Reference:
Alexis Ocampo, Fabio Bella, Jürgen Münch. Developing Services for the Wireless Internet, chapter Software Development Processes, pages 9-32. Springer, 2006.
DOI: 10.1007/978-1-84628-589-9_2
URL: http://link.springer.com/chapter/10.1007%2F978-1-84628-589-9_2

Developing Services for the Wireless Internet:

Software Development Processes

Alexis Ocampo, Fabio Bella, Jürgen Münch
Fraunhofer IESE
{Alexis.Ocampo, Fabio.Bella, Juergen.Muench}@fraunhofer.iese.de

The complexity and novelty of the technology that can be used for developing wireless Internet services (i.e., mobile terminals, mobile networks, mobile interaction, variability of terminals, unstable business/billing models, complete testing environment hardly available, change of technology) and the extreme time-to-market pressure result in insufficient knowledge about development procedures and technical constraints, and therefore insufficient guidance for project managers and software developers on selecting appropriate development processes, techniques, methods, and tools. The end result is poor quality of products, unmotivated developers and managers, and unhappy users.

At the moment, there is very little experience in developing software for such services systematically. Therefore, designing processes for this domain implicates several difficulties:
1) Whereas several standards exist for conventional software development (e.g., IEEE 1074-1997, ISO 12207, CMM, CMMI), no such standards are available for wireless Internet services. 2) The wireless Internet services domain lacks specific experience on particular technologies, their applicability and constraints. 3) The variations of the applications and, as a consequence, possible variations of the development technologies, are not sufficiently understood.

This chapter describes an initial reference process by summarizing guidelines and several hints to take into account for the development of wireless Internet services on the levels of engineering processes and life cycle processes. The reference process is based on experience from the WISE pilot projects (see Chapter 6) and a comprehensive literature survey.

2.1 The Reference Process Model

The lack of knowledge about wireless technologies, the unavoidable growth of this type of applications in the coming years, and the need for a systematic approach for developing these applications are important reasons to justify the creation of such a reference process. This process has been descriptively elicited in a systematic way through the development of pilot projects and literature study [49].

We use the term software process reference model for a process model that integrates consistent and validated empirical and theoretical evidence of processes, products, roles, and tools used for developing software in a domain (in this case, Wireless Internet Services).
Creating a reference process solely from observing software projects limits the level of detail and precision to the abilities of the respective project organizations in which the projects were performed. Therefore, we enhanced the developed reference process carefully with knowledge from other sources (e.g., literature, experience reports from similar projects).

2.1.1 Notation

The following table presents the original entities used by the SPEARMINT® notation [48] and two new symbols needed to understand the reference process model descriptions (see Table 1). The SPEARMINT® notation has the following entities: artifacts, activities, roles, tools. The last two rows in the Table present the symbols that are used for grouping those sets of activities that are considered optional or alternative in the reference process model.

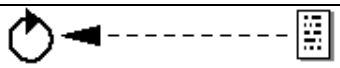
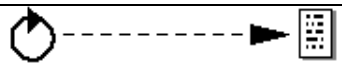
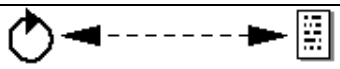
Table 1. Entities and Icons

Entity	Icon
Artifact	
Activity	
Roles	
Tools	
Alternative Box	
Optional Box	

The flow between the artifacts can be depicted by product flow graphs. The product flow graphs contain activities, artifacts, and the relationships between them (see Table 2).

Table 2. Relationships and Icons

Relationship	Icon

The activity consumes the artifact	
The activity produces the artifact	
The activity modifies the artifact	

2.1.2 Reference Process Model Overview

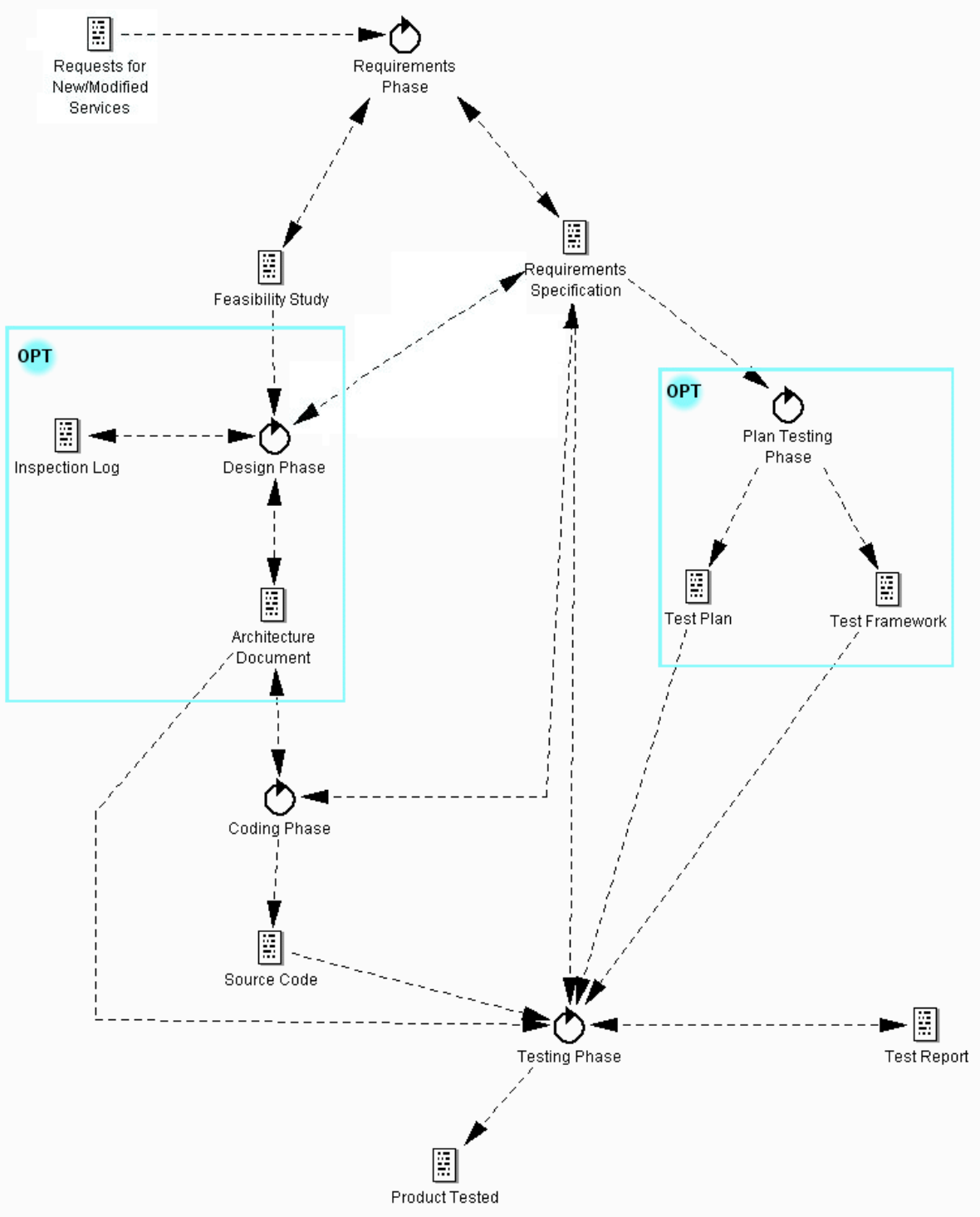


Figure 1. Reference Process Model

The reference process model presented in this chapter is the result of overlapping the original process descriptions elicited from the real experience of the WISE pilots (see Figure 1). It seems like a waterfall model, but in practice it was followed by the pilots like an incremental model. Please note, that in this figure, activities are grouped in a phase. This is because this figure reflects the highest level of abstraction of the model. In the following subsections, the activities for each phase will be detailed.

