UNIVERSITATEA „BABEŞ - BOLYAI”
FACULTATEA DE ŞTIINŢE ECONOMICE ŞI GESTIUNEA AFACERILOR
CATEDRA DE INFORMATICĂ ECONOMICĂ

CONTRIBUŢII LA DEZVOLTAREA ARHITECTURILOR COLABORATIVE A DISPOZITIVELOR INTELIGENTE APLICATE ÎN ECONOMIE

PERVASIVE COMPUTING: COLLABORATIVE ARCHITECTURE APPLIED IN BUSINESS ENVIRONMENT

Summary of the PhD Thesis

PhD Candidate: Sebastian Presecan

Scientific Advisor: Prof. Nicolae Tomai, PhD

Cluj-Napoca, November 2011

Abstract

Personal computers era is nowadays closing, as users are rather focused on information and on the methods of accessing the information then on the tools themselves. Moreover, computational power is distributed on different small devices that are used for helping users to perform various daily tasks. Smart devices owners are using mobile applications as a replacement for desktop applications. Due to limited resources and due to the fact that they must cope with the challenge of being a real replacement for personal computers, mobile devices should be able to run applications that distribute their tasks for execution in the cloud. The proposed framework combines the mobility of smartphone applications with the execution capabilities of the cloud and proves itself as a viable solution for implementing pervasive systems that enable ubiquitous usage of the computational power.

Keywords: pervasive computing, middleware, mobile application, mobile computing

1. Introduction

‘‘The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.’’ (Weiser M. , 1991). With this statement, in 1991, Mark Weiser described his vision about the computer of the 21st century. Computing has since evolved beyond the desktop PC. Computing is no more associated with Personal Computers only. Computers are nowadays embedded into different devices like cell phones, PDAs, different control devices, etc. The cost of hardware significantly decreased, which contributed to the spread of the computing power usage.

The evolution of network technologies made it possible for computers to perform user tasks anywhere, anytime. As time goes by, computers are becoming more ubiquitous. The vision expressed by Mark Weiser is nowadays closer to reality than it was ever expected.

1.1. Motivation

Mobile computing today is a reality, but the computing power is distributed not only to mobile phones. Smartphones are nowadays ubiquitous. Smartphones owners are using the mobile applications as a replacement for desktop applications. As a result, there is an increasing need of using mobile computing in different kind of applications in domains like: medicine, business, etc. It becomes more and more obvious that there is also a need of having architecture for these new conditions. Considering my experience as a developer and architect and the fact that I have been working in designing and implementing large scale distributed systems for more than 12 years, these new technological advances challenged me to try and define architecture for the new generation of pervasive system.

1.2. Research questions

Given the current technological stage when the computing power is spread within different kinds of devices, a new challenge may arise and that is making these smart devices work together. For the moment, the old method of making these devices collaborate presupposes connecting them to some service providers in order to get information from there.

The embedded devices are nowadays smart enough, though they should be able to identify themselves as possible collaborators, in order to perform user tasks or to delegate them to more powerful computers. Baring these aspects in mind, the following questions pop-up:

- Is it possible to improve smartphone applications by sensing the environment?
- Which are the possible ways that can be used to make smart devices communicate?
- What is the best approach to be used in order to let smart devices to transfer their jobs to more powerful computers?
- How should the system be designed in order to assure security, privacy and trustfulness?
- How is it possible to integrate all different types of smart devices running on different operating systems into one single system?

Throughout the current research we have searched for answers to all these questions. Putting together all these answers we have managed to detail a middleware that could be used for implementing any pervasive system. The purpose of the current research is to define a highly scalable architecture to be used in the implementation of different user requirements with little effort from applications developers. The proposed middleware should help the mobile applications become ubiquitous.

1.3. Objectives

One of the objectives of the present thesis is to understand the way in which the current middleware is supporting pervasive computing and to analyse them from the perspective of the implementation of an educational pervasive portal.

The projected result of the current research is a scalable architecture for a pervasive system. We will consider the educational portal from a university campus as a target system to work on. It will thus gain a new dimension, its ubiquity. The usage of the system is meant to be ubiquitous so that the users are not disturbed by the system, except for the situation when, after consuming all the existing possibilities, it cannot be fixed by itself. Even if the proposed architecture has currently been applied to improve an educational pervasive portal, we consider it generic enough to be the backbone in the implementation of any type of pervasive system.

1.4. Contribution

Our PhD thesis raises several novelty issues in the field of Pervasive Computing. The main area of study and research comprises context-awareness, invisibility, security and application mobility. All these aspects have been analysed from the perspective of building a pervasive system middleware. The major contributions are the following:

- identifying the major challenges in the implementation process of a pervasive system and subsequently, defining the evaluation criteria which could be used to analyse different middleware used in implementing pervasive systems;
- creating an innovative solution which would observe the context and gather together all the context providers. They could be used by the mobile application to monitor the environment and to adapt the application's features and functionalities to the context changes, in an invisible way;
- conceiving a cutting-edge solution to achieve the application mobility, by distributing the application services into the cloud in a seamless manner. The application performances could be easily improved through the distribution in the cloud of some of its services. When the distribution is made automatically, without the intervention of the application developer, no additional codes must be implemented. Therefore, we consider this approach as a major benefit for enabling the cloud distribution in a transparent way;
- using a communication channel based on internet de-facto standards which has ben used to integrate different devices into the platform. These devices have been running on different operating systems and supported both the pull and the push of the content;
- building-up a state-of-the-art middleware architecture which provides a generic end-to-end solution for implementing any type of pervasive system which faces all the identified system challenges.

2. Background

This chapter is mainly focused on presenting the key-concepts of Pervasive Computing. It is not intended to be a detailed monograph of Pervasive Computing, but rather to present the key-concepts which are needed in order to perform the analysis of a Pervasive Computing system architecture. I hereby present the Pervasive Computing challenges, some of the systems that are able to solve these particular challenges and the main directions of study within the above mentioned branch of computing science.

2.1. Pervasive computing evolution

Mark Weiser's visionary statement(mentioned above) describes what is expected from pervasive or ubiquitous computing: users access the computational environment anytime, everywhere.

Starting with the definition: "A distributed system is an information-processing system that contains a number of independent computers that cooperate with one another over a communications network in order to achieve a specific objective." (Bapat, 1994) makes it clear that the "pervasive computing" defines a new context for interaction between the end users and the distributed systems.

A.S.Tanenbaum definition of the distributed systems: *"A collection of independent computers that appears to its users as a single coherent system."* (Tanenbaun & Steen, 2006) predicted the pervasiveness of the nowadays computing systems.

Computing has since evolved beyond the desktop PC. Computing is not any more associated only with Personal Computers. Computers are nowadays embedded into different devices like cell phones, PDAs, different control devices, etc. The fact that the cost of hardware has significantly decreased contributed to the spread of the computing power usage.

Given the evolution of computers and the interaction between users and the computing system, we can identify 3 main stages:

- mainframe – which can be defined as *one system multiple users;*

- personal computer – which can be defined as *one user, one system;*
- pervasive computing – which can be defined as *one user, multiple systems.*

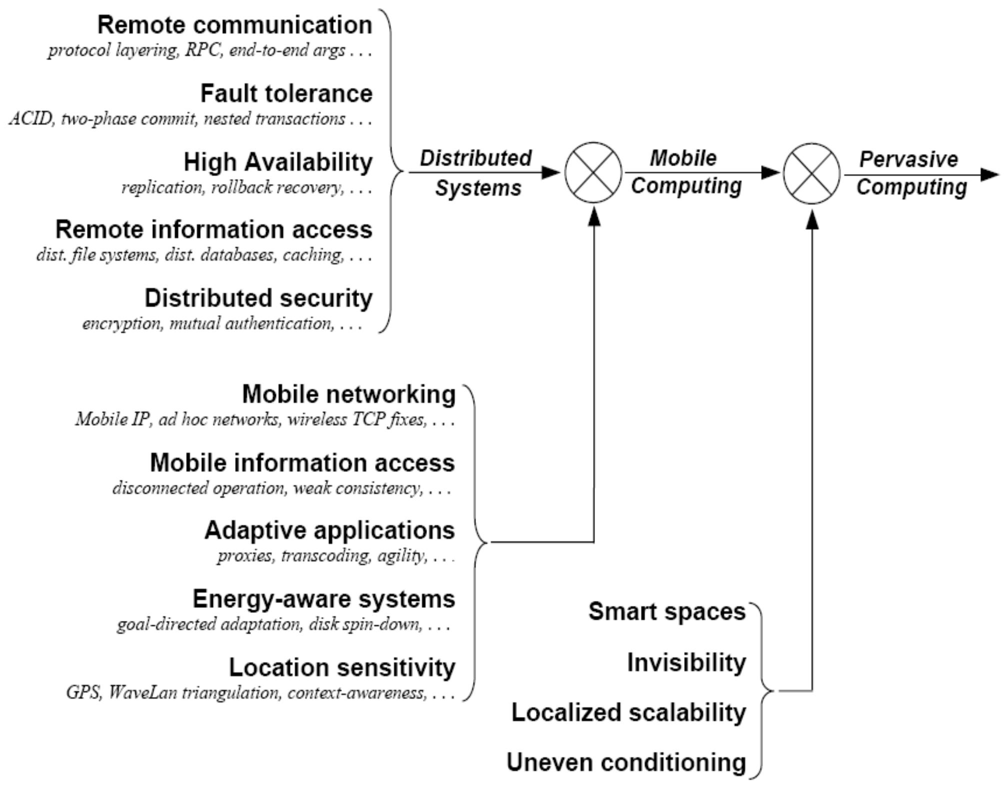
Pervasive computing changed the way people interact with computers. The key idea behind pervasive computing is to deploy a wide variety of computing devices throughout our living and working spaces. These devices coordinate with each other and with network services, aiming at providing people with universal access to their information and seamlessly assisting them in completing their tasks.

2.2. Pervasive systems challenges

Considering the vision of M. Weiser and the user needs expressed into the above scenario, it is clear that a pervasive system must be distributed and must support user and application mobility.

In

Figure 2.1, M. Satyanarayanan describes the main characteristics of a pervasive system.



