The WISE Approach to Architect Wireless Services

Patricia Lago*, Mari Matinlassi**

*Dipartimento di Automatica e Informatica, Politecnico di Torino, Italy
Patricia.Lago@polito.it
**VTT Technical Research Centre of Finland, 90571 Oulu, Finland
Mari.Matinlassi@vtt.fi

Abstract. The Internet is quickly evolving towards the wireless Internet that will be based upon wirelines and devices from the traditional Internet, and will reuse some of its techniques and protocols. However, the wireless Internet will not be a simple add-on to the wireline Internet. From the technical point of view, new challenging problems arise from the handling of mobility, handsets with reduced screens and varying bandwidth. As a result, developing and operating new mobile services will be a challenging software engineering problem. The WISE (Wireless Internet Service Engineering) Project aims at producing integrated methods and tools to engineer services on the wireless Internet. In particular, this paper introduces the WISE approach to service engineering, describes how it is applied to a real world Pilot service and reports initial feedback from project partners when applying the approach.

1 Introduction[1]

The demand for wireless Internet services is growing quickly, partly because 3G mobile phones are entering the market. The development of wireless services faces challenges, like handling to support mobility with continuous communication, handsets with reduced screens and varying bandwidth. This also causes new business models to emerge.

In this changing evolving scenario, the WISE (Wireless Internet Service Engineering) project aims at anticipating the problems of wireless Internet service engineering by producing integrated methods and tools to engineer services on the wireless Internet, capable of reducing costs and time to market. In detail, the WISE Project has the following objectives:

- Deliver methodology and technology to develop services on the wireless Internet.
- Experiment with methodology and technology in real life applications.
- Prepare tools and metrics to evaluate the effect of methodology and technology.
- Disseminate results to the broadest possible audience.

[1] This work has been partially supported by IST Project 30028 WISE (Wireless Internet Service Engineering). URL http://www.wwwise.org

In this paper we introduce the WISE approach, a methodology to be applied to wireless services. The WISE approach is based on the QADA method defined in [1]. The Quality-driven software Architecture and Quality Analysis (QADA) method includes three viewpoints at two levels of abstraction: structural, behavior and deployment. The WISE approach extends the modeling requirements with a definition of the fourth viewpoint, the development viewpoint [2]. According to IEEE Std-1471-2000 [3], "an architectural view is a representation of a whole system from the perspective of a related set of concerns". In the literature, there are several approaches to the design of software architecture that concentrate on different views of architecture. The first of these view-oriented design approaches was the 4+1 approach, developed by Krutchen [4]. After this, several others have approached the jungle of architectural viewpoints. For instance, Jaaksi et al. introduced their 3+1 method in 1999 [5] and Hofmeister et al. used four views to describe architecture [6].

Among these approaches there is no agreement on a common set of views or on ways to describe the architectural documentation. This disagreement arises from the fact that the need for different architectural views and architectural documents is dependent on two issues: system size and software domain e.g. the application domain, middleware service domain and infrastructure service domain. Again, both the system size and domain have an impact on the amount of different stakeholders. Therefore, it is obvious that none of these methods alone is comprehensive enough to cover the design of software architectures for systems of a different size on various domains, or provide an explicit means to create architectural descriptions for all the systems. We are neither trying to cover all the domains, nor to define a catchall set of architectural viewpoints. Instead, we concentrate on the wireless service architectures and the viewpoints needed in wireless service architecture modeling.

WISE Pilots are demonstration environments to experiment with the methodology and technology. Moreover, to simulate the evolution of a Pilot through successive developments, each Pilot undertakes three iterations in which experience gained during previous iteration is taken into the current one. In particular, the Project is now in its first iteration, in the "architecture definition phase". The next step will be to define the reference architecture and reiterate; the final Project goal is the consolidation of a reference architecture for wireless services, refined at each iteration.

The paper is structured as follows. After an introduction to the WISE approach and its architectural viewpoints, we present one of the WISE Pilot services and explain the WISE viewpoints and modeling notation through examples taken from the Pilot. Finally, we draw some initial considerations, and report feedback from industrial partners, when applying the WISE approach to service engineering.

2 The Approach

WISE approach (**Fig. 1**) starts with "pre-studies & analysis" that define the driving requirements for the system and identifies possible constraints for service engineering. Driving requirements are the main requirements for the whole system, i.e. the main quality and/or functional goals the system has to provide. Constraints

may e.g. be related to rules set by laws, enabling technologies or hardware device manufacturers [1].

