2.
State-Of-The-Art

"Ubiquitous computing means placing computers everywhere in the user environment, providing ways for them to interconnect, talk and work together".

Daniel M. Russell and Mark Weiser, "The Future of Integrated Design of Ubiquitous Computing in Combined Real & Virtual Worlds," ACM - CHI, 1998.

In this Chapter, we describe the main paradigms involved in the present research. Considering that a paradigm is an initial reference of knowledge centered on patterns to be followed – i.e. a base model for scientific researches, we focus on Ubiquitous Computing, Multi-Agent Systems (MASs), Goal-Orientation and Software Reuse paradigms from Section 2.1 to Section 2.4. Then, Section 2.5 details the combination of these paradigms, which is used throughout the thesis. Finally, Section 2.6 summarizes the Chapter by presenting some concluding remarks.

2.1.
Ubiquitous Computing

In practice, Ubiquitous Computing is an emergent paradigm, whose applications are embedded in intelligent environments integrated into the physical world and composed of users with different preferences, heterogeneous devices and several content and service providers. Idealized by Mark Weiser (Weiser 1991), Ubiquitous Computing focuses its attention on offering services and contents anywhere and at any time from different devices by assisting the users in their daily activities without disturbing them. Therefore, Weiser proposed the use of Calm Technology (Weiser and Brown 1995) – i.e. computers are personal assistants/servants, which must reduce the "frenzy" of information by helping the users without directly intruding on their life – whose principle is the invisibility. In this field the user satisfaction is one of the main issues. However, the "anywhere and at any time" paradigm poses some other challenges for the

Software Engineering community, such as: the content adaptability and the context awareness. In ubiquitous scenarios, every decision must consider the context under analysis, and all performed activities must be adapted according to the ubiquitous profiles (e.g. user preferences, device features, network specification, and contract information).

Nowadays, the proliferation of wireless technologies combined with the wireless networks allow the users to perform several activities wherever and whenever they might occur. Some examples of access devices are simple cell-phones, palms, Smartphones and **P**ersonal **D**evices **A**ssistants (PDAs). These devices are becoming more and more pervasive by providing resources for the users to perform there daily activities through their own interface. Today, many companies have microprocessors that are small devices, which offer specific services. Interesting and at the same time difficult are the situations, in which the devices must collaborate with each other to provide complex services.

Most of the efforts in Ubiquitous Computing are on the development of powerful devices and on different ways to adequately establish communication with them. Moreover, it is important to reduce the cost and the physical dimension, and to increase the functionalities offered by those devices. Therefore, the main advantages on using mobile devices instead of fixed ones are the access facilities, the possibility of personalizing the content and the service according to the user's device, and the comfort to perform tasks independently of the location (Yunos et al. 2003). Figure 2.1 illustrates some typical situations idealized by Ubiquitous Computing by presenting people in their day-by-day activities.

Figure 2.1 - Ubiquitous scenarios (from petitinvention.wordpress.com[1])

[1] Images obtained from the *petitinvention.wordpress.com*, **Future of Internet Search: Mobile Version**, available *online* at http://petitinvention.wordpress.com/2008/02/10/future-of-internet-search-mobile-version/ (Last Access: January 2010).

The illustrated ubiquitous scenarios emphasize the resources offered by the devices. In addition, they also show that the device usage must be intuitive for the user even if it has various functionalities. However, there are heterogeneous devices that are not easy to use in practice. Therefore, the users normally know and use the essential functionalities, by not utilizing all the device capacity (Schmidt and Beigl 1998). An assistant that represents the user could facilitate the interaction between the user and the device by dealing with the process complexity (e.g. protocols and configuration details). In other words, this personal assistant could encapsulate the complex details in dealing with several services, and stimulates the user in the usage of them. The process complexity invisibility can contribute to the user's satisfaction.

A short time ago, the main use of mobile devices was based on voice. However, the third generation networks (**G**eneral **P**acket **R**adio **S**ervice[2] – GPRS and **U**niversal **M**óbile **T**elecommunication **S**ystem[3] – UMTS), and the recently development of communication and representation protocols (e**X**tensible **M**arkup **L**anguage – XML and **W**ireless **A**pplication **P**rotocol – WAP) are been combined to offer services with high quality for the users (Ekudden et al. 2001; Ralph and Shephard 2001).

Other important topics that must be considered in the development of Ubiquitous Computing are: (i) the necessity of new technologies to identify and specify the requirements in ever-changing environments as the users, the devices and the context under analysis are really different from one another; (ii) the communication channel by using the wireless network; (iii) the balance between the privacy and the personalization issues; (iv) the balance between the transparency and invisibility issues; and (v) the necessity of managing (e.g. capturing, storing, retrieving and excluding) the user and service provider information "on the fly".

According to (Saha and Mukherjee 2003) Ubiquitous Computing is investigated in the academic and industrial areas. It can be confirmed by projects as the ones presented as follows:

---

[2] GPRS is an emergent service to receive information using the mobile phone network.

[3] UMTS is a third generation mobile system, developed by the ETSI with the IMT-2000 framework support from the Union of International Telecommunications. This system proposes the combination of cell-phone, local network, private mobile radio and payment system.

