are executed at the service level by utilizing the middleware layer services and the connectivity layer services. More detailed descriptions of the horizontal layers of the architecture are provided in the following three subchapters.

4.1.1 Service Layer

The service layer is the topmost layer of the architecture and can be seen as an execution framework for services in the service gateway domain and for service instances in the mobile terminal domain. The service layer is the outlook of the service platform from the service point of view. The services of the service platform are accessible via the platform interfaces both in the service gateway domain and in the mobile terminal domain.

The platform interfaces of the domains are not necessarily equal because of obvious reasons: The functionality needed for services in the service gateway domain may differ from the services needed for service instance execution in the mobile terminal domain. However, the only GSEs of the service platform having unequal service interfaces in the service gateway domain and mobile terminal domain are the Service Delivery GSE and the Environment monitoring GSE. The platform interfaces encapsulate all of the functionalities of the domains provided by the middleware layer services into one interface, which is the platform interface.

4.1.2 Middleware Layer

The middleware layer is the layer where the middleware services with different functionalities are present along with the GSEs. The middleware entities on the middleware layer utilize GSEs via the GSE service interfaces to provide support for services in execution on the service layer.

The middleware layer provides a platform interface to the service layer on both domains of the system: Platform Interface(N) and Platform Interface(M). The Service Platform API that is provided for services in the service creation phase

includes all of the needed data types and classes to access both of the platform interfaces. The services are created to utilize the Platform Interface(N) and the separate service instances to utilize Platform Interface(M).

4.1.3 Communications Layer

The communications layer in this work includes all of the layers of the OSI [56] model from the presentation layer to the physical layer. The communications layer is not in the main scope of the work, therefore no standard protocols are presented in the architecture framework. The communication abstraction for the service platform is made to completely separate the platform from any transport or session layer protocol. This is done to guarantee the portability of the platform.

The communications services, both in the service gateway domain and in the mobile terminal domain, implement the customized communication abstraction for the middleware layer services. In other words, the communications services implement the protocol that the service platform uses in its internal communications.

The internal protocol of the service platform defines its own Protocol Data Unit (PDU) and means for sending and receiving the PDUs within the service platform. The communication solution is message-based and does not require session-based communications, however, delivery of the platform PDUs must be guaranteed and no PDU losses are allowed. Because the communications layer is not in the main scope of this work, no further details will be provided about the communication solution implementing the platform communications abstraction.

4.2 Generic Service Elements

The drive for the adaptation supporting GSEs of this work derives from the special characteristics of mobile Internet compared to the Internet at the moment. In the mobile Internet, there are factors that have effects on the services and thereby affect the quality of service and service availability. These issues like changes in connectivity, heterogeneous terminal devices, unreliable wireless

links and handovers between different networks have effects on multiple levels of the overall system. In addition, growing usability and added value service requirements introduce services that provide these qualities by adapting to the user's current situation, also referred to as user context.

If compared to the OSI [56] model, these discrete or continuously changing variables that the services should adapt to have effects on multiple layers of the model, and this would mean cross-layer interfaces, for example, between the transport layer and application layer. This issue is further discussed in [57]. One possible role for a GSE could be to implement this cross-layer interaction by providing well-defined interfaces for different architecture layers to interact in a way that the overall architecture is capable of coping with the special characteristics of the mobile Internet, and still provide a rich user experience from a service point of view.

At the moment there is only one definition available for GSEs. The WWRF WG2 has defined GSEs to be upper middleware layer services [2]. They have also identified eight GSEs needed for supporting adaptive applications, however, it should be noted that in [2], GSEs are specified only at the conceptual level and mostly just initializing requirements are presented for the identified GSEs.

This work acknowledges the definition of GSEs to be upper middleware layer services, but in addition takes the definition further by defining a structure and role for the GSEs in a heterogeneous and distributed computing environment. In the GSE definition in [2], the utilizing of AI methods, ontologies, semantics, models, and learning to support adaptive applications is emphasized. This work takes a more practical approach by identifying requirements, designing, implementing and testing the GSEs proposed to support adaptive mobile services in a service gateway-based service architecture.

The definition for a GSE as used in this work appears in [1] and is provided in the following:

Generic Service Elements (GSE) are upper middleware layer services that provide well-defined functionality and services that are collectively needed and utilized by the services, applications, and other GSEs in a distributed heterogeneous computing environment.

GSEs are architectural concepts that do not define or limit the service provided by a GSE. The definition given sets only two requirements for a GSE. Firstly, the service provided by a GSE should have a certain level of universal applicability to be collectively applicable to be used by the applications, services, and other GSEs. Secondly, GSEs should be able to operate and provide their services in a heterogeneous distributed computing environment. So the service provided by a GSE should be accessible and executable in every computational node of a distributed system where it is needed.

The conceptual structure of a GSE of the service platform and its role as a platform component is illustrated in Figure 25. As it can be seen from Figure 25 b) the service platform GSEs are distributed to the service gateway and mobile terminal domains. So a logical GSE is realized as two separate components on each side of the service gateway-based service architecture. GSE (N) in the service gateway domain, and GSE (M) in the mobile terminal domain, both having their own service and control interfaces realize a logical and functional GSE by interworking. The different parts of the GSEs utilize the communications solution provided as service platform component in their internal communications to provide one logical network transparent GSE from the GSE Client point of view.

Each GSE may use its own protocol that can be standardized or proprietary for its internal communication because the protocol should not be visible outside the GSE domain. The GSE-specific protocols can be based on a customized communications solution provided by the service platform or utilize middleware technologies like CORBA, Java RMI, and RPC. All service platform GSEs are controlled by the platform control components via their control interfaces that are equal.

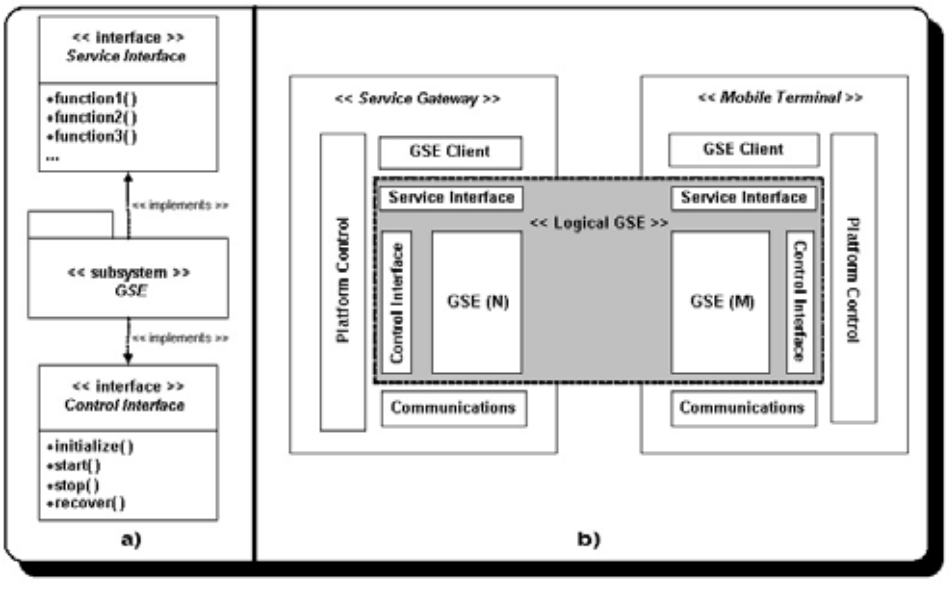


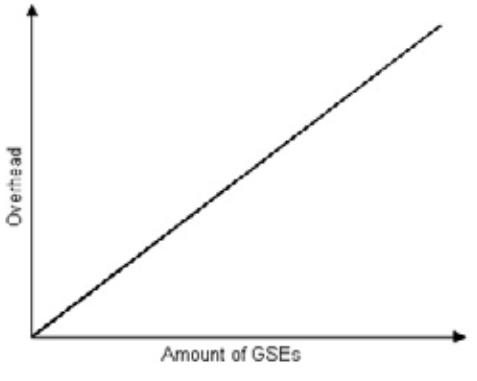
Figure 25. Basic structure of a GSE as part of the service platform.

When designing the GSEs and the architecture based on them, it should be kept in mind that the selection of the task specificity level of the GSE will greatly affect the amount of GSEs needed and the overhead due the amount of the GSEs used. Here, term task specificity refers to the level of universal applicability of a service that an entity is providing.

For a GSE to be generic its functionality must not be too task-specific, but the functionality of a GSE should be well specified in a manner that it is applicable to be used by many different services cooperatively. On the other hand, there are risks in making a GSE as generic as possible too. This is due to the fact that if a component is made as generic as possible, the functionality that the component provides is oversimplified. Therefore, the amount of generic components needed to implement the same functionality as provided by a more task-specific component is higher. If GSEs are designed to be too task-specific or too generic and simplified, the amount of GSEs present in a system will explode and system efficiency will suffer because of the increasing overhead caused by the amount of GSEs present in the system. The overhead is due to the structure and control needed for GSEs. The amount of GSEs as a function of GSE task specificity and the overhead as a function of amount of GSEs is presented in Figure 26 a) and

b). Figure 26 is just an estimated illustration of experiences; it is not based on actual measurements or experiments.

a) Amount of GSEs as a function of task specificity.



b) Overhead as a function of amount of GSEs.

Figure 26. Illustration of effects of GSE task specificity.

The following example illustrates the role of GSEs as the binding entities and basic building blocks of the distributed service platform. Figure 27 provides an example of adapting a service that has already been instantiated and is in use. The need for adaptation in the example is a change in connectivity.

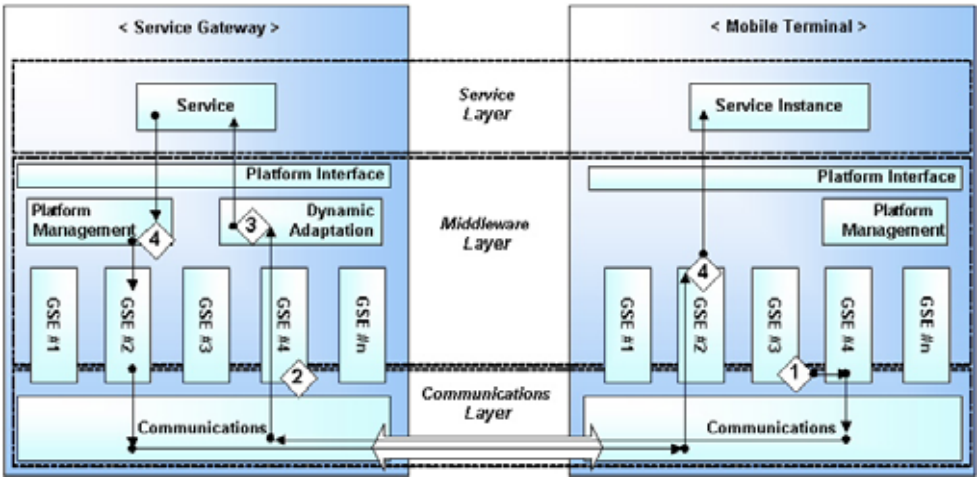


Figure 27. Example usage of GSEs in the adaptation of a service.

The steps marked on Figure 27 with numbers inside a diamond figure are explained in the following:

1. An entity on the communications layer notifies the mobile resident part of GSE #4 about dramatic changes in connectivity. For example, this could be notification that available bandwidth has decreased by half.

2. The notification is immediately transferred to the network resident part of GSE #4.

3. On the network resident side, on the middleware layer, the dynamic adaptation service gets notification of a connectivity change from GSE #4. The dynamic adaptation service notifies the service in execution using network resources about the new situation regarding connectivity.

4. The service adapts to the new situation based on the notification from the dynamic adaptation service. The service creates an updated and adapted service instance and passes it to the middleware layer using the services provided by the platform interface. The platform management component receiving the updated service instance via the platform interface uses GSE #2 to update the service instance to a new one at the mobile resident part of the service platform.

5. The mobile resident part of GSE #2 receives the new service instance from GSE #2 and updates it on the service execution layer.

In the following subchapters, high abstraction level descriptions are provided for the GSEs that were identified and used in this work to provide the needed adaptation support for adaptive mobile services.

4.2.1 Outlining Service Platform GSEs

The architecture of the service platform is based on the GSE concept. In this chapter, the GSEs of the service platform are outlined and functionalities that were identified for the service platform are allocated to six different GSEs. GSEs have already been briefly mentioned in previous chapters, mainly as hosting entities for the profiles and implementing entities for needed platform functionality.

The need for a component providing generic profiling mechanism within the service platform was identified earlier. A Generic Profiling GSE will provide this generic profiling mechanism. The provided generic profiling mechanism will also guarantee the privacy of user context data.

Four different profiles, based on a common profiling mechanism, were identified. These profiles were user characteristics profile, user preferences profile, terminal profile and the environment profile. The first three profiles, which consist of static context information that is partly provided by the user, will be hosted by the Access Control GSE. The environment profile, which consists of dynamic context data that is sensed automatically, will be hosted by the Environment Monitoring GSE.

Regarding user preferences and user characteristics profile, a need for saving data permanently was identified. The platform will provide a Permanent Storage GSE for saving data permanently on the service platform. The Permanent Storage GSE will be useful for services and other GSEs needing to save certain data permanently as well.

The need for event delivery within the service platform and for the services in a distributed environment was also identified. The service platform will provide an event creation and delivery service implemented by the Event Notification GSE. Other GSEs and the services executing on top of the service platform can utilize the service provided by the Event Notification GSE.

A service for delivering service instances from the service gateway domain to be executed in the mobile terminal domain was identified to be needed. A Service Delivery GSE implementing this service will therefore be provided as part of the service platform.

In summary, six different GSEs have been outlined and identified to be the basic building blocks of the service platform. These GSEs are described and designed further next.

4.2.2 Environment Monitoring

The Environment Monitoring GSE is responsible for monitoring the changes of the environmental elements from the user's point of view. The environmental elements are entities of the environment surrounding the user in an I-centric communication model, for example, connectivity, location and sensor information. This information is collectively referred to as dynamic context information. The dynamic context information can also include physical characteristics of the user (e.g. heart rate, respiration rate, and blood pressure).

The Environment Monitoring GSE hosts and manages the environment profile, which always contains up to date information about the user's dynamic context. The environment profile can be retrieved from the Environment Monitoring GSE and used by the dynamic adaptation middleware in supporting service adaptation. The environment profile is updated mainly from the mobile terminal domain part of the Environment Monitoring GSE, because most of the user context information is only available from the mobile terminal domain. The connectivity information is the only context information of the environment profile that can be updated from the service gateway domain. The Environment Monitoring GSE utilizes the service platform basic profiling mechanism provided the Generic Profiling GSE.

The connectivity monitoring solution is embedded in the Environment Monitoring GSE and is based on sending a test PDU with certain time intervals using the communication services provided by the platform. Bandwidth and latency values are calculated from the round trip time of the test PDU. These values are not true measuring results, but are relative to the real values and are therefore usable for estimating connectivity.

There is a basic set of context information that the Environment Monitoring GSE is capable of delivering by default. Also, optional information can be delivered to services by providing the information source software (e.g. sensor measuring software), data types and the middleware entities related to specific information source (e.g. middleware for an optional sensor). In this way, the set of context information that can be delivered via the Environment Monitoring GSE is extendable. To provide a better insight into the role of the Environment Monitoring GSE on the service platform, a sequence diagram illustrating the dynamic context sensing and delivery is provided in Figure 28.

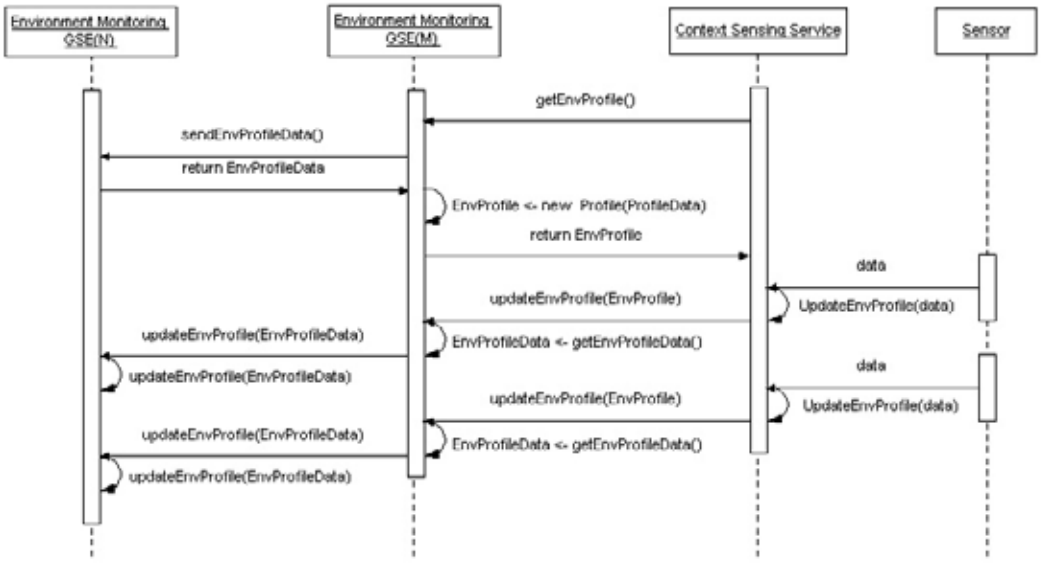


Figure 28. Dynamic context sensing and delivery.