The next step is made up of three tasks that can be carried out in parallel, prior to successive steps. In particular, the aim of the task "survey architectural styles & patterns" [7] is to identify the applicable styles and patterns for the system under development. In addition to requirements engineering, the task "define service taxonomy" assists in the design of the conceptual architecture by identifying the functional domains to which services belong. For instance for the wireless domain, the adopted service taxonomy classifies wireless Internet services into four preliminary domain categories:

- End-user services that are provided directly for the end-users (customers).
- Technology platform services that provide basic functions on which end-user services rely (even though usually not directly used by the end-user as such). Third party service developers can provide these services.
- Application Domain Support Services that are not as generic as technology platform services but specific for the application domain and, on which end-user services also rely. For instance, the domain of multi-user arcade games requires at least a service enabling multi-user communication.
- Management services that are needed to make service available and linked to business processes, even though they are not specific to end-user services.

However, other categories as the mentioned may be used. These categories are used as guidelines defining the structure of the conceptual architecture, and they assist in grouping services into reasonable blocks. Next, the features of the services in these domains are grouped. The features are related to requirements categories of services. The features provided by each domain guide for selecting the general requirements and constraints for service development.

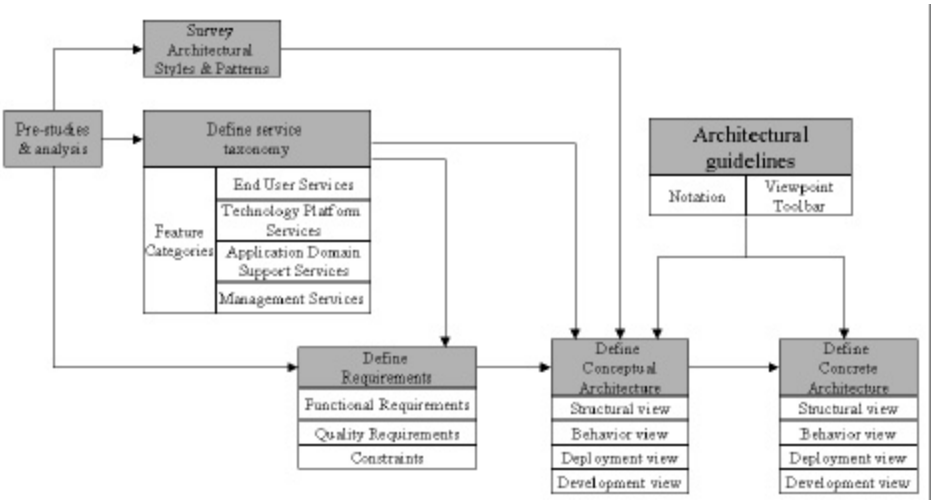

Fig. 1. The main steps of the approach for wireless service engineering

Before development of Pilot architecture, the task “architectural guidelines” defines a set of viewpoints modeling the conceptual and concrete architecture: each viewpoint represents a specific architectural aspect as described below. Guidelines also define each viewpoint’s notation [8].

Conceptual and concrete architectures are both organized into four views, which are: the structural view, the behavioral view, the deployment view, and the development view [3, 4, 5, 6]. The *conceptual* structural view maps functional and quality responsibilities on a conceptual structure, whereas the *concrete* structural view illustrates the component decomposition at the lower aggregation level and extends component and relationship descriptions into inner details. That is, both the aggregation and abstraction levels are affected while moving from conceptual architecture to the concrete one.

The *conceptual* behavioral view defines dynamic actions of and within a system. The *concrete* behavioral view aims to illustrate the behavior of individual components and interactions between component instances. The modeling notation for these views is based on Object Management Group (OMG) Unified Modeling Language (UML) [9]. Diagramming constructs for the first two views are described in **Table 1**.

Table 1. Modeling notations for the structure and behavior viewpoints

	Structural view	Behavioral view
Conceptual architectural methodology - It yields an architectural, specification-oriented perspective of the system	Entities and logical associations - Guidelines	Interactions among entities - Collaboration diagram
Concrete architectural methodology - It yields a technical, design-oriented perspective of the system	- Class diagram (detailed component structure) - Component diagram (shows exported component interfaces, interface usage and each component as a black-box module) - Component diagram (shows complex components as white-box modules with their internal decomposition)	- Sequence diagram (gross level, showing interactions among black-box components) - Sequence diagram (detailed level, showing how interactions are implemented internally to white-box components)

The remaining views of the approach are the *deployment view* and the *development view*. In particular, the *conceptual* deployment view describes the allocation of units of deployment to physical computing units. Units of deployment are atomic units (e.g. a group of components working together) that cannot be deployed across a distributed environment. The *concrete* deployment view focuses on the concrete hardware and software components and their relationships.