- Aura (Aura 2011) – Carnegie Mellon University: the objective is to provide the easy usage of different information services with invisibility and independently of the location;
- Endeavour (Endeavour 2011) – University of California, Berkeley: the objective is to facilitate the understandability of the users in relation to the technology usage by allowing that people interact with different devices and other people, and obtain information;
- Oxygen (Oxygen 2011) – MIT: the objective is to provide the communication in a pervasive way, as part of the people's life; and
- Portolano (Portolano 2011) – University of Washington: the objective is to create an environment to investigate Ubiquitous Computing and their devices in specific tasks.

The main objective of Ubiquitous Computing is to facilitate people's life. Some typical examples in this field are: (i) to make the environment comfortable for people by adjusting the room temperature and configuring its security through a touch of button from your hand-phone; and (ii) to understand where you are by using your personal mobile device. Figure 2.2 illustrates different smart-spaces and daily activities controlled by mobile devices.

Figure 2.2 - Smart-Spaces (from [4])

[4] http://www.tronshow.org/guidebook/2010/tron/e/02/web02_01.jpg

Considering specifically the mobility and the distribution of the users and devices in ever-changing environments, Mobile Computing refers to the systems in which the computational components alter their location in a physical environment. According to (Bellavista et al. 2001), Mobile Computing systems are composed of the user, the host, and the access mobility. The access mobility demands users with an adequate view of their preferences. However, it does not occur in practice. Normally, the users and their preferences are really different from one to another. Moreover, the users' interests are in constant evolution. Therefore, the mobility confers to the user the freedom; and to the software engineers the necessity of developing mobile applications. Ideally, the hosts of the mobile systems allow that the devices move from one host to another by connecting all distributed environments. The access to these systems – by the wireless networks – also allows dynamic adaptations to offer the services according to the user's requirements and the host specifications. These adaptations require a deep investigation/consultation/manipulation of different ubiquitous profiles (e.g. device profile, user profile, network profile, and contract profile).

Mobile Computing has been considered a way to improve the users' satisfaction when they are using their devices in movement (Lyytinen and Yoo 2002). The result of using Mobile Computing in this field is services and contents anywhere and at any time – i.e. Service and Content Omnipresence. However, one of the main limitations for Mobile Computing is the hard work required to adapt computational models while the user is in movement. In order to deal with this limitation, the users normally must control and configure their devices themselves, which negatively impacts on the invisibility and the user satisfaction issues.

Another perspective to improve the presence of computational resources into people's life is Pervasive Computing. The main idea is to allow the dynamic development of a computational model after the environment evaluation. As computing is becoming viable in different environments with the mobile devices, the information is available anywhere, the communication is easier between people and objects, and the interest in Pervasive Computing is also increasing. Pervasive Computing as well as Mobile Computing demand inter-operability, scalability, and adaptability to better satisfy the users in their daily activities.

Ubiquitous Computing can be viewed as the combination of Mobile Computing and Pervasive Computing (Lyytinen and Yoo 2002) – see Figure 2.3. In (Weiser 1993) Ubiquitous Computing is defined as a method to allow the insertion of computing processes into every physical place by guarantying the invisibility principle in the access of services, peripheral and contents. Moreover, it is desired that every device, even if the user is moving, has the ability to construct a dynamic model into the environment, consequently, performing the configuration and adjustments to deal with different services. Therefore, Ubiquitous Computing involves research to, for example: work with small, limited and mobile devices, wireless communication, different users, and other challenges. Some psychological challenges can also be observed, such as: the user privacy, the user preferences, and the multiple users interacting into the environment.

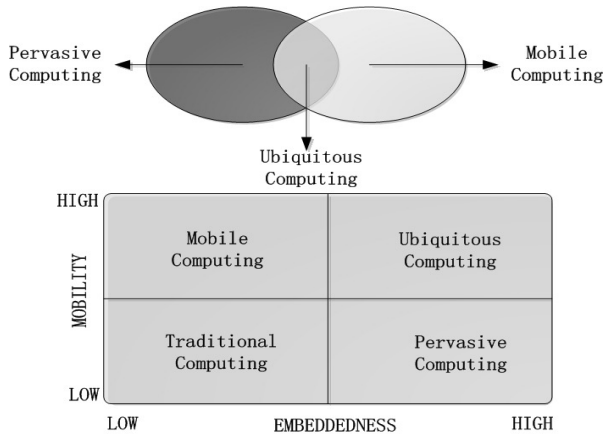

Figure 2.3 - Ubiquitous Computing definition (adapted from (Lyytinen and Yoo 2002))

In (Borriello 2002), the author presents some considerations on the investigation of Ubiquitous Computing paradigm centered on Mobile Computing and Pervasive Computing challenges.