As can be seen from Figure 28, the Environment Monitoring GSE is distributed and has different parts in the service gateway domain in the network (N) and in the mobile terminal domain (M). The Context Sensing middleware service and the sensors are located in the mobile terminal domain. The Context Sensing service first requests the environment profile from the Environment Monitoring GSE in the mobile terminal domain. The Environment Monitoring GSE (M) fetches the current data of the environment profile from the Environment Monitoring GSE (N) and creates a new environment profile from this data. The new environment profile is returned to the Context Sensing service that updates the data it receives from sensors in the profile. The Context Sensing service updates the environment profile with certain frequency by utilizing services provided by the Environment Monitoring GSE (M). The Environment Monitoring GSE (M) updates the environment profile further in the Environment Monitoring GSE (N), where the environment profile used in the adaptation process is located.

4.2.3 Event Notification

The Event Notification GSE is responsible for delivering both service and platform-related events between the service gateway and the mobile terminal domains of the service platform. The services provided by the Event Notification GSE are accessible to services via the platform interfaces for the services. The data types needed for event generation and delivery are provided in the Service Platform API. The event delivery mechanism is designed to be asynchronous so that the event source and consumer do not have to wait for each other because the event generating party is not blocked when sending an event.

The event delivery mechanism is designed to be as simple as possible in order to keep the size of the software residing in the mobile terminal domain small. The actual event content is unknown and irrelevant to the Event Notification GSE, it just delivers the generated events to event listeners, and the event content is not processed in any way.

The implementation of the Event Notification GSE is based on event listeners that are registered with the Event Notification GSE via the platform interface and the event generating service that is also accessible via the platform interface.

The events can be generated and received in both domains of the architecture, and the distributed nature of the Event Notification GSE is fully transparent to the services.

The Event Notification GSE can be used, for example, in the communication between the service and the service instance. It can also be used to trigger or control the adaptation of the service instances.

In summary, the Event Notification GSE has two main responsibilities: delivering events related to internal platform functionality, and delivering service-specific events. An example usage of the Event Notification GSE is provided together with the example use case of Permanent Storage GSE in the next subchapter.

4.2.4 Permanent Storage

The Permanent Storage GSE provides permanent storage for the different layers of the overall architecture to save layer-specific parameters and data. For example, a synchronized calendar service can utilize the Permanent Storage GSE to save data given by the user, and a middleware entity can save its parameters in order to maintain it's internal state in erroneous conditions.

The Permanent Storage GSE uses the platform identification from the platform entities and the service identification from the services for its internal data management. Each client of the Permanent Storage GSE has its own storage(s), which can contain multiple retrievable data containers. These data containers are the basic units of the permanent storage. The size of a data container is not limited, and it can contain any data in serialized format. However, it should be noticed that storing large amounts of data that is needed for service purposes using Permanent Storage GSE is not recommended, because network file systems are available for this purpose. The basic operating principle of the Permanent Storage GSE is to store the data locally whenever possible in order to gain fast access time to the data.

The clients of the Permanent Storage GSE can create and delete storages. Only storage that is empty and does not contain any data containers can be deleted.

The clients can also create and delete data containers as they wish. Data containers can only be created and deleted, not updated. In order to perform an update, the client has to read the old data container and update it and write it to a new data container. After this, the old data container can be deleted. Figure 29 illustrates the hierarchy and structure of the Permanent Storage GSE elements.

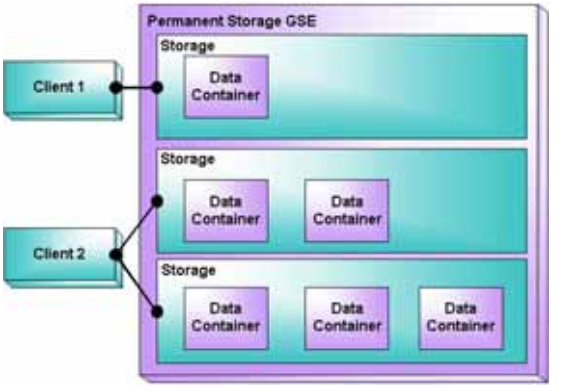


Figure 29. The hierarchy and structure of the Permanent Storage GSE elements.

As can be seen from Figure 29, each client can have one or more storages, which can be empty or contain a certain amount of data containers. Access to another client's storage is prevented by the Permanent Storage GSE. To provide an analogy with file system: storages can be thought as directories and data containers as files.

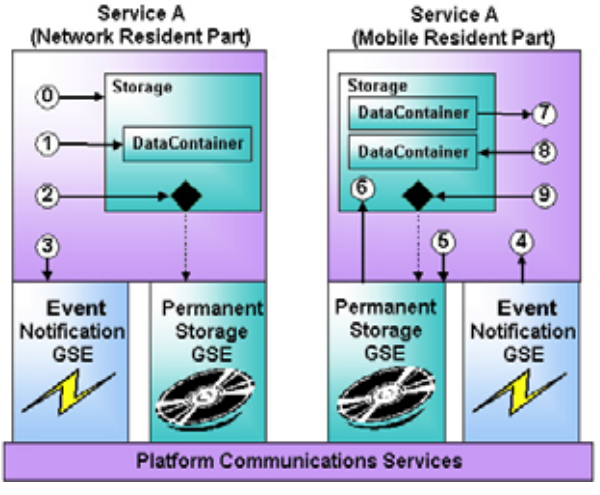


Figure 30. Example of Permanent Storage GSE usage.

In Figure 30, an example use case of the Permanent Storage GSE and Event Notification GSE cooperation can be seen. The service in the use case is distributed across the two horizontal domains of the service platform. In the example use case, the network resident part of Service A (NRA) saves some data using the Permanent Storage GSE and notifies the mobile resident part of Service A (MRA) by generating an event via the Event Notification GSE. The MRA receives this event and retrieves the data that has been saved by the NRA. The MRA can read the data or add its own data and save the data again. Both the NRA and the MRA are able to delete the data that they have saved.

To provide a more detailed picture about the usage and operation of the Permanent Storage GSE, the following sequence explains the use case in detail by following the numbers in Figure 30:

1. The NRA creates a new storage object and a new data container and saves some data in the container. After saving the data in the data container, the NRA saves the data container to the storage object created earlier.

2. To save the data permanently, the NRA calls storage's saving method that will save the storage and the data container permanently via the Permanent Storage GSE service interface.

3. The NRA creates an event to notify the MRA about the saved storage object and sends the event to the Event notification GSE for generation and further delivery. The event includes a reference to saved data.

4. The MRA receives the event, including the reference to the saved data, from the Event Notification GSE.

5. By passing over the reference to the Permanent Storage GSE, the MRA requests the storage including the data container saved earlier by the NRA.

6. The storage object requested is passed over to the MRA.

7. The MRA reads the data saved in the data container by the NRA.

8. The MRA creates a new data container and adds it to the Storage.

9. To save the altered storage containing new data permanently, the MRA calls storage's saving method that will save the storage and the data containers permanently via the Permanent Storage GSE service interface.

4.2.5 Access Control

The Access Control GSE is responsible for user authorization, terminal capability profiling, service authentication and user preferences and characteristics profiling. The profile information about user preferences and the current terminal information of the user is stored and managed by the Access Control GSE. This information is utilized by the static adaptation middleware service of the service platform. The Access Control GSE provides a service for editing the user profile information: a profile editor. The Access Control GSE authorizes the user at login and retrieves his current profiles after authorization.

The Access Control GSE contains functionality for trusted services to gain their trusted status through an authentication process. The authentication of services is based on a user inquiry. When a service wants to gain the status of trusted service, having access to private raw context data, it goes through the service authentication process and the user can either accept or deny the service's trusted status request.

The service authentication process is implemented so that first the service trying to gain the trusted status passes over its ServiceCard including the information about service provider, description of the service, service name and the service id to the Access Control GSE. The Access Control GSE then describes the service to the user and inquires if the service should be trusted. The user is able to see the service provider's name and details of the service described in the ServiceCard. If the user trusts the service provider and is willing to publish his private context information to the service, the user accepts the inquiry and a ServicePass with access rights to raw context data is granted and returned to the service. Using the ServicePass the service can access the raw context data via the platform interface.

The sequence diagram in Figure 31 illustrates the role of the Access Control GSE in service authentication process and in updating information in profiles hosted by the Access Control GSE.

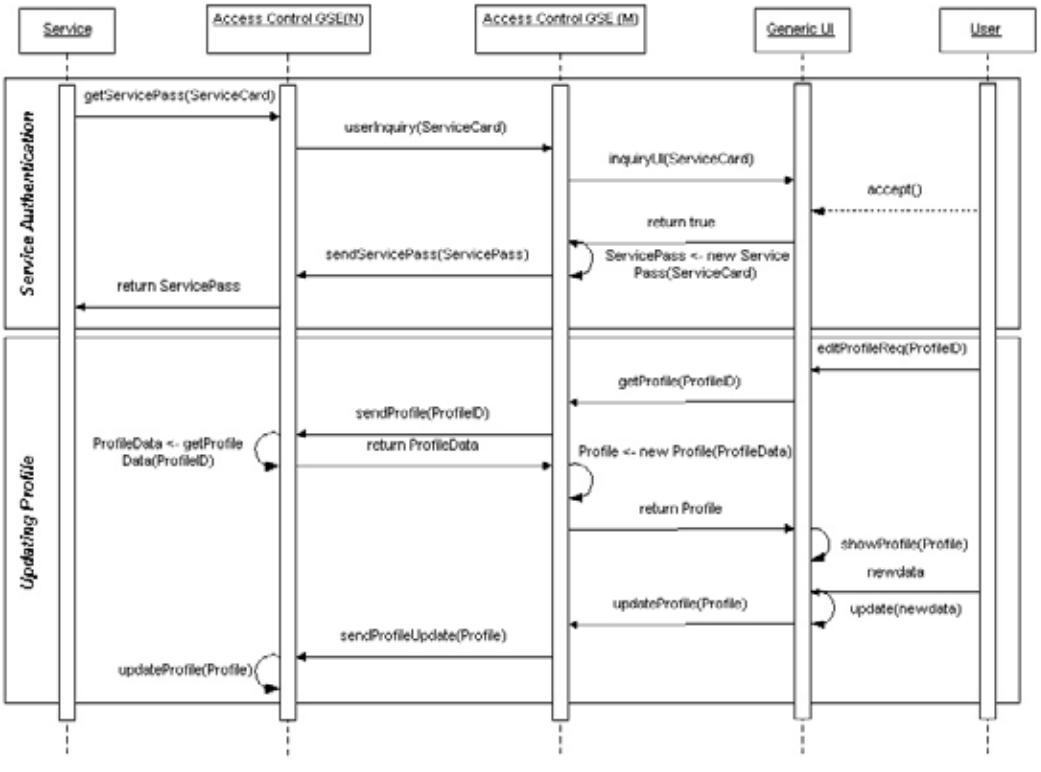


Figure 31. Service authentication and profile information update.

4.2.6 Generic Profiling

The Generic Profiling GSE provides profiling support for the services and other GSEs of the service platform. It implements a generic widely applicable profiling mechanism that is useful and often used by the middleware entities supporting adaptation. Profiling support is also useful for services that need to gather user related information and save it in a structured and easily manageable manner. The services provided by the Generic Profiling GSE include creating, saving, deleting, and retrieving of profiles in the distributed computing environment.

The Generic Profiling GSE provides a couple important classes for profiling; these classes are ProfileSheet and Profile. Profile includes all of the functionality that is needed in the processing of a profile: updating, adding and removing profile elements. The profiling mechanism used supports partially protected profiles, meaning that some of the profile elements are public and some can be protected with a key that has to be known by the requester of the profile in order to have access to the protected profile elements. ProfileSheet is for saving the profile elements of a profile. ProfileSheet can be retrieved from a Profile and new profiles can be created from ProfileSheets. ProfileSheets do not contain any processing functionality, just the profile elements. ProfileSheets can be used in saving and sending profile data as they contain all of the information of a Profile, but are smaller in size. This kind of solution decreases the network load caused by profile sending and updating. The main components and structure of the Generic Profiling GSE is provided in Figure 32.

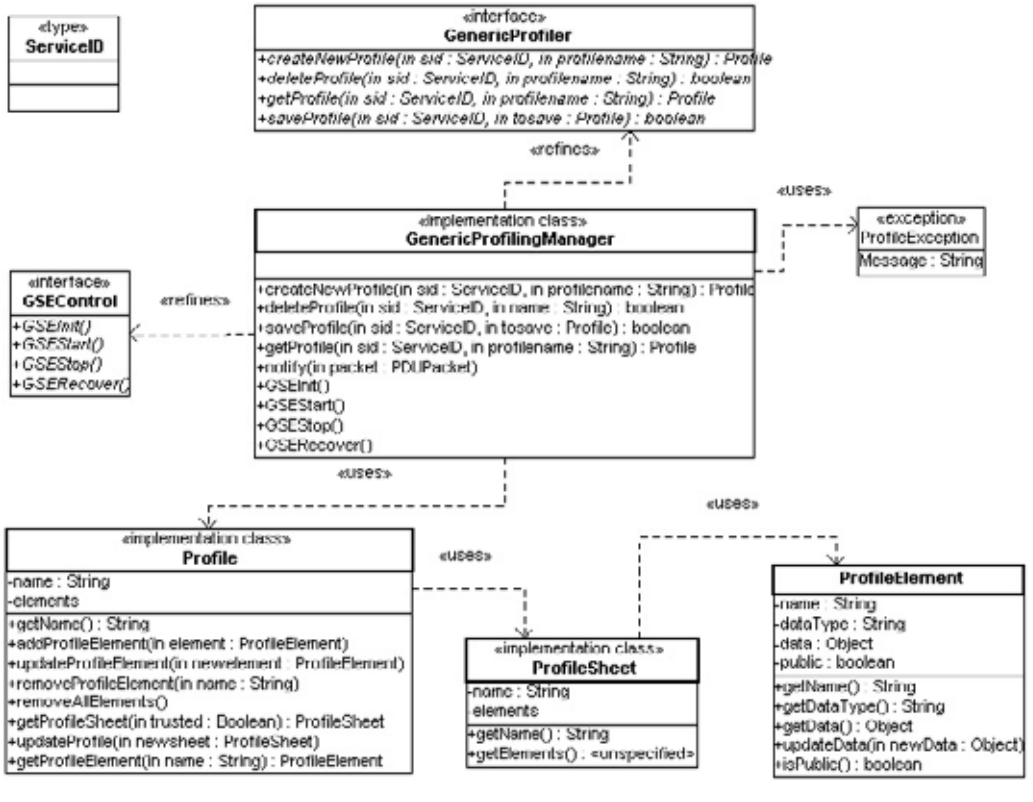


Figure 32. Main components and structure of the Generic Profiling GSE.

4.2.7 Service Delivery

The Service Delivery GSE provides an important function for delivering service instances from the service gateway domain to the mobile terminal domain for execution. When services are installed on the platform, they register their service instances with a platform management component via the platform interface. As the user chooses the service he wants to use, the service instance of a chosen service is delivered to the user using the services provided by the Service Delivery GSE.

The Service Delivery GSE also includes functionality related to service browsing and provides a service browser for the user to choose the services he wants to use. The service browser is provided as a basic platform service and provides access to all of the other services executed on the service platform.

The Service Delivery GSE is a vital part of the distributed service platform that is based on the service gateway-based service architecture. It provides a remote execution environment so that services can be distributed to reside in both of the horizontal domains of the service platform and thereby, the services being too big to be executed entirely at the mobile terminal domain, can be distributed decreasing the workload of the mobile terminal. As an example of Service Delivery GSE operation, an event trace of delivering a service instance from the service gateway domain to the mobile terminal domain in the service browsing process is presented as a sequence diagram in Figure 33.

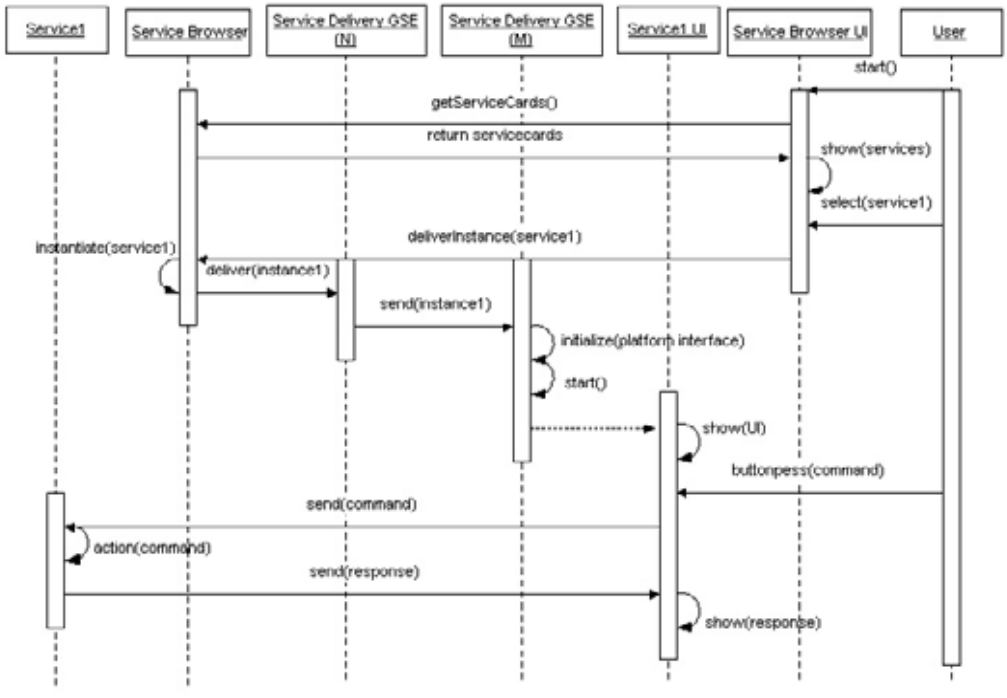


Figure 33. Service instance delivery.