The *conceptual development view* describes the components to be developed and acquired. In addition, it figures out who is responsible for each service and which standards and enabling technologies do the services use. The *concrete* development view illustrates the realization of software components and their interrelationships. **Table 2** summarizes the modeling notations for the deployment and development views described above.

Table 2. Modeling notations for the deployment and development views

	Deployment view	Development view
Conceptual architectural methodology (methods + notation) - It yields an architectural, specification-oriented perspective on the system	Deployment diagram (service development + communication links)	On top of structural VP -Assorting components according to the level of completeness - Allocating development responsibilities
Concrete architectural methodology -It yields a technical, design-oriented perspective on the system	Networked structure Mapping of black-box components on top of the networked structure Business model and service provisioning	Technology details Hardware, software, external resources Templates Guidelines Implementation, configuration, usage procedure

The four views at two levels of abstraction cover the main aspects of a distributed service architecture, from static structure to dynamic behavior, and represent a complete set of views on a system under development. Of course, for simple systems, some views might be omitted because they are trivial or missing: it is up to the architects to identify which views are necessary, and omit the others. In other words, these views can be considered as a view toolbar, or a set of views at disposal for describing all and only those views relevant for the system under development.

3 An Initial Experiment

The WISE approach is applied to develop pilot wireless services (or Pilots). For each Pilot, both conceptual and concrete architectural views are described. For the sake of simplicity, the following shows just some views of the Pilot 2: the wireless interactive gaming service. The complete architecture can be requested on [10] and for the complete Pilot description refer to [11].

3.1 The Problem

The focus of the experiment is on architecting a wireless interactive gaming service, i.e. an arcade game supporting multiple players executing a mission in a dungeon labyrinth. This wireless entertainment service is composed of several smaller services, which together provide the functionality and qualities described below.

Players participate in the game using mobile devices (**Fig. 2**), with limited processing power and memory and therefore, it is obvious to delegate the common, shared playing environment to the server. This, in addition to a capability to manage several wireless network connections at the same time, results in the server to be a robust, thick server, whereas terminals are thin clients only hosting the user applications. This allocation of software conforms to the traditional client-server architectural style [12] with services centralized on a server and clients located across the network.

Fig. 2. The problem overview.

Non-functional requirements for our problem are described in **Table 3**. Each non-functional requirement is associated with a quality attribute refined with requirement definition and scope. That means refining what are the semantics of each quality attribute in this specific system, and where is the requirement context [13]. Non-functional quality requirements drive the definition of service architecture, which again drives the scoping of quality requirements.

Table 3. Non-functional requirements

Quality attribute	**Requirement definition**	**Scope/How**
Scalability	The number of concurrent players in the game is 2 – n.	Client-Server architecture style Component Game Server at server side
Portability	The game can be played with devices supporting either GPRS or UMTS connection. End-user devices support: PalmOS, EPOC, WinCE.	Client application. This requirement is realized by component Communication Manager Layered architecture style Client application includes a virtual machine layer
Extendibility	Features can be enriched, hence ensuring service evolution (in the WISE project, evolution is simulated by carrying out three development iterations).	Extension points in the architecture and code. - On the Server side e.g. quest sending, fights & attacks, manage high score list - On the Client side e.g. quest receiving and handling, fights/attacks/defending, buy/sell items
Modifiability	Services should be easily modified under the evolution of mobile terminals' hardware capabilities.	Client application. By separating communication manipulation and game management, and by splitting logically related functionality, modification is easier. Of course any modification is isolated if interfaces are not influenced.

Concerning development constraints, of particular relevance are the capabilities offered by mobile terminals, which usually put strict limitations on the provided service features. In particular, for wireless terminals we analyzed (for each service) the impact of both the terminal generation (2.0, 2.5 3.0, etc.) and the terminal equipment class (e.g. 10 Lines' screen, video resolution, memory, etc.). As an example, the Pilot service used in our experiment is not limited by terminal generation but it requires support for J2ME technology.

3.2 Applying the WISE Approach

After this short introduction to requirements, we now turn to the illustration and discussion of how the approach was applied in architecting a wireless service.