- Heterogeneous network: the networks can be wire or wireless, and they can follow different communication protocols (e.g. DQDB (**D**istributed **Q**ueue **D**ual **B**us) (DQDB 2011), SMDS (**S**witched **M**ultimegabit **D**ata **S**ervices) (SMDS 2011), and SONET (**S**ynchronous **O**ptical **NET**work) (SONET 2011)). Therefore, it is necessary to establish ways to allow the communication within this heterogeneous environment. A possible way is the use of a protocol that allows sending packages from one band to another. Other protocols are also important to the maintenance and

communication, mainly for wireless networks. This communication can be improved by integrating different platforms;

- Geography versus network topology: the security in the communication of different mobile devices based on a physical perspective differs from a virtual one. It is relevant to focus the attention on the geographic proximity instead of the network topology. In order words, if the mobile device is geographically close, more secure and direct the communication will be;
- Batteries limit the devices use: the device depends on the battery to work. Therefore, there is a concern about saving energy, and creating mechanisms to deal with the device disconnection because of low battery, absence of signal and other communication problems;
- Systems must evolve according to the devices: in order to deal with new commercial trends, the devices are in constant evolution. Therefore, it is necessary that the development of applications based on these devices provides mechanisms to allow their constant evolution; and
- Privacy issues must be considered: in ubiquitous applications, there is information exchanged between the users and the application. In some cases, the users want to protect their personal information. Therefore, it is important to investigate the user preferences and privacy policies in order to guarantee the dependability, accountability, and security of this data.

In order to finalize the presentation about Ubiquitous Computing paradigm, we want to highlight four main technologies adopted by some important corporations/companies to commercially deal with content and services anywhere and at any time. We briefly describe those technologies as follows:

- Jme: Java Platform, Micro Edition, or Java ME is a Java platform for embedded systems. Jme was proposed by Sun Microsystems, now controlled by Oracle Corporation. This is a largely open platform, whose devices implement specific profiles. The most popular of these profiles are the **M**obile **I**nformation **D**evice **P**rofile (MIDP) for some cell-phones; and the Personal Profile for embedded devices, such as PDAs. In our proposal, we particularly focus our development on the Jme platform;

- Apple OS: it is the Apple's mobile operating system. Originally proposed for the iPhone, different extensions have been developed to aim other Apple family's devices, such as: iPod and iPad. This proprietary platform allows programming in Objective C language. However, there are bridges from Java, C# and other languages. It is also necessary a MAC – running the MAC OS – to develop mobile applications based on the Apple OS platform. This technology becomes popular with the iPhone advent. Another relevant consideration is that third-party applications can only be installed from the Apple Store by increasing the development cost as well as the publishing time on the store. Many applications were rejected according to the Apple's policies and business rules;
- Android: it is a mobile operating system, initially developed by Android Inc. Now, it is a subsidiary of Google. This open platform supports Java by translating the byte code to its own custom Dalvik byte code. This code is optimized for mobile devices. Moreover, Android offers the possibility of programming in C and it can run script languages like LUA, Perl and Python. It is possible that it will become a standard in the next few years; and
- Windows Mobile: it is an operating system for mobile devices, developed by Microsoft Corporation. This proprietary platform is based on the Windows CE kernel. It also includes a set of essential applications centered on the Microsoft Windows API. Moreover, third-party initiative focused on the Windows Mobile platform is possible, and the developed applications can be browsed, purchased and downloaded by using a specific service offered by the Microsoft – i.e. the Windows Marketplace for Mobile.

Based on the considerations above, we see that the success of Ubiquitous Computing depends on several other technological solutions. We consider this issue especially from the Software Engineering's viewpoint (e.g. approaches, methodologies and other support inspired by traditional and emergent paradigms). In this field, we have chosen to focus on three specific paradigms: (i) Multi-Agent Systems (Shoham and Leyton-Brown 2008); (ii) Goal-Orientation (Mylopoulos 2008; Lamsweerde 2001); and (iii) Software Reuse.

2.2. Multi-Agent Systems (MAS)

An agent is a software that automatically acts/reacts to execute tasks (Jennings 2000). Most of the agents design is based on the fact that the user just needs to specify her/his main goals, by leaving the decisions of "how" and "when" to the agents. Software agents have particular features that differentiate them from the traditional components, such as: autonomy, collaboration, flexibility, automatic restarting, communication, adaptation, and mobility. However, it is not necessary the presence of all these features for an entity to be an agent. For illustration purposes, some features are described as follows:

- Autonomy: an agent is capable of acting, reacting, and controlling its actions without the human intervention;
- Collaboration: an agent is not obliged to obey commands. It can modify requests by asking for more details in a collaborative way, or even refusing to perform the received request;
- Flexibility: the agent's actions are not pre-defined. It is able to dynamically choose and invoke its actions – in response of an external event – by also determining their execution order; and
- Mobility: an agent can move from one host to another by using multi-platforms, in which it performs its tasks.

A Multi-Agent System represents a collection of software agents (i.e. autonomous components with particular goals). These agents cooperate with one another according to an *Organization* with the purpose to solve tasks. The MAS components can also have the mobility property. Mobile Multi-Agent Systems are composed of a platform of mobility and mobile agents. The platform establishes all mechanisms to provide support for the agents' mobility. As illustrated in Figure 2.4 (adapted from (Jennings 2000)) the *Agents* are responsible for interacting and performing the delegated tasks. The *Interaction* can occur between the agents and the environment as well as between agents by following specific protocols. The organization is inside the environment. Therefore, it determines the tasks that will be performed by the agents to satisfy the users. The *Environment* has resources that can be used by the agents as well as by the organization in order

to conclude the tasks. The *Resources* also impact on the way tasks are performed. Each element (e.g. *Agents*, *Interaction*, *Organization* and *Resource*) contains a *Sphere of Influence* into the environment under analysis. This sphere represents the influence of each element of the environment by emphasizing the importance of them to successfully achieve the goals. The communication allows to deal with the information exchanged between the agents by focusing on the perception and actions involved in this interactive process.