In order to understand the meaning of the optional boxes, it is important to take a brief look briefly at the process models followed by each pilot (for more details, please see chapter 6).
The life cycle model followed by pilot 1 was an iterative process model consisting of three phases: a requirements phase, a coding phase, and a testing phase. One important characteristic of this development process was the absence of an explicit design phase. This was a consequence of reusing the same client server architecture previously used to provide the service on the traditional Internet. The pilot was developed through three iterations. In all of them, design as such was explicitly performed, thus the application prototype and its high level design were documented after development.

In pilot 2, two different teams/organizations were responsible for the development of the client on the mobile device and the multimedia layer on the server side. The organization responsible for the client side followed an iterative life cycle model consisting of four phases: requirements phase, design phase, coding phase, and testing phase. The organization on the server side followed a similar process.

The reference process model was reviewed and approved by software developers from the involved companies in charge of the pilots. The final agreement among developers was to consider the design phase optional. However, pilot 1 developers compromised to establish this phase in the following projects. Optional in this case means, an activity that can be skipped by software organizations that do not feel not mature enough to enact them, while keeping in mind that it should be enacted as soon as possible. The same applied for the plan testing phase.

The following section will provide more detailed guidelines on each of the phases that can be observed in Figure 1, and some hints that can be helpful for developers and project managers in order to overcome the challenges imposed by the special characteristics of the Wireless Internet Services domain.

2.1.3 Requirements

2.1.3.1 Guidelines Based on Experience

The purpose of the requirements phase is to produce or obtain clear and unambiguous requirements. The following are the activities to be followed: Select requirements, perform a feasibility study, and specify the requirements.

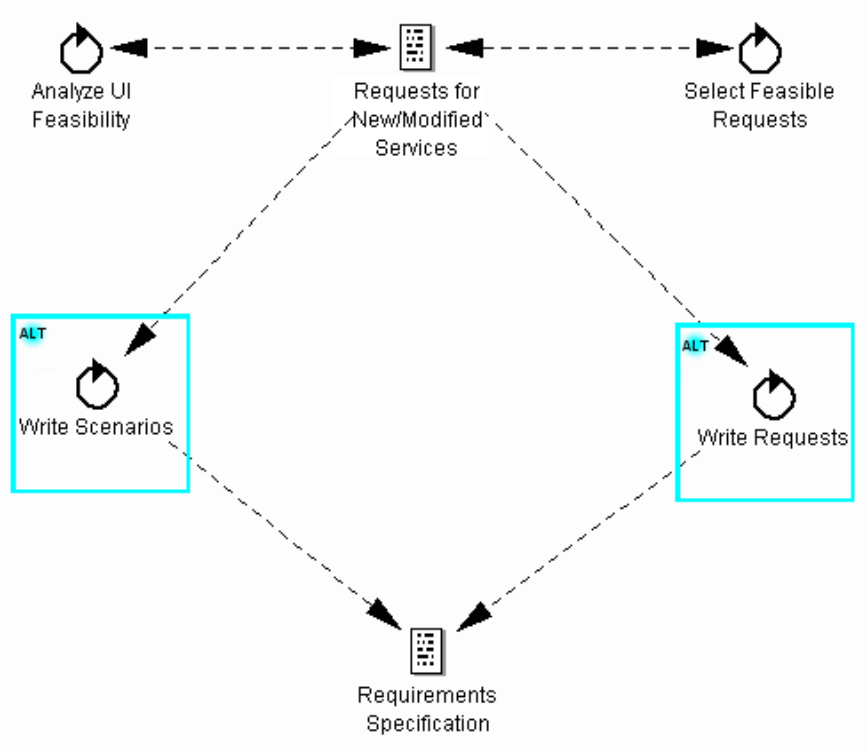


Figure 2. Select requirements

In the observed pilots of the WISE project, the requirements were written down in special meetings with the customer. The marketing group informally communicated usability requirements to developers. Informal meetings were held between developers and marketing personnel, and based on existing knowledge, discussions about best probable usability experience were held. Open questions usually triggered a feasibility study. In the feasibility study, prototypes were developed with the purpose of testing the actual technology (i.e., devices and networks), and understanding its real capability. The interaction between customer and developer as well as the feasibility studies were considered to be of great value by pilot developers, and therefore have been included as pearls in chapter 6 of this book. They are: "Customer requirement gathering", and "Feasibility study".

Figure 2 shows two alternative boxes. The meaning of this is that although both projects produced a document with the application's requirements, the modus operandi was different. Pilot 2 documented at a high level the functions of the service by using UML use cases. Use cases refined the client and server functional requirements. Additionally, performance and usability requirements were specified as part of the nonfunctional requirements. Pilot 1 did not use a special format for documenting the services' functionality. Both options were considered valid and therefore included in the Reference Process Model by developers (write scenarios (UML), write requests). A template and an example were agreed on and produced as part of the Reference Process. They can be found in the appendix of this book.

2.1.3.2 Hints from the Literature

The following methodologies were found in related fields like the Internet or wireless software development, and are included here as additional help for software developers and managers on where to look or learn from. The same applies to the rest of phases of the Reference Process Model.

Index cards are proposed by the usage-centered design approach [16] as a mechanism to specify requirements as part of an agile usage-centered engineering approach for Web applications. Here customers, managers, and developers collect the requirements on cards during a brainstorming session. They sketch the application's purpose from a business point of view, and express their wishes regarding functionality, features, content, and capabilities. The cards are sorted and clustered. The clusters are taken as the basis for specifying user requirements functionality.

Extreme programming [15] proposes user stories as a means to capture functional requirements in a simple, non-formal language. The developer writes them with the collaboration of the customer. The user stories are written on index cards and describe the tasks of the system. These stories are the basis for planning iterations, tracking progress, specifying and testing the functionality. User stories seem to be suitable for requirements that appear late in the development of the application.

The index cards and user stories involve customers under the assumption that they are participative and proactive and that they actually represent the user's needs. This might not be true in all of the cases. A review of the requirements by experts could compensate for this problem.

In order to address usability [17] presents the user-centered approach, where the business group of a software organization is in charge of studying and defining the profiles of its possible users. The profiles are used to determine possible tasks and goals of the users, specifying the functional requirements and creating a prototype for user analysis. The prototype consists of user interfaces that will be discussed with the users and then implemented according to the feedback. In a heterogeneous market like wireless Internet services, this approach could be of great help, because it forces development organizations to consider a wider spectrum of possible profiles.

Combinations of the previously mentioned approaches could be applied depending on the context of the software development organization. For instance, a large software development organization has more resources for looking at the requirements (e.g., the market division) than a small organization.