Conceptual Architecture. For the Pilot we present two main conceptual views, the structural view and the deployment view. The first view (**Fig. 3**) identifies the main elements and their relationships at the conceptual level. These elements are specified out of required macro-functions, defined during requirement engineering and carried out through interviews of industrial partners, and provide the initial functional-to-structural mapping of the service under development.

In particular, we can observe how this view presents the first draft of required domains (e.g. End-User Services, Technology Platform Services), distribution of elements on domains, and, particularly relevant for wireless services, roles in communication across domains (e.g. usage, data exchange, control).

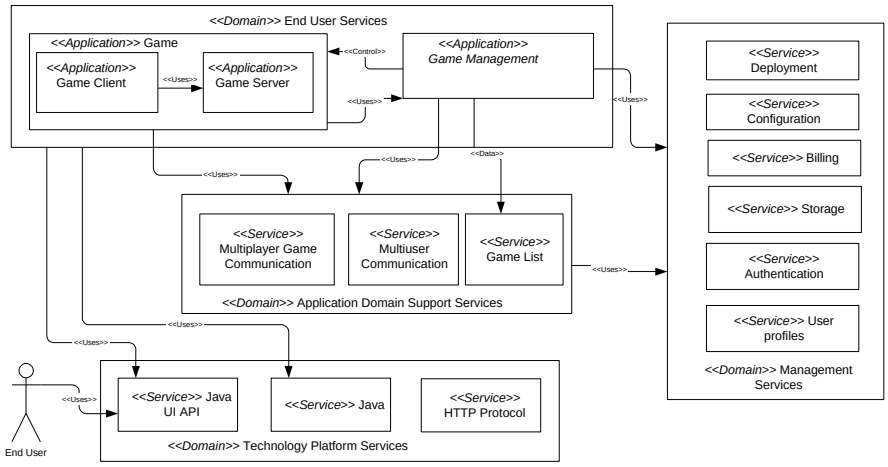

Fig. 3. Pilot conceptual structural view

The next step is to classify structural elements and their communication links. This is provided by the conceptual deployment view (**Fig. 4**).

On the conceptual level, the conceptual elements are treated as services. In this Pilot service, examples of services are Game Management, Game Server and Multiplayer Game Communication.

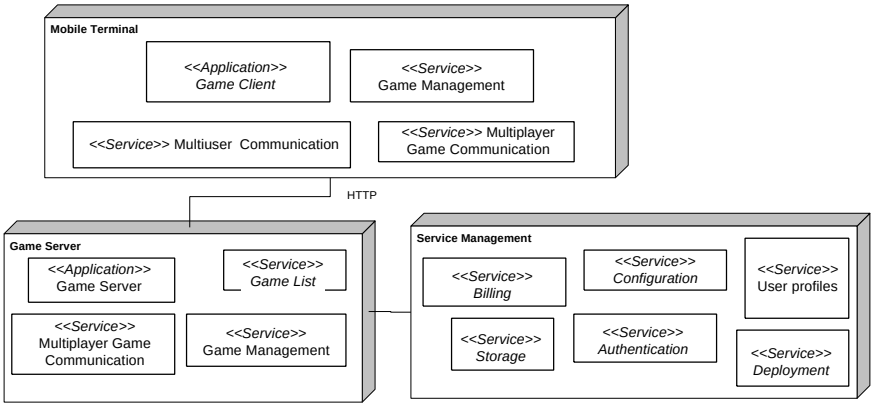

Fig. 4. Pilot conceptual deployment view

Concrete Architecture. For the Pilot, we present the concrete structural, behavioral and deployment views, as follows.

Structural View. The structural view shows the modules (classes and components) making up the service. In distributed architectures, these views provide the structure of distributed components and their interconnections.

On the concrete level, the *conceptual* services will be mapped on *concrete* components. Concrete components implement one or multiple conceptual services. Accordingly, we see that, (**Fig. 3** and **Fig. 5**) the concrete GameManager component (on the client side) implements the conceptual service Game Management, the Game Manager concrete component (on the server side) implements both Game Server and Communication Manager conceptual services, and so on. As a general rule, names assigned to concrete components will match those of conceptual services (if the mapping is one-to-one); otherwise, i.e. if a component implements multiple conceptual services, the concrete component name is decided independently.