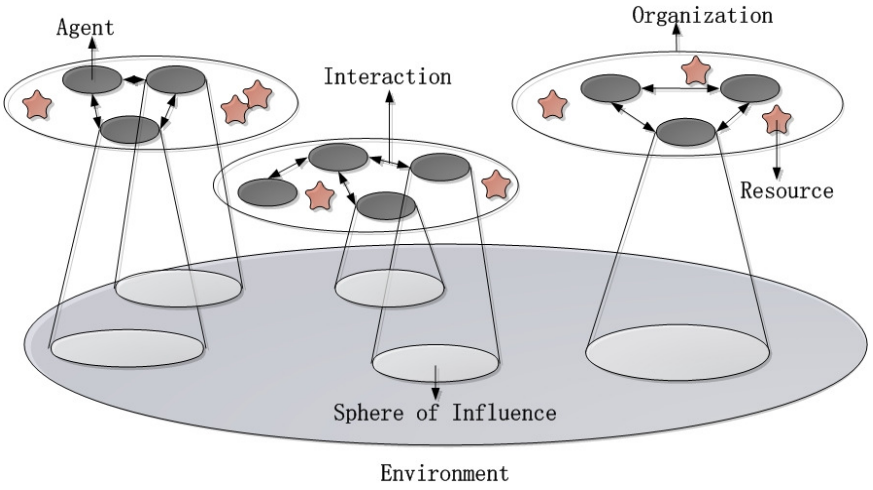

Figure 2.4 - Multi-Agent System (adapted from (Jennings 2000))

It is interesting to observe that various research projects involving the agent concept argue that the agents perfectly perform their tasks in controlled environments. However, it is difficult to know if the agents can deal with tasks by assuming responsibilities in ever-changing contexts, which are dynamic, opened, distributed, and whose components are interdependent. In order to illustrate some agent-based mechanisms that could be improved to deal with these ever-changing contexts' concerns, we focus our attention on mobile agents.

Figure 2.5 shows mobile agents in two hosts, A and B. Normally, these agents move from one host to another, in which the desired data of the application are stored. They select this information based on the user preferences, and then return to the initial host.

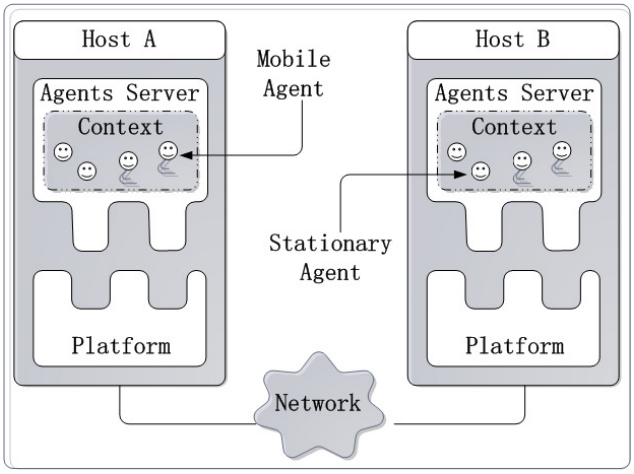

Figure 2.5 - Mobile agents

A mobile agent, in terms of implementation, can be viewed as the combination of code and state. The code is the program with the agent's definition. The state represents a set composed of the agent's internal attributes during the execution. The environment, in which the agent is moving, is distributed, and its access is configured through the server. The server is installed in a host and offers, among other resources and services, an adequate place to the agent's execution. This place is called *Execution Context*.

The interaction between the mobile agents and the server of their correlated platforms are established based on the agent's mobility protocol. Therefore, first of all and in order to establish the agent's transference through the network, the agent's execution is interrupted at the initial host. Then, the agent moves from one context to another into the distributed environment by taking/carrying its code and execution data. The execution is restarted from the interruption point.

In this field, the agent's life-cycle (Figure 2.6) is centered on four stages: (i) *Instantiation*, in which the agent is created; (ii) *Movement*, in which the agent moves from one host to another; (iii) *Initialization*, in which the agent arrives in the new host; and (iv) *Destruction*, in which the agent's state is lost.

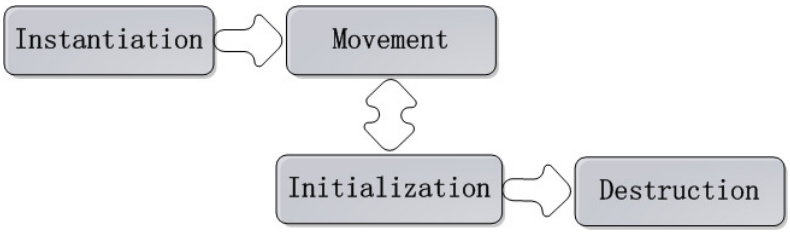

Figure 2.6 - Agent's life-cycle

The circumstances, in which the agents must move, as well as the instants, in which the movement action must be dispatched, are determined by the developers as movement points of the applications. The configuration and the maintenance of the agent's itinerary can be statically or dynamically performed and by respecting the access constraints imposed by the platform servers.

