WIRELESS INTERNET SOFTWARE ENGINEERING IST-2000-30028

Title:
WISE: Analysis of Pilot Architectures

Version: 1.04
Date : 25 Oct 2002
Pages : 13

Author(s):
E. Niemelä

To:
WISE CONSORTIUM

The WISE Consortium consists of:

Investnet, Motorola Technology Center Italy, Sodalia s.p.A, Sonera, Solid EMEA North, Fraunhofer IESE, Politecnico di Torino, VTT Electronics

Printed on:
15-Dec-03 10:48

Status:	**Confidentiality:**	
[ ] Draft	[ ] Public	- Intended for public use
[ ] To be reviewed	[X] Restricted	- Intended for WISE consortium only
[x] Proposal	[ ] Confidential	- Intended for individual partner only
[ ] Final/Released		

Deliverable ID: **D4 C: Architecture Analysis for WISA**

Title:

WISE: Analysis of Pilot Architectures

Summary / Contents:

This document is a part of the deliverable D4 produced in the task 2.1 of the Wise project. Deliverable D4 includes four parts: Part A: Architectural guidelines, Part B: the WISA (Wireless Internet Service Architecture) architectural knowledge base and its reference architecture (WISA/RA), Part C: Analysis of the pilot architectures, and Part D: Handbook of reusable architectural assets.

This document summaries the results of analyses of Pilot architectures (Pilot 1 and Pilot2), versions 1.

TABLE OF CONTENTS

CHANGE LOG

Vers.	Date	Author	Description
00.01	20 Aug 2002	E. Niemelä	The applied analysis approach described.
00.02	26 Aug 2002	E. Niemelä	Analysis results of Pilot 2
	27 Aug 2002	E. Niemelä	Analysis results of Pilot 1
00.03	27 Aug 2002	E. Niemelä	Some corrections
00.04	28 Aug 2002	M. Matinlassi	Figure 1 added, minor changes in text
01.04	25 Oct 2002	E. Niemelä	Version number corrected

APPLICABLE DOCUMENT LIST

Ref.	Title, author, source, date, status	Identification
1	Architecture of Pilot 2, Lago, Tikkala, Forchino	v. 1.2
2	Architecture of Pilot 1, Lago, Kalaoja, Palladino, Piancino	v. 1.3
3	Pilot service 1 requirements, Palladino, Piancino	v. 1.2
4	Pilot service 2 requirements, Boggio, Forchino	v. 1.1

1. INTRODUCTION

This document describes the results of architecture analysis of the first versions of Pilot 1 and Pilot 2.

In the first phase, architectural descriptions of pilots had incomplete information, and therefore, architecture analyses method (like the QADA approach [Matinlassi et al. 2002]) could not be used. However, it is essential to start to discover the items that need improvements in the architecture. Thus, the work has been started by defining a taxonomy for external and internal qualities of Wireless Internet Service Architecture (WISA), i.e. the generic service architecture that is one of the main goals in the WISE project.

The internal qualities are the non-functional properties of a software service (products) that are important for the developers of that part of software in question but may be invisible or unimportant for the other stakeholders involved in the service development. The external qualities are the quality requirements that have to be visible for the stakeholders that use the software when they develop the final product, an Internet based software service, and provision it to end-users.

Various stakeholders of wireless Internet services, i.e. users, application developers, platform (i.e. middleware and infrastructure) service developers and network operators, prefer different qualities (Figure 1). External quality provided by a stakeholder is a prerequisite for internal quality of another stakeholder in the stakeholder 'stack'. The real quality of a service, i.e. how well the service meets all end-user's requirements (cost vs. benefits), defines the **real added value** for an end-user. This quality is achieved only if prerequisite technical and economic qualities are met.

Applicability, i.e. how easily the application can be applied in different contexts, is a quality that the application developers are most interested in. This quality is visible as external qualities through GUI's usability, performance of the application, easiness to use a service by a scaling number of end-users.