In particular, in Pilot 2 architecture, **Fig. 5** depicts in white all components to be implemented, and in darker colors “external” components or technologies (reused or bought). External components are the authentication center and the database. The first component accesses the second, which stores User profile information. Both are located in the domain of a third party Service Provider in charge of managing orthogonal services in outsourcing. As another example, Kjava support is part of the technology already available on the terminal platform, and used by the Pilot client-side.

In the domain of the Service Provider, two components implement service-specific functionality: the Game Server component implements the game and the coordination of each game session. The Service Download Center component is a service-common component, providing support for the end-user to choose and download (on the client side) the application implementing the GUI of the game. In particular, the end-user can in principle first choose a game from an Application Provider (e.g. a game center), and only afterwards decide to play on-line game sessions supported by a Service Provider.

Finally, on the client side there are two components providing service-specific functionality: the Game Manager component implements the GUI and the processing of data related to an on-going game session. It interacts locally with the Communication Manager component that supports distributed communication with the remote Service Provider. Together, these two components implement the game, and are downloaded from a Service Download Center.

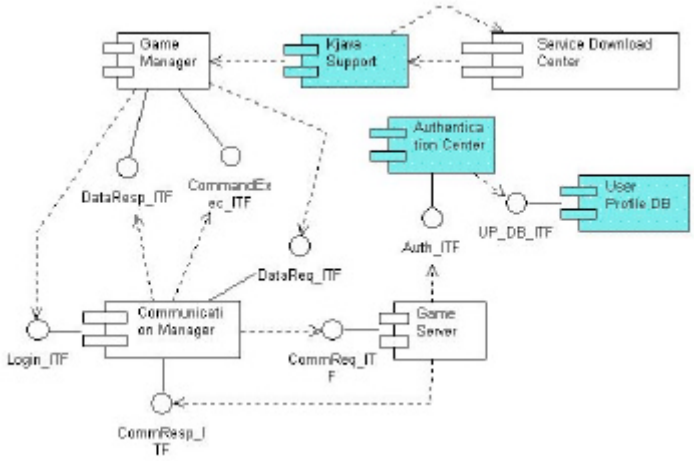

Fig. 5. Concrete structural view: System-level component diagram

Further, detailed component diagrams can be specified to provide the detailed structural view (in terms of internal elements and local interfaces) of those components yielding a complex structure. For example, **Fig. 6** shows the detailed structure of the Communication Manager component on the Pilot client side, i.e. how the Communication Manager is decomposed and how it implements its external interfaces.

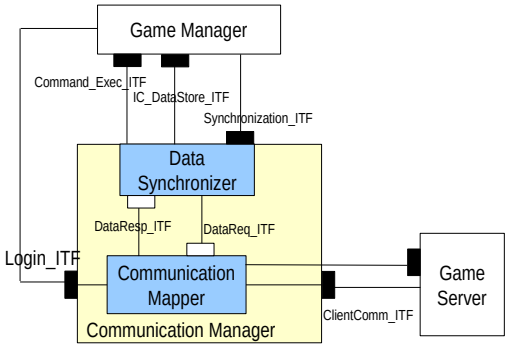

Fig. 6. Concrete structural view: Detailed component diagram for the Communication Manager component

Behavioral View. At the concrete level, Sequence Diagrams [14] model the concrete behavior scenarios: each diagram shows how components implementing the gaming service interact to provide the associated scenario. In particular, interactions among geographically distributed components identify which communication protocol is required on wireless connections, and supported by component implementation.

As wireless communication always involves user devices, the following will focus on the Pilot client side. An example of the Behavioral View is a fragment of the "Game start up" as depicted in **Fig. 7**. In Game start up, the user configures the game session and is ready to play.

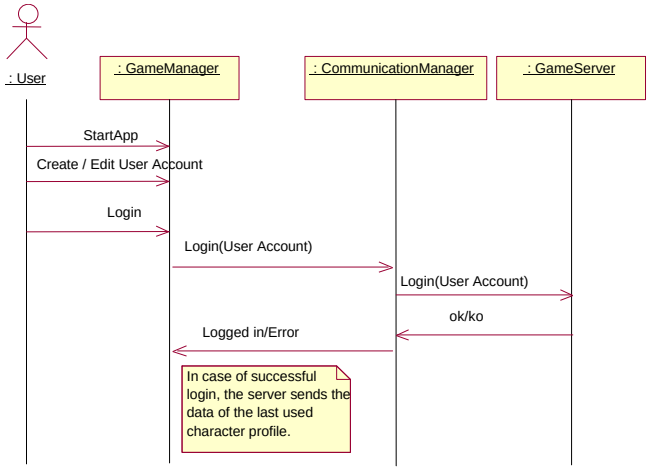

Fig. 7. Concrete Behavioral view: Start Application, create/edit user account, and login