Remote communication
protocol layering, RPC, end-to-end args . . .
Fault tolerance
ACID, two-phase commit, nested transactions . . .
High Availability
replication, rollback recovery, . . .
Remote information access
dist. file systems, dist. databases, caching, . . .
Distributed security
encryption, mutual authentication, . . .
Distributed Systems
Mobile Computing
Pervasive Computing
Mobile networking
Mobile IP, ad hoc networks, wireless TCP fixes, . . .
Mobile information access
disconnected operation, weak consistency, . . .
Adaptive applications
proxies, transcoding, agility, . . .
Energy-aware systems
goal-directed adaptation, disk spin-down, . . .
Location sensitivity
GPS, WaveLan triangulation, context-awareness, . . .
Smart spaces
Invisibility
Localized scalability
Uneven conditioning

Figure 2.1 - Taxonomy of Computer Systems Research Problems in Pervasive Computing (Satyanarayanan, Pervasive Computing: Vision and Challenges, 2001)

2.3. Pervasive systems and middlewares

Even if pervasive computing is a relatively new field, there are already many approaches of defining architectures for pervasive systems. For the purpose of the current work, we have selected those that are representative for niche area. The selection was made considering the world wide appreciation of the system and the different taxonomies used to define these architectures.

The purpose of the assessment is to analyse how these systems meet the requirements defined above, to identify the strength and the weakness of the existing systems and also to identify possible research directions.

The selected systems are:

- GAIA- developed by the University of Illinois;
- AURA- developed by Carnegie Mellon University;
- PICO – developed by the University of Texas;
- One.world – developed by Washington University;
- SODA

2.3.1. Gaia

Gaia is a meta-operating system built as a distributed middleware infrastructure that coordinates software entities and heterogeneous networked devices contained in a physical space (Roman, Hess, Cerqueira, & Campbell, 2002). Gaia is designed to support the development and execution of portable applications for *active spaces* — programmable ubiquitous computing environments in which users interact with several devices and services simultaneously. Gaia exports services to query, access, and use existing resources and context, and provides a framework to develop user-centric, resource-aware, multi- device, context-sensitive, and mobile applications.

2.3.2. Aura

As a consequence of Moore's law, nowadays it is clear that the hardware is no more the bottleneck, but the way in which the user interacts and uses the computing systems.

The intuition behind a personal Aura is that it acts as a proxy for the *mobile user* it represents: when a user enters in a new environment, his or her Aura marshals the appropriate resources to support the *user's task*. Furthermore, an Aura captures constraints that the *physical context* around the user imposes on. (Garlan, Siewiorek, & Smailagic, 2002), (Aura, 2002)

2.3.3. One.World

The One.World architecture was created with the aim of implementing one of the most important pervasive challenges, which is “accessing information anytime and everywhere”. One.world *architecture provides an integrated, comprehensive framework for building pervasive applications. It targets applications that automatically adapt to highly dynamic computing environments, and it includes services that make it easier for developers to manage constant change.* (Grimm R. , 2004)

2.3.4. PICO

PICO’s objective is to meet the demands of time-critical applications in areas such as telemedicine, the military, and crisis management that demand automated, continual, unobtrusive services and proactive real-time collaborations among devices and software agents in dynamic, heterogeneous environments. (Kumar, Shirazi, & Singhal, 2003)
The PICO’s architects imagine a system composed of intelligent autonomous softwares called *delegents* and hardware devices called *camilenus*. The target of PICO is to provide “what we want, when we want, where we want, and how we want” types of services, autonomously and continually.
Camilenus can be of different types and complexities. They are all interconnected using different protocols and technologies. A *delegent* is an intelligent software which works on behalf of *camilenus* or on behalf of the user. For example, a delegent can gather information locally or remotely, with the aim of collaborating with other delegents and form a computing community.

Overall, the PICO system offers a viable solution based on the agents paradigm to implement a dynamic pervasive system. As mentioned previously, in order to match the main pervasive requirements, the system needs to be improved by adding monitoring and dispatching functions either to delegents or to communities.

2.3.5. SODA

To build a pervasive system as we have seen above, there are many challenges that need to be overcome. In order to meet them all, the system is getting extremely complex and expensive. In order to leverage the cost of integration, development and to achieve a good scalability, a solution would be to adopt SOA in pervasive computing.
A Service Oriented Architecture (SOA) is set of principles that define an architecture that is loosely coupled and comprised of service providers and service consumers that interact according to a negotiated contract or interface. These services provide the interfaces to applications in the IT landscape. The primary goal of SOA is to expose application functions in a standardized way so that they can be leveraged across multiple projects. This approach greatly reduces the time, effort and cost it takes to maintain and expand solutions to meet business needs. (Mansukhani, 2005)

SODA is an adaptation of the service oriented architecture (SOA) which integrates business systems through a set of services that can be reused and combined to address changing business priorities. Services are software components with well-defined interfaces, and they are independent of the programming language and the computing platforms on which they run. (Koch, 2005) Thus the purpose of SODA is to integrate the system devices into enterprise software systems. By these the developers will be able to use the embedded devices and sensor as using any other enterprise services. (Sumi, 2006)

2.4. Concluding remarks

Pervasive Computing is a natural evolution of distributed systems. Pervasive Computing users are able to use different applications running on wide range of devices and systems. The distribution of the tasks' execution and the heterogeneity of the running environment make the design of a pervasive system a pretty complex task.

We have identified the main architectural challenges that need to be overtaken by any system performing pervasive tasks: context-awareness, invisibility, service discovery, integration, mobile information access, scalability, security and development and deployment. The following middlewares: *Gaia, Aura, One.World, Pico and SODA* have been analysed from the perspective of full-filling the above-identified architectural challenges. The presented systems provide powerful architecture, addressing some of the challenges. They introduce great concepts and define simple yet powerful ways of fulfilling some of the main key challenges.

The main outcomes from this chapter were published on (Presecan S., 2007), (Presecan S, 2008)

3. Context-awareness

Context-awareness is a key-concept in pervasive systems. The current chapter defines the concept of "context-awareness". It also identifies and describes in full detail the main challenges presupposed by the process of implementing context-awareness systems. At the end of the chapter we have decided to make a presentation of a virtual system architecture that aims at solving the identified context-awareness challenges.

3.1. Context-awareness key concept

"Context-aware software adapts according to the location of use, the collection of nearby people and objects, the accessible devices, as well as changes to those objects over time. A system with these capabilities surveys the computing environment and reacts to changes to that environment". (Schilit, Adams, & Want, 1994)

Context-aware computing devices and applications respond to changes in the environment in an intelligent manner, in order to enhance the computing environment for users. Context-aware applications tend to be mobile applications for obvious reasons:

- the user's context fluctuates most frequently when a user is mobile.
- the need for context aware behaviour is greatest in mobile environment (Adelstein, Gupta, Richard, & Schwibert, 2005).

3.2. Context awareness challenges

In theory, a smart device can get context information by using a large variety of sensors, cameras and microphones, which can be embedded in the device or in the surrounding environment. In practice, this is a very challenging task, if we take into consideration the need of integration of different type of small devices and technologies, the mobility of the user and the reduced computational power of smart devices.

The following challenges can thus be considered important for a context awareness middleware:

- discovery of the source of the context providers.
- query for getting the context information.
- processing of the context information without disturbing the normal system features.
- the system adaptability to the change of context values.
- the integration of different types of context providers and communication channels.

3.3. Context-aware architecture

The context information can be retrieved at least from the following type of sources:

- sensors connected directly to the embedded device.
- sensors connected to a close device or computer.
- actuators which generate context information.

The following architecture is proposed in order to support the context awareness of the pervasive system. Figure 3.1 depicts the main components needed to build a distributed context awareness system.

The key-component is the ContextService. It is responsible for getting all the context information that an application running inside the embedded device might need. It retrieves the context information from the sensors/actuator directly connected to it or it gets them from the “buddy” ContextService.

The “buddies” ContextServices consist of a collection of services distributed on different machines which collaborate in order to retrieve all the context information needed by the embedded applications. The buddies can be defined statically or dynamically. For the static configuration, we can define a list of QoS parameters which need to be used when the buddies are retrieved from the DiscoveryService. The dynamic configuration means that the buddies start working together once there is any request of accessing information about the context from another target.

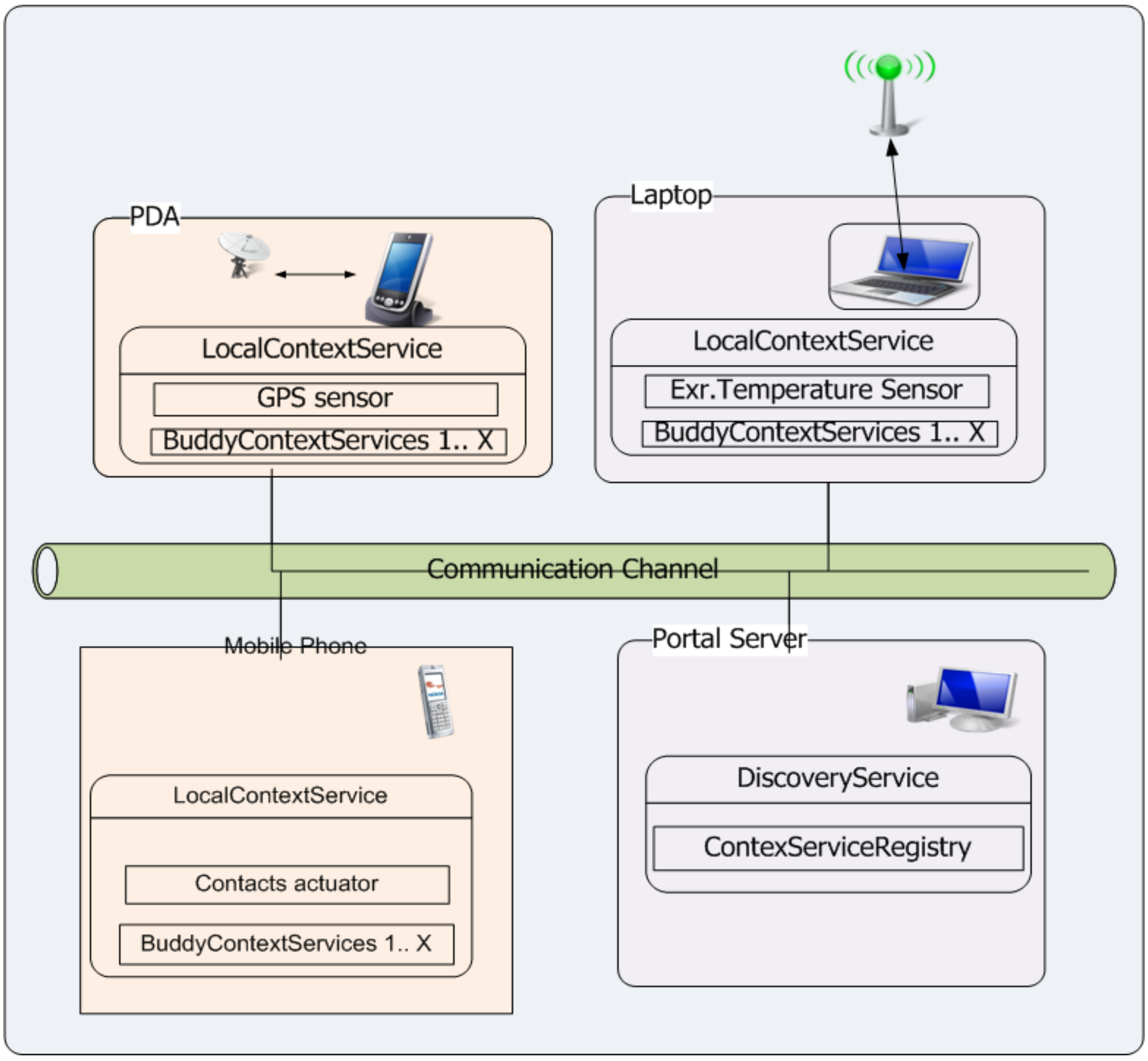


Figure 3.1 - Architecture overview

The buddy ContextService is a smart proxy that is retrieved from the DiscoveryService. The DiscoveryService analyses the QoS parameters transmitted by the buddy ContextService during the requests for a new ContextService and gives the client the ContextService which is most appropriate for his/her needs. The proxy is considered to be smart because the DiscoveryService could replace it in case it does not provide the expected functionality. This way, when the expected QoS are no longer assured by the current buddy, the DiscoveryService is responsible to replace it with another buddy. Consequently, the system assures the local scalability that is an important requirement for pervasive systems.

