

**An Approach for Peer-to-Peer Collaboration between Mobile Devices
Using Web-based Technologies**

Ville Antila

University of Tampere
Department of Computer Sciences
Computer Science, M.Sc Thesis
Supervisor: Roope Raisamo
May, 2009

University of Tampere

Department of Computer Sciences

Computer Science

Ville Antila: An Approach for Peer-to-Peer Collaboration between Mobile Devices

Using Web-based Technologies

Master's Thesis, 85 pages

May, 2009

Communication and collaboration is often independent of time and place. Thus, the collaboration tools should also be available when the users are moving around. The mobile devices have a key role in aggregating information while people are on the go. Therefore it would be beneficial to use that potential also for sharing information collaboratively. In this thesis, one possible solution to this challenge is given. This work provides the theoretical and technical background for mobile peer-to-peer collaboration, by combining and merging results from research on mobile collaboration and collaborative Web services. Moreover, requirements are gathered from these background studies in order to define the scope for this work.

In this thesis the main objective has been to design architecture for mobile collaborative work as well as to implement a prototype system to evaluate the design. Existing technologies were used whenever possible. The proposed solution integrates a lightweight Web Service Architecture and a mobile device equipped with a standard Web server in order to enable a mobile collaborative sharing of information. In addition to the design and implementation, a background survey and a user study has been conducted to evaluate the approach from the user's perspective. The goal was to gain insight for the whole concept as well as to evaluate the architecture and the usability of the proposed client applications. From these evaluations it seems that the approach for mobile collaboration is feasible and usable on the basis of the results. This gives a sound background for future investigations and innovations build on top of this approach.

Keywords: Mobile Web-based applications, Mobile collaboration, Web services, Human-computer interaction

Table of contents

1. Introduction.....	1
1.1. Problem statement and hypothesis	2
1.2. Objectives for the work	3
1.3. Research approach	3
1.4. Structure of the thesis	4
2. Computer-mediated communication and collaboration	5
2.1. Peer-to-peer collaboration.....	6
2.2. Mobile peer-to-peer collaboration	7
2.3. Web-based groupware	8
2.4. Web services.....	9
2.4.1. Collaborative Web services.....	10
2.4.2. Web service architectures.....	10
2.4.2.1 <i>Service oriented model</i>	11
2.4.2.2 <i>Resource oriented model</i>	12
2.5. Design of a mobile collaborative system	13
2.6. Usage scenarios for a mobile collaborative system.....	13
2.7. Available collaborative applications	14
2.7.1. Shared calendar	14
2.7.2. Shared workspace	16
2.7.3. Web logs	17
2.7.4. Discussion forums.....	17
2.8. Summary of related work.....	18
3. Web-based technologies on mobile devices	19
3.1. Overview on technologies for mobile devices.....	20
3.2. Web architecture	21
3.2.1. Web technology stack.....	21
3.2.2. Web runtime environment.....	22
3.2.2.1 <i>Web runtime on a mobile device</i>	24
3.2.2.2 <i>Mobile widgets as task-specific Web service clients</i>	25
3.2.3. Mobile Web server	26
4. Concept architecture	28
4.1. Background	28
4.2. Overview	29

4.3.	Characteristics of the concept.....	30
4.3.1.	Resource oriented server interface.....	30
4.3.2.	Lightweight client for a task-specific purpose	31
4.3.3.	Fast and efficient development	31
4.3.4.	Platform independence and interoperability	32
4.4.	Design objectives.....	32
4.5.	Requirements for the architecture	33
4.5.1.	Requirements from existing work	33
4.5.2.	Requirements for mobile peer-to-peer collaboration.....	34
4.6.	Architecture.....	35
4.6.1.	The resource model.....	36
4.6.2.	Server-side architecture.....	37
4.6.3.	Client-side architecture.....	38
4.6.4.	Communication	40
4.6.4.1	<i>Infrastructure</i>	40
4.6.4.2	<i>Ad hoc</i>	41
4.6.4.3	<i>Ajax</i>	42
4.6.4.4	<i>XML versus JSON</i>	42
4.7.	Summary	43
5.	Concept prototype – Mobile collaborative scheduling.....	45
5.1.	Collaborative scheduling	45
5.2.	Requirements for the prototype.....	45
5.2.1.	User scenario	46
5.2.2.	Use cases	46
5.2.3.	Functional requirements	47
5.3.	Implementation	47
5.3.1.	Server-side implementation.....	48
5.3.1.1	<i>Request handler</i>	49
5.3.1.2	<i>Service model</i>	50
5.3.2.	Client-side implementation	51
5.3.2.1	<i>Client-side components</i>	52
5.3.2.2	<i>DOM manipulation</i>	52
5.3.2.3	<i>Event handling</i>	53
5.3.2.4	<i>Ajax communication</i>	53
5.3.2.5	<i>Web Runtime specific features</i>	54
5.4.	Design alternatives.....	55
5.4.1.	<i>Shared Calendar</i> user interface approach	55
5.4.2.	<i>Group Scheduler</i> user interface approach.....	56
5.5.	Summary	57

6. Evaluation	58
6.1. Methods	58
6.2. Background analysis.....	58
6.2.1. General aspects	59
6.2.2. The use of Web content on mobile devices.....	59
6.2.3. The use of a mobile device for PIM	60
6.2.4. The use of a mobile device for calendar purposes	60
6.2.5. Summary of the analysis	61
6.3. Concept evaluation	61
6.3.1. Mobile collaborative system requirements.....	62
6.3.2. Web services architecture requirements	62
6.4. Prototype evaluation	63
6.4.1. Functional evaluation	63
6.4.2. User evaluation.....	64
6.4.2.1 <i>Background information</i>	64
6.4.2.2 <i>Test setup</i>	65
6.4.2.3 <i>Pilot test</i>	66
6.4.2.4 <i>Test results</i>	66
6.4.2.5 <i>Comments and improvement ideas</i>	69
6.5. Summary	70
7. Discussion.....	71
7.1. Results.....	71
7.2. Findings and considerations.....	73
7.2.1. Architectural considerations.....	73
7.2.2. Communication considerations	74
7.2.3. User interface and interaction considerations.....	74
7.3. Future work	75
7.3.1. Architectural advancements.....	75
7.3.2. Advancements for mobile collaborative scheduling.....	75
8. Conclusions	77
References	79

Table of Figures

Figure 1. CSCW Matrix (adapted from [Baecker, 1995])	6
Figure 2. Web service architecture overview (adapted from [Champion et al., 2002])	9
Figure 3. Service oriented model [Booth et al., 2004].....	11
Figure 4. Resource oriented model [Booth et al., 2004]	12
Figure 5. Google Calendar and Doodle on a S60 mobile Web browser	16
Figure 6. Application Runtime Environments for S60 3 rd Edition [S60, 2007].....	20
Figure 7. Technology stack for Widget development by W3C [Caceres, 2007].....	24
Figure 8. Web framework for S60 platform (adapted from [Nokia Research, Center, 2009])	25
Figure 9. Mobile Web server network architecture [Mobiforge, 2009]	27
Figure 10. Conceptual overview – the distributed service instances (servers) provide the collaborative information for the users of the service (clients).....	29
Figure 11. Architecture overview – each user (peer) can both provide and consume the collaborative information	36
Figure 12. Ad hoc and infrastructure communication comparison.....	41
Figure 13. Group communication – collaboration is done in small groups but the information can be shared to a wider context	42
Figure 14. Communication procedure	43
Figure 15. Implementation architecture – illustrates the logical components of the implemented architecture.....	48
Figure 16. Class diagram for the server-side implementation	49
Figure 17. Client-side components	52
Figure 18. <i>Shared Calendar</i> user interface approach.....	56
Figure 19. <i>Group Scheduler</i> user interface approach	57

1. Introduction

Collaboration is all about re-using and sharing information which is up-to-date. The challenge for mobile collaboration is the availability of that information at any given time. While on the go, a mobile device is often the only available storage for information. Mobile devices are used to save many personal information management items such as calendar entries and contact information. They can also contain information about the user's current activity or act as an aggregator of information from the environment around the user. This information would often be useful to share with other people in a certain group. In addition to this, the mobile devices and the mobility itself provide a wide variety of user scenarios and situations where other means for communication are not available.

The need for collaboration might occur in very different circumstances, thus the needed tools should be available on the go [Haveri et al., 2007]. They also should be scalable to handle different types of communication and collaboration scenarios. For example, there might be a need for collaboration in an environment where there is no established infrastructure for connectivity available [Neyem et al., 2006], or situations where no communication is available, but the information is needed locally (e.g., in an airplane). The use of the information when needed should not be dependent on the current place or situation.

Currently much of the information contained and provided by the mobile device is only usable for the owner of the device itself. The collaborative use of mobile device's resources is not possible with current services available. Nevertheless, the sharing of information in a group is essential for group work and collaboration in general. Collaborative applications are designed to enhance and ease that task [Grudin, 1994]. In addition, the collaborative applications are an interesting field of research due the need for meaningful and efficient interaction between people. This is not an easy task to achieve especially in a mobile environment. On the other hand, there are several emerging technologies that might help.