Regarding device independence, a good starting point for clarifying the concepts is given by [2]. It is a survey that presents a classification of available technologies and their relationship with device independence in the context of wireless Internet services. For example, device attributes like output, input, processor, memory, multimedia objects, application language, or browser language influence the degree of independence of an application. Devices receive content as multimedia objects, application languages or browser languages. Depending on the underlying hardware, devices are able to use different types of content. Therefore, in order to achieve device independence, the content must be sent in a format compatible with a given device.

There are technologies that can be used to adapt the content or application according to the device capabilities. Content adaptation can be done in the server, proxy or client browser. Some examples of these technologies are: HTTP request header files, CC/PP

composite capability preference profile, WAP UAPROF, SyncML, and Universal plug and play.
This survey is a good reference for understanding how each of the above mentioned technologies can help when trying to deliver a device-independent application. The bad news is that at the moment, there is no dominant/unique standard, therefore, choosing a specific technology can imply high risks.
In order to mitigate this risk, the W3C consortium is working on an initiative focusing on device independence and standardization. The idea is that web content and applications are accessible anyhow and anytime. Anytime refers to many access mechanisms (i.e., heterogeneous clients that can provide access anytime), and anyhow refers to many modes of use (i.e., audio, voice, touch, among others). One product of this effort is the device independence principles document [18]. At the moment, the principles are general, but they will be specialized with guidelines and requirements to obtain device independence as well as to concentrate all standardization efforts in one place.

2.1.4 Design

2.1.4.1 Guidelines Based on Experience

The purpose of this phase is to produce high-level and low-level designs that meet the requirements specification. Figure 3 shows high-level and low-level designs. Chapter 4 presents more details on how they should be accomplished. In brief, each level of design can be represented by four viewpoints: structural, behavioral, deployment, and development. The structural viewpoint covers the composition of information and architectural components, whereas the behavioral viewpoint considers the dynamic aspects of the architecture. The deployment viewpoint shows the allocation of architectural components to physical nodes of computing and network environments.

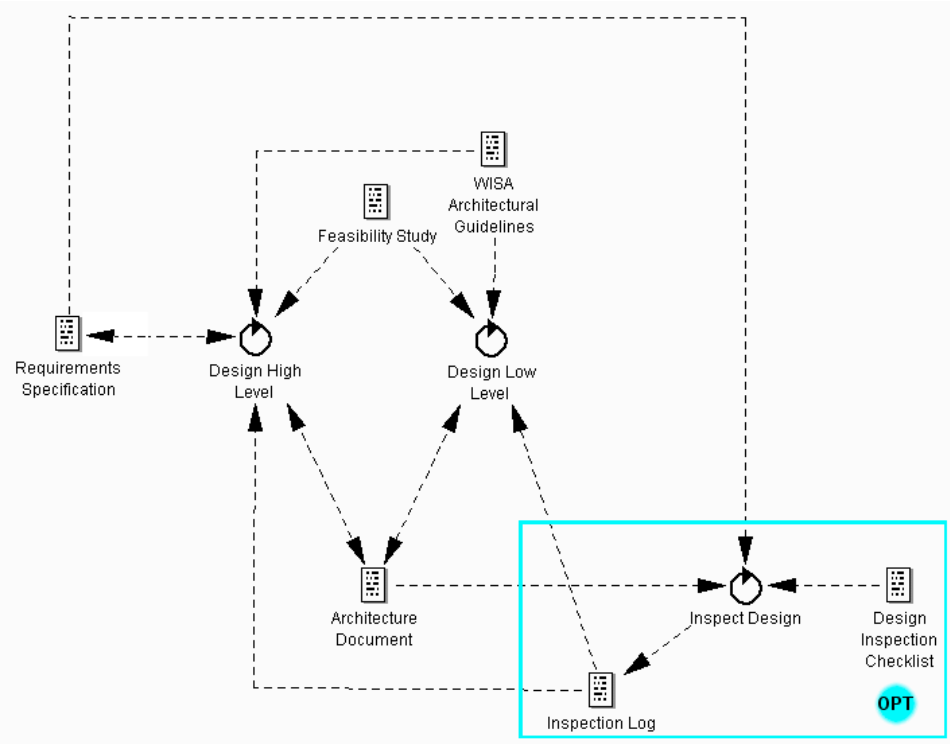


Figure 3. Design phase

The development viewpoint maps the organization and the choice of technologies to services and components.

The design should be inspected based on a checklist and an established procedure. This activity is marked as optional, because it could be skipped once the expertise and knowledge of the domain are mature enough.

Pilot developers recommend designing a user interface that avoids graphical controls that can only be used by a mouse device or touch screen. In other words: when designing the application, keep in mind that it also has to be suitable for the most limited devices. Developers considered this practice of great importance in order to avoid the cost of producing the same application several times for different devices. More references on designing for device independence (this is how this domain is called), are provided in the next section. Additionally, a pearl on “device independence” which summarizes it, can be found in chapter 6 of this book.

Regarding technical features like scalability, quality of service (QoS), and billing models, it is recommended to create components or agents responsible for such issues. One example is the service management component created in the WISE project, which provides services for authentication, authorization, user profile management, provisioning, and billing. Another example is an agent that negotiates the preferences of a client with the limitations of the server to provide the service. Network bandwidth limitations can be addressed by trying to optimize the data exchange rate, and to reduce the amount of data sent over the network to the minimum. More examples of possible components can be found in the taxonomy of services provided in chapter 4.

2.1.4.2 Hints from the Literature

A survey performed at 25 organizations in the UK [6] revealed that formalized design web techniques like hierarchy charts, site flow charts, and storyboards were used in the web domain. Hierarchy charts were used to relate web pages of one site. Site flow charts sketched the decisions to achieve a certain functionality, and storyboards contain the sequence of web pages that a user will encounter within a web site. These techniques were used basically to design the navigation of the structure.

Some of the studied companies had developed website layout standards for using video, animation, graphics, colors, and navigational standards such as where to place the back button, and the use of banners and menus. Standards for designing website content were found, such as the use of specific keywords.

Examples of the use of structured techniques are given by [4], [1], [19], and [20]. Their common feature is the use of object-oriented principles to design static and dynamic views of a wireless Internet service application. Patterns like the MODEL-VIEW-CONTROLLER are recommended for use in wireless Internet service applications by [1], where the logic is concentrated on the server and none or a minimum of the business logic is revealed on the client side. The use of this pattern can have additional benefits such as: All the components are defined logically; each component has a function; interfaces are defined between components; each component can be implemented as another pattern; high reusability, high flexibility, reduced cost, and higher quality. For more details on this pattern, please see chapter 4.

Regarding device independence, [21] provides the Device Independence Web Application Framework (DIWAF). The framework is based on the “single authoring” principle, which consists of designing for the most capable device and automatically adapting content to different device classes. Content, layout, and style are separated for reuse whenever possible.

ScalableWeb is a technique presented by [22] that allows authors to build a device-independent presentation model at design time. ScalableWeb is also based on the single authoring technique, where authors can produce the layout specification for the largest screen size of a given device, and then a rendering system renders the device presentation model into device-specific presentations [27].

Mori et al. [23] present an XML-based approach oriented to design applications that are device independent. One commonality between ScalableWeb and the XML-Based approach is the specification of a task model as input for creating the device presentation model or the abstract user interface, which are later transformed into the device specifics with the help of an automatic/semiautomatic tool.

