December 2002

ARCHITECTURAL ANALYSIS FOR WIRELESS INFORMATION SYSTEMS

Luis Mendoza
Universidad Simón Bolívar

Maria Perez
Universidad Simón Bolívar

Yorka Carvajal
Universidad Simón Bolívar

ARCHITECTURAL ANALYSIS FOR WIRELESS INFORMATION SYSTEMS

Luis E. Mendoza
Universidad Simón Bolívar
lmendoza@usb.ve

María Pérez
Universidad Simón Bolívar
movalles@usb.ve

Yorka C. Carvajal
Universidad Simón Bolívar
ycarvajal@usb.ve

Abstract

The impact on the intense exchange of information makes essential the use of Wireless Information Systems (WIS). Further, the use of standards like the Wireless Application Protocol (WAP) in this kind of system means that the applications can be made available on a massive scale. Before implementing a WIS designed with this type of standard, it is necessary to analyze the different scenarios in which it might be implemented because these systems can be very complex, being very expensive or impossible to improve them once developed; for this reason the importance of carrying out an evaluation architectural of their design before developing it. This paper describes the architectural analysis of a WIS based WAP architecture for an insurance company. The analysis focuses the Modifiability and Portability attributes because they are deemed to be fundamental for the WIS client company. The best architecture was selected using the evaluation method known as the Architecture Tradeoff Analysis Method (ATAM).

Keywords: Wireless Information Systems, architectural evaluation, mobile commerce, wireless/mobile standards, case study

Introduction

The concept of electronic commerce (e-commerce) today not only encompasses the electronic purchase and sale of goods, information and services, but also the use of a network that handles pre and post-sale activities, such as (Turban et al. 2000): (1) advertising; (2) the search for information on products, suppliers; (3) negotiation between purchaser and vendor over price, delivery conditions; (4) pre and post-sale customer service; (5) completion of administrative formalities connected with e-commerce activities; and (6) collaboration between companies with common business interests (on a long or short term basis).

Thanks to the boom in wireless communications, the e-commerce market will be mobile communications: mobile commerce (m-commerce). The challenge will be to transfer all the possibilities of e-commerce to m-commerce, taking into account the limitations of mobile devices and the communications infrastructure. This is where Wireless Information Systems (WIS) fit in. WIS are systems that use wireless technology to communicate a mobile client with another system component and Wireless Application Protocol (WAP) has become the standard for this kind of systems. WIS can be very complex systems.

Therefore is needed a study of the situations and problems that arise when developing and implementing a WIS so that an organization's strategies can be supported through them. This paper presents the use of WAP protocol in an insurance company that had not previously explored this technology. It's important to indicate that WAP protocol was proposed by Consis International; in other words, it was a design requirement for WIS. It also included the definition of an insurance Application Programming Interface (API) with a degree of generality and the evaluation of architecture using the Architecture Tradeoff Analysis Method.

The most important contribution in this paper is that it presents the architectural evaluation of a WIS. This evaluation demands to identify the aspects to be taken into account to the moment to design the architecture of the WIS, in the early stages of the life cycle because is more beneficial and less more expensive within the development process of any system (Bass 1998). On the other hand, ATAM was used to be a architectural evaluation method that facilitates the system understanding, reducing the risks and allowing to modify the design on the base of the quality attributes selected by the company client of the system; in this case *Modifiability* and *Portability*.

Background

Wireless Application Protocol

WAP is a global open standard that allows users of mobile devices (cellular telephones and Personal Digital Assistants -PDAs-) to have access to and interact with information and services available on the Internet (Goldman 2000). It was designed to integrate, enhance and standardize various connectivity approaches of this type of device with the Internet. Wireless Markup Language (WML) is used to specify WAP contents and user interfaces. WML is a markup language based on the Extensible Markup Language (XML) designed to work with small wireless devices with limited resources in terms of memory, power, bandwidth, etc.