In the recent emergence of lightweight Web technologies and Web service models as well as the development of mobile devices and Web runtime platforms, the Web seems to be the most versatile and promising platform for new applications available. Moreover, due to the extensive amount of mobile platforms, it seems that an up rise of runtime interpretable scripting or Widget

type of applications might be more interoperable at least as specific purpose applications [Caceres & van Kesteren, 2007]. These lightweight applications or scripts aggregate content from the native information management and capabilities and combine the information from different sources into a compound knowledge that can be used collaboratively.

Web technologies enable easier and faster user interface development than many of the traditional software techniques. They can also provide a scalable user interface structure that would enhance the interoperability of the applications. In addition, to adapt the user interface even more, context awareness derived from the device's native information sources can be used. The recent development on the Web runtimes and Widgets have been enabling more access to device capabilities and resources through additional extensions, these include for example the location as well as contact lists, calendar events and even data flow from the embedded sensors [S60b, 2009]. This trend could lead the way for device and task specific micro-sized Web applications.

Another important issue is the interoperability of applications. The most prominent way of enabling interoperability between different platforms and applications is to provide open APIs between them. Web services have been studied extensively during the early 2000s and this trend is continuing even more due to the extensive amount of new social Web applications. In a nutshell, Web services provide a distributed environment for Web-based application development. Moreover, it can be claimed that the Web services enable collaboration over a network in a device independent and interoperable way [Haas & Brown, 2003].

In this thesis the objective is to find a solution to enable collaboration in a mobile environment. One interesting possibility is to evaluate if the available Web service architectures combined with lightweight, specific purpose clients can be used for that purpose and if so what are the requirements for them. Moreover, the objective is to implement an example service with an extendable architecture and suitable clients for it. In order to evaluate the functionality of such a system it has to be examined towards the requirements derived from the user scenarios. But before that, the background for this work is elaborated more in the next sections.

1.1. Problem statement and hypothesis

People regularly need to collaborate on a small task within a very brief moment. For example, when a group of people participating in a meeting need to decide the time of the next meeting. These situations are very brief and need to be handled quickly so that the main purpose would not become lost.

Furthermore, people use various devices and services to save their personal information. In these situations the interoperability and seamless interaction between different devices would be very favourable.

The mobile devices have a lot of important information available on them to use in a collaborative way. In addition the Web provides an extendable and interoperable platform for communication and collaboration. The combination of these two technological enablers would be an interesting possibility.

Therefore, the hypothesis for this thesis work is formulated as following:

“The available Web techniques can be applied to the mobile environment to enhance the ability to interact in a group and to share content in a peer-to-peer manner”.

1.2. Objectives for the work

In order to validate the presented hypothesis, there are several objectives for the work. There are some basic objectives to study the possibilities and related work around the topic, as well as to evaluate the most promising approaches to enable the mobile peer-to-peer collaboration.

In more detail the objectives for the thesis work are to:

1. *Research:*
 - The related research and technological possibilities.
2. *Design and construct:*
 - Architecture and a prototype for a peer-to-peer collaboration tool.
3. *Evaluate:*
 - Web service architectures in a mobile environment.
 - Mobile web services combined with task-specific clients as a collaboration platform.
 - The concept prototype service and clients against the requirements for the collaboration tool.
 - The usability of the prototype system by conducting a user study.

These objectives describe the steps in which this work is constructed. Moreover, these objectives are used as the reference points when evaluating the outcome of this work.

1.3. Research approach

The research approach used in this thesis is a combination of constructive and empirical research. The first step is to gather knowledge from the related studies. The related work gives the necessary background for the evaluation

goals and objectives of this work as well as some insight on the results of similar research. Moreover, since the available technology solutions are developing very fast at the moment, the evaluation of the technological possibilities is important.

The constructive part of the research approach includes a prototype of the proposed concept. It serves as the realization of the introduced approach and as a reference implementation of the architecture in case. In order to analyze the feasibility of the approach, there is a need for end-user participation. For involving users in the development, the work in this thesis uses some of the most common techniques of user-centred design process, such as a background survey, usability testing and interviews [Preece et al., 2002].

1.4. Structure of the thesis

In Chapter 2, the background for the thesis is researched and discussed. This includes the available research in the field of computer-mediated collaboration and communication with a specific emphasis on the collaboration in a mobile environment. After that, the available technologies and their possibilities are surveyed in Chapter 3. These include the available Web technologies for mobile devices that can be used to build a mobile collaborative tool.

In Chapter 4, the concept approach, building on the background studies is explained and defined. This includes stating the main characteristics and requirements for the system. To evaluate the approach, the prototype for the concept is explained and discussed in Chapter 5.

In Chapter 6, the functionality and usability of the system is evaluated towards the requirements as well as by conducting a user study. In Chapter 7 and 8, respectively, the discussion and conclusion for the work is given. In these chapters the outcome of the concept prototype and its evaluation towards the goal of the thesis is explained. Moreover, the relation with the related work is discussed further.

2. Computer-mediated communication and collaboration

This chapter relates the topic of this thesis to earlier studies. The focus is on the Web-based collaboration tools such as the collaborative Web services and how they can be used in a mobile peer-to-peer collaborative work.

Collaboration can be work related, family related or events coordinated with a group of friends. Therefore, there are a lot of different user scenarios to be considered when designing a collaborative service or an application. The collaborative applications are emphasizing the use of social networks. A social network can be a group of colleagues, friends or even family. Additionally, a social network can be an online community or a real-world community or perhaps both. In the scope of this thesis, the social networks and communities are merely treated as the user group for collaborative applications.

There are several disciplines that are working on these topics. Computer-Supported Cooperative Work (CSCW) has a long tradition in studying the way people cooperate and collaborate [Grudin, 1994]. Several design goals and guidelines have been developed especially to help design collaborative software. These studies have also given the common ground for later studies. While the scope of the CSCW in general spans many types of collaboration, the focus of this thesis is on *asynchronous communications and collaboration*. This type of collaboration usually happens at a different time and place. The division of different collaboration types is depicted in the “*Time/Space matrix of Groupware*” (cf. Figure 1) introduced originally by Johansen [1988].

Today much of the communication and collaboration is done via the services and applications on the Web. Due to this development, the combination of these two technologies, the mobile Internet, can enable the interaction and communication between people even further [Haveri et al., 2007]. Therefore, from a technological point-of-view, the main focus is on the mobile and Web-based collaborative work.

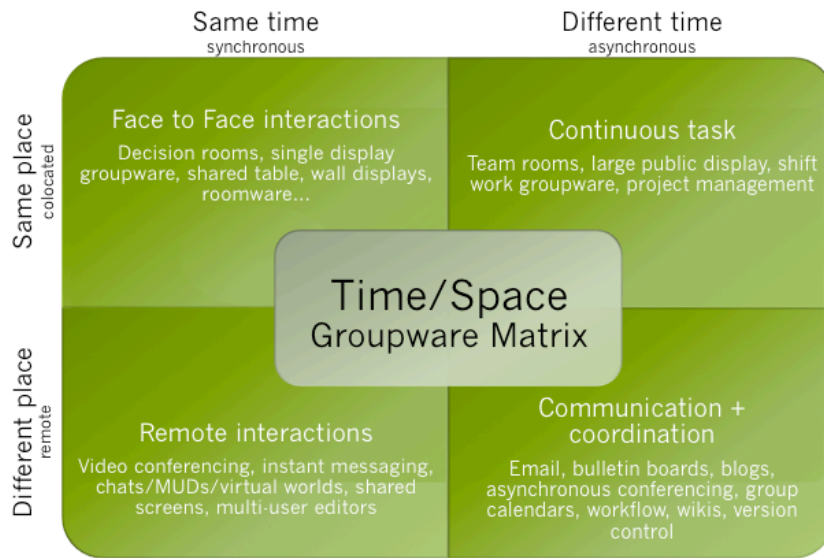


Figure 1. CSCW Matrix (adapted from [Baecker, 1995])

2.1. Peer-to-peer collaboration

Peer-to-peer collaboration uses peer-to-peer networking for a direct person to person communication and collaboration. A peer-to-peer network does not have a dedicated centralized infrastructure that the network would rely on, but rather it depends on the participation of peers to contribute the resources to be shared [Saroiu et al., 2002]. The usage of peer-to-peer type of systems have been growing steadily and the systems have been developing from merely file transferring to more overall communication systems like the Jabber instant messaging system [Oram, 2003].

In this thesis, when talking about peer-to-peer collaboration, the main emphasis is on the collaboration part and not so much on the peer-to-peer networking part. The peer-to-peer technology is merely the enabler for this kind of communication. In this work, the phrase peer-to-peer collaboration is used to describe both the technology aspect, such as the communication paradigm, as well as the way of collaborating. In a way this could be described better by saying that it is direct person-to-person collaboration over a network.

Another aspect of peer-to-peer collaboration, besides the way of communicating, is the way of storing the data. In all collaboration systems the data is a very important part of the system. Whether it is documents, calendar or scheduling events or merely context information, the data is a key part. The

fundamental aspect of peer-to-peer systems is that there is no need for a centralized server. This also means that the data being shared has to be stored in a peer-to-peer manner as well. This has the implication that the data does not have to be synchronized, since it is stored locally and only retrieved when needed. Some other issues are still remaining though, mainly the locking and coupling of objects. Though these limitations only apply to applications which are providing synchronous communication, such as a distributed authoring service.