Each ContextService can be directly connected to one or multiple sensors and actuators. The ContextService from that computer will communicate directly with the sensor, and will publish the sensor values to the entire system. The sensor contains the specific implementation of connecting and retrieving information from the physical sensor.

Actuators are used to push the context information into the system on users' demand. The actuators do not really sense the environment, but they provide information that can help the user's intention. For example: presence actuator notifies the other users that the current one is just joining/leaving the community.

3.4. Concluding remarks

The current chapter aimed at presenting the context-awareness feature, one of the most important challenges for a pervasive system. The architectural challenges triggered by the need of supporting context-awareness have been identified. These are the following: the *integration of different context providers, extensibility, distribution and cooperation, event-based infrastructure, simple domain mobile, simple query interface, dynamic discovery and registration*. Some of the existing pervasive middleware that support context-awareness have been analysed from the perspective of the above identified challenges. The existing solutions can only partially solve the target problems.

A new virtual architecture for implementing a pervasive system which is aware of the context has been detailed. The context-awareness challenges are not new, but the solution proposes some innovative aspects which bring some improvements compared to the existing solutions:

- distributed context provisioning – the context data can be retrieved from the buddy context services and not only from the embedded device.
- it supports local scalability by facilitating the replacement of the context service, based on the QoS parameters.
- transparent context provisioning – the client can ask for a given context value without knowing where that context needs to be retrieved from.

- flexible query mechanism which allows the external application to have a simple and flexible way of getting the context values.
- flexible mechanism for reaction to the context change, which allows the external application to configure their reaction, both statically and dynamically, in case of context change, and to adapt to the new context values without disturbing the user.

The proposed middleware is base on the idea of considering the distribution of the context providers and the possibility to integrate them in a system which is able to monitor their QoS parameters. If the middleware observes that certain provider do not function normally, it replaces the provider by an equivalent one without interrupting the user activities. The context data can be retrieved from the buddy context services which can be located into the near network and not only from the embedded device, as previous middleware had considered. By letting the ContextService instances to be hotreplaced by the DiscoveryService the middleware supports the local scalability. The middleware enables transparent context provisioning, thus the client can ask for a given context value without knowing from where that context needs to be retrieved.

The outcomes of this chapter are summarized in the following studies: (Presecan & Tomai, 2009), (Presecan S., 2009), (Presecan S. , 2009a)

4. Invisibility

Context-awareness is the first step in achieving the “ubiquity” when using computational power. A context-aware system observes the environment and it can react on its changes. The ways in which it reacts could make the system to be really pervasive. The present chapter details the *“invisibility”* as a key-concept in pervasive computing. The ways of achieving “invisibility” are herein detailed and a new solution is presented in order to improve the current available solutions.

4.1. Invisibility concept

Due to the aim of having a proactive reaction to the change of context, the pervasive systems can tend to be annoying, notifying the user with any context change. Adaptation to the environment changes should be made smoothly by the system without too much interaction with the user. It is the role of the pervasive system to facilitate adaptation, which may involve adapting individual software components and/or reconfiguring bindings of components by adding, removing or substituting components. In what the pervasive system is concerned, the adaptation tends to be made more in an application-aware manner and less in a user-aware manner.

Having such a system, the interaction between the client application and the system becomes invisible for the user; most of the adaptation actions are made by the client application, without interrupting the operations performed by the user. The challenge consists in finding the proper method of integrating these heterogeneous technologies into one system, in order to let different connectivity channels open for different types of devices.

4.2. Invisibility Challenges

The pervasive system should be configurable in such a way that only specific conditions could lead to interactions with the user. The user should be able to configure the system in such a way that it should react in various given ways to the change of context.

In order to assure the invisibility and application adaptability, the middleware should face the following challenges:

- *Support for static configuration* for the reaction rules – the application should be able to pre-configure, using a simple configuration language, the reactions to the context change.
- *Support dynamic configuration* for the reaction rules – the external application should be able to register and configure the reaction rules based on simple logic reasoning. The configuration should be done using a simple descriptive language which is open to add new filtering conditions.
- *The rule processing should not affect the overall system performance.* Thus, the system should process the events and perform the reaction to these rules in a smart manner, in order to avoid reducing the overall system performance.

4.3. Detailed design

Starting from these challenges, a simple rule engine was designed with the aim of supporting the application adaptability. Figure 4.1 describes the main classes needed to implement an efficient rule engine.

The key class is the *Rule*. It is used to model a reaction rule. It contains an EventFilter and the Action that need to be performed once the received event passes the filter. The Rule, EventFilter and Action follow the ECA (Event-Condition-Action) pattern. (Ipina & Katsiri, 2001)

The *Rule* expresses a simple IF-THEN-ACTION directive:

If EventFilter then Action

Based on this directive, it is pretty simple to build a pre-configured manual of the system reactions or a dynamic configuration for system reactions. The user can easily define his/her one preference for reactions to the changes in context.

Once the rule is defined, it needs to be registered to the RuleManager. The manager supports 2 ways of registering the rules:

- dynamic based registration – when the application creates the rule and then adds to the manager.
- configuration based registration – at start-up, when the RuleManager reads a configuration where the rules are stored, by using a simple descriptive language.

A simple descriptive language could look like:

rule := name conditions action

*conditions := condition **

condition := contextParam operator value

operator := = | > | < | >= | <=

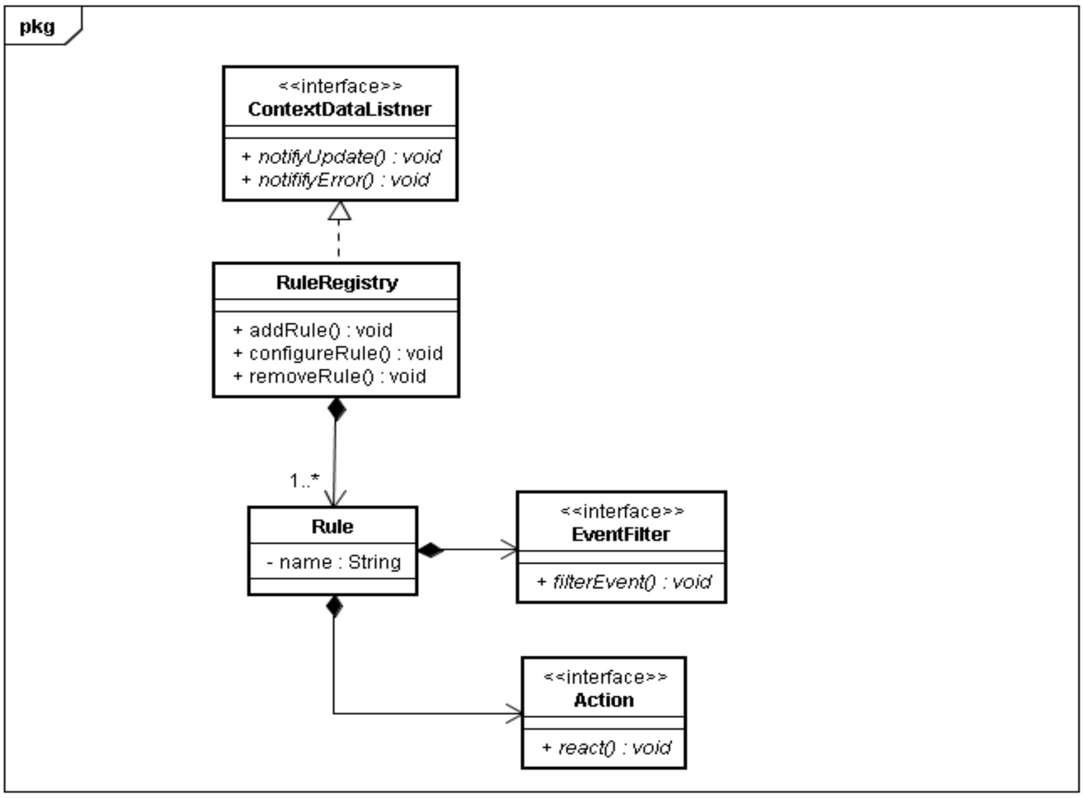


Figure 4.1 - Rule Engine Class Diagram

The RuleManager is a ContextDataListener; during start-up, it finds the appropriate ContextService and registers itself as a ContexDataListener, in order to be able to receive any notification concerning the context change. When the event is received from the ContextService, the RuleManager looks through all the registered rules in order to check for which of them the event passes the configured EventFilter. If it finds some rules for which the event is selected by the EventFilter, it consequently starts to execute the Actions defined for those rules. In order to avoid the computational power, those actions can be executed synchronously or asynchronously, based on a configuration.

4.4. Concluding remarks

Any pervasive system should observe and adapt itself to changes within the environment. Seamless adaptation to the context changes is a must in achieving an “ubiquitous” experience. The current chapter described a simple way of reacting to environment changes. The middleware could adapt to the change based on user’ preferences or based on predefined reaction strategy. The proposed middleware design matches the architecture defined for implementing a context-aware system. By combining the context-aware and the invisibility features together, the proposed middleware helps the application developers with implementing applications which are smart enough to adapt themselves to the heterogenous and dynamic environment specific to a pervasive system.

The outcomes of this chapter are summarized in the following studies: (Presecan S. , 2009), (Presecan & Tomai, 2009)

5. Application Mobility

In a pervasive environment, the users and the devices are in a continuous movement. The information is collected from various sources that could change their location and even more, the data connectivity is moving from one provider to another. In this context, the application and data mobility are key-features that need to be supported by a pervasive middleware.