In Figure 33, the role of the Service Delivery GSE is illustrated as part of the service browsing process. First, the user activates the Service Browser UI, which fetches and shows the ServiceCards that describe all of the available services. The user selects a service from the Service Browser UI and the selected service is instantiated and the instance is passed to the Service Delivery GSE (N) in the service gateway domain. The service instance is further delivered to the Service Delivery GSE (M) residing in the mobile terminal domain, where it is initialized and given the local service platform interface of the mobile terminal domain to have access to services provided by the platform. After initialization, the service instance is started by the Service Delivery GSE (M) and the service UI becomes visible to the user. The user can now interact with the service located in the service gateway domain using the service UI in the mobile terminal domain. The interaction takes place without the intervention of the components used in the service browsing process.

4.3 Platform Control and Management

Both of the domains, the service gateway and mobile terminal domain have their own independent platform control components controlling the communications and middleware layer platform components. Additionally, both horizontal domains of the service platform have their own platform management components for handling service registration, unregistration, and platform interface management.

The platform control components are responsible for the overall service platform control including functions like startup, monitoring and shutdown of the platform, and the management of the service interfaces and control interfaces of the platform components. Platform startup is performed in such a way that the communications layer services are started first. Secondly, the GSEs and the middleware services relying on the communications services are started, and finally the platform interface is made available for the services by starting the platform management components in both horizontal domains of the service platform.

The shutdown of the platform is executed in reverse order. Platform control components also handle any error situations so that the platform can recover from individual component crashes by starting up the component.

Every service component of the service platform has an independent service and control interface, the platform control components use the control interfaces of the components to perform their tasks and pass over the service interfaces to the platform management components. The platform management components manage the service interfaces of other platform components so that the platform interface can be built up from the service interfaces of individual components.

5. Prototype Implementation and Testing

In chapter 3, the requirements for a distributed service platform supporting adaptive mobile services were brought up. In chapter 4, an architecture for such a service platform was presented. Now it is time to provide a description of the prototype implementation of the service platform developed. First, the implemented adaptive mobile services with hardware and software configuration of the prototype implementation are presented. The use cases to be presented are used for demonstration, validation and functional testing of the implemented service platform. Evaluation of the prototype implementation is also provided.

5.1 Implemented Services

Two different services were implemented for service platform validation and functional testing, adaptive context service and adaptive video service. The purpose of these services is to validate and test the service platform by utilizing all of the functionalities provided by the middleware services of the service platform. Short descriptions of the implemented adaptive services and the middleware services that they utilize are provided in the following subchapters.

5.1.1 Adaptive Context Service

The adaptive context service automatically controls the NAs in the surroundings of the user based on the context information provided by the service platform. Adaptive context service provides two different service instances for the user to choose: X.10 Control Service and X.10 Context Service.

The "X.10 Control Service" service instance is provided for the remote controlling of the NAs present in the X.10 network. The service instance adapts to terminal capabilities by modifying the UI configuration. If the user is using the service instance via a laptop, the service instance requests textual input for the house code and the device code of the NA to be controlled. This is due to fact that the service detects that user terminal has a full keyboard available using the terminal profile provided by the service platform. If the user is using the

service from a PDA that has no keyboard, but a pen-based touchscreen for input, the UI provides dropdown menus for the house and the device code selection.

The "X.10 Context Service" service instance is provided for controlling the adaptive context service. The service instance and its UI are very simple containing only functionalities for activating and deactivating the adaptive context service. In this case, the platform services are not used for adapting the service instance, but used in the service itself. Adaptive context service utilizes the service profiling mechanism provided by the service platform to define a priori contexts in which some control actions of a NA are bound to. The adaptive context service utilizes the user preference profiling mechanism provided by the platform in a way that if the user preferences comply with sleep profile (all preference variables have value 1) the context service is automatically deactivated.

5.1.2 Adaptive Video Service

The adaptive video service provides a service for streaming video from a video camera to a user's mobile terminal. Here, the platform services are used for monitoring the available network bandwidth and using the provided adaptation services for adapting the service behavior to the current network connection quality. The adaptive video service utilizes the service profiling mechanism for defining a priori context regarding the connection quality.

Two service profiles are made: one for defining conditions where video streaming is reasonable and one for conditions where video streaming is no longer reasonable. When the conditions correspond to the service profile that defines the conditions for video streaming, the video is streamed to the user. If the conditions change so that video sending is no longer reasonable because of the narrow network bandwidth available, the service adapts its behavior. A notification about the reason for the adaptation is shown to the user, and the user can order still images from the video camera using the adapted service. If the network connection gets better, the service starts the video streaming again.

5.2 Configuration

The prototype implementation included both hardware and software entities that together constituted an execution and demonstration environment for the prototype implementation relevant to the validation of the service platform. In the following subchapters, descriptions of the hardware and software configurations of the prototype implementation are provided.

5.2.1 Hardware Configuration

The hardware configuration of the prototype implementation includes two different user terminals, one network server used as a service gateway, a VTT SoapBox used for context sensing, a video camera, two X.10 Appliance Modules and a X.10 Programming Interface. The hardware configuration is illustrated in Figure 34.

The VTT SoapBox (Sensing, Operating and Activating Peripheral Box) [58] is a light, matchbox-sized general-purpose module with a processor, a set of sensors, and wireless and wired data communications. It has been developed at VTT Electronics as a research tool to be utilized in the prototyping of different systems. The research fields where the SoapBox has been utilized include ubiquitous computing, context-awareness, multi-modal and remote user interfaces, etc.

The SoapBox is a configurable platform that can be utilized for many different purposes. It can be connected to a mobile terminal such as PDA, when it can be used for wireless data transfer and for sensing user actions and the surrounding conditions.

The SoapBox that was used in the prototype implementation was connected to both of the user terminals. Its only function in the prototype was to provide context information from its sensors to the mobile terminal and for the service platform. The wireless communication capabilities of the SoapBox were not utilized in the prototype implementation. The SoapBox was connected to the terminals via a serial port and provided the readings of its sensors once a second. This raw context data was received by the mobile terminal resident part of the

service platform and profiled in the environment profile by the Context Sensing middleware service utilizing the Environment Monitoring GSE services. The context data was abstracted and further utilized in the dynamic adaptation process.

The SoapBox used in the prototype implementation had a three-axis acceleration sensor, light sensor, proximity sensor, and temperature sensor. The only sensor information utilized in the prototype was the light information and the proximity information. The data provided by these sensors was utilized by the adaptive context service implemented in the prototype.

X.10 [59] is a home automation technology that enables normal and simple electrical appliances to be controlled over electric wiring. The setup used in the prototype implementation included two X.10 Appliance Modules that are plugged between the cord of a NA and the electrical wall socket. In addition, a X.10 Control/Programming interface that is plugged into the electrical wall socket and to a serial port of a PC was used to control the NAs constituting a X.10 Network. The programming interface was connected to the OSGi Server.

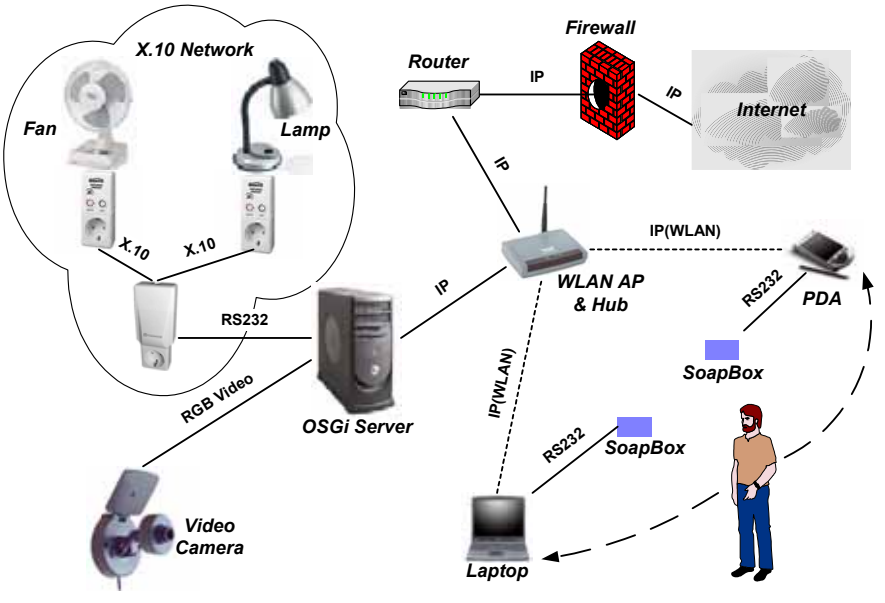


Figure 34. The hardware configuration of the prototype.

As it can be seen from Figure 34, two different user mobile terminals were used in the prototype in order to be able to demonstrate the service instance adaptation to different kinds of terminal capabilities. Both of the terminals also had a wireless network connection in order to demonstrate the effects of variable network bandwidth to the service behavior and adaptation. A VTT SoapBox was connected to both terminals for the sensing of user context information.

As mentioned previously, the natural place for sensing user context information is the PAN, but because of a lack of resources and time, terminal-based context sensing was used in the prototype implementation. A WLAN access point connected to a hub was used for connecting the wired and wireless parts of the network in order to establish one logical interworking network. This network was further connected to a router in order to be able to use the multicast addresses supported by the router. These addresses were needed for video streaming.

One of the services implemented for service platform validation and testing was the adaptive video service, which used a video camera connected to the OSGi server. Another implemented service was the adaptive X.10 context service that uses the X.10 network to control the NAs present in the network. The X.10 network included two X.10 Appliance Modules and on X.10 Programming Interface for network controlling.

5.2.2 Software Configuration

The software configuration of the prototype implementation is divided into two main domains: service gateway domain and mobile terminal domain. The overview of the main software configurations of the domains is illustrated in Figure 35.

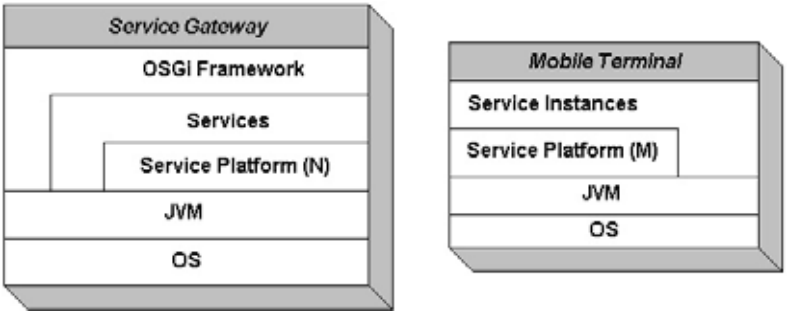


Figure 35. Overview of prototype implementation software configuration.

Figure 35 gives an overview of the main software entities of the prototype implementation. As it can be seen, the only distributed elements in the configuration are the service platform and the services utilizing it. The service platform has two different parts, one residing in the service gateway domain (N) and one residing in the mobile terminal domain (M) and the services and service instances distribution is supported by the service platform.

The OSGi technology that has already been introduced previously was utilized only partly in the prototype implementation. Only the OSGi Framework was used as an application execution framework for the services and the part of the service platform residing in the service gateway domain. The services provided by the OSGi were not utilized. The OSGi implementation that was used was the IBM Service Management Framework (SMF) [60]; the SMF was based on OSGi service platform specification release 2.

The service platform and the proposed architecture are completely independent of OSGi technology and the OSGi Framework was only used to provide a controllable service execution framework, and a clear way of deploying service interfaces. The services of the service platform developed were encapsulated in one service interface that was registered to OSGi and was retrievable for the developed adaptive services that were executed within the OSGi Framework as well.

Java Media Framework (JMF) [61] technology was also used in the prototype implementation to enable the video streaming capability of the adaptive video

service. In other words, the JMF was not an integral part of the service platform, but used only by the services.

The Java Media Framework API enables audio, video and other time-based media to be included in Java applications. The versions of the JMF that were used in the prototype implementation were JMF2.1.1 optimized for windows and JMF2.1.1 cross-platform version that is a pure Java implementation. Windows versions were used in laptop and in the OSGi Server and the cross-platform version was used in the PDA.

5.3 Use Cases

The use cases of the prototype implementation have two different purposes from the validation point of view. The use cases should validate the architecture by proving the solution to be functional in practice, and the use cases should also include adaptation of services, both static and dynamic adaptation, to validate the adaptation solution of the service platform. Some of the use cases demonstrate the usage of the services provided by the service platform itself and some demonstrate the usage of adaptive services implemented for testing and validation of the architecture and service platform functionality.

Use cases are described in the form of short stories in order to illustrate what was really demonstrated in the prototype implementation and how the demonstrations were carried out. Screenshots of the UIs of respective states are provided as well. More detailed descriptions of the use cases are provided in Appendices 2 to 13.

5.3.1 User Authorization

First, the user starts the mobile terminal resident part of the service platform. Once started, a user authorization process takes place. The user is asked to provide his personal user identification and password. When the user identification and password is provided, the mobile terminal resident part of the service platform contacts the part residing in the service gateway domain and sends the user authorization information. The authorization information is

compared to information about authorized users, which is saved previously using the Permanent Storage GSE. If the user is authorized, the communications layer entities establish a connection and the GSE functionality that is distributed is enabled.

At this point, the service platform is initialized and started in both domains and becomes fully functional. When the service platform has been started up, a platform main UI is sent and made visible to the user. Screenshots of the user authorization query and platform main UI are provided in Figure 36.

a) Authorization query UI.

b) Platform main UI.

Figure 36. The user authorization query and platform main UI.

5.3.2 Service Browsing

The platform main UI that is sent to the user contains links to the profile editing UI and a link to the service browser. The user selects the service browser link of the main UI by pressing the Browse Services button, and the service browser is sent and made visible to the user. The service browser contains links to all of the service instances of the services that are using the service platform services and are available to the user to use.

These services in the prototype implementation include adaptive context service and adaptive video service. Links to the service instances of these services that are visible to the user include X.10 Control Service, X.10 Context Service and Video Service. A screenshot of the service browser is provided in Figure 37.

Figure 37. The service browser UI.

A more detailed description of the use case can be found in event trace and its description in Appendices 6 and 7.

5.3.3 Editing Profiles

By selecting the link to the profile editor UI from the platform main UI, the profile editor UI is sent and made visible to the user. From the profile editor UI the user can select either the user preferences profile or user characteristics profile to be edited. A screenshot of the profile editor UI is provided in Figure 38.

The terminal profile is also editable via the profile editor UI; this is due to the fact that in the prototype access for editing, the terminal profile was needed to enable fluent testing. However, terminal profiling in systems other than the prototype should be an automatic process that does not need user intervention or attention. Therefore, the terminal profile editing will not be presented here as a use case.

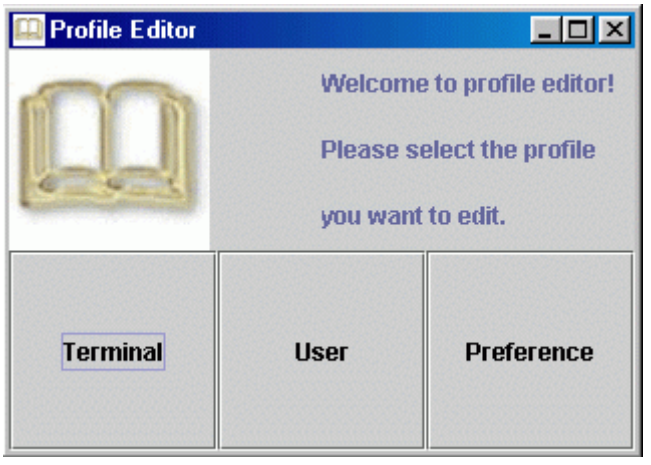


Figure 38. The profile editing UI.

First, the user selects to edit the user preferences profile and an UI for editing the profile appears. A screenshot of a preferences profile editor is provided in Figure 39.

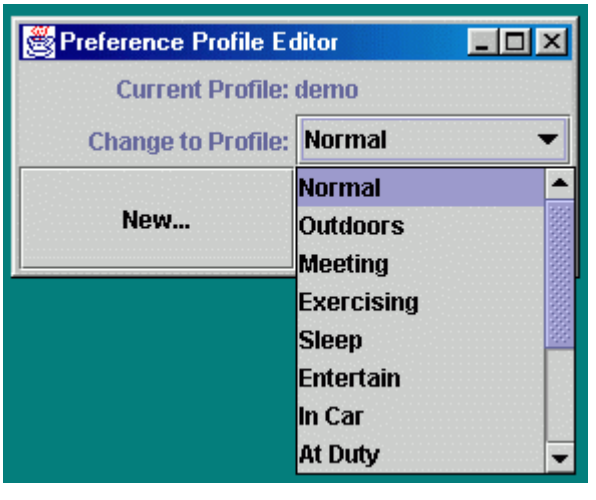


Figure 39. The user preferences profile editor.

The user preferences profile editor includes a set of pre-created profiles for the user to select from. The user can select one of the predefined profiles or if the user wants to define and create a new profile, he can select the "New..." button from the UI and he will get a new UI for creating a new preference profile. The UI for creating new preferences profile is provided in Figure 40.

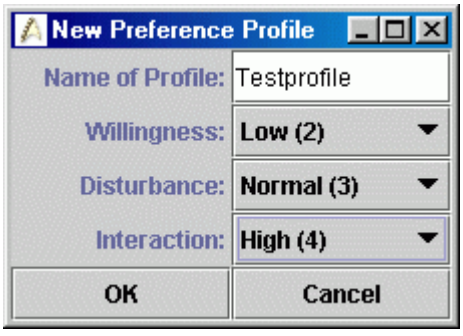


Figure 40. UI for creating new preferences profile.

When the user creates a new preferences profile he has to provide a name and values for all of the preference variables of the profile. When the new profile is ready, it is automatically activated and saved by utilizing the Access Control and Permanent Storage GSEs when the user presses the "OK" button.