2.2. Mobile peer-to-peer collaboration

Peer-to-peer systems have also been studied in a mobile context. Mobility and mobile communication brings new kinds of problems with holistic connectivity issues as well as with a different kind of market structure than the world of the Internet. Due to the unreliable network connectivity, a mobile peer-to-peer collaboration tool should also support disconnected use as well as an ad hoc type of connectivity [Reif et al., 2001]. Reif et al. propose that a mobile peer-to-peer collaboration system should support these three modes of connectivity, *connected mode*, *disconnected mode* and *ad hoc mode*. The different modes have different characteristics and functionality:

- In *connected mode*, a direct connection to the network is available. Collaboration is enabled independent of the location of other peers.
- In *disconnected mode*, there is no connection available, thus a direct collaboration is disabled. Nevertheless some tasks should be able to do offline until a connection to the network is available again.
- In *ad hoc mode*, the connection to the network is weak or not available, but communication with surrounding devices is available via an ad hoc connection. This mode enables working with a group of people close by.

Besides the concepts, some middle layer frameworks have also been developed for mobile peer-to-peer networking and collaboration. An interesting middleware for mobile devices developed by Harjula et al. [2004] is the Plug-and-Play Application Platform (PnPAP). This system wraps the underlying peer-to-peer technologies for mobile devices, such as Jabber, JXME and SIP and thus enables more general use of peer-to-peer networking in mobile applications. It also introduces a holistic connectivity module to enable optimal peer-to-peer protocol – connectivity pair to be used. This approach is very promising and provides many of the features needed for the underlying technology for a mobile peer-to-peer system. On the other hand, much more work should be done on the application level as well. The objective for this

thesis work is to build an application level solution that relies on the basic, commonly available communication protocols.

2.3. Web-based groupware

Groupware is a term defining an application used for collaborative or group work [Ellis et al., 1991]. The term “*Web-based groupware*” is used here to define a group of collaborative applications that are only accessible through a Web interface or a Web browser, and therefore do not have any sort of desktop client interface. The advantage of a Web-based groupware application is that it is usable anywhere, independent of the device the user is currently using. Nevertheless, the user interface structure might apply limitations, for example, on devices with limited screen size, such as mobile devices.

In Web-based groupware both the data as well as the user interface are Web-based and thus are distributed over the network. The use of these systems is not possible without a connection to the Internet at least not in a fully functional form. The Web-based groupware systems are usually relying on a centralized architecture, where a dedicated Web server and possibly a database deals with the representation and data handling as well as the versioning of the system. This kind of architecture is very efficient in a large scale system and it is also relatively reliable, but the user does need a connection to the network available to be able to work with it. It also needs the data to be stored in the centralized server, and thus the synchronization of data becomes an issue.

Nevertheless, most of the groupware applications today are developed for the Web, simply because it provides global connectivity. One of the most interesting and important features of Web-based groupware is that they can provide a Web API for their service, in another words provide a Web service to access, enter and manipulate content available. The goal of Web-based groupware with Web service support is to provide interoperability between many groupware systems [Dustdar et al., 2004]. The trend in the Web-based applications is nowadays to provide these Web APIs in order to extend the availability and functionality of their application or service. These APIs are becoming an important feature in the next generation of Web applications and services. Moreover, there is a trend of providing the existing Web applications as Web services by simply replacing the HTML payloads by something more flexible and machine interpretable format like the XML or JSON [Richardson & Ruby, 2007]. This is one, very simple way of providing a Web service interface, but very effective and extendable as well.

2.4. Web services

A Web service is “a software system designed to support interoperable machine-to-machine interaction over a network” [Haas & Brown, 2003]. A more simplistic description is that a Web service is a Web API that can be accessed over a network such as the Internet. From an implementation point of view though, there is a whole variety of architectural choices and mark-up languages to choose from. The research on these different mark-up languages and architectures has been extensive during the early 2000s. An overview of Web service architecture is depicted in Figure 2.

The importance of Web services as future platform of applications and possibly the future Web as a whole comes from the fact that they can provide a distributed, extendible and flexible architecture for very large scale applications. Moreover, the interoperability of Web applications gives a possibility to use this technology on a variety of devices.

On the other hand the use cases for this approach should be very specific and alternatives well thought about, since in some cases this technology seems not to be very useful or even suitable at all. Nevertheless, as we know from the past, assumptions have often proven to be wrong in the field of Computer Science, especially what comes to the development of Web.

In the scope of this thesis Web services are reviewed as enablers of collaboration over a network. Moreover, the objective is to evaluate which Web service architectures are applicable to the mobile context keeping in mind the scarce resources of mobile devices, such as the limited amount of memory and processing power, as well as the power consumption.

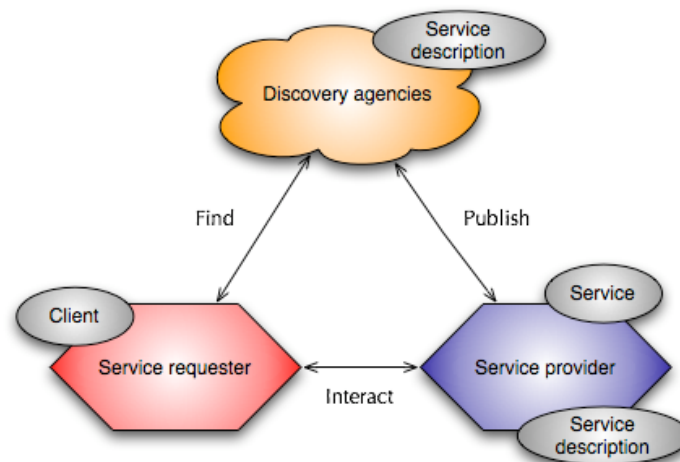


Figure 2. Web service architecture overview (adapted from [Champion et al., 2002])

2.4.1. Collaborative Web services

Since Web services provide a distributed, extendible and flexible architecture for loosely-coupled systems, they give a suitable platform for building collaboration tools as well. Jerstad et al. [2005] presents a framework for collaborative services based on the Service Oriented Architecture (SOA). Service Oriented Architecture is a generic architecture design for service oriented application development and therefore can be easily extended for a more specific use. Basically the architecture for collaborative services presented by Jerstad et al. is a service architecture that provides the means for collaborative work. It provides knowledge sharing in the form of shared documents and items, means for communication and personal interaction as well as tools for organisational management such as calendar and time scheduling.

The approach of Jerstad et al. [2005] has very similar goals than the objectives for this thesis work (cf. Section 1.2). On the other hand the architectural choice might not be so feasible with mobile devices in a distributed environment due to its need for an extensive cross-device communication. There are some other Web service architectures available as well, which can provide a more lightweight implementation. Also the scope of this thesis is not to provide an all-purpose collaboration platform but to enable some of them in the context of mobile devices and mobile collaboration using the existing technologies and protocols.

2.4.2. Web service architectures

Some background for the Web service architectures is presented here in order to choose the best suitable approach for the work. Basically, there are different kinds of Web service architectures available and the decision is mostly dependent on the need of use. The Web service architecture defines the interface and the discovery mechanisms for the Web services (cf. Figure 2). Furthermore, it provides a conceptual model and a context for understanding Web services as well as the relationship between the components involved [Booth et al., 2004]. There are a couple of different concepts of how the services are modelled. The most important models are perhaps the service-oriented model and the resource-oriented model [Booth et al., 2004]. These architectural models does not specify the implementation details of a Web service but rather gives the common ground of concepts to build the implementation upon.

Pautasso et al. [2008] discuss the differences between the service and resource oriented models more specifically. They give some insight into deciding between them based on technical and conceptual comparison. The

main conclusions that Pautasso et al. reached are that the technical characteristics between these approaches are quite similar; meanwhile there are some conceptual differences. One difference is that the resource oriented model is more simplistic and tactical, thus suitable for ad hoc Web integration (i.e. mash-ups), while the service oriented approach provides more overall solution capable of demanding requirements (i.e. enterprise services) [2008]. In the following subsections these two models are explained in more detail.

2.4.2.1 *Service oriented model*

Service oriented model is an architectural style for Web services that focuses on services and actions [Booth et al., 2004] (cf. Figure 3). Web services as a whole are often seen as service oriented, although that is not always the case. Service oriented architecture is one way of designing a Web service. The service oriented Web services are often defined as services described by Web Service Description Language (WSDL) [Christensen et al., 2001] and they often use SOAP [Box et al., 2000] as the communication protocol.

In a service oriented approach the provider of the Web service makes the service available by publishing it. The publishing is done by describing the interface with the WSDL document. Eventually, when the consumer finds the requested service via a discovery agency, then the WSDL document is used to configure the interaction between the consumer and provider [Ferris, 2003]. This is a very simplistic way of describing service oriented Web services without going more into the depths of service discovery mechanisms and semantics involved in the reality.

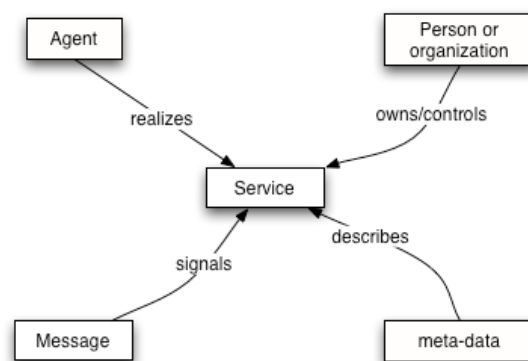


Figure 3. Service oriented model [Booth et al., 2004]

The Service Oriented Architecture (SOA) is aimed at defining a very universal and interoperable interface for large scale systems, such as business processes [Erl, 2005]. It is also a very complex system to implement and thus not very suitable for a mobile environment. Whilst being a very well documented and

studied architecture the Service Oriented Architecture is more of an overall architectural style or reference architecture than a real implementation style. Furthermore, it cannot be easily scaled to a smaller system. Therefore, the use of Web services in a mobile device environment should be implemented with a more lightweight and flexible architecture.