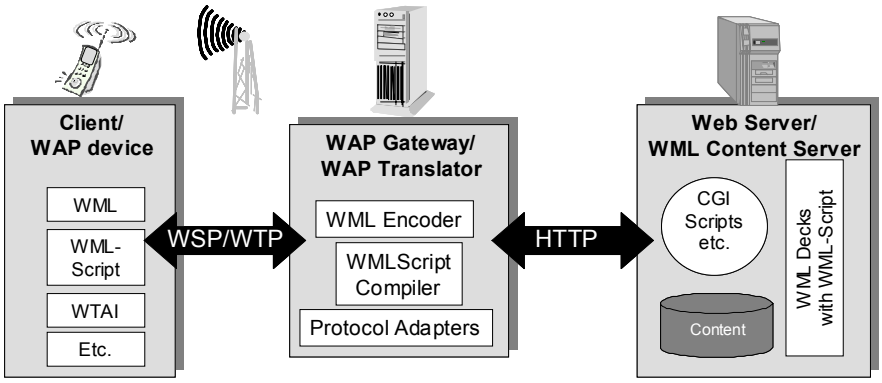

Figure 1. WAP Architecture (Flaherty 1999)

WAP uses existing Web technologies to facilitate its adoption by users and developers. Therefore WAP and World Wide Web (WWW) architecture are quite similar to one another, except that the former is geared to the requirements of wireless communications

Figure 1 shows the basic architecture proposed by the WAP Forum as the reference architecture for WAP. It is a three-tiered architecture: mobile client or WAP device, WAP gateway or WAP translator and Web Server or WML content server.

The interaction between an application designed on the basis of this architecture and the user works like this: (SAS 2002)

1. The client selects a Uniform Resource Locator (URL) through the mobile device, just like a Web page.
2. The request is transferred to the WAP gateway, using the WAP protocol (through the Wireless Session Protocol -WSP- and Wireless Transaction Protocol -WTP-), which is independent from the carrier service.
3. The Web Server processes the Hyper Text Transfer Protocol (HTTP) request, in any valid form of standard URL (an Active Server Pages -ASP-, a Common Gateway Interface -CGI-, or a Servlet).
4. The Web Server returns a file in WML format to the WAP gateway.
5. The WAP gateway codes the WML file and returns it in binary form to the mobile device that requested it
6. Then the device interprets the WML file and shows it on the screen.

Architecture Tradeoff Analysis Method

In large software systems, obtaining quality depends not only on the practical levels of coding, but also on software architecture in general (Bosch 2000). In such systems, quality attributes may restrict the architecture, which explains the interest in evaluating and determining the appropriate one (Bass et al. 1998). There are many qualitative and quantitative techniques for analyzing quality attributes; nevertheless, analyses of attributes are interdependent, in other words each attribute is connected to other attributes.

Architecture Tradeoff Analysis Method (ATAM) was used to make the architectural evaluation, which is the objective of this article. It is a method that analyzes software architectures based on multiple quality attributes. ATAM is a spiral-based method divided into four phases: (1) Scenarios and Requirements Gathering; (2) Architectural Views and Scenarios Realization; (3) Model Building and Analyses; and (4) Tradeoffs. Each of these phases makes one or more contributions to the understanding of the system, reducing risks and modifying the design (Kazman et al. 2000).

Acsel-e® WAP Solution: A Wireless Information Systems

Consis International is currently developing Acsel-e®, its new product release for the integral management of the global insurance business. The company wants some of the services provided be Acsel-e® to be accessible from mobile devices (cellular telephone, PDAs, etc.), so this mobile solution must have a high degree of generality. This is why the solution proposed must consider that the insurance system providing the services would not necessarily be Acsel-e®.

Therefore a WAP protocol based solution is proposed. It will be called the **Acsel-e® WAP Solution (AWS)** and enable some of the services required in the operations of any insurance company to be supported.

So as to guarantee the degree of generality required, it is necessary to define a generic insurance API that would serve as the reference framework for integrating AWS other than Acsel-e®.

Some of the functional requirements of AWS are:
- Allow a customer of an insurance company to:
  - Notify a claim.
  - Consult a claim.
  - Consult the status of the policies held.
  - Quote product policies that the insurance company wants to offer.
- It must notify the status of policies.
- It must handle a unique ID for reported incidents.
- The WAP content generated must be dynamic.