The migration of the agents can be weak or strong (Fuggetta et al. 1998). In both cases, weak migration and strong migration, the agent's execution is restarted without losing the values of the internal attributes. However, on one hand, in the weak migration, the agent restarts its execution in a new host from the first line of its code. On the other hand, in the strong migration, the execution is restarted from the interruption point by following the stack concept.

The mobile agents platforms based on Java normally implement only the weak migration. It occurs because of the independency of this language in relation to the operational systems by requiring a hard work to restart the agent's execution from interruption point (Fuggetta et al. 1998). However, there are environments of the Java platform that implements the strong migration.

Given the intrinsic mobility of Ubiquitous Computing applications, the research on mobile agents is really relevant. To provide an overview, we can consider a service that must be performed (Figure 2.7).

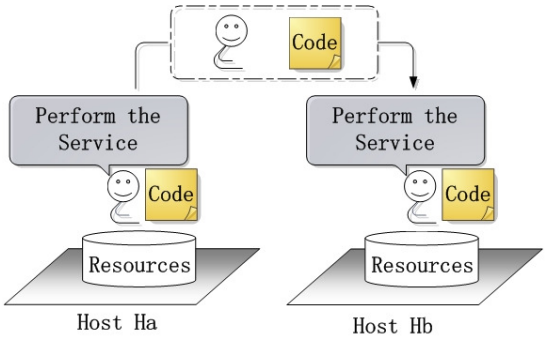

Figure 2.7 - Mobile agents in ubiquitous contexts

The code of this service is specified in the component A, located at the host Ha. However, there is need for resources to perform the service. Some of them are only available at the host Hb. Therefore, the component A migrates to the host Hb by carrying its code, actual state, the service's code, and the partial results obtained until this moment. The component A performs the service by using the resources available at Hb. All this process is based on a protocol that guarantees the communication between the agents and the environment under analysis. This

situation is commonly found in ubiquitous scenarios, for example, to perform complex services (e.g. content adaptation) in a dedicated server. This kind of support is suggested by some authors (Forte et al. 2006) in order to avoid the application server overload.

There are some interesting advantages on applying software agents to the development of ubiquitous applications, such as:

- Asynchronous execution of the services: with the expansion of Ubiquitous Computing, it is important to deal with the mobile devices disconnection and portability;
- Data traffic reduction through the network: in this case it is desired that only the processing requests be transferred through the network instead of a great amount of information. In MAS, the requests are transferred with the agents; and
- User satisfaction improvement: the idea is the use of software agents as personal assistants to represent the users in order to search, analyze, and adapt contents and services according to the users' preferences and their devices features.

The mobile devices (e.g. PDAs and Smartphones) normally work by using expensive and unstable networks. Therefore, tasks that demand an open-connection-based request between the mobile device and the network are not economically and/or technically viable. In this field, mobile agents can encapsulate those tasks, which are sent through the network with the agents. After this process, the agents can return with the results achieved. It is relevant to mention that the synchronization between the connection and the requested tasks' execution is not necessary. In this situation, the mobile agents become independent of the original machine and can asynchronously operate after the transference (White 1995; Lange and Mitsuru 1998; Lange and Aridor 1998).

Some important considerations on using mobile agents must be emphasized, especially with respect to the security issue (e.g. malicious interventions from third machines or agents). It is not cautious to directly accept codes from different machines without a deed investigation of their trustworthiness/reliability. In order to deal with these scenarios, it is recommended the use of techniques to authenticate the elements involved in the transferences. In (Fullam and Barber

2006), the authors describe their investigation of this field by focusing their attention on the agents' reputation. An example is the combination of the mobility platform with reputation-based resources to deal with trustable mobile agents' processing, by increasing the complexity of the ubiquitous applications.

There are several areas in which the Multi-Agent Systems are applied, such as: e-health, e-commerce, educational, computer games, commercial and feature-length films, and defense systems. In medical areas it is important to deal with the scarcity of resources, the irregular distribution, and great distances from the health centers of excellence to areas with poor health assistance. The use of software agents and mobile devices can contribute to this field.

The telemedicine applications focus on the data acquisition to improve the diagnoses as well as on the telesurgery. An example in this scenario is the "Hospital do Futuro" (translating "Hospital of the Future"), in which computational support is used to improve the biomedical areas. Another application is the distance education to help the health professionals by using mobile devices to improve their work in areas with poor health assistance. Figure 2.8 illustrates an intelligent dental clinic, in which various devices are connected to provide different services.