2.4.2.2 *Resource oriented model*

Another architectural style for Web services is a resource oriented model [Booth et al., 2004] (cf. Figure 4). Resource is a fundamental concept in the Web and Web services. In a resource oriented model the Web service is based on resources that have Uniform Resource Identifiers (URIs). They also have an owner, in other words a host. Furthermore, they have a representation of some kind, defined by HTML or XML, for an instance. The resource oriented architecture is often called RESTful Web service architecture [Richardson & Ruby, 2007].

REST (Representational State Transfer) is a basic architectural style on which the whole World Wide Web is built on [Fielding, 2000]. The term RESTful Web service means a Web service interface that is built upon the REST architecture. Basically this means that the Web service provides resources that are defined by URI's and that the mechanism used to interact with these resources are the four methods provided by HTTP: GET, POST, PUT and DELETE. The whole interface can be described by the resources it provides and the applicable methods to interact with them.

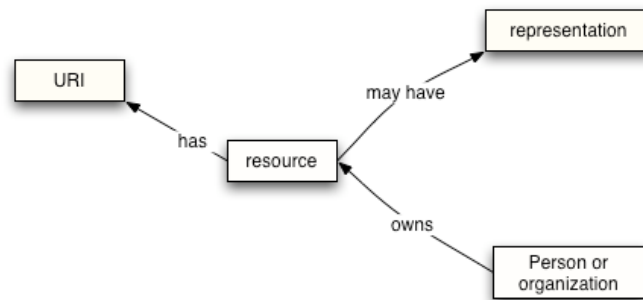


Figure 4. Resource oriented model [Booth et al., 2004]

The power of a RESTful interface is the simple, yet extendable nature of the interface. Moreover, since the whole Web is conceptually based on REST architecture, it is the most common type of Web interface that exists. In this thesis work the objective is to evaluate and implement a Web-based collaborative tool, for which the Web services provide a promising solution. Moreover, since the mobile environment brings some performance issues it should be a simple yet extensive enough to be used for a variety of different

tasks. In this background the usage of RESTful interface would be the most suitable possibility to choose from.

2.5. Design of a mobile collaborative system

Collaborative systems need to have basic functionality to enable the collaboration in a group. Dustdar & Gall [2003] specify some architectural components that are needed in a collaborative system supporting distributed and mobile work. First of all, it needs to have some kind of user and community management, meaning that the people who are using the system have to be defined to enable access control and addressing. Depending on the scale of the system, this might be a centralized unit or a locally configurable module.

Another important feature is the resource management. In a centralized architecture, resources are stored on a dedicated server, which handles the resource management. On the other hand on a peer-to-peer system the resource management has to be done locally and thus provide a way to distribute the resources to others as well as provide a notification service, such as a *publish/subscribe* type of component.

Access control is also an important feature in a distributed system. To enable a secure collaboration between groups of people, there needs to be some kind of way to ensure the control over the access rights. Sometimes this feature can be implied in the very basis of the architecture used; sometimes it needs to be handled separately. In the scope of this thesis, this aspect is acknowledged but not implemented in the example service. Since this architectural concern is very real and a serious task to handle, it is further discussed in Section 7.

Although the architectural concerns listed above are very practical and often much needed, the architecture will often be very different in practice depending on the implementation platform and technology. All of these requirements might not have to be implemented separately as components, but are merely implied in a general architecture of choice.

2.6. Usage scenarios for a mobile collaborative system

In this section, some example usage scenarios for mobile collaborative system are provided in order to give some concrete directions and ideas for the use of this research.

Some of the most interesting examples are:

- *Distributed recommendations and aid for decision making* – This scenario would make use of a “*group presence*” information to decide for example, about scheduling of tasks in a group.
- *Messaging systems* – To enable better asynchronous communication, a messaging system could provide interesting user scenarios when used on mobile device.
- *Shared mobile calendar* – Sharing calendar events in a mobile environment would give flexibility to scheduling and awareness tasks.
- *Shared presence* (i.e. context and location) – The sharing of location and context information can be helpful in a collaborative work. This could be integrated with a messaging system for example.

Other possible and interesting scenarios are in a home environment as well as in a group of friends for example. How the scheduling of events is organized and how items of interest are shared. The personal information can be shared outside these groups as well, even publicly to everyone in the world.

2.7. Available collaborative applications

Today people already use a substantial amount of different Web-based collaboration tools available, such as an asynchronous messaging or calendar services. While most of these services can be used via different devices as well, they all require centralized information storage and are not usually interoperable with other services. They also require a connection to the network to save information which is not always available in a mobile environment. For some information it would be more usable to store it locally and share when it is needed, in a distributed and peer-to-peer manner. Nevertheless, these example applications provide a state-of-the-art for the most important Web-based collaboration tools, and thus provide a scope of scalability for a mobile collaborative platform. These example applications are described in the following subsections.

2.7.1. Shared calendar

There are many commercial calendar applications available that support the sharing of calendar events, such as Microsoft Outlook and Google Calendar. Although being very widely used services, in order for the users to utilize this functionality, everybody in the workgroup has to use the same service provider. Another important issue to take into account, especially in scheduling events is that it often takes place in an ad hoc situation and thus people may not have their computers with them.

Some research has been done to evaluate the need for sharing calendars in a group of users. Carvalho et al. [2008] have been studying the use of calendaring software in the personal and office use. Their results show that sharing of calendar events is not very common in a personal use but on the other hand found very useful in a professional or office use. They state that the participants found sharing of calendar events as useful, but did not extensively use this functionality. The reasons for this remain open, but some effect might be due to the ease of use of the software used or the interaction taken place when managing and browsing the shared views. These issues should also be taken into account when designing a shared calendar view in a mobile collaboration tool.

There are some available Web-based calendaring and scheduling applications that are also available in a mobile browser friendly layout. Probably the most widely used Web-based calendaring and scheduling services are Doodle [Doodle AG, 2009] and Google Calendar [Google Inc., 2009a]. However, these services are very different. Google Calendar is a basic calendaring service, thus it can be used for variety of different tasks which includes a scheduling possibility through sharing of calendar views with a group of people. Doodle on the other hand is a single purpose service providing an easy-to-use and fast service to schedule events. Both of these approaches have their strengths as well as weaknesses. They are also entirely Web-based, thus all the data has to be entered in to the service by the users. Screenshots from these applications used with a mobile Web browser are provided in Figure 5.

To enable these kinds of services in a more peer-to-peer manner, the goal would be to achieve similar services by pulling the information straight from the user's device. This could be done by providing a service from the individual devices that can serve this information to everybody in a group of users. Thus the user does not have to enter the calendaring information twice. These example services also give some reference to how it could be designed.

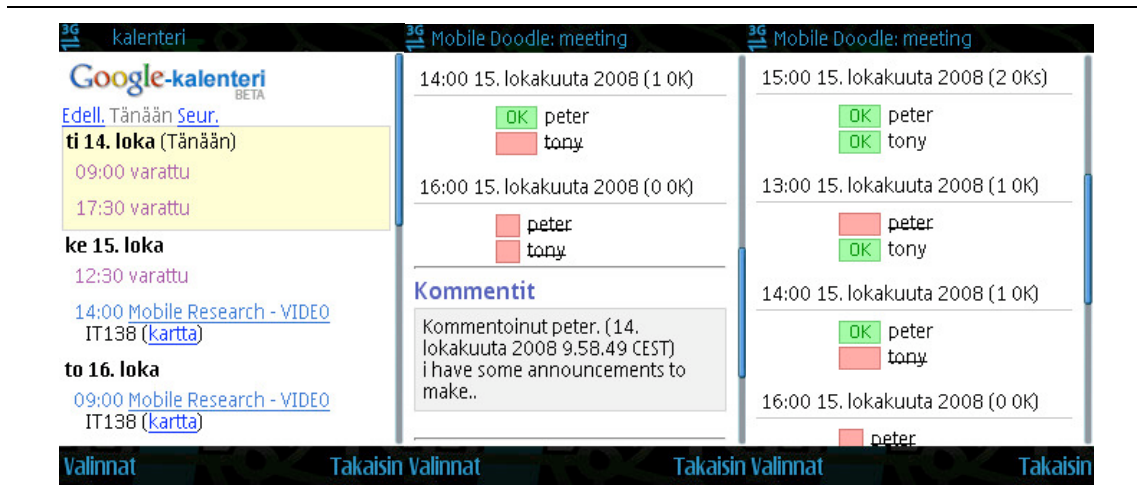


Figure 5. Google Calendar and Doodle on a S60 mobile Web browser

The shared calendar topic has been discussed recently in the literature. Koskela et al. [2007] introduce an example service scenario for their context-aware mobile Web 2.0 service architecture. This service scenario explains a Web-based community calendar that would be accessible through a mobile mash-up service. This kind of approach would enable sharing of calendar events independent of their calendar application, as well as the type of terminal they are currently using. This scenario and the service architecture view are very much related to the goals of this thesis. Unfortunately, this scenario is not implemented yet, and thus can not be evaluated or discussed further here.