A new concept called “cloaudable application” is introduced with the aim of defining a new way of achieving the application and data mobility.
The final section of the present chapter presents a concrete solution for implementing cloaudable applications.

5.1. Mobile cloudable applications challenges

Pervasive computing aims at helping applications to run in mobile environments, by adapting themselves to the environment changes. Any pervasive application should be able to run on mobile devices that have a limited computational power and which lack good Internet connectivity. Due to the limited resources and due to the aim of being a real replacer for personal computers, mobile devices should be able to run applications that are distributing their tasks into the cloud for execution.

In order to make a clear distinction between normal mobile applications (which run on mobile devices and use programmatically resources from certain devices or from the Internet) and the new mobile applications that are performing the user requests by splitting the execution into small tasks, we suggest we call the last type of mobile applications: *“mobile cloudable application”*.

A mobile cloudable application - is a mobile application that is able to migrate into the cloud in order to perform the user tasks in a better way, by using the vast computational power available into the cloud.

Any framework used for developing mobile cloudable applications that are used into the pervasive environment should aim at fulfilling the following requirements:

- *enable the development of context-aware application* – The mobile users are characterized by different contexts: local, social, behaviour. The pervasive applications are using these contexts to understand user's habits and to adapt the task execution to the user's preferences.
- *react on changes within environments* – during their execution, the applications should react on the environment changes. The mobile cloudable applications are embracing the "react-and-adapt" strategy as to adapt their execution to the environment changes.
- *transparent distribution of the execution into the cloud* – the application developer should not be constrained to use custom libraries in order to manage the service distribution. The distribution of tasks into the cloud should be transparent to the application developer and it should be achieved by using the framework.
- *application and data mobility* – the code written for the mobile application should be transparently migrated to the cloud. The code and data should migrate dynamically at runtime and should be requested in order to use the resources of the mobile device optimally.
- *application elasticity* - the cloud services will be used on demand only when the local resource would not be able to meet the expected QoS. The development framework should measure the QoS and should react upon the lack of resources by distributing the execution over the cloud

The mobile cloudable applications are meant to use the mobile device resources optimally and when the device is lacking in computational power or when the execution of tasks within the cloud becomes optimal, they are distributing the tasks execution transparently into the cloud.

5.2. Framework architecture

In order to fulfil the above mobile cloudable application challenges, the framework architecture described in Figure 5.1 could be used for the development of a new generation of mobile applications.

The main components of the herein proposed architecture are:

- *Mobile application* – the application is developed using the mobile platform specific framework and programming language: Java for Android, Objective C for iPhone/iPad, C# for WP7. The mobile application interacts with the user by using a user interface that is specific to the target mobile device. In order to fulfil user requests, the mobile application uses different services located on the mobile devices. The services are either system services or custom services.
- *Local services* – are implementing the granular task used to fulfil user requests. The services could run on local mobile device but could also be run on cloud. The services should implement the pure business logic and should not depend on the user interface.
- *Distribution service* – one of the core components of the architecture. It monitors the QoS of the system and it provides at runtime the appropriate implementation for a given local service. The Distribution Service is not to be used directly by the application. The Distribution Service communicates with the Distribution Service from the cloud in order to manage the lifecycle of the Riders.
- *Rider* – it is used by the Distribution Service in the migration process of the Local Service into the cloud. The Rider contains the running data needed for the execution and the implementation of the local service. Once the Rider installed on the cloud, it will return to the Distributed Service a smart-proxy to the corresponding service running into the cloud. The smart proxy is going to be used by the application as a replacer for the local service. The Rider serves as data and code migration container and is used to support application mobility.

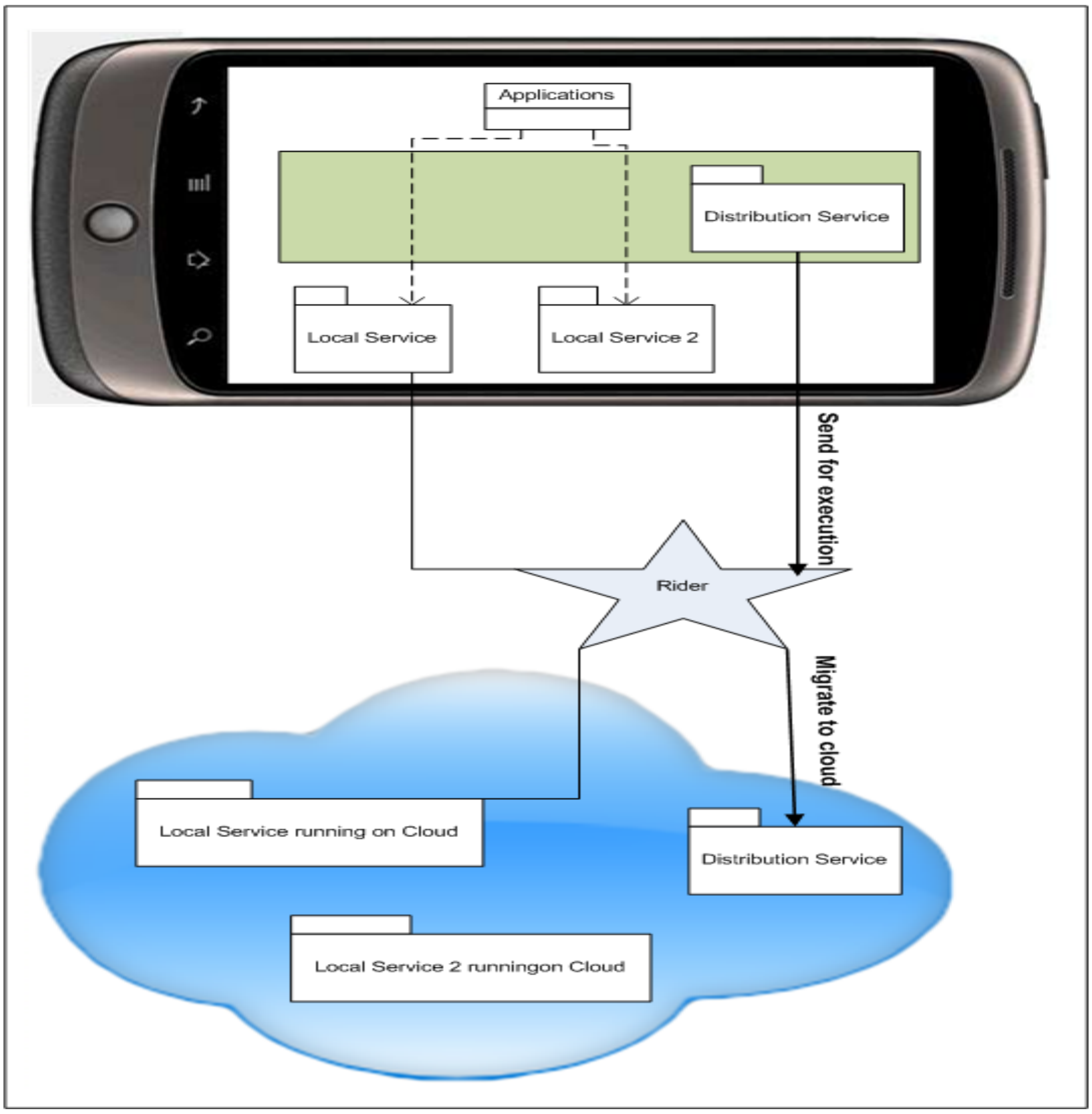


Figure 5.1 - Mobile cloudable framework architecture

The Distribution Service and the Rider, support the application mobility and application elasticity. The Distribution Service is constantly monitoring the system QoS and delegates the calls to the Local Service located on the cloud, if be the case.

The Distribution Service intercepts the calls to local service. From this perspective, the proposed architecture treats the application distribution through the cloud as a crosscutting concern. The cloud distribution is separated from the development of the mobile application. The best way of implementing such architecture is to use the innovative Aspect Oriented Programming Paradigm.

5.3. Concluding remarks

The application mobility is one of the key features of any pervasive application. The pervasive application needs to be able to run in different environments and furthermore, it has to be deployed seamlessly in different environments without the user’s intervention. The current research proposes a solution for the migration of the mobile application into the cloud without the user’s intervention, in a way that is transparent to the application developer.

The resources available within the cloud are incomparable as performance with the resources available on mobile devices. The architecture described in the current study proposed a seamless solution for executing user tasks into the cloud if the local mobile resources are not enough to offer a good user experience. The proposed solution envisages the running on the cloud of the application that has been especially developed for mobile devices. The proposed framework comprises the implementation for seamless integration within the mobile application, as well as the distribution and the deployment of the mobile applications into the cloud.

The proposed architecture presented in the current chapter provides details of an innovative way of achieving application and data mobility. The collaborative architecture that proposes the use of the cloud for performing mobile operations is proved to be viable both from the technical and from the performance point of view. Evolving from the status of experimental solution to the status of commercial product, the presented mobile cloudable application proves it feasibility. Mobile cloudable application is clearly a viable solution for distributing the application services into the cloud into a way that is transparent to the application developer. The proposed solution does not force the application developer to use a very custom framework, it is open and it integrates seamlessly into a mobile application. They need to focus on solving the application business and not how the application is distributed into the cloud. This task in supported by the proposed middleware.

The current research is going to be extended to a more abstract level in order to build an architecture that could virtually be applied cross-mobile platforms.
The outcome of this chapter is published on: (Presecan S. , 2011)

6. Security

Pervasive networks foresee communicating and computing devices embedded throughout our environment. This will cause huge increases in the complexity of network infrastructures and information services available within these infrastructures. Therefore, the challenge of managing information services while maintaining security and privacy will be quite high.

The very same features that make pervasive computing environments convenient and powerful make them vulnerable to new security and privacy threats. Traditional security, mechanisms and policies may not provide adequate guarantees in dealing with new exposures and vulnerabilities.

The Security is a key-feature for a pervasive system in which the devices/users move continuously through a heterogeneous environment. A middleware that could be used as a support for implementing pervasive systems should face the following security challenges: confidentiality, privacy, integrity, availability, authentication, transparency, context-awareness, interoperability, freshness, user mobility, single sign-on. Some of these challenges are specific to pervasive systems, others are general security challenges.

6.1. Detailed design

It is essential that the middleware used for implementing the system within a pervasive environment, support different security strategies. It thus allows the application developer to use some of these strategies based on the security needs of the targeted application. The security should be handled as cross-cutting concern since it should be applied transparently to different operations.