Next the user selects the user characteristics profile editor from the profile editing UI. Using the characteristics profile editor, the user can provide the information requested in the profile. The user can also select any piece of information to be guarded by the platform if he does not want to publish the information to not-trusted services. A screenshot of the user characteristics profile editor is provided in Figure 41.

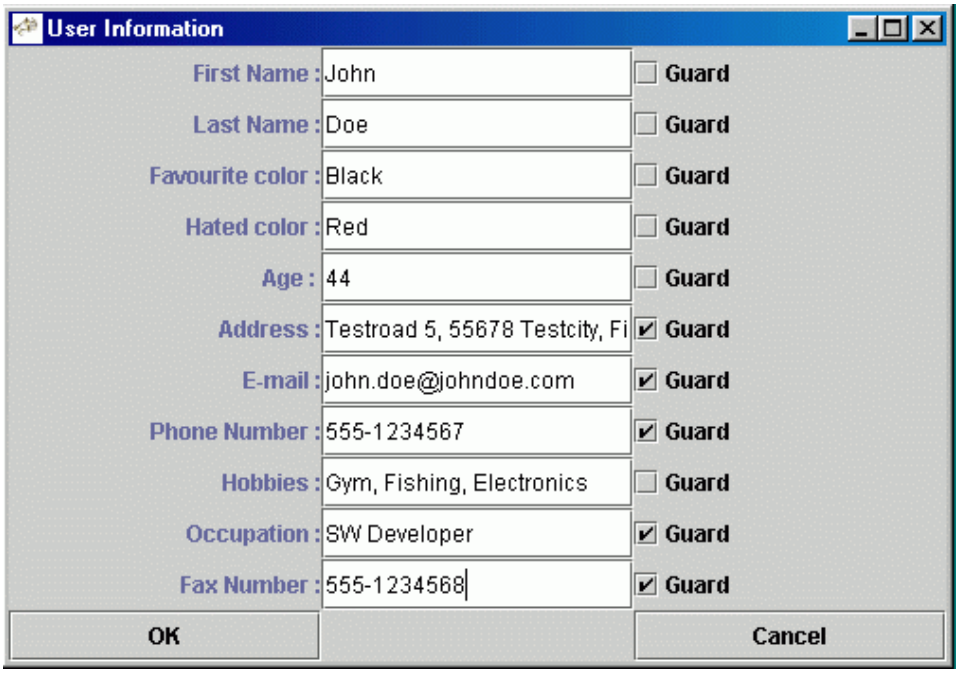


Figure 41. The user characteristics profile editor.

A more detailed description of the use case can be found in event trace and its description in Appendices 8 and 9.

5.3.4 Using Adaptive Context Service

The user selects the X.10 Context Service from the service browser. The control UI for the X.10 Context Service is sent and made visible to the user. Using the control UI, the user can activate or deactivate the adaptive context service. If the user activates the service and the user preferences profile is not set to correspond with the sleep profile, the service is active.

Now the user is using the service via the laptop located on his desk and the nearby desktop light belongs to the X.10 network and is thereby controllable by the adaptive context service. If the lighting conditions in the user's environment are poor, and the user is close enough to the laptop, the desktop light is automatically turned on to provide better lighting for the user. If the user leaves

the laptop, the light is automatically turned of until he returns. The simple control UI for the X.10 Context Service is presented in Figure 42.

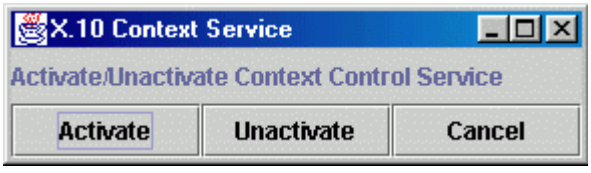


Figure 42. X.10 Context Service control UI.

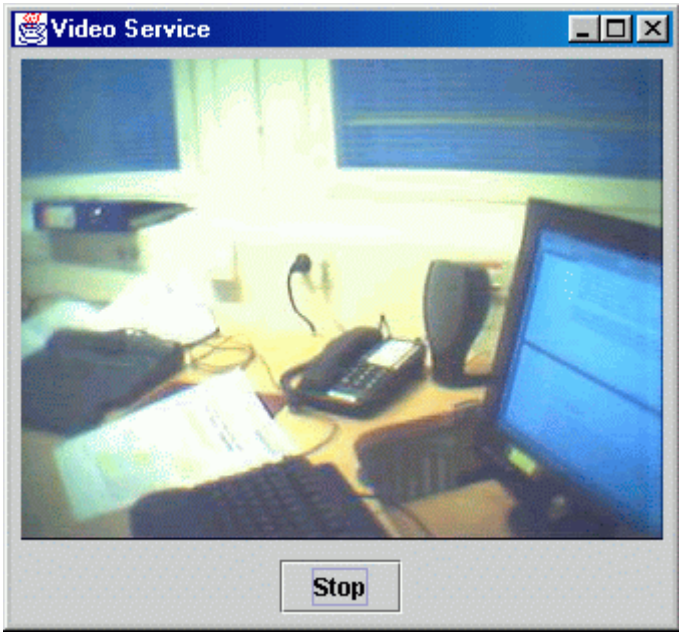
A more detailed description of the use case can be found in event trace and its description in Appendices 10 and 11.

5.3.5 Using Adaptive Video Service

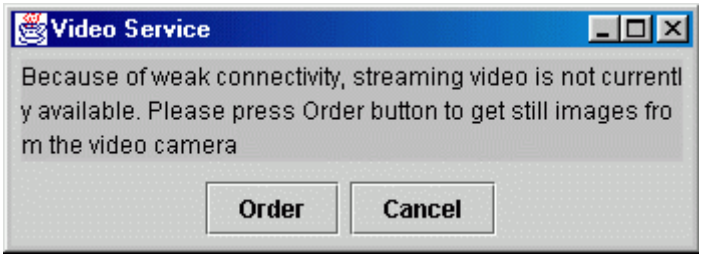
The user selects the video service from the service browser UI and the adaptive video service UI will be sent and made visible to the user. If the connectivity between the mobile terminal and the service gateway is good, the user is able to see the video stream from the video camera.

The adaptive video service UI in this case is presented in Figure 43 a). Now the connectivity changes as the user moves away from the WLAN access point and the available bandwidth decreases. At the point where bandwidth is not sufficient for video streaming, the adaptive video service UI and the operation of the service itself is adapts to new connectivity conditions.

The adapted video service UI can be seen in Figure 43 b). By pressing the Order button in the adapted UI, the user can get still pictures from the video camera.



a) Adaptive video service UI when connectivity is good.



b) Adaptive video service UI when connectivity is poor.

Figure 43. Adaptive video service UIs.

A more detailed description of the use case can be found in event trace and its description in Appendices 12 and 13.

5.4 Evaluation of Prototype

The purpose of the prototype implementation was to validate and test the architecture of the distributed service platform presented in chapter 4. In addition, the prototype implementation is used for the functional testing of the developed service platform.

The evaluation of the prototype implementation is presented in the next subchapters. The evaluation concerns purely the prototype implementation, not the presented service platform architecture, the architecture and some other relevant issues concerning the developed service platform will be discussed in chapter 6.

5.4.1 Context-Awareness

In [12], issues to be addressed when implementing a context-aware system are presented. Answering these questions, where applicable, provides some clarification to the design decisions made in the prototype implementation of this work:

How is context represented internally?

In the prototype implementation, context is represented internally in profiles. The environment profile is the main profile for dynamic context information and in addition, terminal, user preferences, and user characteristics profiles include static context information. At first, when the context data is sensed via the sensors in the user's immediate environment, the data is raw context data that is saved in string format in the environment profile. The context abstractor of the dynamic adaptation service retrieves the environment profile and classifies and abstracts the raw context data according to pre-defined rules. The a priori context definitions made by the services in their service profiles are also internally presented in string format.

How is this (context) information combined with system and application state?

The context information is combined with application state by matching the a priori context definitions that services provide in their service profiles to the current abstracted context data originating from the environment profile, user preferences profile, and user characteristics profile. When the a priori context definitions match the current context, an adaptation request is sent to the service owning the service profile. It is up to the application/service what action is taken when an adaptation request is received.

How frequently does context information have to be consulted?

In the prototype implementation, the raw context information from the sensors is updated once a second in the environment profile, and the context abstraction process is continuously working in the service gateway domain and matching the service profiles against the current context it derives from the environment profile. This is because the dynamic context data is changing all the time and the only way of noticing the changes in this data is to have a continuous process that monitors the dynamic context data. The static context data is consulted at the beginning of the adaptation process for personalization purposes and it can also be consulted by the services when needed.

What are the minimal services an environment needs to provide to make context-awareness feasible?

The minimal services needed in the prototype implementation to make context-awareness feasible are the static and dynamic adaptation middleware, Event Notification GSE, Generic Profiling GSE, Environment Monitoring GSE, Access Control GSE, Service Delivery GSE, and the Permanent Storage GSE. In other words, everything that was implemented is needed to make context-awareness feasible. In addition, concerning context-awareness, the high-level context information is missing from the system, which is a conscious decision to keep the service platform lightweight.

5.4.2 Scale of Prototype

All of the components presented in the service platform architecture and the required functionality identified in chapter 3 were implemented in the prototype implementation. However, because of the lack of time and resources, a few shortcuts were made, but in such a way that the functionality of all of the components of the service platform remained the same and the architecture presented was not neglected. The shortcuts made were inside the individual components and were not visible outside of the components.

The scale of the prototype implementation was big enough to successfully validate the presented architecture and to test all of the required functionality of the service platform. Services were implemented on top of the service platform to test the service platform functionality related to adaptation support and service delivery, and to demonstrate the outlook and capabilities of these kind of adaptive services.

5.4.3 Validation

The validation of the distributed service platform and its architecture presented in this work was successfully achieved by the prototype implementation. In the prototype implementation, all of the platform components presented in the architecture were implemented, and to validate the correct functionality of the platform, a couple of adaptive services were implemented to utilize the services and were executed on top of the service platform.

The validation was seen as successful because the implemented services were functional and the adaptation functionality of these services required very little effort when utilizing the service platform services in implementing their adaptation functionality. However, the service platform architecture presented is designed to be scalable and to include functionalities other than adaptation-related as well, but only adaptation-related components were presented in the architecture that was validated in the prototype implementation.

5.4.4 Testing

The testing of the prototype implementation was carried out in many different phases alongside the development of the prototype. Testing the functionality that the components provide tested the functionality and correctness of the components after implementation with the black-box testing approach. Once the tests were successfully passed and all faults were corrected, the work continued with the implementation of the next component.

The first components that were implemented and tested were the Communications components that implement the communications abstraction for the platform. After this, the implementation and testing of the GSEs was carried out. GSEs utilize the services of the communications components and the services of other GSEs so at this point the first integration tests took place.

After all of the GSEs were implemented, testing all of the GSEs simultaneously through their service interfaces performed an integration test of the GSEs. Once the GSE integration test was successful, the implementation of the adaptation, platform management, and context sensing middleware components was started. All of these components utilize the GSEs, so again after all of the middleware components were implemented, an integration test for the whole platform was carried out.

Implementing adaptive services that utilize the services provided by the service platform and running the defined use cases successfully resulted in the final testing of the service platform functionality.

5.4.5 Shortcomings

One identified weakness of the prototype implementation was the communication solution implementation below the communication abstraction. The communication solution was not in the main scope of the work and minimal resources were used for designing and implementing it. However, the communication solution was vital to be able to demonstrate and test the functionality of the service platform.

The shortcomings of the communication solution were related to the TCP protocol behavior, on which the solution was based, in a wireless environment. Sudden disconnections caused problems in the communications solution. These kinds of sudden disconnections should not be considered as exceptions, but as normal operating modes in future wireless systems. Therefore, more development on the communications solution would have been required to make the prototype implementation invulnerable to disconnections.

5.4.6 Strengths

A clear strength of the prototype implementation was that the main goal of supporting the adaptation of mobile services came through fine. The services implemented to utilize the adaptation support functionality provided by the service platform worked fine and the adaptation functionality was fluent. Also, the main components of the architecture, like GSEs and middleware and control components, worked as expected.

Another strength of the prototype implementation was that its operation was quite stable, except for the shortcomings of the communications solution in a wireless environment.

The prototype implementation also showed the benefits of the service platform supported adaptation functionality and the application-aware adaptation approach. The amount of code making implemented services adaptive was very little, and the usage of a platform ensured that the functionality was fluent and logical both in the implementation phase and when running the adaptive services. The service platform clearly decreased the workload required for the services to be adaptive.

Finally, all of the functionalities that were required from the service platform were implemented and tested in the prototype and came out working fine. For example, there were doubts if the service profiling approach for a priori context definitions would be the way to go, but it turned out to be working well and served its purpose.

One of the objectives set for the work was to have a minimum size prototype implementation after taking the practical design approach. This objective was well met as the prototype implementation of the service platform, Service Platform (N) and Service Platform (M) in Figure 35, resulted in an overall size of less than 360 KB. Of the overall size, approximately one third is allocated to the mobile terminal domain being the resource-constrained domain on the service platform. The surprisingly small prototype implementation proved the benefits of the practical design approach, which enables the developed service platform to be introduced to very resource-constrained mobile computing environments.

In summary, it can be said that regarding the main scope of the work, the prototype implementation was properly scaled and successfully served its purpose in validating the proposed architecture and the adaptation functionality of the service platform presented.

6. Discussion

Related research and technologies related to the domain area of this work were presented in chapter 2. Now it is time to compare and discuss the work completed taking into consideration existing research, and highlight and discuss technologies that seem promising for future implementations and research on service platforms supporting adaptability. Existing research and proposals are especially found in architecture and middleware layer issues.

6.1 Generic Service Elements

Valuable experience about GSEs was gained during the work, especially concerning their structure and the overall service platform architecture utilizing GSEs. It should be noted that the definitions of GSEs in WWRF are still at the conceptual level and the GSEs in this work have the role of implemented middleware layer entities as well as conceptual level entities. Therefore, the abstraction level of the definitions is essentially different. WWRF defines visions and in this work the GSEs have been identified, designed, and implemented in practice. Hopefully the work presented here provides an alternative and complementary approach to the work regarding defining GSEs in WWRF.

As one can see from [2], the GSEs identified by WWRF are partly different from the ones identified in this work. Both have identified environment monitoring and event notification, but all of the other elements are different. One of the goals set for this work was to identify the needed GSE with a practical design approach and, through a prototype implementation, to gain a better understanding of GSEs and their structure and role as middleware services. This goal was achieved, but because the approach of this work was to find a solution with a minimum set of GSEs for a lightweight service platform supporting adaptive distributed mobile services, the identified set of GSEs has differences compared to the ones identified by the WWRF. However, the GSEs in this work are prototyped and validated to be functional. Even though the names of the identified GSEs are different, it can be noticed that the same functionalities are present at least to some extent. A comparison of the GSEs identified,

implemented and validated in this work against the descriptions of WWRF GSEs [2] is provided.

The Event Notification GSE identified in this work is based on the same principles that have been identified in [2]. In WWRF, event notification is used for notifying changes in the context observed by the environment monitoring GSE. In this work, a more generic approach is taken by providing the services of the Event Notification GSE to the services as well as to other GSEs. In this way, the entities of the system can utilize the event notification service for other purposes like the adaptation of service instances or communication between service instance and service. The Event Notification GSE in this work is based on the source-initiated communication model, where the recipients of the events register their interests directly with the component that is capable of publishing the events of interest. The same mechanism is used in the Java event system. The Event Notification GSE of this work utilizes the services of the communication abstraction that provides asynchronous message-based communication and platform-specific PDUs. The Event Notification GSE is therefore immune to disconnections.

The basic operating principle of the Environment Monitoring GSE in WWRF is the same as in this work. It monitors the changes in environmental conditions and provides the information about the surroundings to other software components. In WWRF, the Environment Monitoring GSE is defined to utilize the Event Notification GSE for informing of changes in environmental conditions based on a priori rules. In this work, the Environment Monitoring GSE is based on the profiling of dynamic context data and the changes are delivered to services via the service profiling and context abstraction methods that were presented in chapter 3. The basic operation is still the same, in this work the events are received from the service profiles that include the a priori context definitions that are in the interest of the services, and in the WWRF model these interests are directly expressed for the Environment Monitoring GSE.

The Distributed Application Framework GSE identified by WWRF is used for combining existing services to provide new ones. It is also defined to support the autoconfiguration of services and service discovery. The absolute need for this kind of GSE regarding service adaptability was not identified in this work.

However, it was notified that the kind of functionality would be useful in general for a distributed service platform in a mobile computing environment where the dynamic configuration of services is required.

The Perception Service GSE for the automatic collection of values for learning purposes and the Modeling Service GSE for building a model through learning are also identified by WWRF. In this work, these kinds of services were not seen as compulsory to support the adaptation of mobile services, as the design approach taken was very practical to end up with a lightweight solution. However, these kinds of services may supplement the system services for context-awareness.

The Mobile Distributed Information Base GSE identified by WWRF can be compared to the Permanent Storage GSE identified in this work. However, the main purpose of the Permanent Storage GSE is to provide fast, reliable, and distributed services for permanently saving small amounts of data like parameters or profiles, for example. The Mobile Distributed Information Base is defined in a manner that it could include the services provided by the Permanent Storage GSE.

The Ontology Service GSE identified by WWRF is used for representation, manipulation, and storage of ontologies of varying detail. The Semantic Matching Engine GSE is identified as responsible for using the ontological information to match and reason over instances of ontological knowledge, such as profiles. These services can also be classified as services supporting the context-awareness of the system and were not seen as compulsory from the pure adaptation point of view. However, the adaptation functionality is a central part of a context-aware system and these kinds of services are certainly needed, for example, in utilizing AI methods for deriving higher lever context information from low-level context information.