2.7.2. Shared workspace

Sharing documents and other items are very important in many workgroups. The notion of shared workspace is however very broad. It can be used to describe merely the means of sharing documents as well as a collaborative writing service. Moreover, it can describe an extensive awareness system helping people in collaborative work at large [Dourish & Bellotti, 1992]. In this thesis shared workspace is used as a loose term to describe the means of sharing documents and items.

Commercial Web-based applications for this are available, such as BSCW [BSCW, 2009], and Google Docs [Google Inc., 2009b]. To be able to use these services people have to have an account or in the case of BSCW the workgroup itself has their own service running on a server. These applications are designed for desktop computer use thus they are not generally usable from a mobile device. There has been though, some recent development on mobile collaborative workspace application such as the Nokia Easy Meet [Nokia Oyj, 2009].

Mobile work has its own characteristics. People often need to share items in situ. For these kinds of situations, a more flexible system would be more feasible. Reif et al. [2001] have introduced a Web-based peer-to-peer architecture for collaborative nomadic working. They characterize the needs for nomadic or mobile collaborative work as well as give some example of the needed tools this kind of system should provide. The sharing of knowledge in the form of shared items is very important in mobile collaboration tool. In the system each user has artifacts that are stored on the user's own *resource space*. From these resources the user can make some or all available for other user's. This space is called the *member space* [Reif et al., 2001]. In this work the shared workspace metaphor is used merely for the ability to share items and documents in a group of users.

2.7.3. Web logs

Web logs (blogs) are Web sites that are usually maintained by an individual with regular entries of information. The nature of the entries can be a whole variety of things. They can be personal or corporate types and they usually provide a commentary on a specific genre, for example politics [Wikipedia, 2009a]. From a technical point of view they are Web services that provide a feed of information in a form of entries.

Blogs can be thought of as a type of collaboration tool. The blogs usually provide functionality to link the entry to another in the form of a response. This functionality is called a *trackback* and it is an easy way of linking blog entries to others automatically [Wikipedia, 2009a]. Another way of collaboration by blogging is a collaborative blog. In a collaborative blog there are several authors that provide entries of information [Wikipedia, 2009c]. The topic can be for example project-related news or notifications. The blogging service is taken into account as a way to distribute compound entries of different content to the group of users and thus enabling the sharing of knowledge and items.

2.7.4. Discussion forums

Discussion forums or Internet forums are Web services where people can create discussion threads based on a certain topic [Wikipedia, 2009b]. The threads will have a header and a body of responses. Typically forums have a specific purpose. Forums are an excellent way of finding and distributing knowledge, since usually people are looking for similar information. The content on forums is user generated and the whole maintenance of the forum and thread structure is very much community-based.

There has been some research on distributed forum-like threads used for communication in a mobile environment [Lämsä, 2008]. These threads are

dynamic and are stored locally as opposed to the common centralized architecture. The discovery of threads can be somewhat location based, for example based on Bluetooth connectivity. This kind of distributed thread messaging is dramatically different from the ordinary discussion forum since the nature of communication as well as the content discovery is very much peer-to-peer type. Moreover, the search of content in this kind of peer-to-peer architecture would be very different from an ordinary Web search. These distributed threads would be an interesting feature for a mobile peer-to-peer collaboration tool.

2.8. Summary of related work

In the previous sections the background and the possibilities for the implementation have been surveyed. Some previous studies and approaches have been introduced to give an overview of the work done in the field. Moreover, the objective has been to find a direction for a suitable solution as well as some reference applications to define the scope of scalability. The related work has given the important concepts to build the implementation upon. The available technological possibilities to implement the concept are studied in the next chapter.

3. Web-based technologies on mobile devices

In this chapter the technological possibilities are examined and discussed from two viewpoints. First, the available Web technologies and the possibilities for the application development for mobile devices are examined. Secondly, the Web technologies applicable in the mobile environment and the possibilities for using these technologies in today's mobile devices are discussed.

In the recent development of lightweight Web technologies and Web service models as well as the development of mobile devices and Web runtime platforms, the Web seems to be the most versatile and promising platform for new applications available at the moment. It has been said that the Web will be the platform for most of the application development in the future [Taivalsaari et al., 2008]. Already today some of the biggest and most complex applications and systems are Web-based. On the other hand there are many issues to be settled, mainly concerning the performance of applications as well as the communication.

The Web is not only an application platform but a social phenomenon as well. The Web 2.0 and social networks have accelerated the development of Web-based applications. The term Web 2.0 was introduced by O'Reilly Media [O'Reilly, 2005]. Although being a hype term used mostly in the marketing world, Web 2.0 has some distinctive characteristics that can be defined. The core principals of Web 2.0 and the era of Web applications are to:

- Treat the Web as an application platform
- Encourage user generated content and participation. [O'Reilly, 2005].

The power of Web applications is that they are very interoperable and extendable. They can leverage from the vast amount of data available on the Internet. Web applications often provide additional APIs to achieve the extendibility and interoperability. These APIs provide the functionality and data that the Web application clients need. One way to leverage from these Web services are specific purpose Web clients or so called Widgets [Caceres & van Kesteren, 2007].

Widgets are lightweight, Web technology based applications which are designed for a single specific purpose and quick instant access to Web services or Internet content as a whole. In the recent development of mobile applications the mobile Widget engines or Web runtime have been emerging [Kaar, 2007].

The mobile devices provide a mobile user context to enable a wide variety of user scenarios and situations as well as an ability to enhance the services with context awareness available from the devices [Koskela et al., 2007]. This context-awareness brings more value especially to the user participation principal present in the Web 2.0 ideology. One example of context information that has recently received a lot of attention is the location of a user, but there are many others to be used as well. Furthermore, mobile devices provide a wide variety of personal information management items such as calendar entries and contact information that can be useful to share.

3.1. Overview on technologies for mobile devices

The extending amount of available mobile platforms makes it sometimes impossible to develop an application that would run even on every mobile device or even on the devices with similar capabilities. This might lead into a situation where some of the mobile software development shifts from the installable, native applications into more runtime interpretable and Web-based applications. Of course the need for native applications will remain on the basic level of the operation system, as well as for the critical functionalities and basic information management applications. In addition, there might be an up rise of additional scripting or Widget type of applications that leverage from the native information management and capabilities and perhaps combine information from different sources into a compound and a more specific purpose. The available Runtime environments on the Symbian S60 3rd Edition operating system are illustrated in Figure 6.

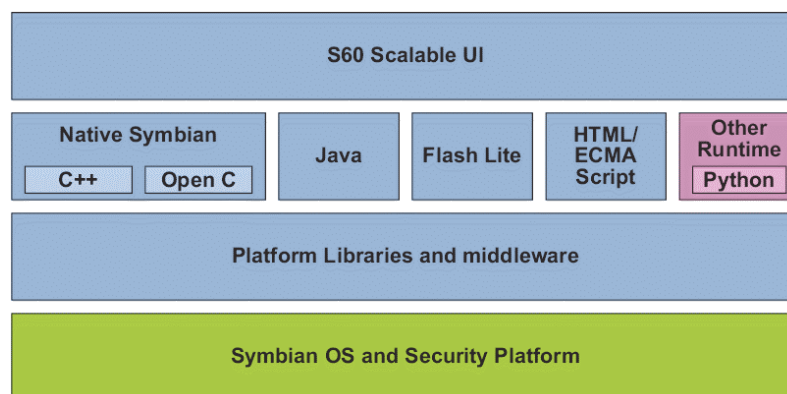


Figure 6. Application Runtime Environments for S60 3rd Edition [S60, 2007]

As the Web services develop, the use of Web runtime as an application platform on the mobile device becomes more and more important. The Web runtime provides an interface and communication capabilities which are, at least in theory, device and platform independent. The Web technologies also

enable easier and faster user interface development as well as a scalable user interface structure than the traditional software techniques.

3.2. Web architecture

In this thesis the Web is considered as an enabler of communication using simple and flexible protocols and architectures. The Internet on the other hand is the network platform for the Web and thus enables communication on the network level. The use of Internet as an extensive source of information is not the goal here, merely the networking abilities it provides. On the other hand the flexible nature of the Web architectures enables the extendibility of the services as well. To have a common background on the terminology used, some of the terms are explained here.

First of all, the World Wide Web Consortium (W3C) states that:

“The Web is an application built on top of the Internet” [Jacobs, 2008]

The Internet on the other hand is architecture with many layers and protocols. The World Wide Web or just Web is one of those application layers of Internet, mainly using the HTTP protocol. The Web architecture is conceptually a network of interconnected resources, which can be documents or other media. An approach by W3C has been to define the architecture of Web from three bases [Jacobs & Walsh, 2004]. These are:

- *Identification*, meaning the identification of resources by URIs [Berners-Lee et al., 2005].
- *Interaction*, meaning the interaction with the resources by the applicable methods of HTTP: *GET*, *POST*, *PUT* or *DELETE* [Fielding et al., 1999].
- *Data formats*, meaning the use of open data formats for mark up, such as HTML [Pemberton et al., 2000], CSS [Bos et al., 1998] and XML [Bray et al., 2000].