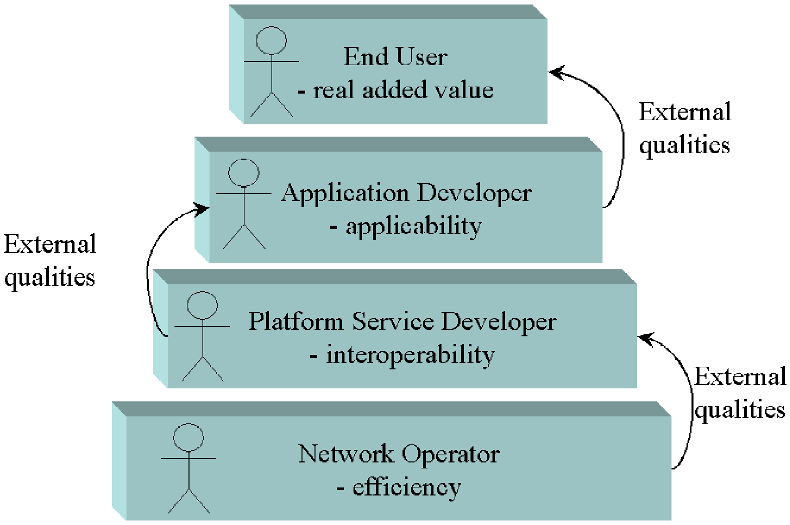

Figure 1. The Real Quality accumulates in co-operation with various stakeholders.

Interoperability of platform services is the criterion a service developer considers as required quality of the software, when a service is provisioned. Interoperability is achieved if platform services (i.e. middleware with communication and management services) are generic and new platform services can be easily integrated by aggregating old ones (horizontal integration). The same platform services should be also able to be used in new sets of applications (vertical integration) and application developers should be able to use them easily (simplicity of provided application interfaces). In order to be profitable (i.e. development cost and time-to-market) from the service provider's point of view, the platform services should also be portable, modifiable for different applications, expandable, maintainable and usable and accessed by application developers. Performance is a quality that has to be considered in a different way from each stakeholder's viewpoint. **Efficiency** that is visible in performance and price, is the quality the network provider is responsible for.

As a summary, the following quality attributes are the most important in the development of WISA (Table 1):

- External qualities: integrability, reusability and simplicity.
- Internal qualities: portability, modifiability, performance and usability.

Some of these qualities can be analysed after the first implementation is ready, i.e. with the user of the platform services but some of them are visible in the service architecture even from the first draft namely;

- Portability
- Maintainability
- Integrability and
- Simplicity.

Therefore, the 1st analysis of the Pilot architectures concentrates on these four quality attributes (see Table 1).

Table 1. A taxonomy of the qualities of wireless services.

Stakeholder	Type of quality	Internal quality	External quality	Means	Importance in WISA
User	Real added value	Connection establishment delay Reliability Context awareness	Total quality	Valuable to the user Price vs. quality	No No
Application developer	Applicability	Modifiability of application Reusability of application	GUI usability (entertainability) Performance of the application Simplicity (easiness) Scalability	Reusable and modifiable application services Required performance and scalability	**Low** **Moderate**

Table 2. A taxonomy of the qualities of wireless services continued.

Stakeholder	**Type of quality**	**Internal quality**	**External quality**	**Means**	**Importance in WISA**
Platform service developer	Interoperability	Portability to various execution environments Modifiability for different application services Platform performance Extendibility of service features Maintainability of platform Usability of platform	Horizontal and vertical Integrability Reusability for a set of application services Simplicity of platform service interfaces (easy to use)	Stable and dynamic platform (at the same time)	**Must**
Network provider/ operator	Efficiency	Latency (round trip delay), transit delay (one-way delay) Throughput (bandwidth (bits/s) IPDV =instantaneous Packet Delay Variation (jitter) Residual error ratio (packet loss %) Cost of CPU, memory Reliability	Performance Reliability Price	Resource friendliness; effectiveness	No

In the following sections, only the quality attributes of most importance (portability, maintainability, integrability and simplicity) are considered based on the requirement specifications and architecture descriptions available from pilot architectures (Doc. Ref. 1-4).

2. PORTABILITY

Portability is the ability of the system to run under different computing systems: hardware, software or combination of the two.