A high-level description of content, style, and interaction that allows adaptation is important in order to achieve device independence.

At the moment, topics of research in the device independence area include, for example, how to balance independence with the usability of the application. One application may appear as expected in the devices, but the usability experience might not be as satisfying for all of them. What are the steps to create abstract, general presentation models? Should generation of the specific presentation models be totally automatic, or just partially, so that authors can manipulate it?

Regarding scalability, [24] introduces a scalability design process based on a set of useful strategies when designing scalable Internet sites. The strategies are based on the design

principles of a scalable architecture: divide and conquer, asynchrony, encapsulation, concurrency, and parsimony. A set of guidelines for system partitioning, i.e., dividing the system into components with a well-defined interface and functionality, is provided. The message is clear. Successful wireless Internet services need to be scalable, but scalability demands a sophisticated architecture, a sophisticated design, and, once implemented, requires monitoring and maintenance.

In order to build seamless mobile services, Friday et al. [20] propose techniques that can be used to adapt the system and improve the quality of service of the network (QoS) at different levels (i.e., user, application, middleware, and transport). For example, the system can allow the user to change from synchronous to asynchronous tasks (user level), or, through proxy services, the application can use local substitute services based on cache information (application level). At the middleware level, the information can be fetched only when needed (on demand), and finally at the transport level, data can be prioritized, reordered, and exchanged according to the bandwidth situation. The adaptation techniques were validated through the development of a mobile collaborative system. One of the final conclusions of the study was that mobile systems must have the support of adaptation techniques at all levels in order to be effective, but architecture to propagate QoS information through the system is still required.

A discussion of billing infrastructure and charging models for the actual and future Internet, and how they could be modified for being used in wireless Internet services, is presented in [25]. One interesting example is the charging model proposed by [26]. This model supposes that the subscriber defines a class as an association between cost and network traffic. First class users have more privileges (e.g., low network traffic), but must pay more. The subscriber could define that he will use the network in first class or second class according to his desired association network traffic-cost. This model introduces complexity to the network behavior, overhead to the subscriber, and what is most important for developers, changes to the software application and extensions to the communication protocols. Therefore, developers should ask themselves during the conception of the application's design how much the model of charging and billing impacts the system's architecture.

The usage-centered engineering [28] approach addresses user interface usability as an important factor in design. Designers must produce a role model, a task model, and an abstract model. The role model groups the common characteristics of user interaction with the system into roles. These characteristics are related to the purpose, duration, attitude toward the system, and information exchange between the user and the system. A task model is a set of task cases and their relationships. A task case lists the steps of the system to provide the desired functionality without assumptions about the user interface. Finally, the abstract model describes the user interface with interaction contexts. The abstract model does not contain details about the look and behavior of the user interface. Designers use these models to create a comprehensive user interface in which all of them are combined. This comprehensive user interface can be validated by the customer.

More specific guidelines for designing user interfaces can be found for devices or families of devices. The big mobile device producers or programming platform providers offer them, for example, the MIDP style guide offered by Sun Microsystems, Inc [29].

The previous approaches gave guidelines for producing usable sites, but how could that be measured? The card sorting technique is proposed by [30] for eliciting quality measures of web pages. It is a technique based on a personal construct theory, whose objective is to elicit and ensure the validity of a measure for a fuzzy attribute like *quality* in a new field such as the Internet. It provides a systematic way to elicit quality measures that the stakeholders consider important. In a new domain like the wireless Internet, this can be of great help, because it minimizes the suppositions about the stakeholder's usability preferences.

2.1.5 Coding

2.1.5.1 Guidelines Based on Experience

The purpose of this phase is to produce and integrate the code that implements the design document and meets the requirement specification. The basic activities to accomplish the purpose are: code, test the units of code, integrate the units of code, and release the code (optional). In this case the release activity was declared optional because it was skipped by one pilot that used a well established configuration management system in the organization.

The activities shown in Figure 4 do no differ much from the code activity of any other domain. However, developers found differences originating from the languages to be used for implementing the services.

Pilot 1 was developed based on the assumption that plenty of Internet services for providing financial, weather, or sport information is available to clients, i.e., services require little user interaction. These services could, in theory, be deployed as wireless devices using the Wireless Application Protocol (WAP). At the moment, the WAP 1.x and 2.0 standards are available [31]. WAP 1.x uses the Wireless Markup Language (WML) for document formatting. WML is a language similar to HTML, specially designed for small clients with small screens and low bandwidth. Complete architecture rework of the service in order to translate HTML into WML was not needed, however, maintenance became a heavy duty because every modification to the HTML version had to be made in the WML version also.

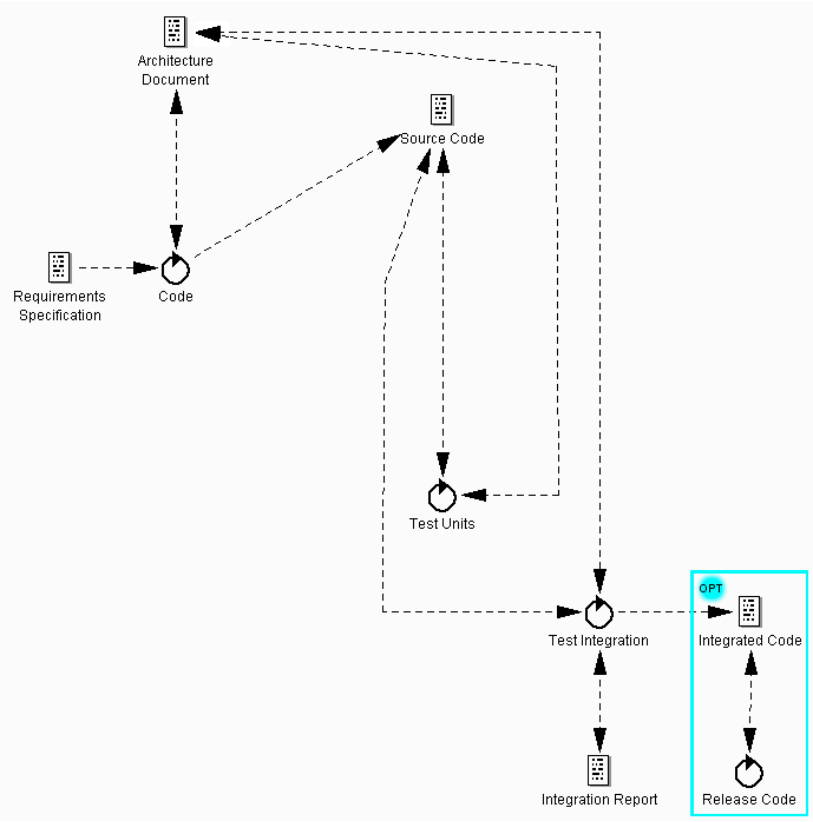


Figure 4. Coding phase

Wireless Internet services that demand interaction of their users, like games, need a more flexible programming platform. Today some wireless devices can be programmed using some sort of C-like language, but C is not a cross-platform language and therefore, portability among different hardware architectures is lost. A dedicated client should be deployed for every possible platform, slowing down time-to-market of the service and increasing costs. JAVA, on the other hand, is commonly used because of its portability. Especially after the release of J2ME, Java can now be deployed on many wireless devices providing a common ground for developers.
Pilot 2 developers used J2ME for implementing a game service. In this case, some devices showed performance problems when the graphics were stored on the client. A proposal for solving this problem was to store the graphics on the server and upload them to the device on demand. This implied the use of caching algorithms, which optimized the performance. Additionally, developers had to obfuscate the code in order to optimize the storage of byte code in memory. Although obfuscation is a technique for protecting the java code from reengineering [33], obfuscators can reduce the size of byte code by renaming classes, methods, and variables in a shorter string.