The most important classes that could be used to define a security framework needed for implementing a middleware that supports pervasive applications are presented in Figure 6.1:

- *Security manager*: it is the main class used to perform the most important security operations: *authentication and authorization.* The SecurityManager could be used directly by the application in order to validate the user access or indirectly by the security interceptors that could be applied as cross-cutting concerns in the execution process of different application calls. The SecurityManager itself is a listener to the context changes. The SecurityManager contains one or multiple *SecurityStrategies* which could be used to perform the operations based on different types of configurations.
- *SecurityStrategy*: contains specific configurations that could be used in order to trigger different behaviours in what security is concerned. Each SecurityStrategy contains different *SecurityProviders* that are in charge with the execution of the security operations. Based on their configuration and based on contextual information, the SecurityStrategy decides whether it could handle the security requests or not and which of the attached providers could be used. The SecurityStrategy contains the logic needed to decide whether the User information is still accurate or not. The SecurityStrategies are able to decide if the user needs to re-authenticate or if his/her authentication is still valid. The SecurityStrategy could poll the information about the context from the *ContextService* defined in Context-aware architecture.
- *SecurityProvider*: it is responsible for the implementation of a specific way of handling the security operations. The SecurityProviders could be shared between applications and they are able to perform the security operations by using a specific protocol/logic. For example: they could perform the authentication/authorization by using an external WebService. By default, the middleware could provide SecurityProviders for open protocols: *OpenId* or *OAuth*. These providers could be used in order to implement the security operations for different systems that are using these standards.

A special *SecurityProvider* is the *AccountManager*. The *AccountManager* could be used to store user access tokens and authorization tokens which could be shared between multiple applications. By using the *AccountManager* the users have to log-in once for a specific system and then they could use different applications to connect to the target system's services. Thus, the "single sign-on" challenge could be easily be supported by the pervasive middleware. The *AccountManager* acts as a shared local storage that stores authentication/authorization information.

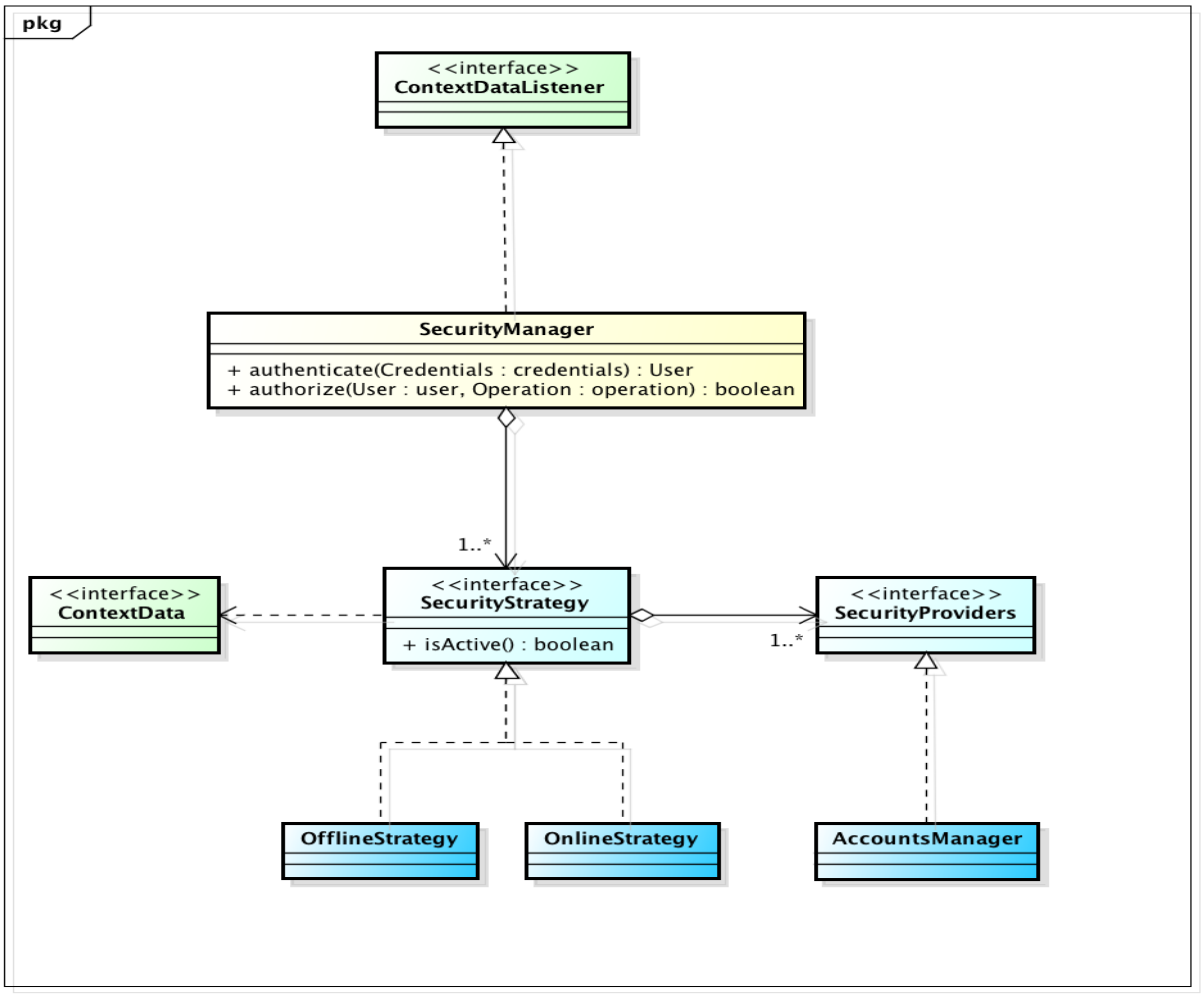


Figure 6.1 - Security framework design

6.2. *Concluding remarks*

The proposed solution aims at solving the security challenges which are specific to context-aware systems. The proposed design supports the development of different security strategies that could be implemented by different context-aware applications. The proposed solution does not address the data encryption or data signing. They need to be addressed by the middleware that ensures the data transport and data integrity.

The proposed solution fits well with the context-aware middleware defined in the Context-awareness chapter. The current solution will be extended and integrated in order to provide a complete middleware which supports all the challenges identified for a pervasive system.

The outcomes of the current chapter have been partially published on (Presecan S. , 2008)

7. Pervasive middleware architecture

The previous chapters have detailed different small architectures used for solving specific pervasive system challenges. The present chapter comprises a case study that aims at identifying the main challenges for a pervasive system used for solving day-by-day problems in a community of students. Once the challenges are identified, an architectural solution is proposed in the view of solving almost all of the identified challenges.

The architectural solution aggregates the small designs proposed by the previous chapters. The result is a generic architecture that could be used for addressing different business domains in which the usage of computational power is ubiquitous.

7.1. Vision

Almost all universities have currently implemented educational portals from where students can get relevant information. Most of these portals are web-centric, exposing the information on the web. In many cases, this is sufficient, but there are improvements that can be made in order to facilitate the collaboration between students and the university. Almost all the e-portals offer one-way interaction. Students directly access the information published on the educational portal, via the web interface.

Considering the current progress in the world of hardware and software engineering, these options are not enough. Nowadays, smart-phones and social networks are ubiquitous. Students want to be informed everywhere and they want to share information with their colleagues in a away that is ubiquitous. The usage of social networks is sometimes much easier than sharing information by using the traditional channels such as the email and the telephone.

The educational system should be also aware about students' profile. Based on these profiles, students should receive certain broadcast messages or have access to certain information.

Students and professors should be linked by a virtual network which is aware of their preferences and profiles and where they could exchange information from any device, from any location. Moreover, the virtual network should provide information security and the information should be easily moved from one device to another.

7.2. Challenges

Starting from the vision and considering the pervasive challenges described in Paragraph 2.2, the following major challenges should be considered by any system developer who aims at implementing a pervasive system which has the features described in the Vision:

- *Context-awareness:* the system should be aware of student's profile/context: geo profile, social profile, environmental context, temporal context.
- *Invisibility:* the system should react on context changes. Each time when the user context is changed, the system is aware about the change and tries to adapt system functionalities to the new context.
- *Application mobility:* On almost all the existing educational portals, data is transferred to the user. The data is mobile, the application is not. There is also a need for moving the applications between difference devices. The users expect to continue their work on different devices.
- *Security:* The need for application mobility and for context-awareness is exposed on different attacks. Users want to perform all the above-described operations in a secured manner.
- *Integration:* A pervasive educational portal is a heterogeneous system which integrates different types of computational systems: servers for storing data and for executing user tasks, laptops and personal computers for accessing the information form fixed locations, smart phones for mobile access. All of these need to be integrated into the pervasive system.
- *Local scalability:* the number of devices connected to the pervasive educational portal can substantially vary based on the location and based on time factors. There are also periods when the system is under the usage stress. The system should support local scalability in order to face all these challenges.

- *Development and deployment:* the system middleware should help developers to implement applications that are able to use the information received from different sources, with different protocols. The middleware should ease the development and the deployment of the application on different types of devices that run different operating systems. The middleware should also provide services and functionalities that help the application developers to reduce the overall costs in the application development.

7.3. Middleware architecture

Starting from the Vision and considering the above-identified challenges, the following architecture is proposed for implementing a pervasive system that could be used for implementing the next version of educational portals. The architecture combines all the designs presented on the previous chapters. The solution aims to solve all the challenges that needed to be solved by a generic middleware used for pervasive systems.

The middleware architecture presented into Figure 7.1, uses the following main components:

- ContextService
- DistributionService
- SecurityManager
- Communication channel

7.3.1. Context Service

ContextService is the component used to implement the context-aware and the invisibility functionalities. *ContextService* observes the environment by the usage of sensors and actuators. The sensors could be embedded into the smartphone such as the GPS sensors, or external sensors. Via the sensors the ContextService is notified whenever the information collected by these sensors are changing. ContextService is also connected to actuators that provide information on-demand about different contexts such as: social context, functional context, temporal context.