It must also fulfill the following non-functional requirements:
- WAP contents must be concise and small (in view of the current restrictions of WAP devices).
- It must use release 1.2 of WML.

Moreover, in order to be able to define the API, the following aspects had to be considered:
- Identification of basic operations in the world of insurance.
- Identification and specification of the services offered by the API.
- Logical separation and organization of the services.
- Guarantee the generality aspect required.

The following section describes the solution proposed.

WAP Solution Proposed

Figure 2 shows the WAP solution proposed for the company, based on the basic architecture (see Figure 1) and the requirements mentioned above for AWS.

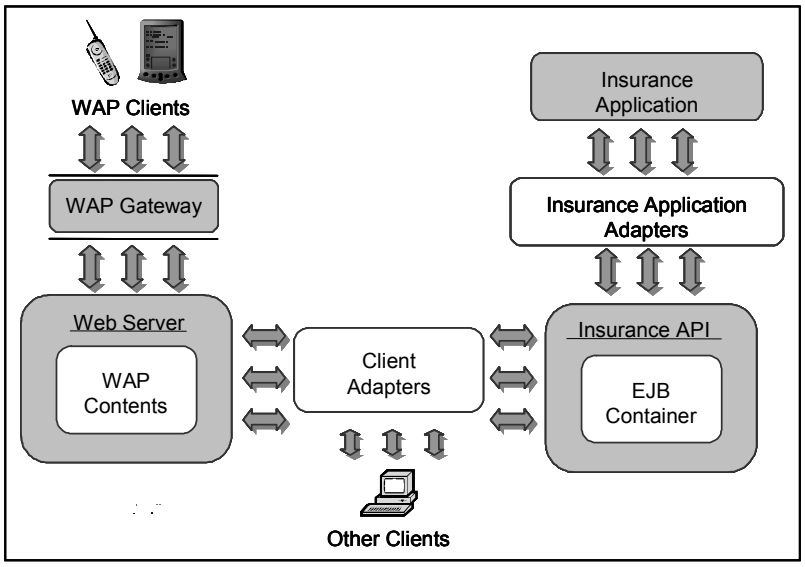

Figure 2. Proposed Architecture Based on WAP for AWS

In Figure 2:

- **WAP Clients:** Are those capable of communicating with a Web server that serves WAP contents.
- **WAP Gateway:** Its purpose is to convert Hyper Text Markup Language (HTML) content into WML so WAP clients can request it. It also compresses the content of the response from the Web server to the WAP client from pure WML to compressed WML
- **Web Server/WAP Contents:** This is the Web and WAP content server source. It can generate HTML contents or directly into WML.
- **Client Adapters:** These are adapters used by clients who wish to communicate with the insurance API.
- **Other Clients:** All clients who have a non-Web interface, such as JAVA, standalone or Common Object Request Broker Architecture (CORBA) clients. These clients can reside on conventional computers, but not on WAP devices.
- **Insurance API/EJB Container:** The real implementation of components and services for the insurance API.
- **Insurance Application Adapters:** Enable the generic API insurance application to be adapted. Allow the proposed solution to be integrated with applications other than Acsel-e®.
- **Insurance Application:** Refers to the insurance application that has all the business logic and functionality.

It should be noted that not all the components present in this solution are mandatory. Nevertheless, this design is justified for the following reasons: (1) there are different kinds of WAP clients, with different needs and different design criteria; (2) there are different types of clients for the insurance API, with different needs, depending on their design paradigm: standalone, Web or CORBA; and (3) flexibility is a very important issue for the WAP solution. It needs to be adapted to different situations and specific needs.

So the optional components of the solution are:

- **WAP Gateway:** If WML is generated, or a compressed format WML, straight from the source server.
- **Client Adapters:** If it is assumed that the potential insurance API clients wish to interact with it directly.
- **Insurance Application Adapters:** If the insurance application is supposed to interact directly with the insurance API.