The developers who participated in the projects performed their unit tests using emulators for WAP and J2ME. One fact noted after unit testing was that the features of emulators are still far from the features of the real device. Developers had to program new code in order to replace functionality provided by the emulator but non-existent in the real

device. Thus, it is suggested to introduce the use of real devices early in the development process. Developers performed unit testing in an informal way. They did not document the test cases or the results. If a defect was found, there was no record of it, and the developer was expected to fix it immediately. The developers performed their unit tests using the "J2ME™ Wireless Toolkit 2.1". This is a client emulator that allows running midlets on top of it, and emulates the functionality of a mobile device. Specific components such as network modules were tested on cell phones, since they were critical for the rest of the application. The graphic library components were tested separately before including them in the application.

2.1.5.2 Hints from the Literature

Pair programming is a technique exploited by eXtreme Programming, where two developers produce code in one machine [15]. One person concentrates on the strategy to produce the code and the other one on whether the approach could work and how it could be simplified. He also has control of the computer. Pair programming can be seen as difficult to implement in an industrial context, but in a new domain such as the wireless Internet, it could perhaps have a good effect in order to resolve new and uncertain problems thanks to its person-to-person communication mechanism.

Zettel et al. [7] propose the LIPE process model to develop e-commerce services on time to market, based on other eXtreme Programming techniques like refactoring.

Although eXtreme Programming techniques seem to be flexible enough, there are some drawbacks. In the case of refactoring, small teams formed by great developers can be successful [14]. But what if the developers are not so experienced? It has been concluded by [10] that eXtremme Programming techniques are not intended for larger teams. Although success cases for larger teams have been presented [34], this assumption must be considered especially in the wireless Internet services domain, where projects can become large and complex.

In order to optimize the use of the device power, tips, guidelines and techniques can be found in the J2ME/WAP developer discussion groups. One example can be found in [32].

Security for WAP applications can be assured through the Wireless Transport Layer Security (WTLS). In the case of J2ME applications, power and memory constraints make security a challenge.

WAP 2.0 uses the extensible hypertext markup language (XHTML) for formatting the documents [31]. In theory, WAP2.0 allows developers to create richer applications that handle multimedia and animation, among other features. Almost all available browsers can display XHTML, but not all HTML features can be converted into XHTML.

cHTML is the content development language for i-mode (NTT Docomo's Wireless Service). cHTML is also similar to HTML, but is optimized for wireless networks and devices.

More details on the programming languages mentioned above and others are provided in chapter 3.

2.1.6 Test Planning

2.1.6.1 Guidelines Based on Experience

The purpose of this phase is to produce test cases for each of the requirements specified and set up the needed hardware and software in order to execute the test cases.

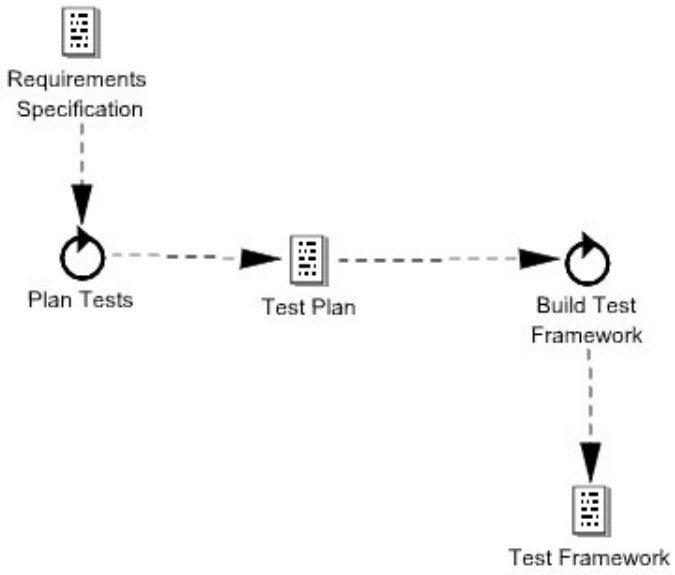


Figure 5. Plan Tests

In order to do this, the following activities should be performed: Plan the tests to check the structure, functionality, performance, content, and usability of the service, and set up the needed hardware and software in order to execute the planned tests (see Figure 5).
The assumption at the beginning of the projects - that setting up a real test environment for the application, i.e., the target device(s), server, and network, was a complex task - led to the inclusion of a special activity in the process to accomplish this objective. The fact is that the effort for setting up the testing environment of a wireless Internet service is considerable and should not be underestimated.
Finding mechanisms or tools for simulation or emulation can help to solve the problem and reduce costs for setting up a realistic testing environment. Technology providers provide emulators for many mobile devices on the market.
Emulators approximate the functionality of the target device. In practice, using network emulators can be of great help for creating tests, collecting the results, and repeating them under different network conditions.
In general, the following issues must be taken into account, especially before setting up the environment for testing a wireless Internet service.

- Look for a suitable simulator/emulator, or design test cases that reflect the behavior of the network.
- Make sure that the mobile device is available. If this is not the case, use the device emulators, but keep in mind that usually the functionality and API of such emulators may be slightly different from the functionality and API of the target devices.

The developers of the pilot projects performed their unit tests using emulators for WAP and J2ME. They had to program new code in order to replace functionality provided by the emulator but non-existent in the real device. Thus, it was suggested that future projects should try to introduce the use of real devices early in the development process. This activity was considered to be of great relevance by the developers of the pilots, especially because there are hidden risks (e.g., high costs, lack of suitable technology) that could make a project fail. Therefore, a list of basic steps to "build a test framework" has been produced and included in chapter 6.

2.1.6.2 Hints from the Literature

Markov models [44] are mathematical models that can be used for the validation of the mobile application's usability. This mathematical construct is based on finite state machines, and can be helpful in analyzing the usability applications implemented in push button devices such as mobile phones or PDAs. Finite state machines represent the whole system as a set of states and transitions. Finite state machines are particularly effective at modeling, as they are simple, quick to produce, and scalable to any size. Also, as well-defined mathematical objects, it is possible to perform complex reasoning on the model to produce reproducible, quantitative results.

A sophisticated technique to plan the tests of wireless Internet services is the Model-Based Testing Technique [41]. Just like the previously mentioned technique, this technique also uses finite state machines and directed graphs or state transition diagrams as a basis for creating models to test the functionality of the application. Benefits of model-based testing include the written structure of states and transitions and the possibility to automate tests, which gives a general understanding to all team members on how the application should work. It is still a topic of research to find out if model-based testing is a cost-appropriate technique for finding defects, due to the fact that the effort invested by developers on building the model is not negligible.