Diversity of implementation technology

In Pilot 1, the architecture is tightly intertwined round the used technologies: WAP and HTTP. Conceptual architecture describes layers for domains (e.g. technology platforms and domain support services) but these layers are not visible in the concrete architecture. However, handling of the diversity of end-user information is allocated to one component, Intelligent Gateway. Because the end-user service should be able to run in fixed networks and wireless networks, the differences between HTML and WAP browsers should be supported by portability mechanisms inside Intelligent Gateway.

The Pilot 2 architecture is heavily based on existing implementation technology; Java MIDlets on the client side and Java (J2EE or J2SE) on the server side. Although portability is not considered in the requirements specification, from the Pilot 2 architecture description can be seen that Game Manager is the only component that is directly connected to the Kjava support component. Therefore, portability of client software could be supported by a technology platform specific layer that provides required services for communication, graphical user interfaces etc. This layer has been defined at the conceptual level but is not visible at the concrete level.

Suggestion:

- The layered architecture style should be applied to support portability both in the client and server side. This can be implemented as insulator library components.

Rationale:

- When technology-specific interfaces are concealed inside a layer, a change in the implementation technology does not affect other parts of the service (scope of the change is known).
- As library components, the layer can be shared and used by several service engineers; this provides the starting point for generic components needed in WISA.

Diversity of communication

In Pilot 1, the client connection will be implemented with WML language and GSM/GPRS connections in the first phase. If UMTS will be used or how its use affects on architecture has not been considered.

In the requirement specification of Pilot 2, diversity of communication is defined as GPRS and UMTS. GPRS is used in the first phase, UMTS in the second and/or third phase. Although this is a fundamental issue it is not considered at the architecture level. An initial communication description between a client and the server has been defined but how the change over from GPRS to UMTS affects on service development is not considered.

Suggestion:

- Analysis of wireless network topologies, i.e. what the wireless Internet really means and how it affects on software architecture.
- GPRS and UMTS specific parts of software on the client and server part should be identified and isolated into separate components in the pilot architecture (places of variations are known).

Rationale:

- Impact analysis of wireless Internet is necessary in order to anticipate changes required in architecting.
- Isolation provides an ability to run the same end-user service in both environments with minimum extra work.
- Candidates for generic services (middleware or application family related) might be identified.
- New features provided by UMTS can easily and fast be utilised in the application development and enables using new business opportunities provided by UMTS.

3. MAINTAINABILITY

Maintainability is the ease with which a software system or component can be modified to correct faults, improve performance, or other attributes, or adapt to a changed environment [Dobrica & Niemelä, 2000].

As it can be seen from the definition, maintainability is related to portability. However, it is broader concept considering the whole life-cycle of a service and its execution environment.

Easiness to make architectural modifications depends on at least, the following prerequisites exist:

- Architecture is described in the same way; meaning of terms and notation is shared and descriptions are done in same accuracy.
- Architecture should be carefully documented; necessary information is available, dependencies between architectural elements have been identified and defined, traceability between descriptions is supported and reasons behind design decisions are documented.
- Implementation conforms to the architectural descriptions and defined standards.

Maintainability is not mentioned in the requirement specification and the architecture descriptions of Pilot 1 and Pilot 2.

Meaning of terms and notation

Although in WISE we defined a short vocabulary and guidelines how the pilot architectures should be defined, there are misunderstandings and weaknesses. The term of architecture is not absorbed as it is intended to use in the WISE project. This may be due to defective tutoring, unwillingness to change design practices and/or adapt the QADA method that has documented guidelines with examples but it is a new method and approach in WISE.

Suggestion:

- The structure of the architecture document (a map view) will be (re)defined and the architecture of Pilot 2 is used as an example architectural documentation. (Now only the client side of the architecture has been described).
- The architectural descriptions of pilots will be refined according to the new document map. This work will be done by pilot partners until the end of the phase I or in the beginning of the phase II.
- The notation will be strictly fixed and anomalies are not allowed without good reasons (if any).
- A tutorial inside the WISE consortium will be arranged within the next meeting.
- The tutorial can be used as a teaching material in industrial companies (by their own).

Rationale:

- Structure, terms and notation form the common language the architects communicate with.
- Document structure is essential in maintenance when the software engineers need to understand the existing software in order to determine what to change and how it can be done.
- Improving knowledge of software architecture is critical for WISE in order to get the generic architecture (i.e. WISA) validated by the pilots.