From a futuristic point of view, AWS could grow as a result of: (1) new clients' incorporation (attention to bigger quantity of users); (2) new adapters' incorporation (fomenting this way the system scalability when allowing the use of new types of wireless devices), and (3) integration with other applications different to Acsel-e® (fomented the portability).

Since the architecture proposed has optional elements, it had to be evaluated so that the ideal configuration could be selected. This is why the ATAM architectural evaluation method was chosen. As mentioned above, this method covers four phases, but because of space limitations, we shall only describe the most significant deliverables for the purposes of this article. These are: the Architectural Proposals, the Analysis of the Quality Attributes and the Tradeoffs.

Most Important ATAM Deliveries

Architecture Proposals

The AWS solution allows for at least five variants or architecture proposals. Each will be described below:

1. All the components listed in the architecture make up the solution.
2. All the components listed in the architecture make up the solution, except that no WAP gateway is used.
3. All the components listed in the architecture make up the solution, except that no client adapters are used. Inside access to API services from the WAP client is direct.
4. All the components listed in the architecture make up the solution, except that no adapters are required at the level of the insurance application.
5. All the components listed in the architecture make up the solution, except that no adapters are required at the level of the insurance application or at the client level.

Analysis of the Quality Attributes

For Consis International the most important quality attributes to be considered for AWS were *Modifiability* and *Portability*. *Modifiability* (sometimes called *Maintainability*) is the ability to make changes quickly and cost effectively follows directly from the architecture (Bass et al. 1998). Portability is the ability of the system to run under different computing environments (Bass et al. 1998). These attributes are directly related to the objective of AWS, i.e. it must be generic and adaptable, able to work with any insurance system and able to interact with any WAP client (or any other client).

Following the brainstorming technique with the AWS stakeholders, the quality attributes were broken down into a Utility Tree (Kazman et al. 2000). This is necessary in order to analyze the architecture proposed. The Utility Tree is a hierarchical process. Its first level represents the quality attributes in the form of nodes; at the second level there are different scenarios that can be seen in each attribute; and in the third level the scenarios are instantiated. Figure 3 shows the Utility Tree for AWS.

With the Utility Tree, the possible scenarios were analyzed. The analysis is described in the next section.