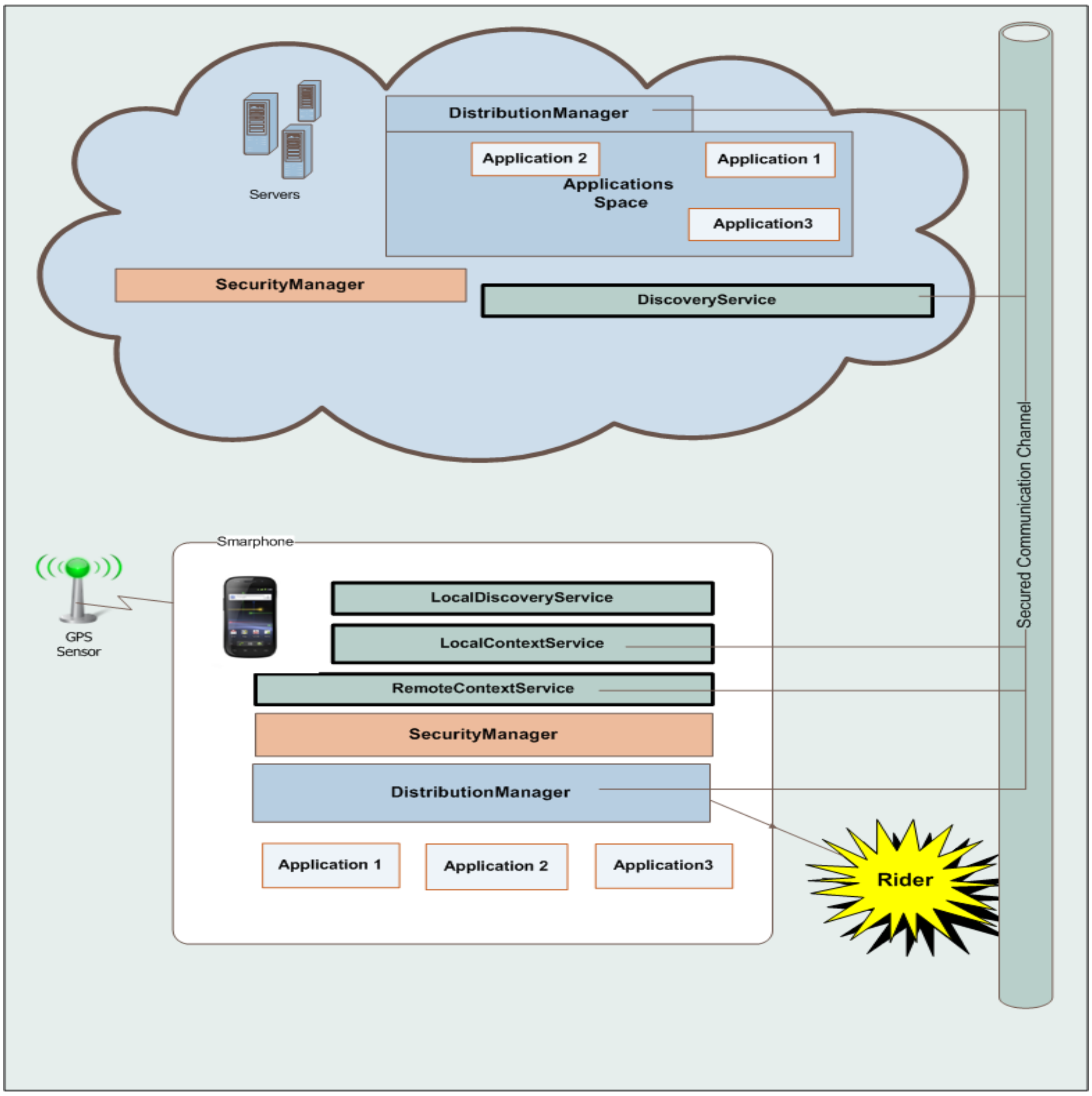


Figure 7.1 - Middleware architecture

7.3.2. Distribution Service

The users expect to continue their activities by using different computational power environments. They want to have the application and data migrated from one device to another-one, and they want to have the proper usage of the computational power existing into a certain system. The middleware is in charge of migrating the data and the application to the corresponding device/server for performing the user tasks. The application mobility is transparent for the user. They users do not want to be bothered

with this kind of activities; they simply want to perform certain tasks by using at maximum the computational power.
The *DistributionService* performs the distribution of application together with their data and context. The DistributionService knows how to find a better place for performining certain tasks by monitoring the QoS values of different devices and servers connected into the system.

7.3.3. Security Manager

SecurityManager performs all the operations need to support: authentication, authorization and confidentiality. SecurityManager uses different strategies to authorize the user access. Into a pervasive system the security is a crosscutting concern that need to be applied transparently to different system's operations. SecurityManager supports the implementation of these concerns providing the following security operations: authorization and authentication.

7.3.4. Communication channel

An educational portal that is ready for a pervasive environment needs to integrate different types of devices, sensors, actuator, computers. Each of them uses different protocols and communication channels. The integration efforts without having defined a clear communication channel, is a nightmare. The communication abstraction used to integrate different application running on different devices is a major challenge for pervasive computing.

The communication channel is able to deliver securely data that are retrieved on demand by the external system clients or by different devices that are connected into the system. The polling interface is exposed as RESTfull webservices over HTTP/HTTPs. Via the RESTfull webservices the data could be read/updated/deleted by the devices or subsystem connected to the middleware.
The context notification over HTTP might not be that easy. By default, the HTTP does not use the push of the data. There are several technologies/frameworks that have solved this problem.

One of the solutions is to use the HTTP long connections and use these connections for sending bidirectional message. Such example is **Bidirectional-streams Over Synchronous HTTP (BOSH)** BOSH, the technology defined in this specification, essentially provides a "drop-in" alternative to a long-lived, bidirectional TCP connection. BOSH is designed to transport any data efficiently and with minimal latency in both directions. For applications that require both "push" and "pull" semantics, BOSH is significantly more bandwidth-efficient and responsive than most other bidirectional HTTP-based transport protocols and the techniques. (Paterson, Smith, Saint-Andre, & Moffitt, 2010)**.**

The technique employed by BOSH achieves both low latency and low bandwidth consumption by encouraging the connection manager not to respond to a request until it actually has data to send to the client. As soon as the client receives a response from the connection manager it sends another request, thereby ensuring that the connection manager is (almost) always holding a request that it can use to "push" data to the client.

The Figure 7.2 details the ways of interacting with the communication channels:

- ContextService expose the data by the usage of QueryInterface. The QueryInterface is defined as a RESTfull webservice which runs over HTTPS
- QueryInterface is able to register notification listeners by using HTTP notification. The HTTP notifications are achieved by using the BOSH protocol. By using the BOSH protocol the application could be notified anytime when certain context values are changed and the notification is sent over the standard HTTP.
- Certification Authority could be used by certain applications to issue certificates which could be used then to sign sensitive information
- Applications/users are connecting over HTTPS to the RESTfull services in order to get the information exposed by different public services

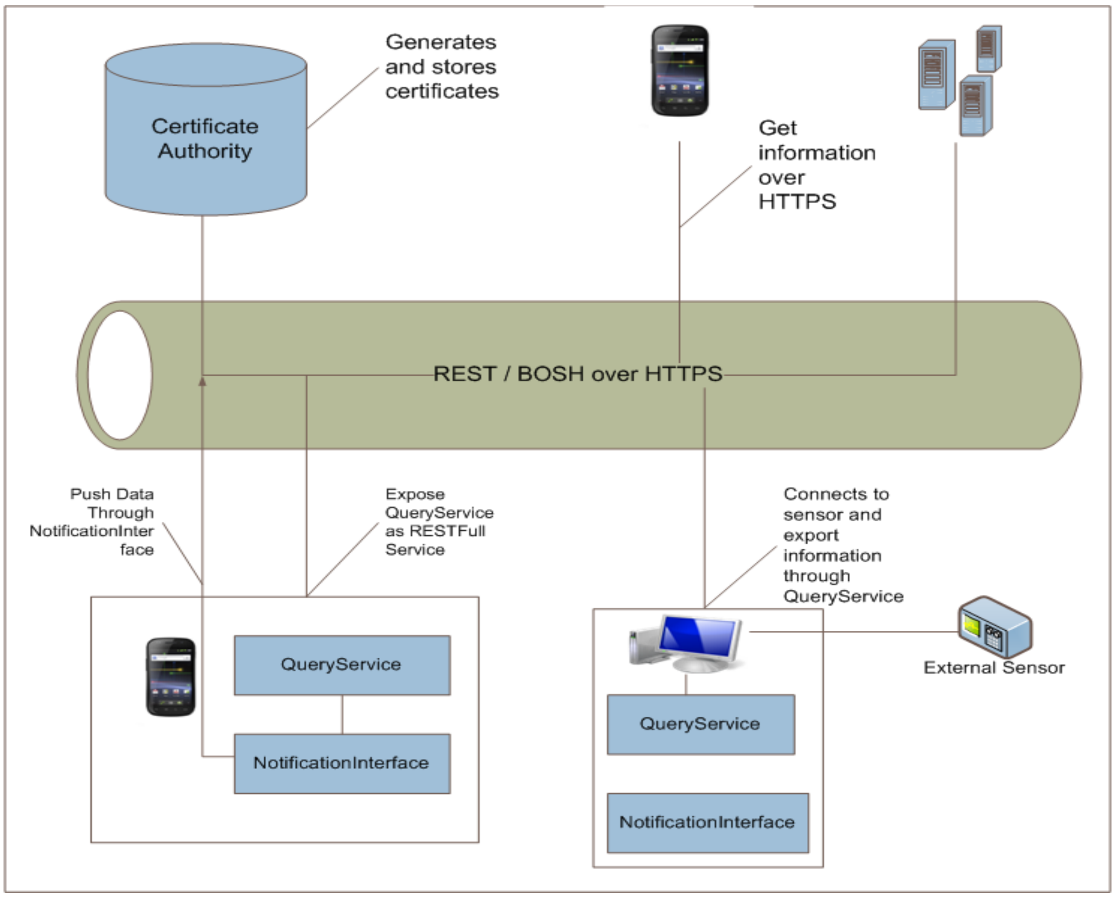

Certificate Authority
Generates and stores certificates
Get information over HTTPS
REST / BOSH over HTTPS
Push Data Through NotificationInterface
Expose QueryService as RESTFull Service
Connects to sensor and export information through QueryService
QueryService
NotificationInterface
QueryService
NotificationInterface
External Sensor

Figure 7.2 - Communication Channel

The purpose of any middleware is to support implementation of the overall system features and to ease the life of the developers. Having a communication channel that is based on HTTP and BOSH, it is good from the integration point of view. Thus the application developers could easily written pervasive application which could connect each other in order to perform the user tasks.

7.4. Concluding remarks

Nowadays, the personal computer is disappearing; it becomes a tool that is used by the users to perform certain tasks. Currently the focus is on information and on the applications/systems that facilitate the access to the information. The pervasive education portal is a challenging task, which starts first with a vision that is detailed into this chapter. Following the vision, we have identified the challenges that need to be solved in order to implement a pervasive educational system.

Starting from the vision and challenges, we have proposed architecture for middleware that could be used for implementing a pervasive educational portal.

The proposed architecture combines all the components described into the previous chapters, and provides consistent ways of solving identified challenges. The proposed middleware is generic, thus it could be used for implementing any type of pervasive system.

The proposed architecture solves some of the most important pervasive challenges: context-awareness, invisibility, integration, application mobility, security and deployment. The proposed middleware is based on open-standards and does not lock-in the developers for using custom framework. The proposed standards have a big community support, which ease the overall costs of the solution.. The proposed communication protocols (HTTP and BOSH) for enabling both Push and Pull of the date are innovative and helps the applications to be used almost in any network, since HTTP is the internet de-facto standard and BOSH works on top on HTTP.