Central-Control Wireless Emulators abstract the entire mobile wireless network to a model with a set of parameters, thus emulating end-to-end applications and protocols. The emulator applies network conditions and traffic dynamics to each packet passing by to reproduce the network effects, thus testing the performance of the applications and protocols. This type of emulation is conducted by connecting mobile hosts such as handheld devices and computers to the central-control emulator.

Examples of such emulators are ONE [35] and Dummynet [36].

VINT/NS [37] is one of the commonly used simulation combined wireless emulators. The emulation facility in VINT/NS is able to capture and direct traffic into the simulator. Within the simulator, protocol modules, algorithms, and visualization tools can be incorporated in an automatic fashion. In addition, arbitrary mobility can be generated with the help of the simulator. Its advantage is that it offers a large amount of simulation resources in the central simulator, compared with the central-control approach that only

has a limited number of network parameters available. However, this approach lacks the support for the evaluation of real topology-related protocols.
The trace-based mobile network emulator [38] also emulates the characteristics such as performance and bandwidth in a real environment. However, the approach in this emulation is different. This approach consists of three distinct phases: data collection phase, trace distillation phase, and modulation phase. The first phase collects traces from a target mobile wireless network. Trace collection logs every outgoing and incoming packet, along with the time at which it was sent or received. Other protocol related information such as sequence number flags are also collected. The second phase constructs a wireless network model with the collected traces. This network model makes it possible to compute the parameters. The last phase reproduces the traced network effect in a wired network by using the model and computed parameters.
Another emulator named flying emulator [39] emulates the physical mobility of wireless devices by using the logical mobility of software-based emulators of the devices and target software. The emulator is implemented as a mobile agent; it dynamically carries the target software to each of the sub-networks to which its device is connected on behalf of the device, allowing the software to interact with other servers in the current sub-network. That is, it can test software designed to run on a wireless device in the same way as if the software were disconnected from the network, moved with the device, and reconnected to and operated on another network. Moreover, the framework allows emulators to easily simulate other characteristics of wireless networks by using a runtime byte code rewriting technique.
One distributed network emulator system, EMPOWER [40], provides a mechanism to emulate the mobility of a wireless network in a wire line network. EMPOWER allows the user to define packet latency and bandwidth as parameters and test a given topology wireless network.

2.1.7 Testing

2.1.7.1 Guidelines Based on Experience

The purpose of this phase is to execute the planned test cases and validate that the specified requirements are met by the service implementation. As observed in Figure 6, the mandatory activity is the system test, while the rest is optional. The reasons for declaring these other activities optional were based on the following:

- Even though the activity ‘test usability’ is very important, it was declared optional, since it demands effort and resources (as observed in the WISE project) that some organizations cannot afford.
- The ‘acceptance test’ was declared optional due to the fact that demands tight collaboration and great communication channels between the customer and the developer, which cannot always be realized.
- Finally, the activity ‘analyze defect’ was declared optional because in the context of the WISE project, the most mature partners did follow it, while the less mature ones did not. Instead, defect analysis and recording of the actions to solve the problem were performed in the test system activity.

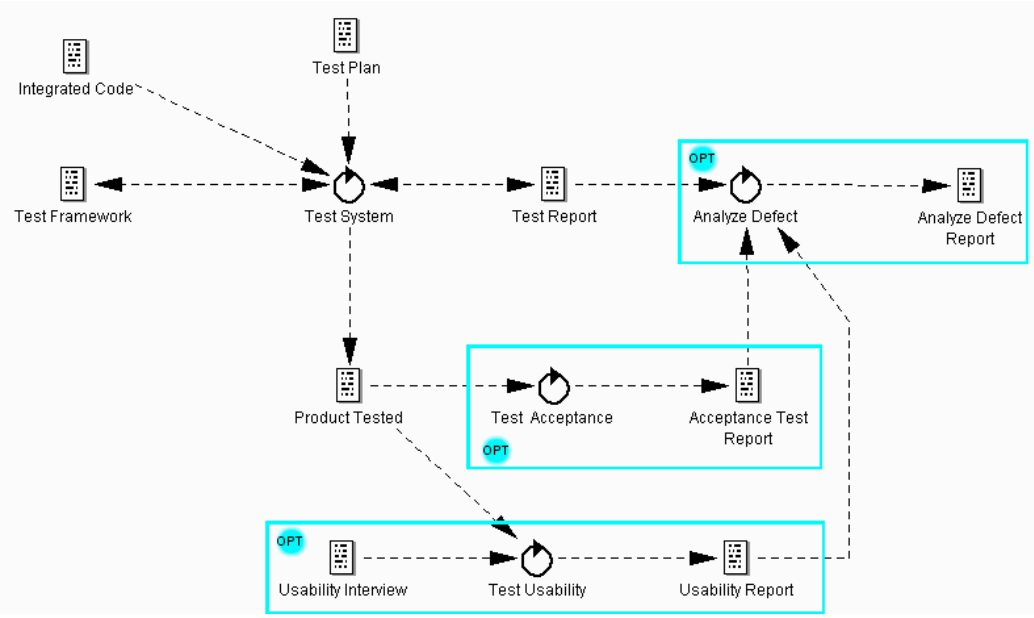


Figure 6. Testing phase

The complete functionality of the system (client and server) was tested using the real environment. Real devices and emulators were used during these tests. Tests on the real device were more time-consuming. One example was to test what happened when the application was interacting with other phones and a call came in, or when the battery was empty.

The activity 'test usability' is based on the paper by Nielsen [50] in which experts guided by a set of usability heuristics evaluate whether user-interface elements such as dialog boxes, menus, navigation structure, online help, and others conform to the user needs. The activity comprises the following sessions:
1. The briefing session: Experts are told what to do. A prepared script is useful as a guide and to ensure that each person receives the same briefing. The project manager, developers of the pilot, and the process engineer developed the heuristics.
2. The evaluation period: Each expert typically spends 1-2 hours independently inspecting the product using the heuristics as guidance. The experts need to take at least two passes through the interface. The first pass gives a feel of the flow interaction and the product scope, the second one allows the evaluator to focus on specific interface elements in the context of the whole product. While evaluating the interfaces, the evaluator must record the problems.
3. The debriefing session in which experts come together to discuss their findings, prioritize the problems they found, and make suggestions.
The main problem of the usability testing deployment was that it required many people involved in the test, who needed to have access to the application, or have it installed on their PCs. This was a time-consuming requirement in the case of desktops and real mobile devices. Therefore, developers performed a lighter version of usability testing once they saw problems with interaction or presentation while programming the

graphical user interface. A list of usability items was recorded and used to improve the user interface. Examples of how this information was recorded are:

USABILITY PROBLEM 1

ISSUE. The "Logged in the game community" screen does not show enough information. It is not clear which user is logged in and if a character has been loaded or not.
SOLUTION. Add such information below the bottom icons, in a space that is already used as status bar.

USABILITY PROBLEM 2

ISSUE. When starting a new game, a presentation of the situation could help. Otherwise the player feels lost and without aim.
SOLUTION. Add a dialog and content for it. A great part of the presentation dialog can be reused for this implementation.

USABILITY PROBLEM 3

ISSUE. When a user is hit, he/she doesn't have the impression of being hit. The display of server notifications is really poor.
SOLUTION. Add a damage image and sound, and also improve the event notifications. The Dungeon Master will notify the players of the game event or actions (i.e., "You have been damaged").