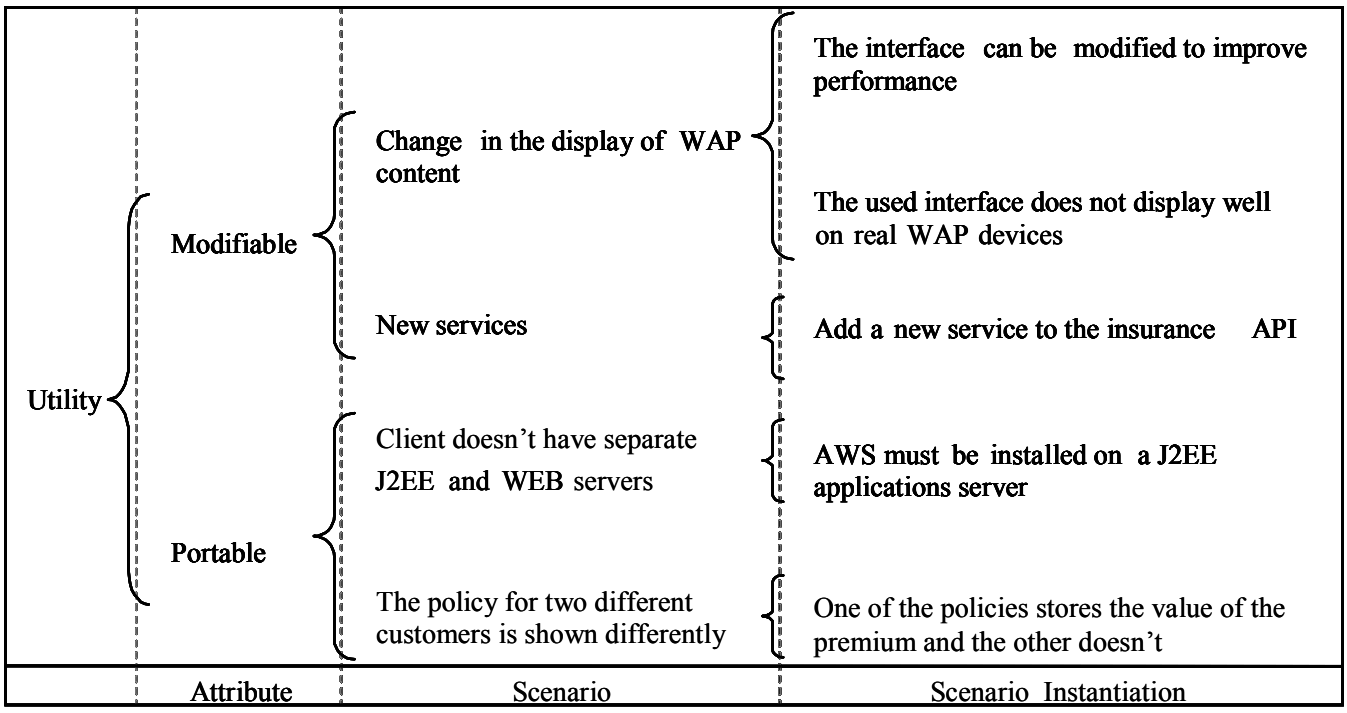

Figure 3. Utility Tree

Analysis of the Architecture Proposals

Tables 1 and 2 summarize the analysis of how the architecture proposals solve the scenarios shown in the Utility Tree.

Attribute: Modifiable

Table 1 . Analysis of the Architecture Proposals in Relation to the *Change in Display of WAP* to the *New Service Contents* Scenario

Proposal	Scenarios	
	Change in display of WAP contents	*New services*
1	If the WAP content interface is to be changed, the changes would be limited to modifying the display of the WAP contents, which, in the reference implementation, are JSP pages. This is reduced to adding, removing or modifying simple JSP pages, without affecting the other architecture components. This kind of change in AWS tends to be local and low risk.	For this case, the main component affected will be the one the new services should be incorporated into, i.e. if the service corresponds to an operation on policies, the component affected will be the API insurance policy server. The client adapters may also have to modified to make the new service accessible.
2	Lack of the WAP gateway only matters in the case of changing interface, for reasons of performance, since the gateway would assist in compressing the WAP contents, thereby improving performance. However, the modification continues to be localized and low risk.	Same reasoning as Proposal 1.
3	The clients' adapters are used to give access to the application to clients non WAP and an architecture that it doesn't count on them it doesn't affect this scenario.	Same reasoning as Proposal 1, only that when not having clients' adapters, the modifications should be made directly on the client, that which is negative for this scenario.
4	The lack of adapters for insurance applications doesn't affect the interface change directly, since these adapters allow the integration with other applications different to Acsel-e.	Just as in the analysis of the proposal 1, the main affected component will be that to the one that is incorporated the new service.
5	Just as it was mentioned in the analysis of the proposals 3 and 4, the lack of these adapters doesn't affect this scenario.	In the same way that in the proposal 3, the lack of clients' adapters makes that the scenario is affected negatively.

From Table 1, we can see that the *Change in display of WAP contents* scenario is not affected by any of the proposals. This makes the application highly *Modifiable* for this scenario. Proposals 3 and 5 are less suitable than the other architecture proposals for solving the *New services* scenario. It can be concluded from the previous analysis presented in Table 1 that the application's *Modifiability* is negatively affected only with proposals 3 and 5.