Deployment View. The Deployment view focuses on the execution environment where the service will fit. This view shows two aspects: the first aspect is the business model [15] instantiated for the service, and the second is the deployment diagram mapping the system-level component diagram of the structural view, on the instantiated business model. For example, the instantiated Business Model (**Fig. 8**) and the Deployment Diagram (**Fig. 9**) compose the Pilot's concrete Deployment View.

The instantiated Business Model concentrates on those Business Roles and Business Relationships relevant to the Pilot[2]. Business roles in dark play some task in the operation of the Pilot service. This task can either involve service provisioning (see those roles inside the dashed box) if there will be some software components deployed in a networked structure, or not involve service provisioning (see roles outside the dashed box) if they have a business relationship prior to service provisioning (e.g. Technology provider). The latter seems particularly relevant for customers, who need to represent the complete service business chain.

[2] The business model specific to a selected Pilot architecture is instantiated from a generic business model (called the WISE Business Model [16]) defined for the wireless service domain. Project iterations aim at refining this along with the generic WISE architecture.

In particular, two issues need special attention:

- **The business relationship of Application provider to Service user/provider (ApplicProv):** it models "game download" prior to game provisioning. Game download supports the acquisition from the end-user of the application needed to play the game (i.e. client components). Download can be carried out from both a fixed node (e.g. using any Internet browser) and a mobile node. This aspect is under refinement, representing wired-wireless inter-operation.
- **The business relationship between Service providers (Peer):** the Pilot considers authentication and user profile storage as management services supported by a third party service provider. This aspect will be detailed during current Project iteration.

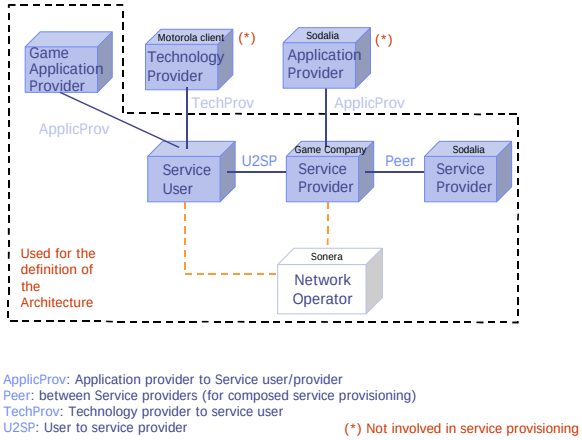

Fig. 8. Concrete deployment view: The WISE business model instantiated for the Pilot

The second part of the concrete Deployment View is the Deployment Diagram (**Fig. 9**): once defined, our instantiated business model, the Deployment Diagram maps the component diagram on top.

In particular, the diagram proves the following important issues:

1. The domain associated with role Service User, can be deployed on a fixed node (e.g. a PC connected to Internet) for game download[3], or on a mobile node (e.g. a 3G cellular phone or a 4G mobile device).

2. On the Service Provider side, there are two types of nodes mapped on two different domains playing the same role: The service node provides the game control, and service core components are deployed on the service node. Management service node provides the outsourced management services, on which service-common components are located. In particular, these components implement orthogonal services, like user profile access and storage, and authentication.

[3] Game execution can be also carried out on a fixed node. This scenario along with the analysis of QoS and development differences will be possibly investigated later in the Project.