6.2 Architecture Review

In chapter 2, two interesting architectures from the field of pervasive computing were presented - Project Aura and the Gaia metaoperating system. Even if these architectures were designed for active spaces, there are some similarities with

the architecture provided in this work. These similarities are found in the provided functionality, as the binding factor of the three architectures is the need for adaptation.

In addition, there are characteristics in Aura and Gaia that could bring additional value to the service platform presented in this work. The comparison of the architecture presented in this work to Aura and Gaia architectures, and possible complementary characteristics of the architectures are discussed in the next subchapters.

6.2.1 Comparison to Aura

In Aura, the lowest layer of the architecture is the intelligent networking layer, which supports network weather monitoring and network proactivity. This kind of networking layer would clearly be useful for adaptive services that would be able to anticipate the network's state and adapt their behavior from this information. Aura's architecture also includes the Coda [52] file system, whose goal is to offer clients continuous access to data even in the presence of server and network failures. The adaptation strategy adopted in Coda is application-transparent adaptation. The principles of the Coda file system provide a good starting point for distributed file systems in mobile computing platforms as well.

The adaptation approach of Coda was, however, seen as inadequate for some situations and a component supporting application-aware adaptation called Odyssey [53] was designed. Odyssey is based on extending UNIX with a small but powerful set of extensions for mobile computing. The functionality provided by Odyssey can be compared to adaptation functionality provided by the service platform presented in this work. The fundamental idea in both systems is to provide relevant information for applications in order to enable application-aware adaptation.

Aura also has a component implementing remote execution environment for applications called Spectra. In this work, the Service Delivery GSE and the distributed nature of the service platform enable the remote execution of service instances. The task layer of Aura, also called Prism, performs user task monitoring, user intent monitoring and high-level proactivity. This kind of layer

is missing from the architecture presented in this work and is worth considering as an approach for deriving the high-level context information and achieving proactivity in the service platform. However, the exclusion of high-level context information usage in the service platform developed was a conscious decision that was made due to the practical design approach targeting lightweight implementations.

6.2.2 Comparison to Gaia

The Gaia metaoperating system architecture also has some similarities with the services provided by the service platform of this work. In particular, the Gaia kernel components: Event Manager, Context Service, Presence Service, Space Repository, and Context File System can easily be seen as "relatives" of the GSEs presented in this work.

The Gaia Event Manager uses a technique referred to as event channels. Event channels forward supplier's events to the event consumers registered with the channel. The event manager has a single entry point and one or more event channel factories. In Gaia, applications can also define their own event channels for application state change monitoring. The functionality of the event manager of the Gaia kernel is comparable with the Event Notification GSE in the architecture presented in this work. Event channeling is not used in the service platform presented, but would clearly be a possible method for separating the platform and service-specific event delivery and therefore adding security to the event delivery solution.

The Gaia Context Service lets applications query and register for context information, which helps applications adapt to their environment. This is also the functionality that the Environment Monitoring GSE, together with adaptation middleware components, provides in the service platform of this work. The context model of Gaia is based on first-order logic and Boolean algebra, which enables simple rule writing to describe context information. The same approach is also used in the service profiling mechanism used in the adaptation of mobile services in this work.

The Gaia Presence Service and Space Repository are related to the resource management of active spaces. Therefore, they have no comparable entities in the service platform architecture presented in this work.

The Gaia Context File System enables the use of context information to distinguish meaningful information from irrelevant information. The service platform presented in this work does not provide a context sensitive file system, but a service for permanently saving data implemented by the Permanent Storage GSE.

Aura and Gaia architectures are from the field of pervasive computing, designed for active spaces, and take advantage of context-awareness and adaptation. Therefore, as seen in this chapter, there are clear similarities in the services that these architectures provide with the services provided by the service platform of this work. This chapter has introduced a few targets for development and these among others are discussed next.

6.3 Targets for Development

Now that we have presented the architecture and prototype implementation of a distributed service platform supporting the adaptability of mobile services, it is time to look back and raise some targets for development regarding the developed service platform. Also, future research issues regarding the developed service platform and its architecture are to be raised.

6.3.1 Representation of Context Information

Further developments are needed to enhance the scalability of ontology and the semantics that are used for representing context information. The approach taken in the prototype implementation was to base ontology and semantics on string representation of context data. However, another approach would be to utilize more flexible technologies like XML for context information representation. In this way the ontology and semantics would become more fluently extendable.

On the other hand, the use of XML introduces XML parsing that is quite a resource-consuming process and can produce an overwhelming workload for resource-constrained mobile terminals. So compromises have to be made when choosing the methods for context information representation in mobile computing systems.

6.3.2 High-Level Context Sensing

There is a clear need for further research on interpreting higher-level context information from the lower-level context information. This is an issue that was not researched extensively because the focus of the work was in the adaptation and not context-awareness in general.

However, for deriving high-level context information, the methods of AI could be used. These methods include use of, for example, Bayesian networks and classifiers as can be seen in [27]. Rule-based systems are also one possible approach to derive high-level context information [26]. Additionally, introducing a new architecture layer: a task layer that is found in the Aura architecture for deriving high-level context information and providing proactivity for the system is an approach worth considering. Future research regarding the high-level context interpretation is needed and especially from the practical design approach point of view that aims to keep the system lightweight and small in size.

6.3.3 Event Notification Enhancements

The Event Notification GSE of this work was designed to be as simple as possible to keep the workload small. However, further enhancements are required regarding the security of the event delivery mechanism. In the prototype implementation, all of the events were visible to all of the clients of the Event notification GSE. This should not be the case because of security reasons. The event delivery of platform-related events should be invisible to the services. Also, the event delivery between a service and a service instance should not be visible to other services.

One possible approach to achieve this is to use event channels that are also used in the Gaia metaoperating system. By using event channels the delivery of platform-specific events could be separated from service-specific events. And each service could also have its own secure event channel in use not visible to other services.

6.3.4 Communications Solution Enhancements

The main purpose of the communications solution of this work was to implement a communication abstraction separating the platform from any specific transport layer protocol by providing platform-specific PDUs and services for communicating between the different domains of the service platform. The communications solution also provided services for deriving the available network bandwidth and latency.

However, lots of enhancements valuable for adaptive applications can be made. For example, the Aura architecture includes the Intelligent Networking layer that has the capability of providing information about current network conditions and also provides network proactivity anticipating the network's state in the near future. In particular, network proactivity would be a valuable enhancement to the communications solution made in this work.

6.3.5 Technologies

In chapter 2, a set of technologies from the domain of this work was presented. However, the requirement analysis and architectural design took place in a technology-independent way focusing on needed service platform functionality. However, there are technologies that could have been used in the prototype implementation phase, these were not used, however, as the prototype implementation was made for concept and architecture validation and demonstration purposes only.

For example, it would be worth studying to see if CC/PP Profiles can be used in profiling in general and also for the terminal profiling carried out on the service platform. Also, the usage of SOAP would be one possibility to implement the

communications components of the platform. Even the taken approach for communications abstraction implementation was customized and message-based middleware, the RPC-based solutions could have been used as well.

6.3.6 Service Platform Adaptability

A lightweight distributed service platform supporting the adaptation of mobile services has been presented in this work. However, during the work it was noticed that future research is needed to make the service platform and its components adaptive as well. So further research is required regarding the adaptability of the GSEs and other platform components.

From the architecture point of view, one possible way to implement the adaptation of platform components could be to control the adaptation via the control interfaces of the components. The work completed provides a good starting point for research on service platform adaptability.

6.4 Achievement of Objectives

This chapter discusses how the objectives that were set for the work in chapter 1.3 have been reached. The first objective was to identify the basic requirements and perform a requirement analysis for future mobile services and service provisioning in distributed and heterogeneous computing environments. In particular, the interest was in identifying the requirements that future mobile computing environments and users generate regarding the adaptation of mobile services.

This objective was successfully reached via the wide literature review that concluded the adaptation, personalization, and context-awareness are the basic requirements for future mobile services. Additionally, it was noticed that these requirements and terms are partly overlapping in a way that adaptation is an integral part of both personalization and context-aware functionality. Therefore, the personalization and context-awareness, together with the heterogeneous distributed computing environment, set the requirements for adaptation functionality that is needed in future mobile services.

From the requirements identified, an adaptation process that consisted of static and dynamic adaptation was designed and presented. The static adaptation process can support service personalization and the dynamic adaptation process enables context-aware applications to monitor context continuously. The overall adaptation process was taken as the starting point for designing the internal functionality of the service platform components.

Another objective was to develop a distributed service platform providing adaptation support and alleviating the design and implementation of adaptive mobile services. The service platform should be based on the requirements gathered from the literature review.

This objective was also successfully reached via the architectural design and prototype implementation of the service platform. The GSE concept covered in the literature review was taken as the starting point for the architectural design of the service platform. The GSE concept was taken further by defining GSEs in more detail and designing, implementing and testing the GSEs as basic building blocks of the service platform. The service platform architecture supporting the adaptation of mobile services, and alleviating the service development to meet all the requirements identified was presented and validated in a prototype implementation.

An additional objective, aside from the actual service platform functionality, was to use a practical design approach that should result in a lightweight service platform that could be introduced to resource-constrained mobile computing environments.

This objective was also successfully reached as the prototype implementation of the service platform, service gateway and mobile terminal resident parts together, resulted in an overall size of less than 360 KB, of which approximately one third is allocated to the mobile terminal. This proved that the service platform developed could be introduced to very resource-constrained mobile computing environments. In summary, it can be said that all the objectives set for the work were met satisfactorily.

7. Conclusions

In this work, a distributed service platform for adaptive mobile services has been the main interest of research. At first, an introduction was given about the research activities that have lead to the development of this kind of service platform for a service gateway -based service architecture. The service gateway-based systems were selected for the research framework because in preceding research they have been identified as one possible solution to future service management issues [3].

An insight to the domain of the work and related research was given via the wide literature review carried out to perform a requirement analysis for future mobile services and service platforms supporting service adaptation. The trend of providing services individually and personalized way instead of mass-produced services was clearly emphasized in the literature review. As a result of the literature review, the three basic requirements for future mobile services adaptation, personalization, and context-awareness were identified.

It was also identified that the definitions and usage of these terms are partly overlapping and that the features of personalization and context-awareness utilize adaptation in their overall operation. Therefore, the features of personalization and context-awareness together with the computing environment set the requirements for the adaptation.

Further requirements for the adaptation support functionality were gathered from the literature with a practical design approach targeting a lightweight service platform. The basic requirements and operation principles for a lightweight service platform supporting the adaptation of mobile services were presented. An adaptation process consisting of static and dynamic adaptation was presented. Also, profiling mechanisms and profiles to describe user preferences, user characteristics, terminal capabilities, dynamic context, and a priori context definitions were outlined and designed. Perhaps the two most vital profiling techniques presented were the user preference profiling that enables the representation of user preferences towards the service domain and the service profiling that enables the services to express their interests regarding the context.

Having gathered the requirements for the lightweight service platform supporting the adaptation of mobile services in a service gateway-based service architecture, the architectural design of the service platform took place. The GSE concept covered in the literature review was taken as the starting point for the architectural design. Defining the GSE in a more detailed yet universally applicable way in the work and in [1], advanced the state of the art regarding GSEs. The work also provided an alternative and more practical approach to GSEs by identifying, designing, implementing, and testing six GSEs needed to support the adaptation of mobile services in service gateway-based service architecture. The architecture for a lightweight distributed service platform that is based on utilizing GSEs was also provided.

The designed architecture was validated by a prototype implementation that proved the advantages of the practical design approach as the prototype implementation resulted in an overall size of less than 360 KB, of which approximately one third is allocated to the mobile terminal domain. This proves that the developed service platform and its architecture can be introduced to very resource-constrained mobile computing environments, which was one of the goals set for the work.

The fundamental research problem that this work focused on was how to support future mobile services to meet the manifold requirements set for them. The requirements are set by the raised user expectations of personalized and context-aware services and by the distributed heterogeneous computing environments. This work shows that adaptation of mobile services can be supported providing the support as middleware services that are part of a lightweight distributed service platform that hides the heterogeneity and distributed nature of the underlying computing environment. This kind of service platform considerably alleviates the development of future mobile services to meet the requirements of personalization, adaptation, and context-awareness.

References

[1] Pakkala, D. & Latvakoski, J. (2004). Distributed Service Platform for Adaptive Mobile Services, to be submitted to The 2004 International Conference on Pervasive Computing and Communications (PCC-04), June 21–24, Las Vegas, Nevada, USA.

[2] Raatikainen, K., Hohl, F., Latvakoski, J., Lindholm, T. & Tarkoma, S. (2003). Generic Service Elements for Adaptive Applications, WWRF WG2 White Paper, p. 12.

[3] Pakkala, D., Välitalo, P. & Latvakoski, J. (2003). User Centric Peer-to-Peer Service Environment for Interaction with Networked Appliances. In: Proceedings on 23rd International Conference on Distributed Computing Systems Workshops, 19–22 May, Providence, Rhode Island, USA. Pp. 242–247. ISBN 0-7695-1921-0.

[4] Pakkala, D., Välitalo, P., Pääkkönen, P. & Latvakoski, J. (2003). User-Centric Peer-to-Peer Service Environment for Interaction with Networked Appliances. ERCIM News, July, No. 54, pp. 12–13.

[5] Latvakoski, J., Pakkala, D. & Pääkkönen, P. (2003). An Interaction based Approach to Mobile System Construction. In: Proceedings on 3rd Workshop on Applications and Services in Wireless Networks, 2–4 July, Bern, Switzerland. Pp. 243–252. ISBN 3-9522719-0-X.

[6] Latvakoski, J., Pääkkönen, P., Pakkala, D., Tikkala, A., Remes, J. & Välitalo, P. (2002). Interaction of All IP Mobile Internet Devices with Networked Appliances in Residential Home. In: Proceedings on 22nd International Conference on Distributed Computing Systems Workshops, 2–5 July, Vienna, Austria. Pp. 717–722. ISBN 0-7695-1588-6.

[7] Wireless World Research Forum Working Group 2 (19.11.2003). Home Page, URL: http://www.comtec.e-technik.unikassel.de/content/conference/wwrf-wg2/

[8] Weiser, M. (1991). The Computer for the 21st Century. Scientific American, September, vol. 265, issue 3, pp. 94–104.

[9] Weiser, M. (1993). Some Computer Science Issues in Ubiquitous Computing. Communications of the ACM, vol. 36, pp. 75–84.

[10] Abowd, G. D. & Mynatt, E. D. (2000). Charting Past, Present, and Future Research in Ubiquitous Computing. ACM Transactions on Computer-Human Interaction, vol. 7, pp. 29–58.

[11] Bagrodia, R., Chu, W. W., Kleinrock, L. & Popek, G. (1995). Vision, Issues, and Architecture for Nomadic Computing. IEEE Personal Communications, vol. 2, pp. 14–27.

[12] Satyanarayanan, M. (2001). Pervasive Computing: Vision and Challenges. IEEE Personal Communications, vol. 8, pp. 10–17.

[13] Garlan, D., Siewiorek, D. P., Smailagic, A. & Steenkiste, P. (2002). Project Aura: Toward Distraction-Free Pervasive Computing. IEEE Pervasive Computing, vol. 1, pp. 22–31.

[14] Roman, M., Hess, C., Cerquiera, R., Renganathan, A., Campell, R. H. & Nahrstedt K. (2002). A Middleware Infrastructure for Active Spaces. IEEE Pervasive Computing, vol. 1, pp. 74–83.

[15] Satyanarayanan, M. (1996). Fundamental Challenges in Mobile Computing. In: Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing, May, Philadelphia, Pennsylvania, USA, ACM Press, New York. Pp. 1–7.

[16] Meer, S. v. d. & Arbanowski, S. (2001). From Unified Messaging Towards I-centric, Services for the Virtual Home Environment. In: Proceedings of IEEE Intelligent Network Workshop (IN 2001), Boston, MA, USA. Pp. 489–492. ISBN 0-7695-1198-8.

[17] Steglich, S. & Popescu-Zeletin, R. (2001). Towards I-centric User Interaction. In: IEEE International Conference on Multimedia and Expo, Tokyo, Japan, 2001.

[18] Biemans, M., Kranenburg, H. v. & Lankhorst, M. M. (2001). User Evaluations to Guide the Design of an Extended Personal Service Environment for Mobile Services. In: Proceedings of IEEE Fifth International Symposium on Wearable Computers, Zurich, Switzerland. Pp. 94–101.

[19] Campadello, S. (2003). Middleware Infrastructure for Distributed Mobile Applications. PhD Thesis, Department of Computer Science, University of Helsinki. Helsinki, Finland. 155 p.

[20] Lauer, H. et al. (2003). Personalization, Wireless World Research Forum Working Group 2 White Paper. 36 p.

[21] Arbanowski, S. & Meer, S. v. d. (1999). Service Personalization for Unified Messaging Systems. In: Proceedings of 4th IEEE International Symposium on Computers and Communications, ISCC'99, Red Sea, Egypt. Pp. 156–163.

[22] Dey, A. K. & Abowd, G. D. (2000). Towards a Better Understanding of Context and Context-Awareness. In: Proceedings of Computer-Human Interaction 2000 (CHI 2000), Workshop on The What, Who, Where, When and How of Context-Awareness, April 3, Hague, Netherlands. 12 p.

[23] Korhonen, I., Pärkkä, J. & Gils, M. V. (2003). Health Monitoring in the Home of the Future. IEEE Engineering in Medicine and Biology Magazine, vol. 22, issue 3, May/June, pp. 66–73.

[24] Lee, S.-W. & Mase, K. (2002). Activity and Location Recognition Using Wearable Sensors. IEEE Pervasive Computing, vol. 1, issue 3, pp. 24–32.