Attribute: Portable

Table 2. Analysis of the Architecture Proposals in Relation to the *Client Does Not Have Separate J2EE and Web Servers* and *Two Different Customers' Policies Displayed in Different Ways* Scenario

Proposal	Scenarios	
	The client doesn't have separate J2EE y Web servers.	Two different customers' policies are displayed in different ways (variability of the policy attributes).
1	In this case the J2EE applications server offers everything necessary to install both AWS Web components and the Enterprise JavaBeans (EJB) components that display the insurance API. The difference lies in the centralized management of all these architecture components. Dealing with this situation does not cause much problem for AWS, as it involves a change in the installation process and not in the solution logic, which is maintained.	As the insurance API considers all the business objects as configurable objects, this change is limited to considering the policy in each installation with the adequate properties.
2	As the gateway it is not in none of the servers, the proposal doesn't affect this scenario.	The nonexistence of the WAP gateway only affects the services availability to the clients WAP, but it doesn't affects the WIS portability and therefore it doesn't affect this scenario.
3	Although the proposal affects the existence of clients non WAP, it doesn't impact this scenario.	The portability is not affected with this proposal within this scenario since this it only limits the type of the client's device.
4	This proposal affects the attribute of quality negatively, but it doesn't affect the scenario.	Since the application adapters are independent of the logic of the system, this scenario is not affected by the lack of this component.
5	As this proposal it includes the proposal 4, the portability is affected.	Of the analysis of the proposals 3 and 4 is derived that the proposal 5 doesn't affect this scenario.

None of the proposals affects Portability in relation to *The client does not have separate J2EE and Web servers* scenario. As can be seen in Table 2, implementation of the insurance API solves this scenario. From the analysis presented in Tables 2, it can be concluded that the architecture of the AWS application is *Portable*, regardless of the architecture proposal adopted.

Architecture proposals 1, 2 and 4 respond in the same way in relation to the scenarios proposed. Based on this, a cost-benefit analysis was conducted (which is not presented in detail due to space reasons). The result showed proposal 2 to be the most economical, because in this proposal the WAP gateway is excluded, at the same time that it is possible to implement an executable prototype of the architecture that satisfies the requirements of quality (*Modifiability* and *Portability*). Its implementation is described in the following section.

Wap Solution Implemented

For the purpose of this article, the WAP solution implemented is presented with a description of the WAP based architecture implemented for AWS, the architecture's executable prototype and the final interface of AWS.

WAP Based Architecture Implemented for AWS

Lastly a solution capable of satisfying the needs of particular WAP clients was implemented. It can be seen in Figure 4, where:

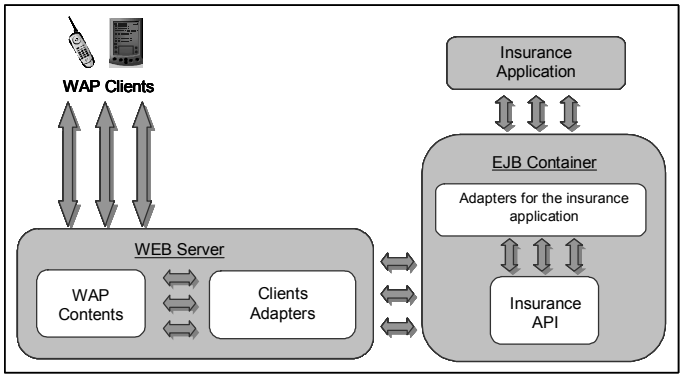

Figure 4. WAP Solution Implemented

- **WAP clients:** These are displayed in WML pages, on the WAP device or emulator, ready for use by the system users.
- **Web Server:** In this component, the WAP component is dynamically generated, in WML, using JSP pages.
- **Client adapters:** These adapters were implemented as JavaBeans from the JSP pages.
- **Insurance API:** Was implemented using EJB technology. In particular using session beans.
- **Adapters for the insurance application:** These adapters were implemented as an EJB session that encapsulates the functionality of Acsel-e®
- **Insurance application:** Consis International's Acsel-e® insurance application.

This architecture displays the following characteristics:

- Given the nature of the adapters, the client adapters move to reside on the Web server (where the WAP contents are); and the insurance application adapters become part of the EJB container, where the insurance API is implemented.
- For the client adapters, it is very convenient to use JavaBeans, due to the natural integration between this technology and the JSP pages used to display the contents.

Executable Prototype of the Architecture

Having defined the architecture for implementing WAP based AWS, an executable prototype of the architecture was defined with the following preliminary functionalities: (1) basic navigation between JPS pages that represent the WML contents; (2) JavaBeans as client adapters; and (3) EJB components that represent an implementation of the insurance API, providing the simplest services. Figure 5 shows some pages of AWS. This prototype evolved until it became the final release of the functional prototype required to obtain the WAP solution proposed which is described below.