Dependencies and traceability

Two major weaknesses of recent documentation of pilot architectures are the incomplete interface descriptions of components and lacking traceability between the viewpoints of the same abstraction level and between the conceptual and concrete levels.

The aim is that the conceptual level captures commonalties and variabilities and provides overall information without technical details. This description is used for communication between heterogeneous stakeholders (i.e. managers, developers etc.). The objective of the concrete architecture is to provide component descriptions with strictly defined interfaces. The concrete architecture is used as a specification when the needed components are allocated to the software developers in or outside organisation, i.e. to be developed by the company itself, ordered from a subcontractor or bought from a commercial market place.

Suggestion:

- The strictly defined interfaces are part of the development viewpoint that also includes the dependency diagram. Dependencies on selected technologies are defined in a separate diagram in the development viewpoint. (Now development viewpoint is missing).
- Mapping between abstraction levels and maybe even a new viewpoint (for mapping) is required.
- A large amount of Message Sequence Charts (MSC) diagrams should be avoided by preferring collaboration diagrams and interface descriptions of good quality.
- The name of a component is a part of its identification and therefore, the name should remain the same in every description it is mentioned.
- Design rationale will added to each viewpoint, might be necessary in each diagram, too.

Rationale:

- A service is developed in co-operation with several actors and interface descriptions are required in work-break-down and communication during the whole life-cycle of a service. In WISE, phases I...III simulate incremental development and new work-allocation may be possible during the project.

- Platform services (i.e. middleware and infrastructure services) are intended to be in long-term use and therefore, maintenance should be started from the first step, too.
- Due to poor traceability, the documents are difficult to understand, modify and use, and that is costly to service developers and service providers.

Conformance with the architecture

Conformance with the architecture is crucial in service engineering when an application (service) is added and maintained 'on-the-fly' during the life-cycle of the platform services. Because detailed design information or implementation of components is not available yet, this part of analysis has to be postponed.

Pilots 1 and 2 define a protocol, HTTP 1.1 that is used for communication between the client and the server. However, the reasons behind the selection of the client – server style have not been documented.

Suggestion:

- Setting of the quality goals is essential in service engineering in order to get the benefits from quality-oriented software development. Reasonable quality means that return on investment (ROI) should be considered for the different parts of software, i.e. the services that are used only in one application and the others that are used for a family of services. The reasoning rules of economic benefits might be input information from business models to the architecture development.
- Conformance testing should be included as a new activity to the WISE approach. This should be considered in the WISE process models as well as architecture guidelines in the second iteration phase.

Rationale:

- The purpose of WISA is to support the development of successful end-user services. A generic architecture will not be adopted in industry if it does not provide guidelines how to adopt the WISA architecture and how to ascertain that all components included to the service architecture follow the same 'design rules'.
- This might lead to the contributions that can also be valuable in the standardisation bodies.

4. INTEGRABILITY

Integrability means the ability to make the separately developed components of the system work correctly together [Dobrica & Niemelä 2000]. Interoperability is a special case of integrability that measures the ability of a group of parts that constitutes a system to exchange information and use the one exchanged. Interoperability is here omitted because portability has a similar overall purpose. On the other hand, integrability has been separated into two parts, namely horizontal integrability and vertical integrability. The purpose is to classify the service developers and their products into two categories: those that aim at global software markets with generic service products and those that provide customised services to end-users.

In Pilot 1, the main goal is to convert the existing end-user service to the wireless network. Therefore, the architecture description emphasises on application development. However, due to several connections between Web Server and Server Farm, User Profile DB and Quots Cache DB, the Web Server component seems to be a natural place where the architect should take account of horizontal integrability. On the other hand, vertical integrability should be provided by the Intelligent Gateway component, and interfaces between it and Web Server. Especially, interface descriptions require more emphasise in the next phases.

In Pilot 2, horizontal integrability is considered at the conceptual level but due to missing definitions of the management services only some observation can be made. The interface between Communication Manager and Game Server has been defined at the semantic level. In order to get the separately developed components integrated the interfaces and protocols need to be defined strictly. Vertical integrability is difficult to see from the concrete architecture. An assumption is that the Communication Manager provides a 'generic' communication service to the Game Manager that is tightly coupled with the game application service. This means that the communication service might be used for other services too but the Game Manager service has to be developed separately for each game.