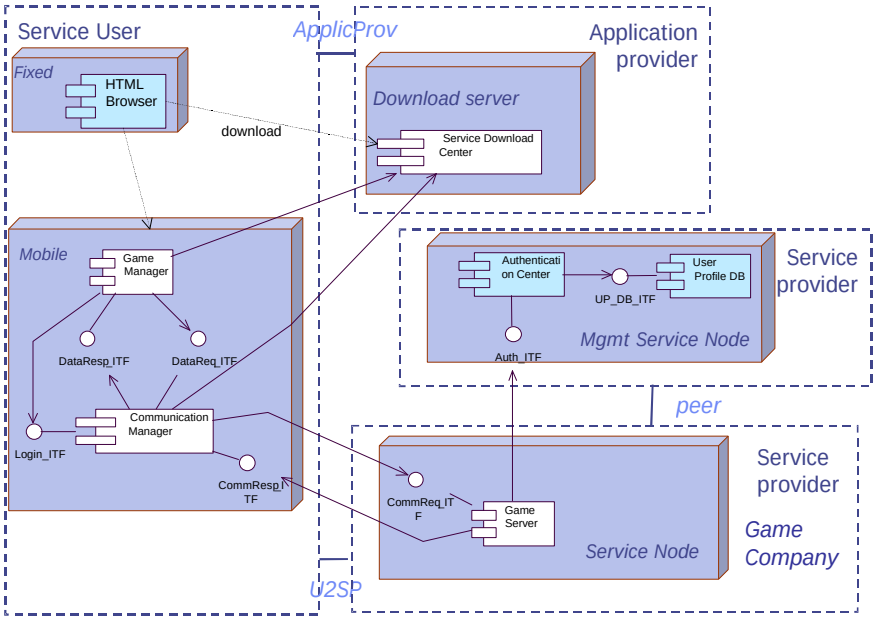

Fig. 9. Concrete deployment view: Deployment diagram

4 Discussion

As introduced in the first part of the paper, the WISE Project is still in its first iteration that concludes in the end of year 2002. At the moment of writing this paper, industrial partners are working on the Pilot implementation phase. Therefore, we do not yet have the final results for the complete development of wireless service Pilot architectures. Nonetheless, we can already draw some initial conclusions, and report feedback from these industrial partners, in particular when applying the WISE approach to service engineering.

4.1 Learning curve

At the beginning of the first iteration, the WISE approach has been designed and explained in a document stating definitions of terms, guidelines for applying it, and the semantics and notation of viewpoints. The whole approach has been explained to industrial partners in special meeting sessions especially conceived for training. This, in consideration of the well-known fact that one of the most critical steps in introducing new working procedures is people resistance to changes.

In addition, flexibility in learning new concepts decreases with time. Nonetheless, in this scenario, the use of examples of toy wireless services has been of great help, as

has the fact that the approach viewpoints are based on OMG UML, which is familiar to all industrial partners, hence providing a common initial background.

After the "training phase", service engineering started with interview sessions in which specialists in the proposed approach elicited requirement and design information from people in charge of service implementation. This kind of "tutored architecting phase" has been fundamental as a starting point for overcoming people resistance and learning how to proceed. Afterwards, developers continued the work on their own, with some sporadic off-line consultancy. Finally, we could observe that technical people already used to innovation are more receptive than business-oriented people accustomed to settled working methods.

4.2 Tool support

We said that previous knowledge of UML among people has been of great help in overcoming resistance to a new working approach. In this respect, the use of a software tool for requirements engineering and design, has been particularly helpful in the learning process. After an analysis of available tools and their potentialities, as well as familiarity of tools, we decided to adopt a commercial UML tool that is familiar to most of them and widely available, and to adapt the tool to support viewpoint diagrams in addition to pure UML diagrams. In spite of the difficulties in drawing, developers applied the approach to their work (not without any difficulties). We can say that current state of Pilots' wireless service architecture is maturing fast. Even industrial partners not used to service engineering at all, are now using the approach effectively.

4.3 Market- and business-related issues

Among industrial partners, business people found that business views focusing on market analysis are missing. Used to industrial service engineering in which market-driven motivation is necessary, they experienced a gap between the four viewpoints defined by the WISE approach, and "real world" aspects like market analysis and revenues. It must be underlined that the approach is meant for technical purposes only (i.e. to engineer the software architecture of services), and business issues are supposed to be worked out and solved in advance. Nevertheless, diagrams like the business model provide a natural interface from service architecture to market analysis and business-related issues.

4.4 Network viewpoint

During requirements analysis, industrial partners made use of informal drawings to depict the networked structure of devices, machines and terminals involved in service provisioning. This informal network diagram seems to be very effective in understanding the execution environment of the service under development, and in spite of its informality, it provides a valuable common communication mean. This aspect should be included in the development view (it is under investigation if at the

conceptual or at the concrete level). Its role will be mainly to identify hardware technologies, network configuration, communication protocols and external software components.

4.5 Development viewpoints

Another ongoing work in the WISE Project is the definition of the contents and the border between concrete and conceptual development viewpoints. We can agree that this viewpoint provides (at the concrete level) service documentation when service development concludes or at least is ongoing (examples of development documentation are installation and release notes, usage instructions, hardware and software pre-requisites, etc.). Nonetheless, also at the conceptual level technology adoption or hardware/software limitations must be considered in the development process as early as possible. An important issue then, is to decide what belongs to which level, and to avoid overlapping.