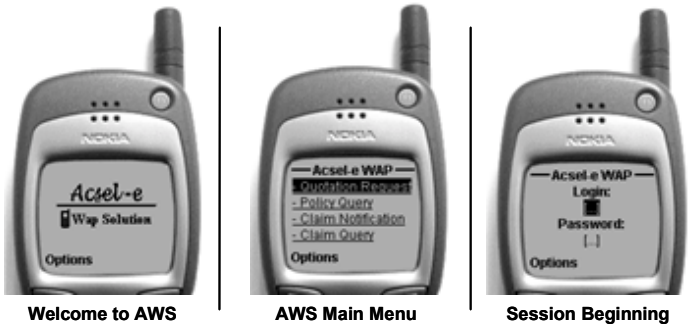

Figure 5. Some Pages of AWS

Description of the Final Interface of AWS

We shall begin by describing the criteria under which the interface of the WAP contents of the solution was designed. It is necessary to stress the importance of the interface design for WAP contents to be displayed on mobile devices. Due to the current limitations of WAP devices, both in terms of screen and keyboard, and of some input devices, as well as the small bandwidth of wireless networks, it is essential for the interface to meet the following requisites: (1) it must be concise, expressing what has to be said in the least number of words; (2) it must have as few graphics as possible; (3) it must take into account there must not be more than two options for the user to choose from (most WAP devices at present have no more than two buttons for selecting options); (4) the contents should be as short as possible, not exceeding 512 bytes, due to the memory limitations of current WAP devices; and (5) the number of steps or screens needed to execute an operation must be kept to a minimum, otherwise the user will get soon tire and stop using the service (Openware™ 2001a; Openware™ 2001b). Figure 6 shows the final navigation map for the solution.

Lessons to be Learned

1. The m-commerce forces to build very complex systems. The obtaining of attributes of quality as *Modifiability* and *Portability*, demand architectural evaluation in the design stage of this type of systems. If the system is already implemented it can be extremely expensive or impossible to obtain this type of attributes. Of there the importance of evaluating the architecture in the design (early stages of the life cycle) to verify if the architecture favorable the required attributes of quality

2. The identification of the attributes of quality should be as a reminder to decide the definitive architectural design.

3. Several architectures can implant the same solution; however, they should be evaluated to select the one that more promotes the wanted attributes of quality.

4. To include in the stage of architectural design the evaluation of the architecture, with base to the attributes of quality, it represents a significant improvement in the development process of this type of systems.

5. ATAM was appropriate for this evaluation kind.

6. The Utility Tree technique facilitates that the stakeholders makes decisions about the most important attributes of quality for a system.