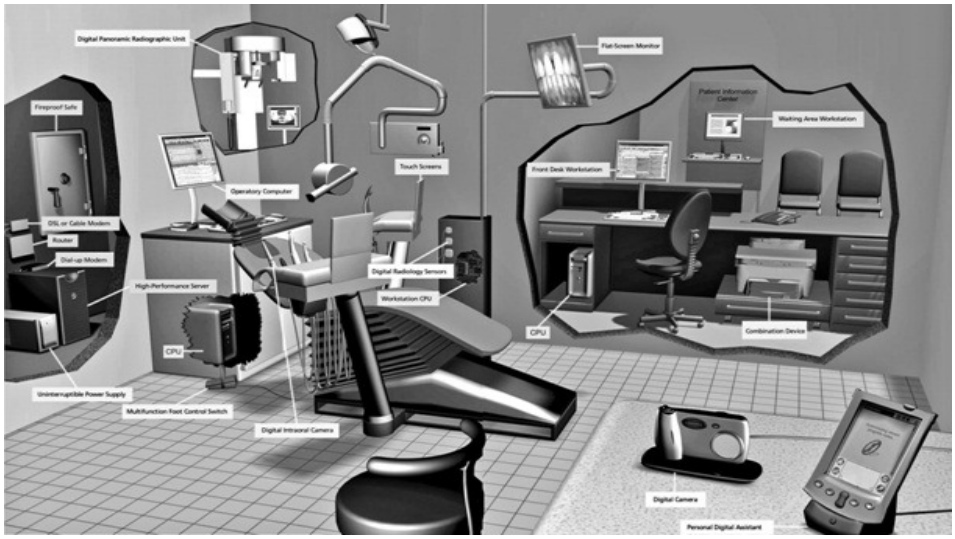

Figure 2.8 - Intelligent dental clinic (from technologically well-equipped dental office[5])

[5] Images obtained from the technologically well-equipped dental office. DSL: Digital subscriber line. CPU: Central processing unit. Artwork by Titus Schleyer, Heiko Spallek and Stephan Hempel, available *online* at http://jada.ada.org/cgi/content/abstract/134/1/30 (Last Access: January 2010).

In the e-commerce domain, software agents can assume different roles, such as: salesmen, manager, critics and customers. The agents can also negotiate by representing the salesmen and/or the customer. In most of the cases, the agents obtain a list of stores with the price, the product availability and other information for future analysis. Figure 2.9 shows the agents' way-of-working in an e-commerce scenario.

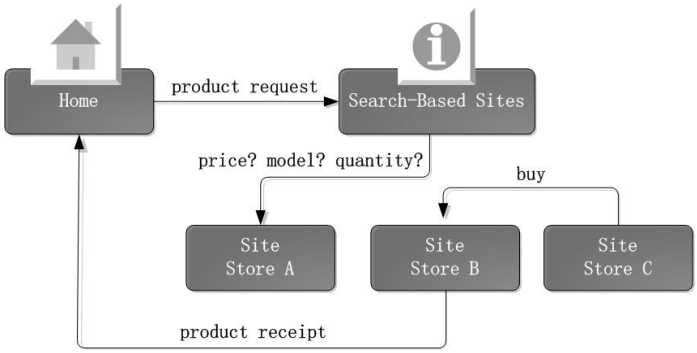

Figure 2.9 - E-commerce-based scenario

Another area in which the use of mobile agents is interesting is the search for information. In applications involving search for information, mobile agents can be transferred to different information sources to search the desired information by returning to the original host with the obtained results. Figure 2.10 illustrates this process.

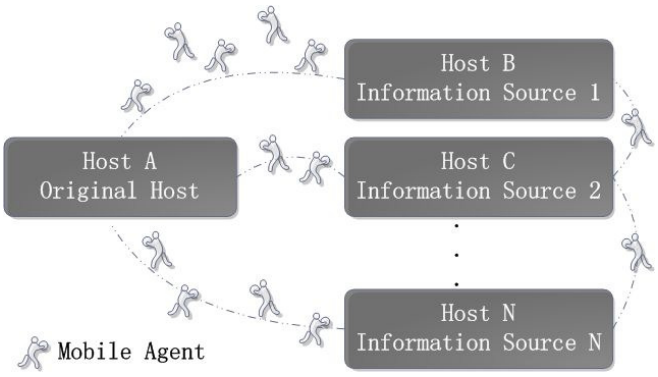

Figure 2.10 - Search for information by using mobile agents

A software agent can also perform the role of a personal assistant that has the purpose to execute tasks by representing the user. For example, the user can desire that her/his personal agent interacts with other agents (e.g. that represent different entrepreneurs). The agents can negotiate the scheduling of a meeting. Reasoning and learning techniques can improve this interactive and collaborative process.

The monitoring and notification can also be performed by agents. This kind of activity is commonly used in commercial sites. They represent examples of asynchronous tasks. The agents can be transferred through the network in order to notify the users that an information or product is now available to consultation or sale. Furthermore, agents are constantly applied to manager networks to reduce the data traffic and the response time.

Research and applications in the areas of agents and Multi-Agent Systems go on and on and we restricted our description to issues that will be covered by the case study presented in this thesis as well as others developed by us to compose an adequate view of the ubiquitous challenges.

2.3. Goal-Orientation

In the early 1980s, the Goal-Orientation was originated in the educational psychology literature based on studies of primary school children in order to explain the differences in classroom learning styles (Atkinson 1964; Eison 1979) – i.e. why some students were focused on achieving superior grades (Grade-Orientation), while others were focused on the learning process (Learning-Orientation).

Recently work (Mylopoulos 2008) emphasizes the importance of using goals to capture the stakeholders' purposes, which are related to functional and non-functional requirements. It is relevant to understand the social and organizational context. In this field, some goal-oriented approaches (Yu 1997) provide resources to social analysis through the modeling and the analyses of strategic relationships among multiple stakeholders. These stakeholders are modeled as actors of the system-to-be. Moreover, they depend on each other for goals to be achieved, tasks to be performed and resources to be shared, exchanged or consumed. This modeling is based on the intentionality abstraction (Yu 1997).