These three bases give the conceptual basis on the development of Web services and applications. Although being a very simple architecture from an overview perspective, there is a wide variety of interfaces and mark up languages on top of it. The challenge here is to keep it simple.

3.2.1. Web technology stack

The Web architecture is inherently client-server architecture. Therefore, the Web technology stack is also twofold. The client part of the Web architecture can be a Web browser, a Widget or any client software able to communicate with the server using HTTP. The HTTP protocol gives the basic interface methods that describe the communication, these being the *GET*, *POST*, *PUT*

and *DELETE*. The communication is usually stateless and the request – response cycles are synchronous.

From the user interface point of view though, the client can communicate with the server also by using asynchronous communication or Ajax (Asynchronous JavaScript and XML) [Garret, 2005] as it is commonly named. This means that the UI is not locked during the request to the server, this being especially important in the Widget applications. The interface language used in the communication with the server can be XML-based language [Bray et al., 2000] or a more logic oriented language like JSON (JavaScript Object Notation) [JSON.org, 2009]. The server part of the Web architecture is responsible for delivering and storing the content for the Web application, thus it can be called the Web service provider. The most important architectural aspect in the Web service implementation is the structure of the interface it provides for the clients.

In the recent development of Web architectures there has been a shift in the direction from a *multi page interface* model into a *single page interface* model [Mesbah & Deursen, 2007]. The new approach leveraging from the Ajax communication [Garret, 2005] provides more interactive applications. This means that the Web interface is more modular, as if built from smaller units. Therefore, the architecture should be more modular as well. In an Ajax type Web application the interface consists of dynamic elements which are updated when needed. This approach can be implemented through the original Web architectures as well, by treating the interface objects as resources that are fetched and updated from the server and vice versa.

In comparison to conventional application design, Web-based applications are more focused on the content and information and not so much on the computational abilities. The key concept of Web-based applications is that they provide a uniform, extendable interface to a specific content that can be created by the user.

3.2.2. Web runtime environment

The Web browser has been the sole platform for Web applications for a long time. This has been changing though with the recent development of Web runtimes or Widget engines as they are more commonly named. Web runtime is a runtime for locally installable Web applications also called Widgets. Web runtimes are usually capable of rendering HTML and CSS as well as running scripting languages like JavaScript, but some available engines use their own proprietary XML-based languages for building the Widgets.

Since these Widgets are installed locally, the basic content structure and representation of the Widget is static. On the other hand these Widgets usually

depend on the dynamic Web content. This has the consequence that the use of dynamically changing DOM (Document Object Model) [Le Hors et al., 2000] structure and Ajax [Garret, 2005] communication for updating that content is a necessity in the Widget development.

Moreover, these Widgets are lightweight applications which are usually developed with basic Web technologies. They are designed for a specific purpose and usually access the content from the Internet using Web 2.0 API's and techniques such as the Ajax [Kaar, 2007]. Although capable of accessing Web content, they can also provide local data, such as time, calendar or contacts. Some of the available Widget engines from major software and device vendors are listed in Table 1. The main differences between the engines are the naming conventions and the used mark-up languages for the development. Four out of the six mentioned engines use standards based HTML, CSS and JavaScript for the development, so these should be interoperable. For example, the porting of Apple Dashboard widgets onto the latest Nokia S60 devices is fairly straightforward [Kaar, 2007].

Engine	Manifest file	Extension	UI Markup	OS
Yahoo! Widget Engine	*.kon	.widget	Proprietary XML	Windows, Mac OS X
Windows Sidebar	gadget.xml	.gadget	HTML + CSS	Windows Vista
Google Desktop	gadget.gManifest	.gg	Proprietary XML	Windows, Mac OS X
Opera Browser	config.xml	.zip	HTML + CSS	Windows, Mac OS X
Apple Dashboard	info.plist	.wdgt/ .zip	HTML + CSS	Mac OS X
Nokia Web Runtime	info.plist	.wgz	HTML + CSS	Symbian S60

Table 1. Available Widget engines from major vendors

The W3 Consortium specification defines Widgets as following:

“Widgets are a class of client-side Web applications for displaying and updating local or remote data, packaged in a way to allow a single download and installation on a client machine or device.” [Caceres, 2008]

This definition does not limit out the use of proprietary XML for the development, although the use of open standards should be encouraged by W3C. The proposed technology stack for the Widget development by the W3C is illustrated in Figure 7, which depicts the variety of technologies and standards involved in this development. This specification still has a *Working Draft* –status though [Caceres, 2007].

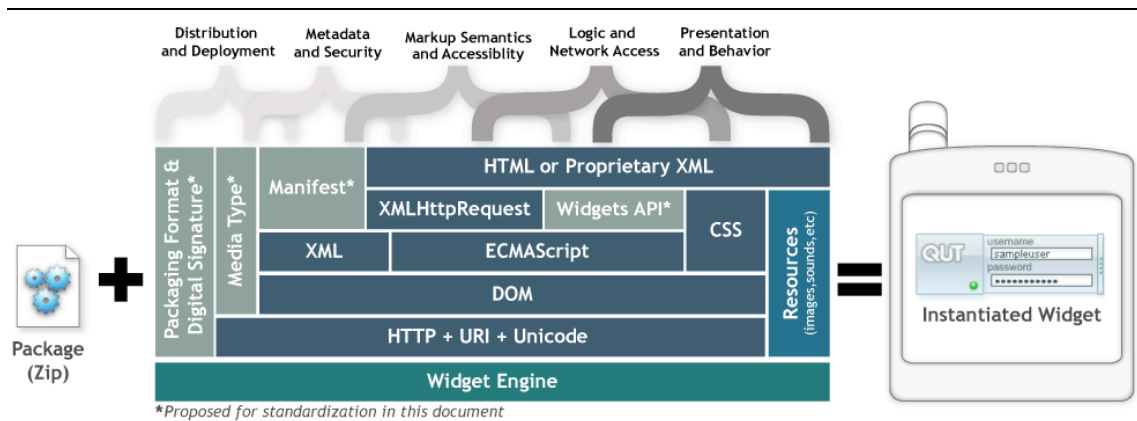


Figure 7. Technology stack for Widget development by W3C [Caceres, 2007]

3.2.2.1 *Web runtime on a mobile device*

The mobile Web runtimes or Widget engines have been around for a while now. The first Widget engines were implemented as Java ME applications capable of running Widgets written in dedicated, proprietary XML languages. This has had the implication that the Widgets are not installable as stand alone Web applications but are run inside the Widget engine application. The first real Web runtime for a mobile device was introduced to Symbian S60 provided by Nokia. It enables the development of Widgets in standard Web technologies, more specifically HTML, CSS and JavaScript. It is also more integrated with the operating system, thus it enables installing and running stand alone Widgets [Kaar, 2007].

The S60 Web runtime was introduced to Symbian S60 platform to the 3rd Generation Feature Pack 2 version [S60, 2007]. It is based on the S60WebKit Web framework, the same framework that the S60 Web Browser is based on (cf. Figure 8). The S60 Web runtime provides the same basic functionality as all the WebKit-based Web runtimes and browsers. This enables the interoperability of Widgets in a variety of devices that support the same technologies and interfaces. It also eases the porting of Widgets to other devices and platforms [Kaar, 2007]. Moreover, the Web runtime provides some additional API's to the device specific capabilities and resources. In the 5th Generation S60 platform the additional API for Widgets is extended from the previous release to enable more access to the device's capabilities.

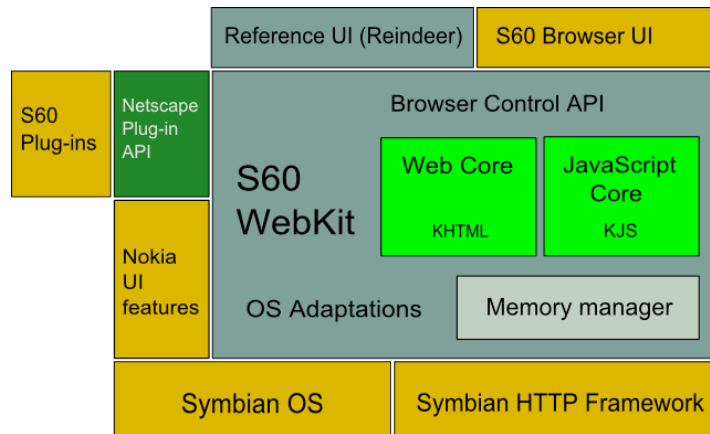


Figure 8. Web framework for S60 platform (adapted from [Nokia Research, Center, 2009])

Capabilities of Web runtime environment in Symbian S60 platform:

- S60 3rd Generation [S60, 2007]
 - Widget specific UI functionality (i.e. transitions)
 - Access to some platform information
 - Ability to launch native applications
- S60 5th Generation [S60, 2009b] (Added functionality to platform capabilities and resources)
 - The application manager
 - Access to Calendar records, Contacts records, Log information and Media gallery
 - Access to Device's location, Landmarks and Sensors
 - Access to some of the system information

The capabilities provided by the platform are likely to increase in the future, thus this development environment provides an interesting platform for future Web-based and Web-enabled applications.