[25] Ranganathan, A. & Lei, H. (2003). Context-Aware Communication. IEEE Computer, vol. 36, issue 4, pp. 90–92.

[26] Chen, G. & Kotz, D. (2001). A Survey of Context-Aware Mobile Computing Research. Dartmouth College Department of Computer Science, Dartmouth. Technical Report TR2000-381, 16 p.

[27] Korpipää, P., Mäntyjärvi, J., Kela, J., Keränen, H. & Malm, E.-J. (2003). Managing Context Information in Mobile Devices. IEEE Pervasive Computing, vol. 2, issue 3, July–September, pp. 42–51.

[28] Kaasinen, E. (2003). User Needs for Location-aware Mobile Services. Personal and Ubiquitous Computing, issue 7, pp. 70–79.

[29] Davies, N., Cheverst, K., Mitchell, K. & Efrat, A. (2001). Using and Determining Location in a Context-Sensitive Tour Guide. IEEE Computer, vol. 34, issue 8, pp. 35–41.

[30] Intille, S. S. (2002). Designing a Home of the Future. IEEE Pervasive Computing, vol. 1, issue 2, April–June, pp. 76–82.

[31] Göker, A. & Myrhaug, H. (2002). User Context and Personalisation. In: Workshop on Case Based Reasoning and Personalization, September 4th, Aberdeen, Scotland. 7 p.

[32] Colouris, G., Dollimore, J. & Kindberg, T. (1994). Distributed Systems Concepts and Design. 2nd edition, Addison-Wesley, London. 615 p.

[33] Vinoski, S. (2002). Where is Middleware? IEEE Internet Computing, vol. 6, issue 2, March–April, pp. 83–85.

[34] Raatikainen, K. (19.11.2003). Middleware and Protocols for the Future Mobile Internet, Nokia Featured Arcticle, http://www.nokia.com/nokia/0,,41256,00.html

[35] Zou, H., Wang, B. & Mao, W. (1999). User Profiles for Access to Telecom Services. In: Proceedings of IEEE Fifth Asia-Pacific Conference on Communications, vol. 2. Beijing, China. Pp. 1114–1117.

[36] Object Management Group, Inc. (19.11.2003). Common Object Request Broker Architecture, URL: http://www.corba.org

[37] Object Management Group, Inc. (19.11.2003). Home Page, URL: http://www.omg.org

[38] Sun Microsystems, Inc. (19.11.2003). Java Remote Method Invocation, URL: http://java.sun.com/products/jdk/rmi/

[39] Wall, T. & Cahill, V. (2001). Mobile RMI: supporting remote access to Java server objects on mobile hosts. In: 3rd IEEE International Symposium on Distributed Objects and Applications, Rome, Italy. Pp. 41–51.

[40] World Wide Web Consortium. (19.11.2003). Extensible Markup Language, URL: http://www.w3.org/XML/

[41] World Wide Web Consortium. (19.11.2003). Simple Object Access Protocol, URL: http://www.w3.org/TR/SOAP/

[42] World Wide Web Consortium. (19.11.2003). Composite Capability / Preference Profiles, URL: http://www.w3.org/TR/NOTE-CCPP/

[43] World Wide Web Consortium. (19.11.2003). Resource Description Framework, URL: http://www.w3.org/RDF/

[44] World Wide Web Consortium. (19.11.2003). Home Page, URL: http://www.w3.org

[45] Open Services Gateway Initiative (OSGi). (19.11.2003). Home Page, URL: http://www.osgi.org

[46] Open Services Gateway Initiative (OSGi). (19.11.2003). OSGi Service Platform Release 3 Specification, URL: http://www.osgi.org/resources/spec_download.asp

[47] Sun Microsystems, Inc. (19.11.2003). Java 2 Platform overview, URL: http://java.sun.com/java2/whatis/

[48] Helal, S. (2002). Pervasive Java. IEEE Pervasive Computing, vol. 1, issue 1, January–March, pp. 82–85.

[49] Lawton, G. (2002). Moving Java into Mobile Phones. IEEE Computer, vol. 35, issue 6, June, pp. 17–20.

[50] Sameshima, S., Arbanowski, S. & Suzuki, J. (2001). Super Distributed Objects, Object Management Group (OMG), Super Distributed Objects DSIG White Paper, July. 21 p.

[51] Steglich, S., Vaidya, R. N., Gimpeliovskaja, O., Arbanowski, S. & Popescu-Zeletin, R. (2003). I-Centric Services Based on Super Distributed Objects. In: IEEE 6th International Symposium on Autonomous Decentralized Systems (ISADS´2003), Pisa, Italy. Pp. 232–239.

[52] Satyanarayanan, M., Kistler, J. J., Kumar, P., Okasaki, M. E., Siegel, E. H. & Steere, D. C. (1990). Coda: A Highly Available File System for a Distributed Workstation Environment. IEEE Transactions on Computers, vol. 39, issue 4, April, pp. 447–459.

[53] Satyanarayanan, M. (1996). Accessing Information on Demand at any Location: Mobile Information Access. IEEE Personal Communications, vol. 3, issue 1, February, pp. 26–33.

[54] The Bluetooth Special Interest Group (SIG), Inc. (19.11.2003). The Official Bluetooth Website. URL: http://www.bluetooth.com/

[55] OBP Research Inc. (19.11.2003). ReaGeniX tool overview, URL: http://personal.inet.fi/business/obp/intro.pps

[56] International Organization for Standardization. (1984). The Open Systems Interconnection (OSI) Model, ISO 7498.

[57] Prehofer, C., Kellerer, W., Hirschfeld, R., Berndt, H. & Kawamura, K. (2002). An Architecture Supporting Adaptation and Evolution in Fourth Generation Mobile Communication Systems. KICS Journal of Communications and Networks, vol. 4, issue 4, December, pp. 1–9.

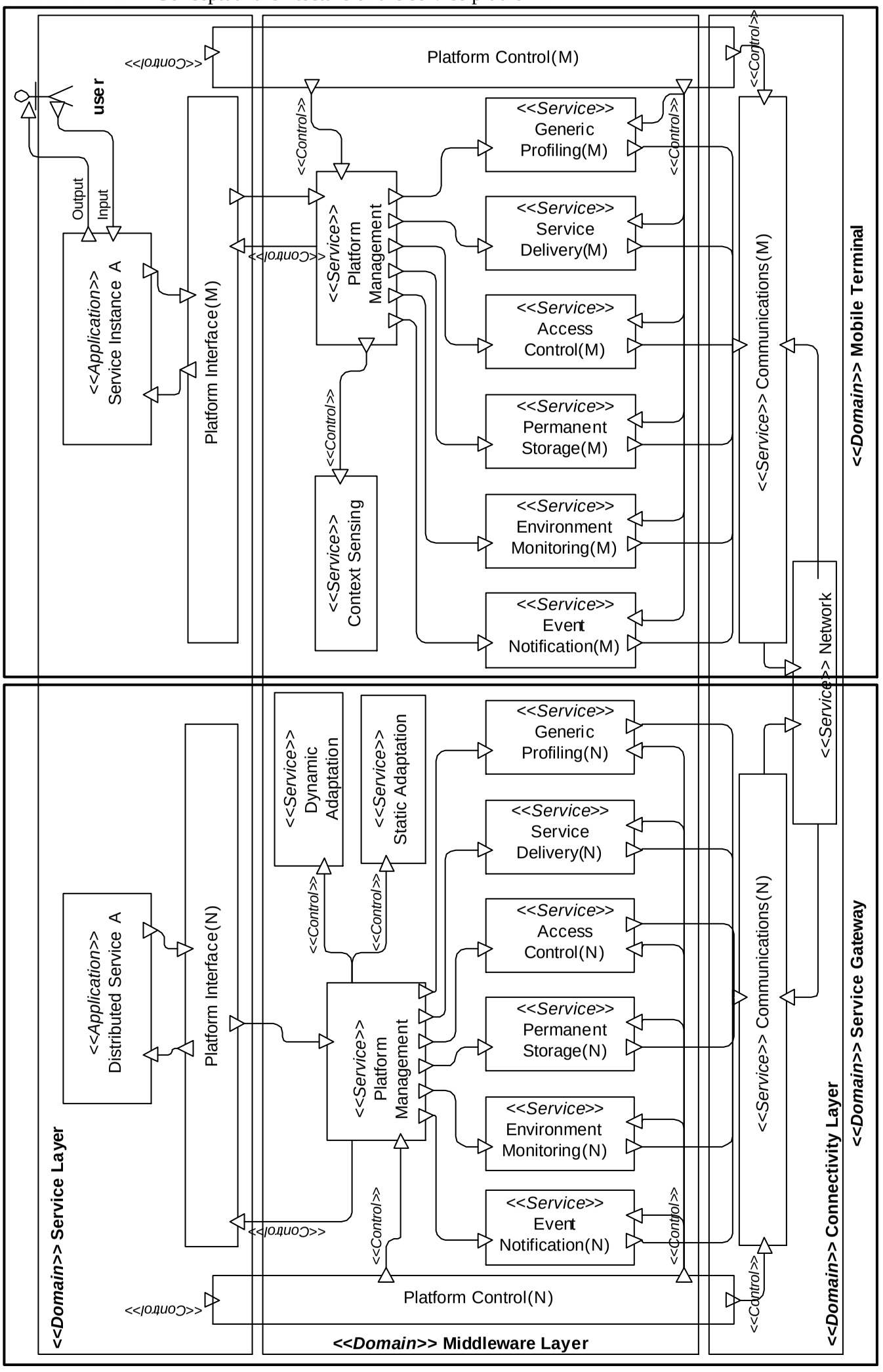
[58] Tuulari, E. & Ylisaukko-oja, A. (2002). SoapBox: A Platform for Ubiquitous Computing Research and Applications. In: Lecture Notes in Computer Science 2414: Pervasive Computing. Zürich, CH, August 26–28, Mattern, F. Naghshineh, M. (eds.). Springer-Verlag. Pp. 125–138.

[59] Smarthome Inc. (19.11.2003). X.10 Overview, URL: http://www.smarthome.com/aboutx10.html

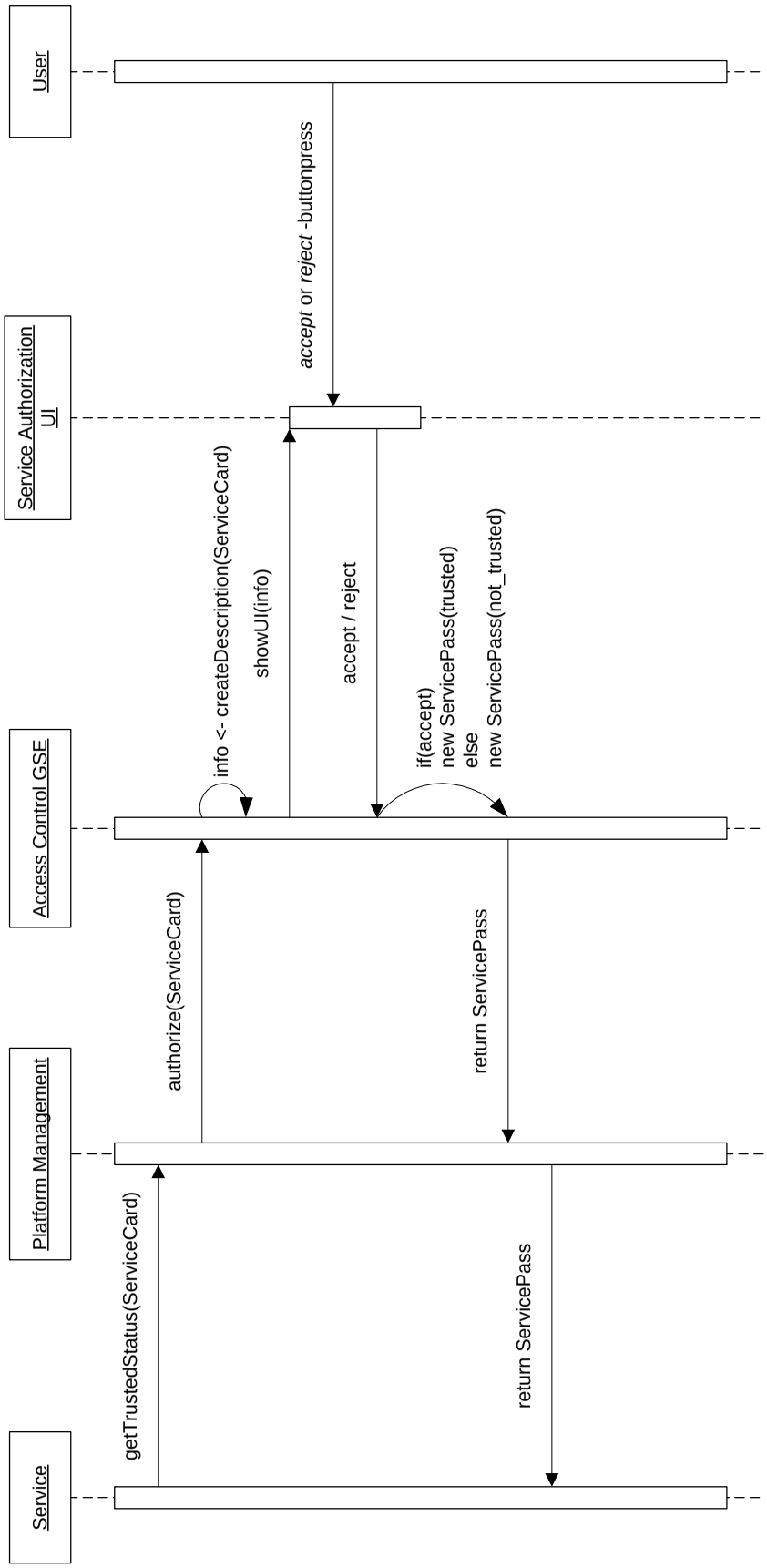
[60] International Business Machines (IBM), Corp. (19.11.2003). IBM Service Management Framework overview. URL: http://www-3.ibm.com/software/wireless/smf/

[61] Sun Microsystems, Inc. (19.11.2003). Java Media Framework (JMF) Home Page. URL: http://java.sun.com/products/java-media/jmf/

APPENDIX 1 Conceptual architecture of the service platform



APPENDIX 2 Service authorization event trace



APPENDIX 3 Service authorization event trace description

1. First, a service wanting to gain trusted status within the service platform and thereby have access to guarded private data starts the service authentication process by passing its ServiceCard to the Platform Management component as a parameter in the getTrustedStatus function call. The service makes its function call via the platform interface from where the call is forwarded to the Platform Management component.

2. The Platform Management component calls the authorize function of the Access Control GSE and passes over the ServiceCard it received in step 1.

3. The Access Control GSE creates a service description from the ServiceCard including the service's name, provider, description of private data usage and all other relevant information needed to describe the service.

4. The service description is shown in the Service Authorization UI and the user is asked if the service described should be trusted to have access to private data.

5. The user either accepts or rejects the services trusted status query by pressing either accept or reject button of the Service Authorization UI.

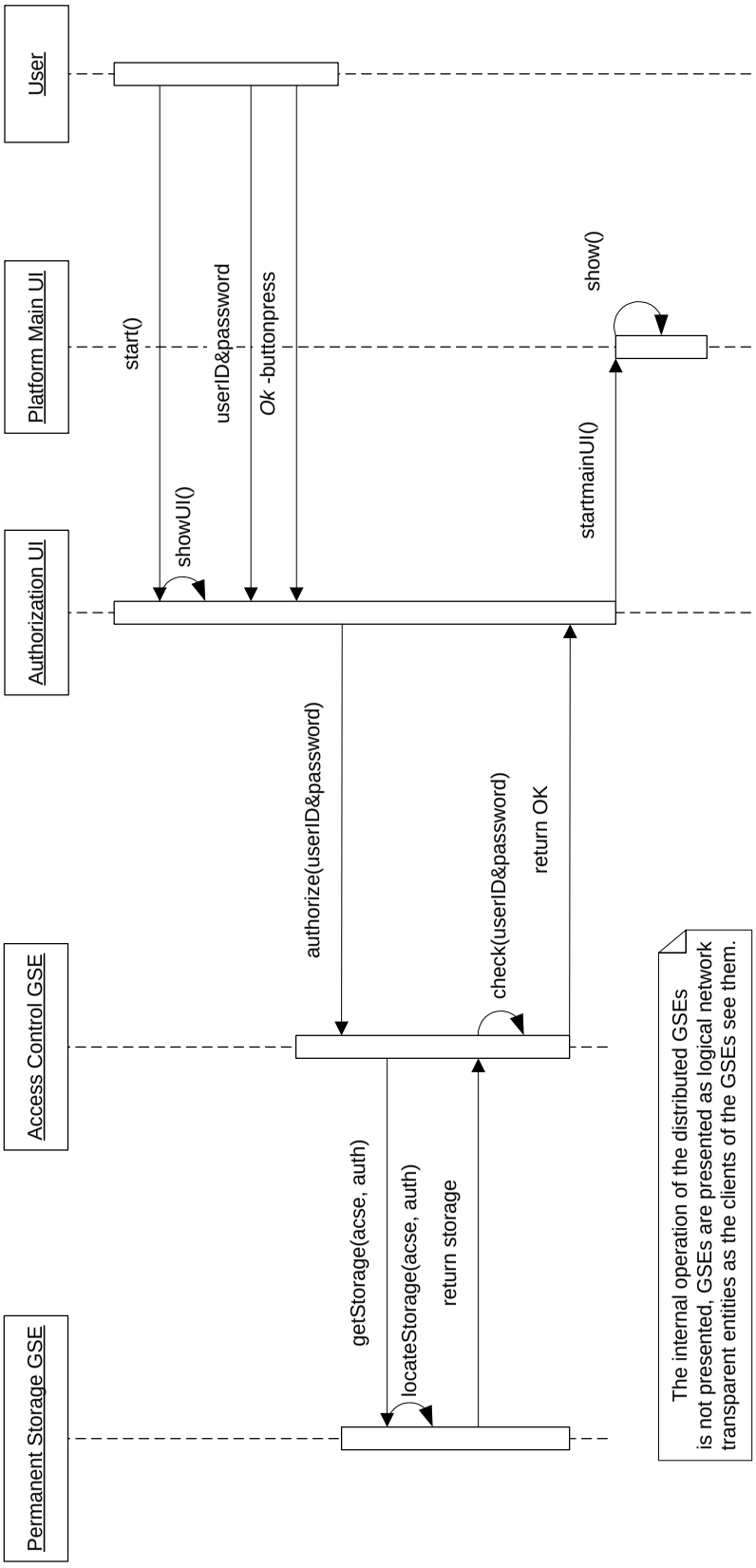
6. The Access Control GSE gets the result of the query. If the user has accepted the query a ServicePass with trusted status is created, otherwise a ServicePass with not-trusted status is created.