2.1.7.2 Hints from Literature

The usability of mobile devices as a medium to deploy WAP applications has been criticized because of their physical limitations by [46]. Their usability test and studies reveal that bigger and more capable user displays improve user interfaces, and therefore acceptance by users. An interesting study done by [47] was focused on detecting usability problems through testing, and afterwards improving the application. In this study, the users tested alternatives of a user interface with the same functionality. Users were gathered and performed the same activities within a wireless Internet service. Meanwhile the time and number of interactions were measured. Finally, users were interviewed about their experiences and suggestions. The measures and information were taken as a basis for deciding which the most suitable interface was and for creating guidelines to be used in future projects. Looking at the previous references can help software managers and developers to better understand the user's problems when interacting with mobile devices.
The web usability assessment model [43] includes eleven usability attributes, which have been identified as significant in assessing a customer's perceived usability. The usability attributes include design layout, navigation, personalization, design consistency, design standards, reliability, security, performance, information content, accessibility, and customer service. An automated usability-testing tool named Usability Enforcer tool based on the web usability assessment model implements a set of usability rules for a

targeted customer profile, the specified computing environment, and the strategic goals of the wireless application.
One visualization-based approach to improve the usability of a web site, and a predictive model to locate problem areas, is introduced by [42]. The approach underlines the importance of using visualization techniques to understand the behavior of users on a web site and to identify unreachable places. Visualization techniques can also be used to analyze the past behavior of a site, and to understand the impact of new changes.
The software that runs on mobile devices can also be validated with the amount of power consumed by the components of the device when the application is in use [45]. Applications interact not only with the display, but also with various other hardware devices: processor, memory, network interface, and possibly hard drive. All these devices consume energy during operation. This gives an opportunity to monitor and record power levels during the test suite execution.

2.1.8 Life Cycle Process Models

This section provides a brief discussion of life cycle process models that are commonly referenced in the software engineering literature and which may be suitable for developing wireless Internet services.
The throwaway prototype model and the incremental development model [3], which were found suitable for domains of similar characteristics like the Internet and mobile phones [4], [5], [6], [7], [8], [9] prescribe to initially provide essential operational functions, and then more capable versions of the system. Increments are usually defined as an agreement between the customer and the development organization. This allows development organizations to get feedback from the final customer during the development of the increments until the final version of the solution is delivered. Additionally, monitoring and controlling the project plan can be done more precisely, and the quality of increments can be assured with the established verification and validation activities.
Agile development practices and techniques aim at finding a balance between flexibility and structure in the actual business environment, where volatility and uncertainty increase [10], [11]. One of them, adaptive software development (ASD), proposes to face uncertainty with short delivery iterations; new requirements and technical information with intensive collaboration among managers, customers and developers; and process improvement with reviews after each iteration and project retrospectives.
The dynamic systems development method (DSDM) [10] contains three major phases: functional model iteration, design and build iteration, and implementation. All three of them are iterative. DSDM is similar to ASD because it allows to introduce new functionality (new requirements) late in the project. Additionally, prototypes built for each feature are preferred over long documents as documentation.
One risk of agile approaches is that they rely on the tacit knowledge and experience of developers [12]. This should be carefully considered, especially because developers themselves continue to learn due to the immaturity of the wireless Internet services domain. Additionally, issues like scalability and performance have to be carefully designed.
The spiral model [13] assumes risks as the driving force of software projects. This model proposes ongoing refinement of the system specification into source code components. Refinements are made through cycles, and each cycle is risk assessed. A risk assessment

determines if a project continues or is cancelled. The nature of the spiral model seems reasonable to apply in an uneasy domain like wireless, but the real cost of identifying, analyzing and maintaining risks can be high, which is not so suitable for small and medium-sized companies.
Boehm [14] proposes a combination of agile and plan-driven methods through the risk-driven spiral method, which is intended to balance flexibility and discipline. The rationale behind that is that although the market changes constantly and puts pressure on development organizations to deliver their products rapidly, increasing dependability of systems and applications demands high quality products.

2.2 Conclusions

The Reference Process Model presented in this chapter, although it does not differ significantly from traditional iterative process models on the life cycle level, provides a deeper insight into the wireless characteristics that must be evaluated by project managers and developers before and during the implementation of wireless Internet services. The representation of alternative and optional parts provides the basis for a better understanding of the possible options for developing wireless Internet services, taking into account the context of the project. For instance, it is now known that planning the tests of wireless Internet services can be a complicated task, influenced by the heterogeneity of devices and networks. Therefore, it is an optional task for small software development organizations, which may not have enough budget to invest into this. On the other hand, by leaving aside this activity, software managers and developers know the risk and must look for contingency plans to mitigate it, e.g., the negotiation of testing facilities before signing the contract.
Many hints were provided not only from various literature sources but also from the experiences of pilot projects that summarize the most important points to take into account for new projects.
One outstanding observation was the need of performing design feasibility studies in order to better understand the technology and the possibilities for implementing a service. Experiences from the pilots showed that the requirements phase was difficult to control due to the novelty of the domain and the fact that low level requirements and, particularly, usability-related requirements (e.g., how to represent large tables on small displays) were often not well understood. Feasibility studies proved to be a good means to handle the related uncertainty.
Testing proved very challenging due the great diversity of devices available on the market, the unreliability of device specifications, the low degree of automation of the testing procedures on real devices, and the unreliability of the available emulators.

2.3 References

[1] Kovari, P., Acker, V.B., Marino, A., Ryan, J., Tang, L.K., Weiss, C.: Mobile Applications with Websphere Everyplace Access Design and Development. IBM SG24-6259-00 (2001).
[2] Buttler, M.H.: Current Technologies for Device Independence. HP Laboratories, HP-2001-83, Bristol., (2001).