3.2.2.2 Mobile widgets as task-specific Web service clients

The Web is full of small applications. There are various Web sites and portals that host small Web Widgets for multiple purposes; one of these services is iGoogle [Google Inc., 2009c]. iGoogle is a Google front page embedded with small Web Widgets that provide for example, weather information, news headlines, calendar events and email notifications. Another trend emerging is the mash-ups of different Web services, for example location specific data overlaid on a map service [Wikipedia, 2009d]. Some Web sites even provide their own application development APIs, one of the most successful has been

Facebook [Facebook, 2009]. The Web is full of small applications designed for a single purpose. Will mobile Widgets fit into this emerging development?

The main point of using Widgets as clients for Web services is that the Widgets are designed to enable a single specific functionality that is the initial elegance and purpose of them. The other important feature is the strong relationship or dependency on Web technologies. They provide an interface to plug into a Web service API and thus to enable a specific functionality for a single set of tasks. Another important issue is the fast and relatively easy development of Widgets, which gives the possibility to develop various specialized, single purpose clients for the needed tasks with relatively low development costs. Furthermore, they encourage a wide developer community, also called the Long Tail development [Anderson & Andersson, 2007].

3.2.3. Mobile Web server

The Web server is responsible for delivering and storing the resources and thus providing the Web service to the clients. The Web service architectures and interface implementations were discussed in Subsection 2.4.2, thus the emphasis here is more on the technical implementation and the available technologies for the mobile environment.

To be able to serve Web content, the Web server has to be online, in other words connected to the Internet. This might sound obvious and relatively easy to achieve, but in the mobile environment the connectivity is often an issue. Thus when using a Web server on a mobile device one has to take into account its own unique problems.

Mobile devices are becoming more and more powerful computing devices and they are capable of running similar applications to PCs a couple of years ago. Wikman et al. [2008] have introduced a commonly used Web application development stack which is installable on a mobile device. More specifically they have introduced a port of an AMP stack (Apache, MySQL and PHP/Python) for Symbian S60 platform. Moreover, this Apache port for S60 has been released as an open source project as well as a proprietary package including a gateway service to enable connection from the World Wide Web [Ilkka & Vainio, 2007]. The network architecture for the Mobile Web server is illustrated in Figure 9.

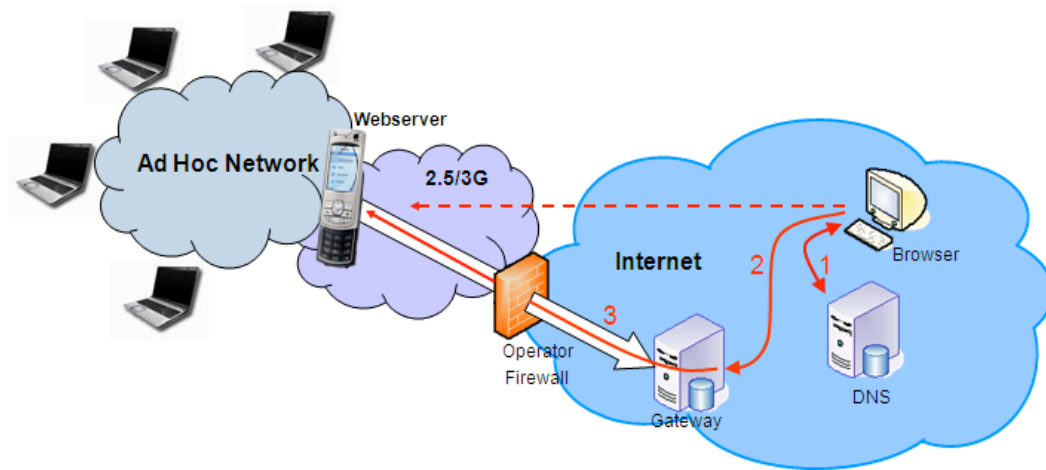


Figure 9. Mobile Web server network architecture [Mobiforge, 2009]

The use of a Web server on a mobile device might sound slightly awkward or inefficient at the very least. Whilst the latter might be true in the traditional way of using a Web server, other usage scenarios can be more relevant for Web servers on mobile devices. These use cases could include ad hoc networking and mobile situations where other communication is not available, or the use of context and personal information stored on the mobile devices. The content served by the mobile Web server should also be more modular, lightweight and context-adapted. The use of a mobile web server is part of the concept of a mobile peer-to-peer collaboration tool introduced in the next chapter. It formalizes the proposed approach more in order to gain understanding on the possibilities and constraints it may have.

4. Concept architecture

In this chapter the main focus is in describing the concept approach for sharing content and context information from an individual mobile device. When used among a group of people this concept approach makes the collaboration between people possible using mobile devices. This concept is a combination of several technical and theoretical approaches and techniques described in Chapters 2 and 3. More specifically this approach tries to adapt the work done on the field of Web services and architectures to be used in a mobile environment enabling the communication and collaboration means investigated in Chapter 2. From a technological point of view the approach tries to capture the latest development on mobile Web technologies and techniques as well as harness those possibilities. Moreover, the concept does not imply a single development environment or a technological solution, but it does describe some particular implementation aspects in order to make the approach more concrete.

This chapter will give an explanation of the core concept, described by some key characteristics. It gives an overview on the background of the selected approach and describes the requirements for the architecture. In more detail, it applies the resource oriented Web-based architecture on a small-scale for mobile devices in a distributed environment.

4.1. Background

The emergence of Web technologies on mobile devices has happened very quickly, continuing from the mobile Web browsers capable for “*real Web content*” into integrated Web runtimes that enable the access to the device capabilities as well. Moreover, the Web has become one of the key development platforms for applications that require distributed content and communication. In addition, there is a continuous standardization of Web technologies by a wide collaboration of corporations and institutions, which will keep the development going in the future as well.

The key drivers of Web technologies are as follows:

- *Wide developer community* – there is continuously growing amount of developers using Web technologies
- *Interoperability* – using common and accepted protocol increases interoperability

- *Open standards* – the Web is built on open standards and widely accepted conventions
- *Platform independent development* – the Web applications are developed on top of a “Web Runtime” environment providing platform independence.

The mentioned background and key drivers are the main reasons why the Web technologies have been taken as the target technologies for the design and implementation of this collaborative platform approach for mobile devices.

4.2. Overview

This section provides an overview on the system and how it can be perceived. One way of describing the approach is to divide it into logical components which characterize the approach. In general, the system could be described as a distributed client-server model [Berson, 1996]. This means that there are possibly several clients and servers interacting with each other when using the system. The core of the proposed concept is to use the available information resources of a mobile device in a collaborative work by using mobile Web services in a *person-to-person* manner. Each person will provide their own information through a mobile Web service. This Web service can then be used by a group of users in a collaborative work. The communication is done in a peer-to-peer manner by accessing the individual content providing servers directly using their IP-addresses. While the communication and addressing issues are a very important aspect on this approach, they are not the main focus on this thesis. Existing solutions and technologies are used when possible, thus building the work on earlier research and development.

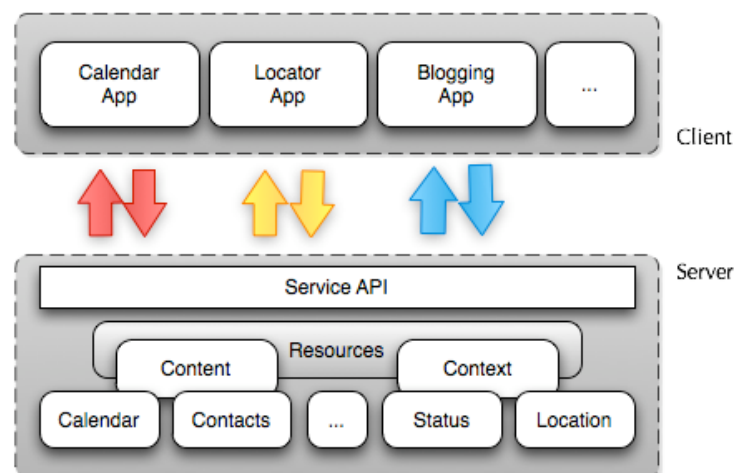


Figure 10. Conceptual overview – the distributed service instances (servers) provide the collaborative information for the users of the service (clients)

The conceptual overview of the system is depicted in Figure 10. It illustrates the logical components of the system and how the information is flowing. The main elements are the client-side applications that leverage from the common, server-side API. The server-side API provides the access to the device resources, such as the location, contacts, documents or calendar events. This kind of separation is needed for the distributed use of resources, for example for collaboration tasks. The separation also makes it possible to keep the service interface uniform, thus making it possible for the server- and client-side to evolve independently.

4.3. Characteristics of the concept

The system has some distinct characteristics that define the design space as well as the functionality and features it provides. These characteristics serve both as the design goals as well as the defining aspects of the concept itself. These characteristics are inspired by various other studies on the fields of mobile software development and Web service architectures, trying to combine ideas from both. Moreover, these characteristics are meant to be general enough but still descriptive to define the goals for this approach. These key characteristics are listed in the following four subsections.

4.3.1. Resource oriented server interface

First of all the concept approach is built upon a unified Web Service API structure that provides the needed resources for the client applications. This API is designed to comply with the resource oriented ideology of the Web itself. Basically this means that the resources are identified and accessed using URIs and interacted with the basic HTTP methods: *GET*, *POST*, *PUT* and *DELETE*. This architectural style is the very basic architecture for any Web-based system and can be applied quite easily for many implementations [Fielding, 2000]. It provides a lightweight and scalable framework for the server interface as well as a design model that fits well to the application area.