7. The ServicePass created is returned from by the Access Control GSE as an authorize function return value to the Platform Management Component.

8. The Platform Management component returns the ServicePass it receives to the service as a return value to the getTrustedStatus function call.

If the user accepted the service's trusted status query, the service can now access the guarded and private data using the ServicePass it was given as a result of the service authorization process. The ServicePass is always required when a service tries to retrieve private and guarded data. Services can obtain the ServicePass with trusted status only by going through the service authorization process as described.

APPENDIX 4 User authorization event trace



APPENDIX 5 User authorization event trace description

1. When the mobile terminal side of the service platform is started, the Authorization UI is the first thing visible to the user.

2. The user gives his personal user identification (userID) and password as input to the Authorization UI.

3. When the user has given the input required he presses the "OK" button in the Authorization UI.

4. The Authorization UI passes the userID and the password to the Access Control GSE as parameters to the authorize function call.

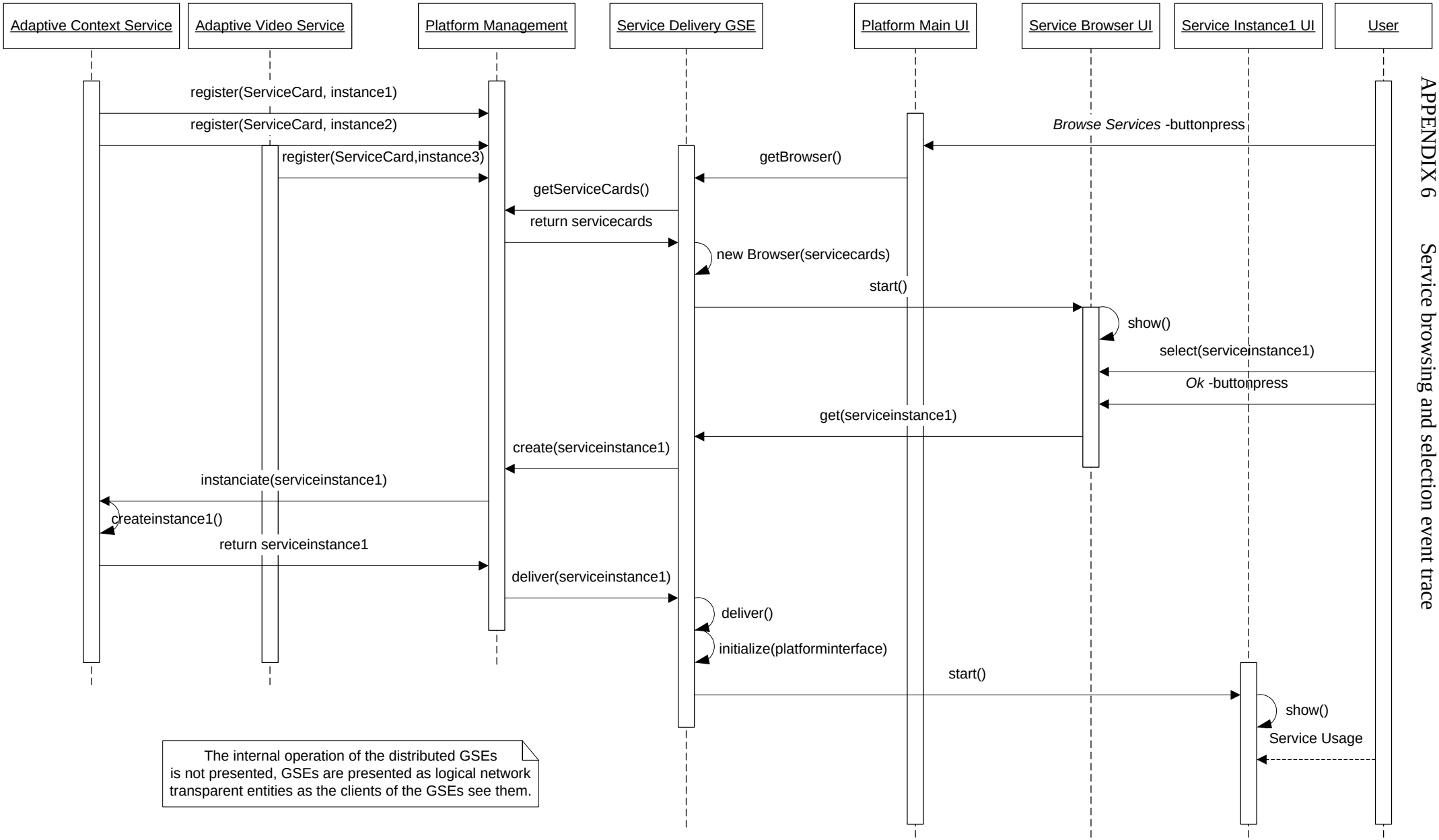
5. The Access Control GSE retrieves a Storage containing the information about authorized usersIDs and passwords from the Permanent Storage GSE by calling the getStorage function with the ServiceID of the caller and storage name as parameters.

6. The Permanent Storage GSE locates the requested Storage, which can be located in either domains of the service platform, and returns the Storage as a return value to the getStorage function call.

7. Possessing the information about authorized users, the Access Control GSE compares the userID and password given by the user with the ones retrieved from the Storage. If they match, OK is returned as a return value to authorize the function call. If the userID and password are not valid, !OK is returned.

8. If OK is received by the Authorization UI it starts and shows the Platform Main UI. If !OK is received, the user is requested to give the userID and the password again.

If the user passes the user authorization process he is given the Platform Main UI that provides full access to all of the services running on the service platform, including the platform services for profile editing and service browsing.

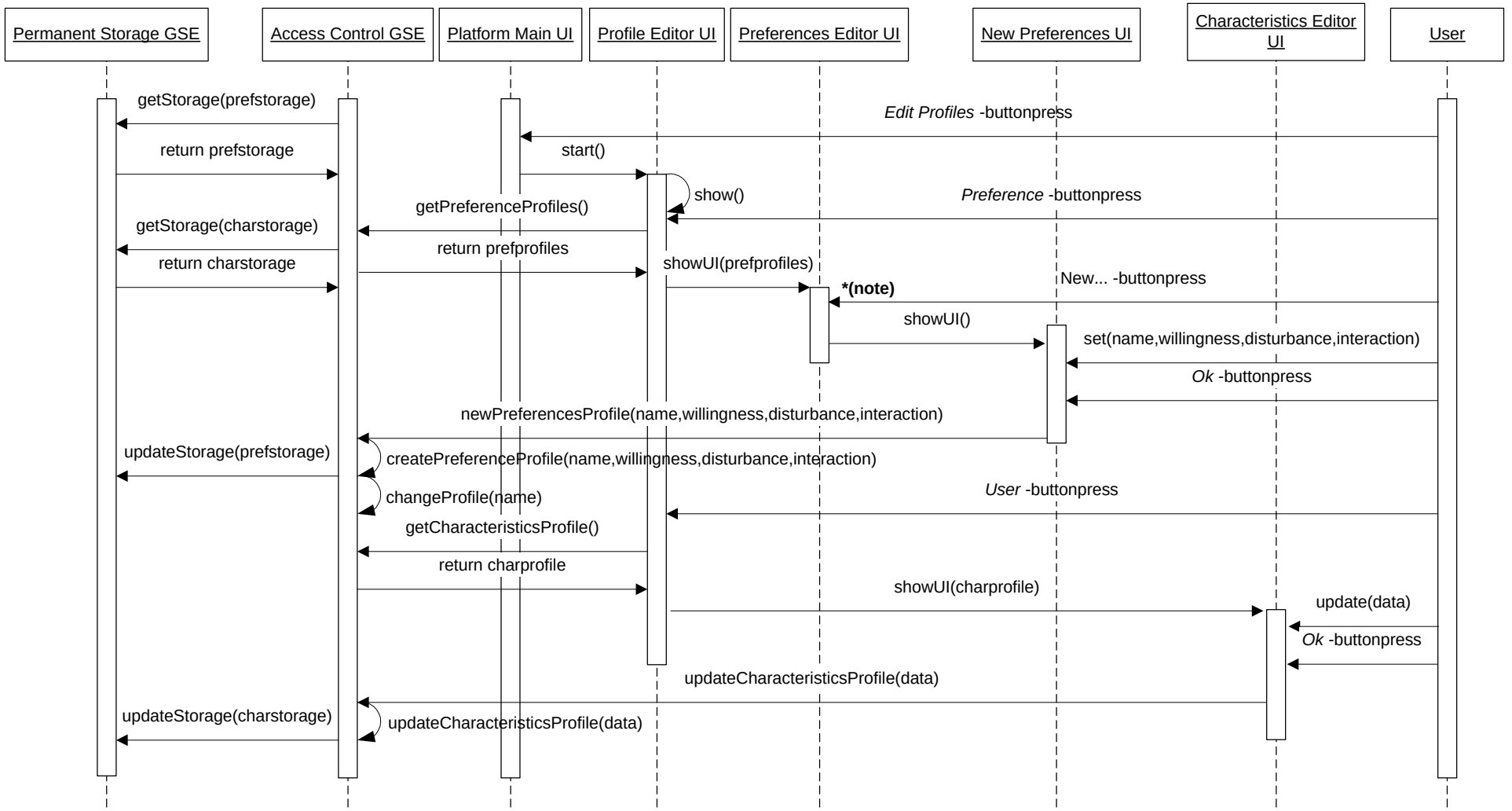


APPENDIX 6 Service browsing and selection event trace

APPENDIX 7 Service browsing and selection event trace description

1. When the Adaptive Context Service and the Adaptive Video Service are installed on the service platform and service gateway they register all of their service instances by calling the register function of the Platform Management component and passing the ServiceCards describing the service instances to be registered as parameters. The services call the register function via the platform interface from where the call is forwarded to Platform Management component.
2. The user starts the service browsing process by pressing the Browse Services button in the Platform Main UI.
3. The Platform Main UI calls the getBrowser function of the Service Delivery GSE in order to activate and start the Service Browser UI.
4. The Service Delivery GSE retrieves all of the registered ServiceCards from the Platform Management component by calling the getServiceCards function.
5. The Platform Management component returns all of the registered ServiceCards describing the registered service instances.
6. The Service Delivery GSE creates a Service Browser UI, which includes the list of available services in a dropdown menu and makes it visible to the user to select the service instance he wants to use. (*)
7. The user selects a service instance from the Service Browser UI and presses the "OK" button. The Service Browser UI calls the get function of the Service Delivery GSE with selected service instance name as the parameter to activate the service instance that the user selected.
8. The Service Delivery GSE calls the Platform Management create function with selected service instance name as the parameter. The call is forwarded to the service having registered the service instance selected.
9. The service creates the service instance and returns it to the Platform Management component that calls the deliver function of Service Delivery GSE with the created service instance as the parameter to deliver the service instance to the mobile terminal domain for execution. (The static adaptation of service and service instance takes place at this point)
10. The Service Delivery GSE delivers the service instance to the mobile terminal domain where it is initialized and given the local platform interface to have access to service platform services.
11. The service instance is started and made visible to the user by the Service Delivery GSE and now the user is able to use the service instance that he selected in step 7.

(*) The ServiceCards include more information about the service instance than just a name. Thereby the service browser could have an option to view this information that describes the service instance and the service. However, this option was not implemented in the prototype implementation.



The internal operation of the distributed GSEs is not presented, GSEs are presented as logical network transparent entities as the clients of the GSEs see them.

***(note)**

At this point the user could also select one of the existing preference profiles and it would be activated by Access Control GSE

APPENDIX 9 Profile editing event trace description

1. The user presses the "Edit Profile" button in the Platform Main UI, which results in the startup of the Profile Editor UI that is shown to the user.

2. The user presses the "Preference" button, which results in the Profile Editor UI retrieving the currently active and previously created preference profiles from the Access Control GSE by calling the getPreferenceProfiles function.

3. The Access Control GSE retrieves the existing preference profiles from the Permanent Storage GSE and returns these and currently active profiles to the Profile Editor UI, which creates Preference Editor UI showing the currently active preferences profiles and all of the existing preferences profiles. The user can select one of these profiles to be activated or create a new preferences profile that will be activated automatically after creation.

4. The user wants to create a new preferences profile and presses the "New..." button, which results in the startup of the New Preferences UI into which the user can define a new preferences profile.

5. The user gives a name for the new profile and defines the three preference variables, as he likes. After this, the user presses the "OK" button, resulting in the newPreferencesProfile function call of the Access Control GSE with parameters given by the user.

6. The Access Control GSE creates and activates a new preferences profile with the parameters received. The new profile is saved using the Permanent Storage GSE.

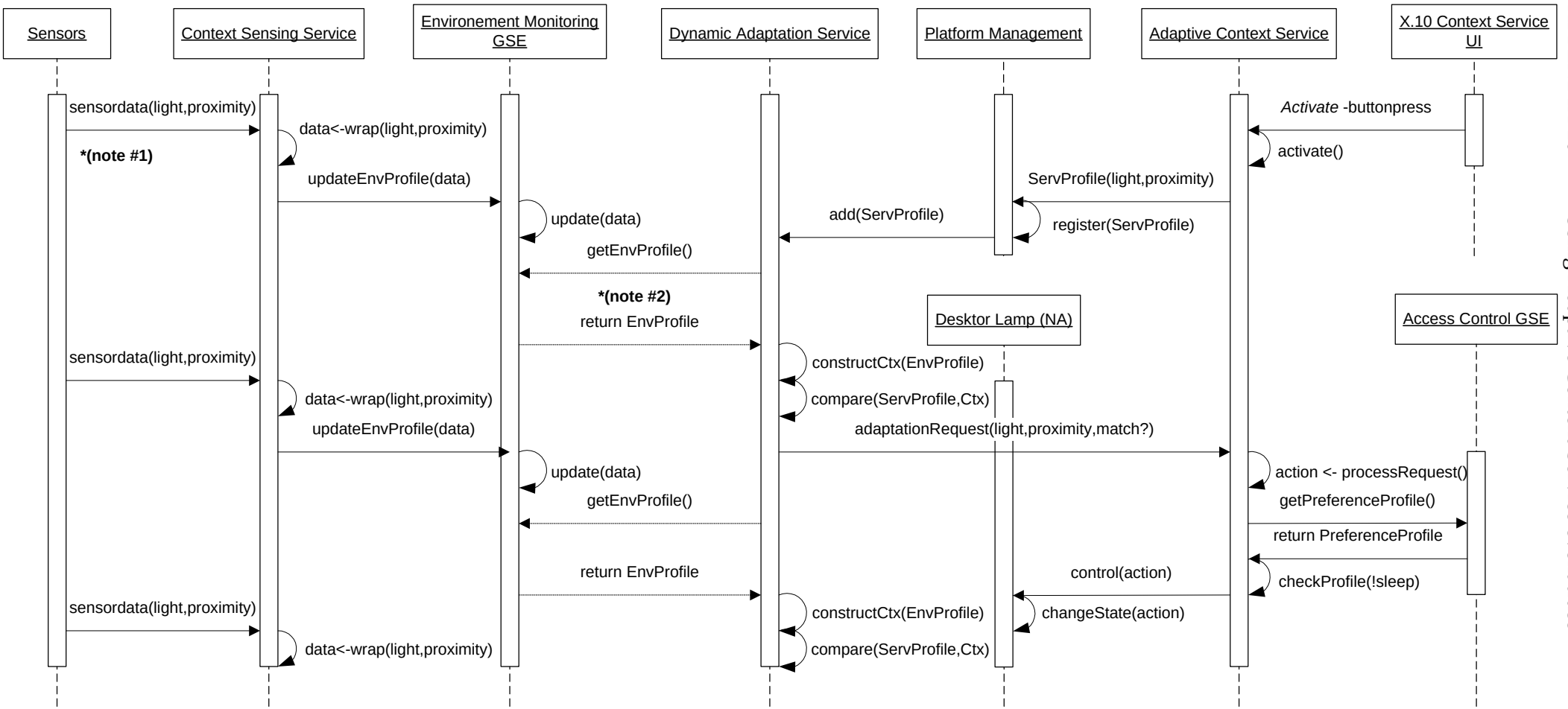
7. Now the user wants to update the user characteristics profile and presses the "User" button in the Profile Editor UI, resulting in the getCharacteristicsProfile function call of the Access Control GSE.

8. The Access Control GSE returns the user characteristics profile that it has saved using the Permanent Storage GSE to the Profile Editor UI.

9. The Profile Editor UI creates and makes visible the Characteristics Editor UI containing the current user characteristics data. The user updates this data and presses the "OK" button.