[3] McDermid, J.A., Rook, P.: Software Development Process Models, Software Engineer's Reference Book, Ed., Boca Raton, FL: CRC Press, pp. 15.26 – 15.28. (1994).
[4] Adamopoulos, D.X., Pavlou, G., Papandreou, C.A.: An Integrated and Systematic Approach for the Development of Telematic Services in Heterogeneous Distributed Platforms. Computer Communications, vol. 24, pp. 294-315 (2001).
[5] Karlsson, E., Taxen, L.: Incremental Development for AXE 10. ACM SIGSOFT Software Engineering Notes, vol. 22, No. 6 (1997).
[6] Taylor, M.J., McWilliam, J., Forsyth, H., Wade, S.: Methodologies and Website Development: A Survey of Practice. Information and Software Technology, vol. 44, No. 6, pp. 381-391 (2002).
[7] Zettel, J., Maurer, M., Münch, J., Wong, L.: LIPE: A Lightweight Process for E-Business Startup Companies based on Extreme Programming. Proceedings of the Third International Conference on Product-Focused Software Processes Improvement (PROFES), pp. 255-270, (2001).
[8] Nilsson, A., Anselmsson, M., Olsson, K., Johansson, Erik.: Impacts of Measurement on an SPI Program. Q-Labs (http://www.q-labs.com/files/Papers/SPI99_Imp_of_Meas_on_SPI.pdf).
[9] Yau, V.: Project Management Strategies and Practices for Wireless CDMA Software Development. Proceedings of the IEEE International Conference on Industrial Technology, (1996).
[10]Highsmith, J.: What is Agile Software Development? The Journal of Defense Software Engineering. October, (2002).
[11]Maurer, F., Martel, S.: Rapid Development for Web-Based Applications. IEEE Internet Computing, vol 6, No 1, pp. 86-90 (2002).
[12]Boehm, B.W.: Get Ready for Agile Methods, with Care, IEEE Computer, vol 35, No 1, pp. 64-69 (2002).
[13]Boehm, B.W.: A Spiral Model for Software Development and Enhancement, IEEE Computer, vol 21, No 5, pp. 61-72 (1988).
[14]Boehm, B.W.: Get Ready for Agile Methods, with Care, IEEE Computer, vol 35, No 1, pp. 64-69 (2002).
[15]Beck, K.: Extreme Programming Explained: Embrace Change. Addison Wesley, (2000).
[16]Constantine, L.L., Lockwood, A.D.L.: Usage-Centered Engineering for Web Applications. IEEE Software, vol. 19, No. 2, pp.42-50 (2002).
[17]Hammar, C.M.: Designing User-Centered Web Applications in Web Time. IEEE Software, vol. 18, No. 1, pp. 62-69 (2001).
[18]http://www.w3.org/TR/2001/WD-di-princ-20010918/
[19]Schwabe, D., Mattos, G.R., Rossi, G.: Cohesive Design of Personalized Web Applications. IEEE Internet Computing vol. 6, No 2, pp 34-43 (2002).
[20]Friday, A., Davies, N., Blair, G.S., Cheverest, K.W.J.: Developing Adaptive Applications: The MOST Experience. Integrated Computer Aided Engineering: ICAE, vol 6, No 2, pp. 143-158 (1999).
[21]Giannetti, F.: Device Independency Web Application Framework. In: W3C Device Independent Authoring Techniques Workshop, 25-26 (2002).

[22]Wong, C., Chu, H., Katagiri M: A Single Authoring Technique for Building Device Independent Presentations. In: W3C Device Independent Authoring Techniques Workshop, (2002).
[23]Mori, G., Paterno, F., Santono, C.: An XML Based Approach for Designing Nomadic Applications. In: W3C Device Independent Authoring Techniques Workshop, (2002).
[24]Roe, C., Gonik, S.: Server-Side Design Principles for Scalable Internet Systems. IEEE Software, vol.19, No. 2, pp. 34-41 (2002).
[25]Cushnie, J., Hutchison, D., Oliver, H.: Evolution of Charging and Billing Models for GSM and Future Mobile Internet Services. Lecture Notes in Computer Science, vol. 1922, pp. 312-323 (2000).
[26]Odlyzko, A.: Paris Metro Pricing: The Minimalist Differentiated Services Solution. AT&T Laboratories Research, April 1999.
[27]Wong, C., Chu, H., Katagiri, M.: GUI Migration Across Heterogeneous Java Profiles. Proceedings of ACM SIGCHI-NZ'02. (2002).
[28]Constantine, L.L., Lockwood, A.D.L.: Usage-Centered Engineering for Web Applications. IEEE Software, vol. 19, No. 2, pp.42-50 (2002).
[29]http://java.sun.com/j2me/docs/alt-html/midp-style-guide7/preface.html
[30]Upchurch, L., Rugg, G., Kitchenham, B.: Using Card Sorts to Elicit Web Page Quality Attributes. IEEE Software, vol. 18, No. 4, pp. 84-89 (2002).
[31]Read, K., Maurer, F.: Developing Mobile Wireless Applications. IEEE Internet Computing, vol 7, No 1, pp. 81-86. (2003).
[32]http://wireless.java.sun.com/midp/tips/appsize/
[33]Colberg, C.S., Thomborson, C.: Watermarking, Tamperproofing and Obfuscation-Tools for Software Protection. IEEE Transactions on Software Engineering, vol. 28, No. 8, pp. 735-746, (2002).
[34]Cockburn, A., Highsmith, J.: Agile Software Development: The People Factor, Computer, vol 34, No 11, pp. 131-133. (2001).
[35]Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek M. F.: The Click Modular Router, proceedings of ACM Transactions on Computer Systems, vol. 18, pp. 263-297, (2000).
[36]Rizzo, L.: Dummynet: a simple approach to the evaluation of network protocols, proceedings of ACM Computer Communication Review, vol. 27, (1997).
[37]Fall, K.: Network Emulation in the VINT/NS Simulator, In Proceedings of 4th IEEE Symposium on Computers and Communications, (1999).
[38]Noble, B.D., Satyanarayanan, M., Giao, T.N, Katz, H.R.: Trace-Based Mobile Network Emulation, Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication.
[39]Satoh, I.: Flying Emulator: Rapid Building and Testing of Networked Applications for Mobile Computers, Proceedings of 5th International Conference on Mobile Agents (MA'2001), Lecture Notes in Computer Science (LNCS), vol. 2240, pp.103-118, Springer, (2001).
[40]Zheng, P, Ni, M.N: EMPOWER: A Network Emulator for Wireless and Wireline Networks. In Proceedings of IEEE INFOCOM (2003).
[41]El-Far, I.K., Thompson, H,H., Mottay, F.E.: Experiences in Testing Pocket PC Applications. In Proceedings of the Fifth International Software and Internet Quality Week Europe, (2002).

[42]Chi, E.: Improving Web Usability Through Visualization. IEEE Internet Computing, vol. 6, No 2, pp. 64-71 (2002).
[43]Becker, A.S.: A Usability Perspective on Wireless Internet Technology, Computer Science & Software Engineering, Florida Institute of Technology, (2001).
[44]Thimbleby H., Cairns, P., Jones M.: Usability Analysis with Markov Models, ACM Transactions on Computer-Human Interaction, Vol. 8, (2001).
[45]Sinha, A., Chandrakasan, A.: JouleTrack - A Web Based Tool for Software Energy Profiling, Proceedings of the 38th Design Automation Conference, (2001).
[46]Nielsen, J.: Graceful Degradation of Scalable Internet Services, WAP: Wrong Approach to Portability, Alertbox 31/10/1999 at http://www.useit.com/alertbox/991031.html
[47]Buchanan, G., Farrant, S., Jones, M., Thimbleby, H., Marsden, G., Pazzani, M.J.: Improving Mobile Internet Usability. In proceedings World Wide Web 10, pp. 673-680, (2001).
[48]Becker-Kornstaedt, U., Hamann, D., Kempkens, R., Rösch, P., Verlage, M., Webby, R., Zettel, J.: Support for the Process Engineer: The Spearmint Approach to Software Process Definition and Process Guidance. Proceedings of the Eleventh Conference on Advanced Information Systems Engineering (CAISE '99), pp. 119-133. Lecture Notes in Computer Science, Springer-Verlag. Berlin Heidelberg New York (1999).
[49]Becker-Kornstaedt, U., Boggio, D., Muench, J.,Ocampo, A., Palladino, Gino.: Empirically Driven Design of Software Development Processes for Wireless Internet Services. Proceedings of the Fourth International Conference on Product-Focused Software Processes Improvement (PROFES), (2002).
[50]Nielsen, J., Mack, R.L.: Usability Inspection Methods / John Wiley & Sons, Inc; (1994).