8. Conclusions

The present research aimed at contributing to the theoretical approaches within the domain of pervasive computing, but in a way that could provide tangible results, as well as assistance for those specialists who will actually implement pervasive systems. Pervasive computing is still a young field of research and there are no de-facto standards for developing pervasive systems. In order to have a significant contribution by providing a middleware which could be used for implementing pervasive systems, we have established two main objectives: understanding the current status of the pervasive middleware and design a generic middleware which could be easily implemented by any pervasive system.

The first few chapters of the present thesis focus on understanding the general challenges one must fulfill in implementing a pervasive system. We have thus identified the key challenges for a pervasive system: context-awareness, invisibility, service discovery, integration, security, application mobility, development and deployment. These key concepts are detailed in chapter 2. The second chapter also focuses on presenting the status of the latest research in the domain of pervasive middleware and the way in which theoretical research can provide solutions in implementing systems that could solve the above listed challenges. The second chapter does not claim to be a complete analysis of all the existing pervasive middleware. Nevertheless, we have tried to focus on those that we considered to be the most important middleware designed for pervasive computing. The existing middleware are analyzed from the method perspective, that is to say, the way they overcome the identified challenges. By analyzing them thoroughly, we have come to the conclusion that these middleware actually represent a good starting point for gathering ideas about how the ideal middleware should look like.

The second objective of the present research is designing a pervasive middleware and we have focused on fulfilling our goal starting with the third chapter. Context-awareness is a major key concept in pervasive computing; it is one of the key differentiators between mobile computing and pervasive computing. Our endeavor of creating a smart middleware for pervasive computing has started with providing the design for a particular part of the middleware that aims to solve the context-awareness challenges. The proposed design brings forward the idea of having a “buddy” ContextServices which could collaborate for providing information about the environment. The distribution of the “buddies” is transparent to the application developer and the middleware is able to discover and link the buddies together. The DiscoveryService which monitors the QoS could replace the “buddies” ContextProviders at runtime. This dynamic replacement also ensures the localized scalability which is a major challenge for a pervasive middleware.

Together with Context-awareness, Invisibility is a mandatory key feature of any pervasive system. The fourth chapter provides details about the Invisibility challenges and about the ways of supporting it. The proposed middleware could contain configuration of reactions to the context changes. The RuleManager itself is a ContextListener, therefore, it is notified whenever the monitored context values are changed. The RuleManager could be configured statically or dynamically with the reaction rule. Thus, the system is learning the user’s behavior and preferences and it could react on context changes on behalf of the user. The system adaptation without user’s intervention is the key feature in supporting invisibility.

Applications Mobility is a key differentiator between distributed and pervasive systems. It is a challenge pretty difficult to achieve, given the heterogeneity of devices and the security concerns. In the fifth chapter, we have proposed to develop an innovative way of supporting the application migration from smart devices to other devices or even in the cloud. By using the DistributionService, the Rider and the custom ClassLoader, the application services could migrate to a cloud or to any nearby more powerful device in order to have access to a higher computational power for executing user tasks. The application mobility is supported by the middleware and it is implemented transparently for the application developer. By using the proposed AOP paradigm, the distribution in the cloud is implemented as a cross-cutting concern. By using the distribution advice, the application developers could annotate their services as being distributed in the cloud or on more powerful devices. The distribution in the cloud could be decided automatically by the middleware through monitoring the QoS services. To the end-user, the services distribution in the cloud is thus made in a transparent and silent way. By making the distribution when then QoS of the services are under certain agreed values, the middleware support the localized scalability.

Chapter 6 provides detailed information regarding Security implications in the pervasive world. The discussions concerning Security are further taken to presenting our proposal for the supporting Security in a pervasive middleware. Security is perceived as a cross-cutting concerned, thus the authorization could be implemented by using AOP advices.

The middleware architecture is defined and detailed in Chapter 7. The architecture combines all the designs presented in the previous chapters and proposes a generic middleware that could be implemented for any pervasive system. The middleware aims at overcoming the pervasive computing challenges defined for an educational pervasive system. The middleware supports context-awareness, invisibility, security, application mobility, local scalability and integration. It does worth mentioning the innovative way of pushing the context values to the devices/systems that are interested in their values. Using the BOSH http push protocol makes the push of the events containing context values. The events are thus distributed to the subscribers by using the HTTP transport. The proposed middleware is based on open-standards such as: REST, BOSH, HTTP(s) and therefore, it does not lock-in the developers by forcing them to use a specific standard. The usage of open-standards is the way of supporting the integration of different types of devices running on different operation systems.

Comparing the proposed middleware to the analyzed existing systems: AURA, Gaia, One.World, Pico and SODA, it is obvious that it has a better coverage in terms of supporting different pervasive specific features. The analyzed systems are able to fulfill only certain challenges; they have not been created with the aim of providing an end-to-end solution for pervasive systems.

Through the present research thesis, we have proposed a state-of-the-art solution for a middleware which focuses on covering all the major aspects of pervasive computing. Our solution aims at being complete and we believe it is a unique one because of its innovative proposals, such as:

- buddy context providers which are interconnected into a net of providers and which are replaced dynamically whenever one of them does not perform well or is disconnected from the network
- transparent tasks distribution into the cloud for achieving local scalability and application mobility
- using AOP for implementing a seamless application distribution
- using BOSH over HTTP for distributing the events

Even if the solution has been implemented and tested partially as lab examples and partially is part of a pretty successful commercial product, the current research needs to be extended in order to support multiple programming languages and multiple platforms. The lab examples have been implemented using Java and Android, but in order to prove its completeness, the current work might be enriched and further developed in order to ease the integration and development of the application using different platforms.

9. Bibliography

[1] Adams, A. (2003). A whole picture is worth a thousand words. *SIGCHI Bulleting (Supplement to Interactions)* , Volume 2003, May-June , pp. 121-124.

[2] Adams, A. (1999). Users' perception of privacy in multimedia communication. *Conference on Human factors in computing systems* (pp. 53-54). ACM Press.

[3] Adams, A., & Sasse, M. (1999). Privacy Issues in Ubiquitous Multimedia Environments: Wake Sleeping Dogs, or Let Them Lie?, INTERACT, pp. 214-221.

[4] Adelstein, F., Gupta, S., Richard, G., & Schwibert, L. (2005). *Fundamentals of Mobile and Pervasive Computing.* New-York: McGraw-Hill.

[5] Anthony, D., Joshua, M., & Tauber, A. (1997). Mobile computing with the rover toolkit. *IEEE Transactions on Computers*. Volume 46, pp. 337 – 352

[6] Appweb. (2007, 01). *AppWeb Server*. Retrieved 08 2011 from AppWeb Server: http://appwebserver.org/

[7] Aura, P. (2002). *Project Aura*. Retrieved 07 2008 from http://www.cs.cmu.edu/~aura/

[8] Bapat, S. (1994). Object-Oriented Networks, Models for Architecture, Operations and Management. New-York: Prentice-Hall International.

[9] Becker, C., & Schiele, G. (2003). Micro-broker- based middleware for pervasive computing, *Pervasive Computing and Communications (PerCom),* pp. 443 - 451

[10] Bellotti, V. (1997). Design for privacy in multimedia computing and communication environments. Massachusetts: MIT Press.

[11] Bellotti, V., & Sellen., A. (1993), Design for Privacy in Ubiquitous Computing Environments, *European Conference on computer supported cooperative work*, pp. 77-92

[12] Birrell, A., & BJ, B. N. (1994). Implementing remote procedure calls. *ACM Trans Computer Systems ,* Volume 2 (1), pp. 39-59.

[13] Brandram, J., & Hansen, T. (2004). The AWARE architecture: supporting context-mediated social awareness in mobile cooperation, *Proceedings of the*

2004 ACM conference on Computer supported cooperative work, pp. 192 – 201

[14] Campbell Roy, Al-Muhtadi J., Naldurg P., Sampemane G. and Dennis M. (2003), Towards Security and Privacy for Pervasive Computing, *Software Security — Theories and Systems,* Volume 2609/2003, p. 77-82

[15] Carriero, N., & Gelernter, D. (1986). *The s/net's linda kernel.* ACM Trans Comput Syst .

[16] Chan, E., & Bresler, J. (2005). Gaia Microserver: An Extendable Mobile Middleware Platform. *Third International Conference on Pervasive Computing and Communication*, Santa Cruz: IEEE, pp. 309-313

[17] Chappell, D. (2006). *Understanding .NET (2nd Edition).* Addison-Wesley Professional .

[18] Cox, P. A. (2011). *developerWorks*. From Mobile cloud computing: http://www.ibm.com/developerworks/cloud/library/cl-mobilecloudcomputing/

[19] Crane, D., & McCarthy, P. (2008). Comet and Reverse Ajax: The Next-Generation Ajax 2.0. Apress.

[20] Dey, A., & Abowd, G. (1999). *Towards a better understanding of context and context-awareness.* Georgia: Georgia Institute of Technology.

[21] Doceur J. (2002), The Sybil Attack, *Peer-to-Peer Systems,* Volume 2429/2002, pp. 251-26

[22] Edu, Berkely (n.d.). *Endeavour System*. Retrieved 07 2008 from http://endeavour.cs.berkeley.edu

[23] Edu, M. (2003). From Oxygen System: http://oxygen.lcs.mit.edu/

[24] Eugster, P., Felber, P., & Guerraoui, R. (2003). *The many faces of publish/subscribe.* ACM Computer Surveys, Volume 3, pp. 114 - 131

[25] Fielding, R. (2000). *Phd. Disertation Architectural Styles and the design of Network-based Software Architecture*. Retrieved 2008 from http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[26] Forum, U. (2008). *Universal plug and play device architecture*. From UPnP: . http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20080424.pdf

[27] Garlan, D., Siewiorek, D., & Smailagic, A. (2002). Project Aura: Toward Distraction-Free Pervasive Computing. *Pervasive Computing*, Volume 1(2), pp. 22 - 31

[28] Google. (2010). *Robo Guice*. From http://code.google.com/p/roboguice/

[29] Google. (2011, 04 29). *The Mobile Movement: Understanding Smartphone Users*. Retrieved 04 2011 from http://googlemobileads.blogspot.com/2011/04/complimentary-copy-of-mobile-movement.html

[30] Google. (2008). *What is Android*. From http://developer.android.com/guide/basics/what-is-android.html

[31] Google, A. (2009, 12). *Trip Journal winner of Android Development Challenge Organized by Google*. From Android Development Challenge: http://code.google.com/android/adc/gallery_travel.html

[32] Gravelle, R. (2008, 03). *Webreference*. From Comet Programming: Using Ajax to Simulate Server Push: http://www.webreference.com/programming/javascript/rg28/

[33] Grimm, R. (2004). One.world: experiences with a pervasive computing architecture. *Pervasive Computing*, Volume 3, pp. 22 - 30

[34] Grimm, R., Anderson, T., Bershad, B., & Wetherall, D. (2004). System Support for Pervasive Applications. *Trans. Computer Systems , 22* (4), -.