10. The Characteristics Editor UI updates the changed information by calling the updateCharacteristicsProfile function of the Access Control GSE and giving the updated data as the parameter.

11. The Access Control GSE updates the user preferences profile and saves the updated profile using the Permanent Storage GSE.



The internal operation of the distributed GSEs is not presented, GSEs are presented as logical network transparent entities as the clients of the GSEs see them.

***(note #1)**

The sensor periodically update the data they measure to Context Sensing Service that processes it further.

***(note #2)**

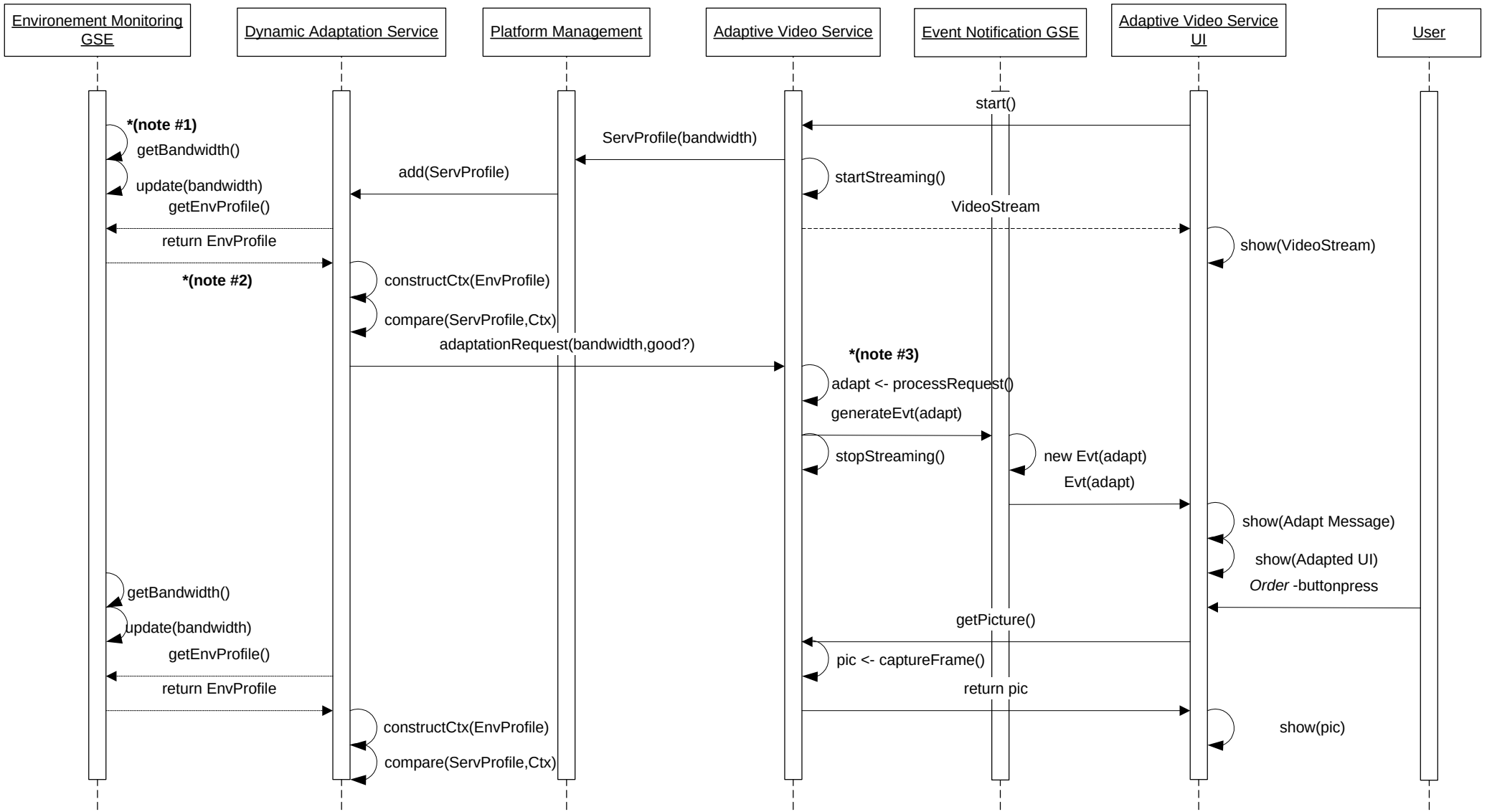
The Dynamic Adaptation Service periodically fetches the environment profile from the Environemnt Monitoring GSE and constructs the current context based on it.

APPENDIX 11 Using Adaptive Context Service event trace description

1. The user has gone through the service browsing and selected X.10 Context Service -service instance. Now the user presses the "Activate" button in the UI of the selected service instance.
2. The Adaptive Context Service receives the button press (*) and activates. This results in the Adaptive Context Service to register its ServiceProfile for monitoring lighting and proximity context information using the platform interface.
3. The registration of the ServiceProfile is received by the Platform Management component that registers the ServiceProfile and adds it to the Dynamic Adaptation Service for monitoring.
4. The Sensors in the user's surroundings send measurement results periodically to the Context Sensing Service that wraps the measurement data and updates it in the environment profile by calling the updateEnvProfile function of the Environment Monitoring GSE with wrapped data as the parameter.
5. The Environment Monitoring GSE updates the received data in the environment profile.
6. The Dynamic Adaptation Service periodically fetches the environment profile by calling the getEnvProfile function of the Environment Monitoring GSE and constructs the current context from the environment profile data.
7. The Dynamic Adaptation Service compares the current context with the a priori context definitions made in the ServiceProfiles. If the definitions and current context match an adaptation request with a matching ServiceProfile, identifications will be sent to the service owning the ServiceProfile. In this case the lighting conditions are not good and the user is near his terminal PC, this results in an adaptation request to be sent to the Adaptive Context Service.
8. The Adaptive Context Service receives the adaptation request and processes it possibly resulting in an action command to be executed. At this point the Adaptive Context Service consults the Static Adaptation Service by fetching the user preferences profile and checking that the user has not set a sleep profile. If not, the Adaptive Context Service turns on the Desktop Lamp near the user's PC by calling a control function with an action command as the parameter.

(*) Even if not drawn in the event trace diagram, the X.10 Context Service -service instance and the Adaptive Context Service can use the service provided by the Event Notification GSE of the service platform.

To summarize, the Adaptive Context Service automatically turns on the desktop lamp if it senses that lighting conditions are not good near the user and the user is near the terminal. Also, if the user leaves his terminal, the desktop lamp is automatically turned off until the user gets near to the terminal again.



***(note #1)**

The Environment Monitoring GSE internally monitors the available network bandwidth by periodically updating the bandvisth value to the environment profile.

***(note #2)**

The Dynamic Adaptation Service periodically fetches the environment profile from the Environemnt Monitoring GSE and constructs the current context based on it.

***(note #3)**

In this case the available bandwidth is below the value defined a priori in the service profile and therefore adaptation takes place. If the value would be above the service profile definition adaptation would not take place.

APPENDIX 13 Using Adaptive Video Service event trace description

1. The user has selected the Adaptive Video Service -service instance to be started, resulting in the Adaptive Video Service to register its ServiceProfile for monitoring available network bandwidth. The ServiceProfile is registered using services in the platform interface from where the registration call is forwarded to the Platform Management component, which adds the Service Profile to Dynamic Adaptation Service for monitoring.
2. The Adaptive Video Service starts the video streaming and the video starts playing in the Adaptive Video Service UI.
3. The Environment Monitoring GSE includes functionality for available bandwidth monitoring and it periodically updates the available bandwidth value to the environment profile it hosts.
4. The Dynamic Adaptation Service periodically fetches the environment profile by calling the getEnvProfile function of the Environment Monitoring GSE and constructs the current context from the information in the environment profile.
5. The constructed context is compared to a priori context definitions in the ServiceProfiles. If the definition matches up with current context, an adaptation request will be sent to the service owning the matching ServiceProfile. In this case, the available bandwidth has dropped below the value defined in the ServiceProfile of Adaptive Video Service resulting in an adaptation request to be sent for it.
6. The Adaptive Video Service receives the adaptation request and processes it resulting in an adaptation event to be generated by calling the generateEvt function of the Event Notification GSE. (*) Video streaming can be stopped at this point.
7. The Adaptive Video Service -service instance receives the adaptation event and adapts accordingly by showing the reason for the adaptation to the user and changing its operation to picture mode. In picture mode, the user can order still pictures taken from the video camera.
8. The user presses the "Order" button in the Adaptive Video Service UI that results in the getPicture function call of the Adaptive Video Service.
9. The Adaptive Video Service captures a frame from the video camera and returns it to Adaptive Video Service UI as a return value to the getPicture function call.
10. The Adaptive Video Service UI receives the picture and shows it to user.

(*) If the available bandwidth increases above the value defined in the ServiceProfile, a new adaptation request is sent resulting in the video streaming being started and the UI changing to video mode again.

Published by

Series title, number and report code of publication

VTT Publications 519
VTT–PUBS–519

Author(s)
Pakkala, Daniel

Title

Lightweight distributed service platform for adaptive mobile services

Abstract

Distributed computing environments are becoming more heterogeneous due to the integration of different wireless and fixed networks and variety of terminal devices that can be used to access content and services on the Internet. Also, increasing user expectations of personalizable, adaptive, and context-aware mobile services bring complexity to the development of future mobile services. The distributed and heterogeneous computing environment, together with increasing user expectations, set requirements for future mobile services that are difficult to meet without service platform support alleviating service development.

In this work, a lightweight distributed service platform that has been designed with a practical design approach to support the adaptation of mobile services, and partly also personalization and context-aware functionalities of the services, is presented. A requirement analysis for future mobile services is carried out by an extensive literature review. The mobile service adaptation support functionality of the service platform is designed based on the identified requirements. Further architecture based on the concept of Generic Service Elements is designed for the service platform. The validation of the architecture is achieved by a prototype implementation. The validation was successfully achieved by the prototype implementation. It also proved the service platform's applicability to resource-constrained distributed mobile computing environments as the prototype resulted in an overall size of less than 360 KB, of which approximately one third is allocated to the mobile terminal domain.

Keywords
adaptation, middleware services, Generic Service Elements (GSE), context-awareness, service personalization, pervasive computing architectures

Activity unit
VTT Electronics, Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland

ISBN			Project number
951–38–6269–0 (soft back ed.) 951–38–6270–4 (URL:http://www.vtt.fi/inf/pdf/)			
Date March 2004	Language English, Finnish abstr.	Pages 145 p. + app. 13 p.	Price D
Name of project		Commissioned by	
Series title and ISSN VTT Publications 1235–0621 (soft back ed.) 1455–0849 (URL: http://www.vtt.fi/inf/pdf/)		Sold by VTT Information Service P.O.Box 2000, FIN–02044 VTT, Finland Phone internat. +358 9 456 4404 Fax +358 9 456 4374	

Julkaisija

Julkaisun sarja, numero ja raporttikoodi

VTT Publications 519
VTT–PUBS–519

Tekijä(t)
Pakkala, Daniel

Nimeke

Kevytrakenteinen hajautettu palvelualusta mukautuville liikkuville palveluille

Tiivistelmä

Langattomien ja kiinteiden verkkojen yhdentymisen sekä moninaisten Internetin sisällön ja palveluiden käyttöön tarkoitettujen päätelaitteiden ansiosta hajautetut tietokoneympäristöt ovat epäyhtenäistymässä. Lisäksi käyttäjien lisääntyvät odotukset koskien räätälöitäviä, mukautuvia ja tilannetietoisia liikkuvia palveluita tuovat monimutkaisuutta tulevaisuuden liikkuvien palveluiden kehitykseen. Tulevaisuuden lisääntyvät käyttäjien odotukset sekä epäyhtenäiset hajautetut tietokoneympäristöt asettavat vaatimuksia liikkuville palveluille, jotka niiden on vaikea täyttää ilman palvelun kehitystä helpottavia palvelualustoja.

Tässä työssä esitellään kevytrakenteinen hajautettu palvelualusta, joka on suunniteltu käyttäen käytäntöön perustuvaa lähestymistapaa. Palvelualusta on suunniteltu tukemaan liikkuvien palveluiden mukautuvuutta, sekä osittain myös palveluiden räätälöitävyyttä sekä tilannetietoisuutta. Laajassa kirjallisuuskatsauksessa on tehty vaatimusmäärittely tulevaisuuden liikkuville palveluille. Palvelualustan liikkuvien palveluiden mukautuvuutta tukeva toiminnallisuus on suunniteltu kirjallisuuskatsauksessa tunnistettujen vaatimusten pohjalta. Palvelualustan arkkitehtuuri on suunniteltu pohjautuen ideaan yleisistä palveluelementeistä. Suunniteltu arkkitehtuuri on vahvistettu toimivaksi tekemällä palvelualustasta prototyyppi, joka toimi odotetusti. Lisäksi palvelualustan prototyyppi toteutus osoitti alustan soveltuvan resursseiltaan rajoitettuihin hajautettuihin liikkuviin tietokoneympäristöihin, koska sen lopullinen koko oli alle 360 Kt, josta noin kolmasosa sijoittuu liikkuvaan päätelaitteeseen.

Avainsanat
adaptation, middleware services, Generic Service Elements (GSE), context-awareness, service personalization, pervasive computing architectures

Toimintayksikkö
VTT Elektroniikka, Kaitoväylä 1, PL 1100, 90571 OULU

ISBN			Projektinumero
951–38–6269–0 (nid.) 951–38–6270–4 (URL: http://www.vtt.fi/inf/pdf/)			
Julkaisuaika Maaliskuu 2004	Kieli Englanti, Suom. tiiv.	Sivuja 145 s. + liitt. 13 s.	Hinta D
Projektin nimi		Toimeksiantaja(t)	
Avainnimeke ja ISSN VTT Publications 1235–0621 (nid.) 1455–0849 (URL: http://www.vtt.fi/inf/pdf/)		Myynti: VTT Tietopalvelu PL 2000, 02044 VTT Puh. (09) 456 4404 Faksi (09) 456 4374	

VTT PUBLICATIONS

499 Kololuoma, Terho. Preparation of multifunctional coating materials and their applications. 62 p. + app. 33 p.

500 Karppinen, Sirpa. Dietary fibre components of rye bran and their fermentation *in vitro*. 96 p. + app. 52 p.

501 Marjamäki, Heikki. Siirtymäperusteisen elementtimenetelmäohjelmiston suunnittelu ja ohjelmointi. 2003. 102 s. + liitt. 2 s.

502 Bäckström, Mika. Multiaxial fatigue life assessment of welds based on nominal and hot spot stresses. 2003. 97 p. + app. 9 p.

503 Hostikka, Simo, Keski-Rahkonen, Olavi & Korhonen, Timo. Probabilistic Fire Simulator. Theory and User's Manual for Version 1.2. 2003. 72 p. + app. 1 p.

504 Torkkeli, Altti. Droplet microfluidics on a planar surface. 2003. 194 p. + app. 19 p.

505 Valkonen, Mari. Functional studies of the secretory pathway of filamentous fungi. The effect of unfolded protein response on protein production. 2003. 114 p. + app. 68 p.

506 Mobile television – technology and user experiences. Report on the Mobile-tv project. Caj Södergård (ed.). 2003. 238 p. + app. 35 p.

507 Rosqvist, Tony. On the use of expert judgement in the qualification of risk assessment. 2003. 48 p. + app. 82 p.

508 Parviainen, Päivi, Hulkko, Hanna, Kääriäinen, Jukka, Takalo, Juha & Tihinen, Maarit. Requirements engineering. Inventory of technologies. 2003. 106 p.

509 Sallinen, Mikko. Modelling and estimation of spatial relationships in sensor-based robot workcells. 2003. 218 p.

510 Kauppi, Ilkka. Intermediate Language for Mobile Robots. A link between the high-level planner and low-level services in robots. 2003. 143 p.

511 Mäntyjärvi, Jani. Sensor-based context recognition for mobile applications. 2003. 118 p. + app. 60 p.

512 Kauppi, Tarja. Performance analysis at the software architectural level. 2003. 78 p.

513 Uosukainen, Seppo. Turbulences as sound sources. 2003. 42 p.

514 Koskela, Juha. Software configuration management in agile methods. 2003. 54 p.

516 Määttä, Timo. Virtual environments in machinery safety analysis. 2003. 170 p. + app. 16 p.

515 Palviainen, Marko & Laakko, Timo. mPlaton - Browsing and development platform of mobile applications. 2003. 98 p.

517 Forsén, Holger & Tarvainen, Veikko. Sahatavaran jatkojalostuksen asettamat vaatimukset kuivauslaadulle ja eri tuotteille sopivat kuivausmenetelmät. 2003. 69 s. + liitt. 9 s.

518 Lappalainen, Jari T. J. Paperin- ja kartonginvalmistusprosessien mallinnus ja dynaaminen reaaliaikainen simulointi. 2004. 144 s.

519 Pakkala, Daniel. Lightweight distributed service platform for adaptive mobile services. 2004. 145 p. + app. 13 p.

Tätä julkaisua myy	Denna publikation säljs av	This publication is available from
VTT TIETOPALVELU PL 2000 02044 VTT Puh. (09) 456 4404 Faksi (09) 456 4374	VTT INFORMATIONSTJÄNST PB 2000 02044 VTT Tel. (09) 456 4404 Fax (09) 456 4374	VTT INFORMATION SERVICE P.O.Box 2000 FIN–02044 VTT, Finland Phone internat. +358 9 456 4404 Fax +358 9 456 4374

ISBN 951–38–6269–0 (soft back ed.)
ISSN 1235–0621 (soft back ed.)

ISBN 951–38–6270–4 (URL: http://www.vtt.fi/inf/pdf/)
ISSN 1455–0849 (URL: http://www.vtt.fi/inf/pdf/)