In addition, softgoals are used to systematically deal with quality attributes (i.e. non-functional requirements), while the dependencies among actors bring about opportunities and vulnerabilities. Finally, dependencies are analyzed based on a qualitative reasoning. In other words, alternative configurations of

dependencies are explored by the actors, during the design phase of the system, to determine their strategic positioning in a social context.

On account of their Goal-Orientation nature, in the described approaches for all actions are given meaning, direction, and purpose by the specified goals that individuals look for. Only to illustrate, we can consider the previous discussion between grade-orientation-based style and learning-orientation-based style. In grade-orientation-based style, the orientation is defined as goals focused on an individual's set of beliefs that represents the reasons why someone engages in academic tasks to achieve superior grades. In learning-orientation-based style, the definition focuses on task completion and understanding, as well as on new skills developing.

As some final considerations, we can also mention that: (i) the goal-oriented design enhances the possibility of guaranteeing the design decisions traceability for further and deeper investigation; (ii) a goal-oriented approach is adequate to deal with different viewpoints as its abstractions (e.g. actors, goals, tasks, resources, softgoals, beliefs, and dependencies) can be used to model social contexts by taking into account the stakeholders' practical reasoning centered on their goals and alternative plans to achieve them; and (iii) goal-oriented approaches (e.g. TROPOS) was evaluated as being very supportive of non-functional requirements. They deal with non-functional requirements by representing them as softgoals. The softgoals are constructed by considering different ways to reason and evaluate alternatives to *satisfice* (Yu 1997) – i.e. *satisfy in a certain degree* – one or more non-functional requirements. Therefore, the combination of MAS and Goal-Orientation paradigms results on MAS enriched by the intentionality concept, which can be used to improve the cognitive ability of the agents in the interpretation of, for example, the human practical reasoning (Bratman 1999).

2.4. Software Reuse

Software reuse is the process of developing new software from reusable artifacts instead of building this software from the very beginning. In the literature it is possible to find discussions about reuse classification schemes (Biggerstaff and

Richter 1987; Krueger 1992; Prieto-Díaz 1993). This classification tries to distinguish between various approaches to reuse. Basically, these publications consider the reuse in different abstraction levels by taking into account that everything associated with the software project can be reused, such as: knowledge, models, documentation, architectures, design, procedures and code. According to (Krueger 1992): (i) reusable artifacts for the higher abstraction levels reduce the effort required to go from the initial concept of a system to its representation, i.e. from its requirements to its design; and (ii) reusable artifacts for the lower abstraction levels reduce the effort required to go from the system's representation to its executable implementation, i.e. from its design to its code.

The systematic software reuse is a software engineering technique that involves the use of reusable artifacts from existing systems, literature and other information sources in order to systematically build new systems by trying to facilitate their construction, to improve their quality and to reduce their cost and development time. However, the development of a reusable artifact requires additional development effort compared to development of non-reuse setting. This artifact needs to be developed in a generic way that allows their reuse in different software projects.

Investigating various ubiquitous applications, it is possible to identify commonalities among them – i.e. common ubiquitous concerns. As previously mentioned (Section 2.1), some of these concerns are the device heterogeneity, the environment distribution, the intrinsic mobility, the content adaptability need, the context awareness need and others. Therefore, it is interesting to develop reusable artifacts to deal with these concerns.

This thesis investigated the reuse of conceptual models, frameworks, libraries, patterns and architectures, organized in reuse-oriented artifacts called *Building Blocks*. We propose building blocks to promote reuse at different abstraction levels, from the requirements to code. In higher abstraction levels, we focus our attention on architectures, conceptual models and models reuse. In lower abstraction levels, we offer frameworks, platforms, libraries and persistence support. Our reuse-oriented building blocks were developed based on an extensive investigation of different ubiquitous applications to compose an adequate view of common concerns in ubiquitous contexts centered on intentional Multi-Agent-

Systems. This step of our proposal, described in Chapter 4, is called *Domain Engineering of Ubiquitous Applications*.

2.5. The Paradigms' Combination

The combination of the described paradigms can contribute to deal with ubiquitous challenges, such as the ones illustrated in Figures 2.11, 2.12 and 2.13. These figures present some situations that commonly occur in ubiquitous contexts and how goal-oriented agents – i.e. intentional agents – can deal with them by using reusable support sets.

When the users are moving from one intelligent environment to another (Figure 2.11), their respective agents must coordinate this migration by temporally suspending the operations in order to negotiate an appropriate support at the new intelligent environment.