[35] Group, O. M. (2008). *Common object request broker architecture (corba/iiop)*. From OMG: http://www.omg.org/spec/CORBA/3.1/

[36] Hyun Jung La, S. D. (2010). A Conceptual Framework for Provisioning Context-aware Mobile Cloud Services. *3rd International Conference on Cloud Computing.* Miami: IEEE, pp. 466 - 473

[37] Interface 21, S. F. (2006). *Aspect Oriented Programming with Spring*. From http://static.springsource.org/spring/docs/2.0.x/reference/aop.html

[38] Ipina, D., & Katsiri, E. (2001). An ECA Rule-Matching Service for Simpler Development of Reactive Applications. *Published as a supplement to the Proc. Of Middleware,* IEEE,

[39] Jetty. (2003). *Jetty webserver*. Retrieved 08 2011 from Jetty webserver: http://jetty.codehaus.org/jetty/

[40] Kagal, L., Finin, T., & A., J. (2001). Trust-Based Security in Pervasive Computing Environments. *IEEE Computer* , Volume 34 Issue 12

[41] Knorr, E., & Gruman, G. (2010). *What cloud computing really means*. From Inforworld:
http://www.infoworld.com/d/cloud-computing/what-cloud-computing-really-means-031?page=0,1

[42] Koch, C. (2005). How SOA Really Works. *CIO*, http://www.cio.com/article/10591/How_SOA_Really_Works

[43] Krill, P. (2010, 10). *AJAX alliance recognizes mashups*. Retrieved 08 2011 from Infoworld: http://www.infoworld.com/d/developer-world/ajax-alliance-recognizes-mashups-559

[44] Kumar, M., Shirazi, B., & Singhal, M. (2003, July-Sept). PICO: A Middleware Framework for Pervasive Computing, *IEEE Pervasive Computing,* Volume 2, pp. 72 - 79

[45] Laerhoven, K. V. (1999). Online Adaptive Context Awareness, starting with low- level sensors. Free University of Brussel.

[46] Mangoose. (2008). *Mangoose*. Retrieved 08 2011 from Mangoose Web Server: http://code.google.com/p/mongoose/

[47] Manjunatha, A., Ranabahu, A., Sheth, A., & Thirunarayan, K. (2010). Power of Clouds In Your Pocket: An Efficient Approach for Cloud Mobile Hybrid Application Development. *Cloud Computing Technology and Science.* IEEE.

[48] Mansukhani, M. (2005). *Service Oriented Architecture.* Hewlett Packard .

[49] Mei, L., Chan, W., & Tse, T. (2008). A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues. *Asia-Pacific Service Computing.* IEEE, pp. 464 - 469

[50] Microsystems, S. (2006). *remote method invocation (rmi) - related apis and developer guides.* . From Java Website: http://java.sun.com/javase/6/docs/technotes/guides/rmi/index.html

[51] Mitchell, K. (2002). *Supporting the Development of Mobile Context-Aware Computing.* Department of Computing. Lancaster University.

[52] Monnox, A. (2005). Rapid J2EE™ Development: An Adaptive Foundation for Enterprise Applications. Prentice Hall.

[53] Newsome, J., & Shi, E. (2004). The Sybil attack in sensor networks: Analysis and Defense. *International Symposium on INformation Processing in Sensor Networks*, US: ACM Press, pp. 225-229

[54] Nokia. (2006). *Mobile Web Server*. Retrieved 2011 from Nokia Labs: http://research.nokia.com/page/231

[55] Norman, A. (1998). *The Invisible Computer.* Cambridge. Massachusetts: MIT Press.

[56] One.world. (2003, 02). *One.world*. Retrieved 07 2008 from http://cs.nyu.edu/rgrimm/one.world/

[57] Orlando, D. (2009, 01). *Cloud computing service models*. Retrieved 04 2011 from developerWorks: http://www.ibm.com/developerworks/cloud/library/cl-cloudservices1iaas/

[58] Oxygen, M. P. (2002). *Pervasive Human-Centred Computing*. Retrieved 12 2006 from MIT Laboratory for Computer Science: http://oxygen.lcs.mit.edu/Overview.html

[59] Paterson, I., Smith, D., Saint-Andre, P., & Moffitt, J. (2010, 07 02). *Bidirectional-streams Over Synchronous HTTP (BOSH)*. Retrieved 08 2011 from XMPP Standards Foundation: http://xmpp.org/extensions/xep-0124.html

[60] Ponnekanti, S., Johanson, B., Kiciman, E., & Fox, A. (2003). Portability, extensibility and robustness in iRos. *Pervasive Computing and Communications(PerCom 2003).* IEEE, pp. 11 - 19

[61] Presecan, S. (2009). Distributed context provisioning middleware. Proceedings of the International Conference on Knowledge Engineering, Principles and Techniques Conference (KEPT2009), Section 'Knowledge Processing in Economics', Babes-Bolyai University of Cluj-Napoca

[62] Presecan, S. (2011). Mobile Cloudable Applications - New Way of Distributing Mobile Tasks Into the Cloud - TO BE PUBLISHED. *International Workshop on Machine-to-Machine Communications, IEEE GlobeCom.* Huston: IEEE.

[63] Presecan, S. (2009). Pervasive context-aware middleware. *Proc. 9th International Conference on Informatics in Economics.* Bucharest: ASE.

[64] Presecan, S. (2007). Pervasive Education Portal - Architectural Challenges. *Proc. 8th International Conference on Informatics in Economics* (pp. 108-113). Bucharest: ASE.

[65] Presecan, S. (2008). Security Challenges for Pervasive Computing. *The Romanian Workshop on Mobile Systems, Economy Informatics.* Cluj-Napoca: FSEGA.

[66] Presecan, S. (2008). SOA based architecture for pervasive systems. *The Romanian Workshop on Mobile Systems, Economy Informatics*, Cluj-Napoca: FSEGA.

[67] Presecan, S., & Tomai, N. (2009). Distributed Context Provisioning and Reaction Middleware. *Intelligent Computer Communication and Processing.* Cluj-Napoca: IEEE Computer Society Press, pp. 351 - 354

[68] Qian Wang, D. R. (2009). SOA's Last Mile-Connecting Smartphones to the Service Cloud. IEEE international conference on *Cloud Computing.* Bangalore : IEEE, pp. 80 - 87

[69] Ricky K., K. M.-L. (2010). A Stack-on-Demand Execution Model for Elastic Computing. *39th International Conference on Parallel Processing.* San Diego: IEEE, pp. 208 - 217

[70] Rivest, R., & Stanajo, F. (2002). *Security for ubiquitous computing.* US: Willey.

[71] Robinson P., Vogt H. and Wagealla W. (2005), Some Research Challenges in Pervasive Computing, Privacy, *Security and Trust within the Context of Pervasive Computing,* Volume 780, pp. 1-16

[72] Roman, M., & Campbell, R. (2000). GAIA: Enabling active spaces. Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system, pp. 229-234

[73] Roman, M., Hess, C., Cerqueira, R., & Campbell, R. (2002). Gaia: A Middleware Infrastracture to Enable Active spaces. *IEEE Pervasive Computing Magazine* , pp. 74 - 83

[74] Sasse, M., & Adams, A. (1999). Taming the wolf in sheep's clothing: privacy in multimedia communications. *ACM international conference on Multimedia* Orlando: ACM Press, pp. 101-107

[75] Satyanarayanan, M. (2001). Pervasive Computing: Vision and Challenges. *IEEE Personal Communication* , pp. 10-17.

[76] Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The Case for VM-Based Cloudlets in Mobile Computing. *Pervasive computing , 8* (4).

[77] Schechter, B. (1999). Seeing the light: IBM's vision of life beyond the PC. IBM Press.

[78] Schilit, B., Adams, N., & Want, R. (1994). Context-aware computing applications. *Workshop On Mobile Computing Systems and Applications*. Santa Cruz: IEEE, pp. 89-101

[79] Scholtz, J. (2001). Workshop on Evaluation Methodologies for Ubiquitous Computing. Ubicom.

[80] Source, S. (2003). *Aspect Oriented Programming with Spring*. (I. 21, Producer) From http://static.springsource.org/spring/docs/2.0.x/reference/aop.html

[81] Sousa, J., & Garlan, D. (2002). Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. *3rd Working Conference on Software Architecture*, US: IEEE, pp. 102-106

[82] Sumi, H. (2006, Oct-Dec). SODA: Service-Oriented Device Architecture. *Pervasive Computing* , Volume 5, pp. 94 - 96

[83] Tanenbaun, A., & Steen, M. (2006). *Distributed Systems: principles and paradigms.* New-York: Prentice Hall.

[84] UDDI. (2000). *Universal Description, Discovery and Integration of Business for the Web*. From Universal Description, Discovery and Integration of Business for the Web: http://www.uddi.org

[85] Vaquero, L. M.-M. (2008, 12). A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun.* , pp. 50-55.

[86] W3C. (2008). *Simple Object Access Protocol*. From SOAP: http://www.w3.org/TR/soap/

[87] W3C, (2008a). *Web Services Description Language (WSDL)*. From Web Services Description Language (WSDL): http://www.w3.org/wsdl

[88] Waldo, J. (1999). The jini architecture for network-centric computing. *Communications of the ACM*, Volume 42, pp. 76-82

[89] Washington, U. o. (2002). *Portolano: An Expedition into Invisible Computing*. Retrieved 08 2008 from http://portolano.cs.washington.edu/

[90] Weiser, M. (1991). The Computer of the 21st Century. *Scientific American ,* Volume 265, pp. 94-101.

[91] Weiser, M., & Brown, J. (1998). The coming age of calm technology. *Beyond calculation: The next fifty years of computing*, pp. 75-82.

[92] Wejchert, J. (2000). *The Disappearing Computer.* European Commission, Information Document, IST Call for proposals. Brussels: European Commission.

[93] Xiao, Y. (2007). Security in distributed, frid, mobile and pervasive computing. US: Auerbach Publications.

[94] Zhang X., Kunjithapatham A., Jeong S. and Gibbs S. (2011). Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Computing , *Mobile Networks and Applications.* Springer link, Volume 16, issue 3, pp. 270 - 284

[95] Zhou, Y., & Jiannong, C. (2007). A Middleware Support for Agent-Based Application Mobility in Pervasive Environments. *27th International Conference on Distributed Computing Systems Workshops.* IEEE, pp. 9 – 9o