Suggestion:

- Integration should be supported by a separate interface description with the protocol definitions in the development view.
- Integration interfaces should be generic and the first ones to be fixed in WISA.

Rationale:

- The goal of WISE is to contribute wireless Internet service engineering. The means should be a quality-driven design method that can easily be followed and adopted in industry, and standard architectural descriptions (i.e. WISA) that can be adopted as a partial solution, e.g., in OMA (Open Mobile Architecture).

5. SIMPLICITY

Simplicity here is defined as an easiness to apply and use the architecture descriptions and therefore, closely related to usability and reusability [Dobrica & Niemelä 2000]. However, simplicity in this context means the ability to use the platform services for different kinds of end-user services, and add or create new platform services when richer applications require more powerful platform services or new adopted technology makes it possible to develop new support services or simplify their implementation.

In summary, this ability is lightly considered at the conceptual level in both pilot architectures. Due to missing interface descriptions and API for the game family this ability should be addressed much more in the next two phases. The same is in Pilot 1.

Suggestion:

- In the phase II, variability of services should be described in the most important services. The potential service categories that need variability support are: user interface services of mobile terminals, communication services, authentication services, and (game) application management services.
- Due to simplicity, a generic model, i.e. a pattern, is preferred.
- The scope of WISA will be limited to the selected service categories.

Rationale:

- WISA will include only the parts of pilot architectures that have most potential for being generic, standardised, widely accepted and adopted.

6. THE FIRST STEPS TO BE DONE

There are some other problems and weaknesses in pilot architecture descriptions. Only the most serious ones are listed here.

1. Components in the concrete structural view do not conform to the components of the conceptual structure. Architectural styles are not visible at the concrete level. *Conclusion:* In all views, especially in the structural view design rationale has to be documented so that also other designers know the reasons behind design decisions. Styles selected at the conceptual level should be followed at the concrete level as well as selected design patterns.
2. Collaboration should describe the actions that have to be taken in order to provide desired functionality and map these actions to services and components. Instead, collaboration model is confused with the Use cases and the MSCs and therefore, not used at al. *Conclusion:* Collaboration model **maps** the most essential use cases or groups of use cases (less than 5) to the conceptual structure diagram. Collaboration model does *not* equal with the information provided by the use cases or message sequence charts.
3. Duplicate information should be avoided. MSCs are widely used but interfaces are still undefined. *Conclusion:* Instead of external components strictly defined interfaces should be defined in the concrete structural view. Instead of a large amount of MSCs strict interfaces should be used.
4. Notation is complicated; there are three types of components that have two or three differences in their symbols (e.g. colour, shape, naming style, colour of an interface symbol), *Conclusion*: Notation should be simplified.
5. Concrete architecture descriptions are too vague; component implementation requires stricter specifications and the third design phase, detailed component design, is too much. *Conclusion:* Concrete architecture descriptions should be refined concerning interface descriptions, communication protocols between components (including data format), internal structure of components (e.g. Intelligent Gateway) and design rationale.
6. Information of different abstraction levels is mixed in the same descriptions. *Conclusion:* In phase II, more emphasise will be put on tutoring of pilot service engineering.
7. The same component or service may have two or three different names. *Conclusion:* Traceability has to be addressed more on in each design phase. The same components should be identifiable in the code, too.
8. The goal of WISE; development of wireless application services or development of generic services for wireless applications. *Conclusion:* A couple of slides that highlight the objectives of pilot architectures in a compact way. For example, what is the main architectural objective of each pilot (work break-down), what has to be the contribution of each pilot in each phase (place in the WISA context) and how the targets can be achieved (tutoring).

7. REFERENCES

Matilanssi, M., Niemelä, E., Dobrica, L. 2002, Quality-driven architecture design and quality analysis method. A revolutionary initiation approach to a product line architecture. Espoo: VTT Technical Research Centre of Finland, VTT Publications 456, 128 p. + app. 10 p.

Dobrica, L., Niemelä, E.. 2000. A strategy for analysing product line software architectures. Espoo: VTT Technical Research Centre of Finland, VTT Publications 427, 124 p.