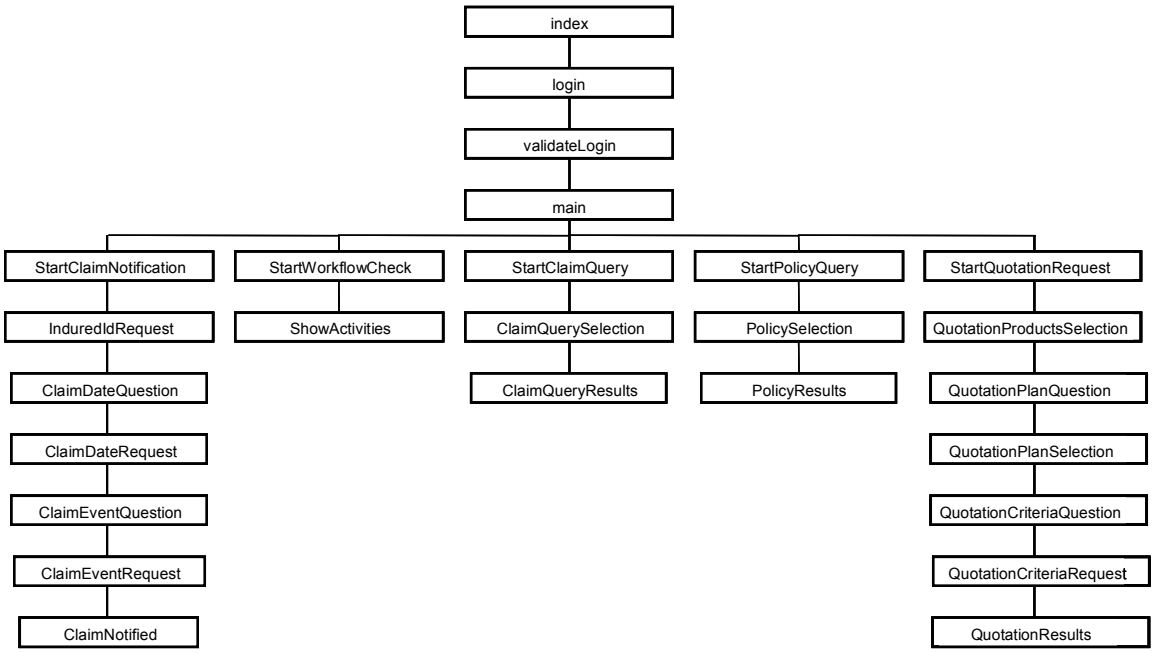

Figure 6. WAP Content Navigation Map

Conclusions

- Development of WIS with the WAP protocol enables training and development time to be reduced.
- Architecture proposals were analyzed using ATAM for a WIS system that uses the WAP protocol.
- Construction of the Utility Tree proposed by ATAM enabled the most important quality attributes for the application, according to the stakeholders, to be identified, prioritized and refined.
- Evaluation of the architecture in relation to the scenarios described in the Utility Tree enabled the most suitable architecture to be selected, saving cost and showing that the architectural design proposed strengthens the *Modifiability* and *Portability* quality attributes for the type of system described here.

Acknowledgements

The authors wish to thank to E. Ascanio for his contribution to this research.

References

Bass, L., Clements, P. and Kazman, R. *Software Architecture in Practice*. Addison Wesley, Reading, MA, 1998, pp. 76-85.
Bosch, J. *Design and Use of Software Architectures*. Addison Wesley, Reading, MA 2000, pp. 85-93.
Flaherty, N. "Wireless Application Protocol Technical Overview", in *Wireless Developer '99*, Wireless Application Protocol Forum Ltd. , Monterrey, CA, May 1999, p. 4, http://www.1.wapforum.org/what/docs/WD99Technical990506.ppt.

Goldman, S. "Wap Today and Tomorrow", in *Internet World UK*, Wireless Application Protocol Forum Ltd., May 2000, p. 2, http://www.wapforum.org/new/IWUK-WAP-Goldman.ppt.

Kazman, R., Klein, M. and Clements, P. *ATAM: Method for Architecture Evaluation*, CMU/SEI-2000-TR-004, Carnegie Mellon University, Pittsburg, PA, August 2000, pp. 1-84, http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00tr004.pdf

Kazman, R., Klein, M. and Clements, P. *Evaluating Software Architectures: Methods and Case Studies*. Addison-Wesley Pub Co; 2002, pp. 19-42.

Openware Systems Inc. "Graphical Browser Application Style Guide" in *Openware ™web site*, Redwood City, CA, August 2001, pp. 3-86. http://developer.openware.com/doc/50/stule_guide.pdf

Openware Systems Inc. "Usability Interface" *in Openware™ web site*, Redwood City, CA, November 2001, pp. 3-12, http://developer.openware.com/technotes/WP_OUI_112001.pdf

SAS Institute Inc. "webAF's WML TransformationBeans for WAP/WML Applications", in *SAS Institute web site*, Cary, NC, January 2002, p. 1, http://www.sas.com/rnd/appdev/webAF/server/wmloverview.htm.

Turban, E., Lee, J., King, D., and Chung, M. *Electronic Commerce: a Managerial Perspective*. Prentice-Hall, Englewood Cliffs, NJ, 2000, pp. 14-16.