5 Conclusions

The aim of this paper was to introduce an approach to architect wireless services. In addition, it was to illustrate the adoption of this approach with an example. The example was wireless interactive gaming service under development in the WISE Project. This paper also presented the initial experiences gained during the first development iteration.

The WISE approach identifies development steps, and a set of architectural viewpoints and associated visual notation. The main steps are to survey architectural styles and patterns, to define service taxonomy, and to define service requirements. These are grouped into functional and quality requirements and constraints. With the influence of these initial steps, we define the conceptual architecture. Conceptual architecture is described with up to four architectural views: structural, behavioral, deployment and development. Conceptual architecture is then refined into the design-oriented concrete architecture, which describes the similarly named four views at more technical level with specific details and lower-level decomposition. This paper represented selected views and diagrams, as examples.

In the WISE Project, these steps will be processed iteratively three times. The first development iteration is described here, and it comprises only the basic features of the game; these basic features will be enriched in the two future iterations.

We are currently adopting the WISE approach in the industry and results have been encouraging. The resistance to change is already overcome with successful help of training, consulting and example toy services. In addition, the UML-based notation has been easy to learn and also there is also tool support by adapting familiar commercial UML tools.

Except for already achieved good results, we have few issues to solve or improve, especially in relationship to the network viewpoint and the development viewpoint as explained in the previous section.

Acknowledgements

We would like to thank all partner members of the WISE project, especially Mr. Kalaoja, Mr. Tikkala, professor Niemelä and Mr. Boggio who are the co-authors of the architecture guidelines and the game service architecture.

References

1. Matinlassi, M. Niemelä, E. and Dobrica, L: Quality-driven architecture design and quality analysis method. A revolutionary initiation approach to a product line architecture, VTT Publications 456, Technical Research Center of Finland, Espoo, FI, 2002.
2. Purhonen, A., Niemelä, E., Matinlassi, M.: Views of DSP Software and Service Architectures. Submitted to Journal of Systems and Software. 30 p.
3. IEEE Std-1471-2000: IEEE Recommended Practice for Architectural Descriptions of Software-Intensive Systems. IEEE Computer Society.
4. Krutchen, P. B: The 4+1 View Model of Architecture, IEEE Software Vol. 12(6), 1995.
5. Jaaksi, A., Aalto, J-M, Aalto, A. and Vättö K..: Tried & True Object Development. Industry-Proven Approaches with UML, Cambridge University Press, New York, 1999.
6. Hofmeister, C., Nord, R. and Soni, D.: Applied Software Architecture, Addison-Wesley, Reading, MA, 2000.
7. Gamma, E.: Design Patterns: Elements of Reusable Object-oriented Software, Addison-Wesley, Reading, MA, 1994.
8. Niemelä, E., Kalaoja, J., Lago, P., Tikkala, A.. and Matinlassi, M.: Conceptual Architecture and Guidelines to Use It, V. 1.0, IST 30028 WISE Project, Deliverable D4, April 2002.
9. OMG: The Unified Modeling Language (UML) Resource Page. On-line at http://www.omg.org/uml
10. WISE: The "IST WISE Project home page". On-line at http://www.wwwise.org
11. Lago, P., Boggio, D., Tikkala, A. and Forchino, F.: Architecture for Pilot 2, V. 1.2, IST 30028 WISE Project, Deliverable D11, June 2002.
12. Bass, L., Clement, P. and Kazman, R.: Software Architecture in Practice, Addison-Wesley, Reading, MA, 1998.
13. Matinlassi, M. and Niemelä, E.: Designing High Quality Architectures, International Conference on Software Engineering, Workshop on Software Quality, Orlando, FL, May 2002.
14. Lago, P.: Rendering distributed systems in UML, In: Siau K. and Halping D. (Eds.): Unified Modeling Language: Systems analysis, design and development issues, Idea Group Publishing, 2001.
15. TINA-C: "TINA Business Model and reference Points", TINA-C Baseline, v4.0, May 1997. On-line at TINA Consortium (Telecommunications Information Networking Architecture) Web site, http://www.tinac.com
16. Kallio, P.: Business Models in Wireless Internet, V. 1.0, IST 30028 WISE Project, Deliverable D11, May 2002.