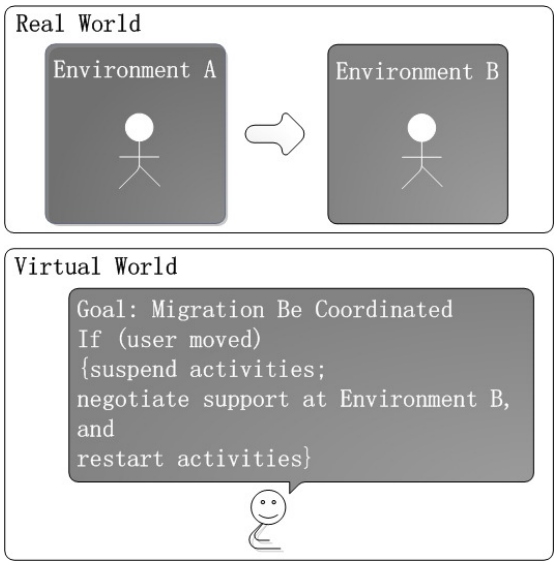

Figure 2.11 - Situation in which the user is moving

After this negotiation, the agents can return to their activities by recovering its previous stage. The complexity of the process is invisible to the user. However, it is possible to maintain the trace of the agents' activity for further investigation in case of unapproved performance. This resource helps on balancing the invisibility and the transparency issues, which is desired in ubiquitous contexts. Moreover, it also contributes to the location awareness and accountability issues in ever-changing contexts. Furthermore, complex content adaptations can be

performed in a dedicated server by using the same mobility-based mechanism. It avoids, for example, the application server's overload. Therefore, the described mechanism can compose a reusable support set that will be used or instantiated or extended in the development of ubiquitous applications to deal with the mobility issue with positive impacts on invisibility, transparency, location awareness and complex content adaptation issues.

When the environment is modified, the agents must maintain the same quality of the offered services. If the quality is not compatible with the desired one, the agents must find alternatives – i.e. environment must be re-configured – to guarantee the services with the same quality (Figure 2.12).

The desired quality is obtained by investigating the ubiquitous profiles centered on user preferences, device features, network specifications and contract information. It is also possible to use the non-functional requirements as quality criteria represented by fuzzy variables and its respective sets, which are investigated "on the fly" by using fuzzy conditional rules to improve the agents' decisions. We propose a reusable building block centered on a fuzzy logic library to improve the cognitive capacity of intentional agents on dealing with non-functional requirements at runtime. This building block is discussed in details later in this thesis.

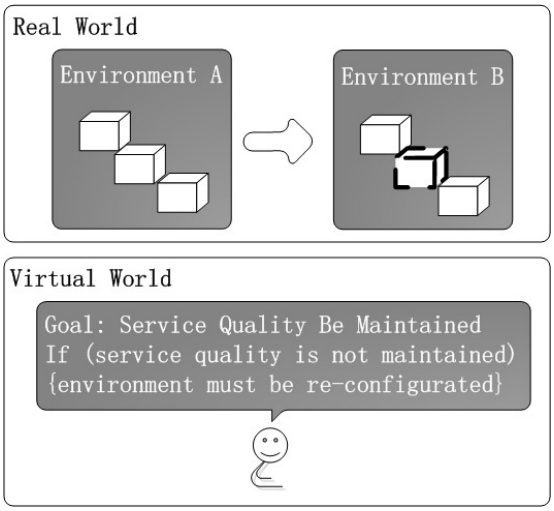

Figure 2.12 - Situation in which the environment is modified

When the context is modified, the agents must identify the performed operations based on this modification, and then adapt them to the new context. The operations have requirements that depend on the context (Figure 2.13).

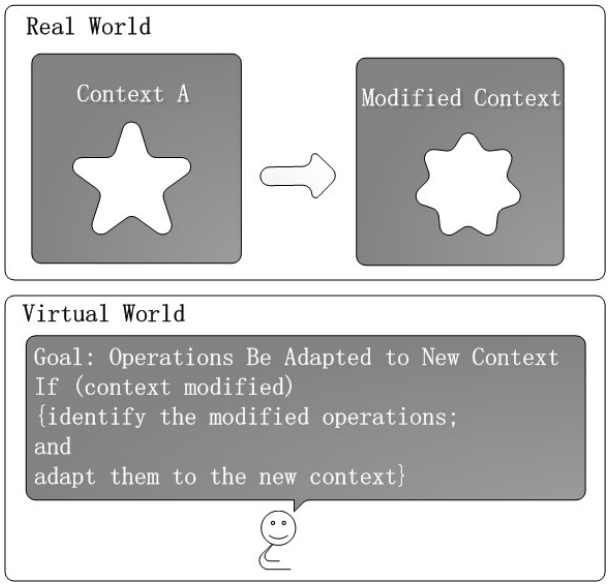

Figure 2.13 - Situation in which the context is modified

2.6. Closing Remarks

This Chapter provides an overview of some paradigms: (i) **Ubiquitous Computing** idealized a world in which the users can perform their daily activities (e.g. download media contents and pay a bill using a specific service) from heterogeneous devices and anywhere and at any time those activities might occur; (ii) **Multi-Agent-Systems** represent an organization composed of different autonomous entities that cooperate with one another to achieve specific goals by performing specific tasks. This organization has some interesting properties (e.g. autonomy, mobility, reactivity, proactivity, flexibility and adaptability); (iii) **Goal-Orientation** is an emergent paradigm that uses goals to capture the stakeholders' purposes, which are related to functional and non-functional requirements of a social and organizational context; and (iv) **Software Reuse** is a paradigm centered on the creation of new systems from existing artifacts. It focuses on building reusable artifacts in different abstraction levels to facilitate other systems construction as well as to reduce the cost and the team's efforts in developing it. Finally, the Chapter shows how to combine these paradigms – by presenting some typical ubiquitous scenarios – to compose a suitable reuse-oriented approach in order to deal with ubiquitous challenges, specially mobility and ever-changing situations.