Another key aspect of the resource oriented approach is the uniformity of the interface, since the interface only consists of the available resources and the basic methods for interaction, the both sides (client and server) can evolve independently from each other. In a distributed and possibly very rapidly developing and changing environment the uniformity of the interface is an important aspect in the development.

The design of a resource oriented interface might be trivial in a simple and concise case, but with a more extensive set of resources the design will become an issue. Therefore, it is important to have a common, understandable

procedure when transforming the information from the application requirements into a resource oriented interface. The design and development of a resource oriented server interface from the requirements through an information model into the resource model is discussed later in Subsection 4.6.1.

4.3.2. Lightweight client for a task-specific purpose

In order to make use of the resources provided by the server interface, a client application is needed. This concept approach does not imply any particular client-side technology for the development of these *micro-sized Web service clients*. On the other hand, to give an example of a possible technology to be used, the mobile Widgets [Kaar, 2007] are introduced as an example.

Widgets are lightweight Web applications developed using the common Web techniques such as HTML, CSS and JavaScript [Caceres, 2008]. These Widgets are designed to be used for a specific purpose and often rely on an access to Web services. Because the Widgets are designed for a specific use they can maintain a lightweight and simple user interface structure and the user interaction can be kept to a minimum. This is essential in a mobile environment taking into account the various usage situations as well as the available screen space for the user interface.

Although this concept of *lightweight Web service clients* was inspired by the development of Widgets for mobile devices, these clients can also be developed using several other technologies and programming languages. These include, for example, simple native applications or Flash based applications. The only requirement is that it can use HTTP connectivity to enable the communication within the framework. In any case the main idea is to provide a client interface for a simple task, which can benefit from the ability for a group communication. These simple tasks could be characterized as “*micro services*” that provide an interface as well as the service back-end to enable a decentralized use.

In addition to the computational lightness the client applications should provide an easy and fast user interaction. Mobile devices are often used on the go and thus the user is required to pay attention to multiple things at the same time [Oulasvirta et al., 2005]. This requires the mobile applications to provide fast and simple user interaction. The single task-oriented approach also suits this requirement.

4.3.3. Fast and efficient development

In order to enable enough capabilities the system must have several lightweight, special purpose clients that connect to the resource oriented API,

to retrieve and save data. The main goal for the development of these lightweight micro service clients is that they are fast and efficient to develop as well as to use. The user interface and functionality can be optimized for a specific use, thus the interface and interaction design is easier and quicker to test and evaluate.

Another aspect supporting this characteristic is the use of Web technologies or other fast development technologies for the client-side development as well as the use of traditional Web server back-end technologies such as Apache Web server. This enables a faster learning curve and can attract many traditional Web developers to develop applications for mobile devices [Ilkka & Vainio, 2007; Jones, 2007]. The trend of using Web techniques for stand alone applications is fast growing and so is the development of the platforms for those applications.

4.3.4. Platform independence and interoperability

The concept approach itself can be seen as platform independent because it is conceptually Web-based. The only prerequisite is that the platform serves the possibility for communication over the Internet using HTTP. An even more important feature than platform independence is the interoperability between different devices. For example, a useful scenario would be to enable the use of desktop user interface to access mobile device data and capabilities, such as calendaring and messaging capabilities [Ilkka & Vainio, 2007]. By using existing Web standards for both client- and server-side implementations as well as for the communication; the same distributed application can be used on different platforms and devices.

4.4. Design objectives

Design objectives guide the selection of a solution from the number of available possibilities [Coley Consulting, 2009]. It is done by listing the desired features and ordering them by importance. This helps to choose the platform, architecture, and the technology used. Today, there exists many different approaches in software development and this number is more likely to increase than decrease in the future. By ordering the desired features, the trade offs and possibilities may be more visible in the design of a system.

In this thesis the design objectives are used to guide the decision from approaches researched earlier. This section will give the basic objectives that have lead to the decision of architecture and platform.

Objectives:

1. *Extendable* – the approach should be extendable for several uses.

2. *Lightweight* – The approach should be enough lightweight to be used with mobile devices.
3. *Fast and effective interaction* – The approach should enable fast and effective interaction, in order to be usable on a mobile device.

4.5. Requirements for the architecture

The architectural requirements specify the requirements for the overall system. These can be thought to be the requirements for the concept in case. In more detail, these specify the requirements for the overall use of the system spanning several application instances. These architectural requirements are adapted from the design objectives described on the previous section. The objectives are taken into account when deciding the architectural requirements and the implications of those decisions. The architectural requirements are also divided into server-side and client-side requirements.

First of all the system should have an *extendible architecture*. This basically means that the framework should be usable for several collaboration tasks, such as the sharing of calendar events, documents and items et cetera. In addition, the architecture should be interoperable between different types of devices. This requirement should be taken into account when deciding the server-side architecture.

Secondly the architecture should be *lightweight*. This could mean a couple of things. It is very desirable that the system does not use extensive amounts of device resources, but this is more of an implied requirement for any system or application designed for a mobile device. In this context the *lightweight* architecture means that the server-side architecture should be implemented in such a way that it does not require any overhead processing to serve the requests, thus the interface implementation should be as straightforward as possible.

Thirdly, the architecture should enable *fast and effective interaction*. This requirement specifies the behaviour of the system more than the architecture, but since it relates very closely to the client-side architecture and implementation, it is listed here as well. On the other hand this should be taken into account more when designing the individual client-side applications for the services. All of the mentioned requirements are taken into account when designing the architecture for the system.

4.5.1. Requirements from existing work

In addition to the mentioned general requirements based on the design objectives there are some more specific requirements derived from earlier

research. Since the concept can be characterized as a *mobile collaboration platform* based on *Web service architecture*, the requirements also have to be gathered from these two research domains: *mobile collaborative systems* and *Web service architectures*. Since there is earlier work on defining the requirements for mobile collaborative systems, these are used as a basis for the requirements introduced here. In addition, the use of Web services for collaboration requires taking into account the work of W3C on Web service specifications.

These requirements for mobile collaborative systems are adapted from the list of requirements for nomadic team-working introduced by Reif et al. [2001].

These requirements include:

- **RMCS1:** *Knowledge sharing* – the users may act as providers and consumers of information on the system
- **RMCS2:** *Device-independence and interoperability* – the system has to support the use different devices and platforms
- **RMCS3:** *Communication and collaboration* – the system has to support the use of different modes of connectivity, these being: connected, disconnected and ad hoc connectivity's and different means of communication, such as voice and text-based messaging.

In addition, the requirements for Web services architecture are adapted from the W3C specifications for Web services architecture [Austin et al., 2004].

The requirements for Web services architecture (WSA) are:

- **RWSA1:** *Interoperability* – the WSA should enable the development of interoperable Web services
- **RWSA2:** *Reliability* – the WSA must be reliable and stable over time
- **RWSA3:** *Integration with the Web* – the WSA must be consistent with the evolution of the World Wide Web
- **RWSA4:** *Security* – the WSA must provide a secure environment
- **RWSA5:** *Scalability and Extensibility* – the WSA must be scalable and extensible.

4.5.2. Requirements for mobile peer-to-peer collaboration

The requirements listed above can be used to conduct a list of requirements for a mobile peer-to-peer collaboration platform using Web services. These requirements are thus derived and adapted from the requirements introduced earlier by Reif et al. [2001] and Austin et al. [2004] for this specific use, and to be used as a reference in future use of this work.

The requirements for mobile peer-to-peer collaboration are:

1. *Information sharing in a peer-to-peer manner* – the users may act as providers and consumers of information while using the system and the information is used in a peer-to-peer manner
2. *Device and platform interoperability for services* – the system has to support the use of different devices and platforms and should enable the interoperability of the developed services among different platforms
3. *Support for different communication types* – the system has to support the use of different modes of connectivity, these being: connected, disconnected and ad hoc connectivity's and different means of communication
4. *Reliable and secure platform for collaboration* – the platform must be reliable and stable over time and it must provide a secure environment for the collaboration
5. *Scalable and extensible platform to enable different services* – the platform must be scalable and extensible to enable several collaboration scenarios
6. *Integration with services in the World Wide Web* – the platform should provide a consistent way for integration with the services in the World Wide Web.

The above mentioned requirements are taken into account when designing the architecture and when implementing the proof of concept prototype. In addition, the concept architecture is assessed towards these requirements in the evaluation chapter of this thesis.

4.6. Architecture

The basic architecture consists of a client and server applications used with a mobile device. A user providing a service will have a server application running on their mobile device. In addition, each user who accesses these services through a client application, for example a Web browser or a Web widget. To put in short, the architecture is a distributed system between the peers consisted of several clients and servers interacting with each other. The amount of peers in the group can be relatively small, depending on the use. Although the architecture is ordinary client-server architecture it is kept very lightweight in order to be usable with the scarce resources of mobile devices. The overview on the architectural components is illustrated in Figure 11.

The most important aspects of this architecture consist of a resource oriented server interface that provides services to access the device capabilities. The client-side applications are designed for a specific, task-oriented purpose and thus are lightweight and fast to interact with. They provide the means to

access and interact with the server interface in an asynchronous, interactive way. More details of the server and client-side architectures are given in the following subsections.

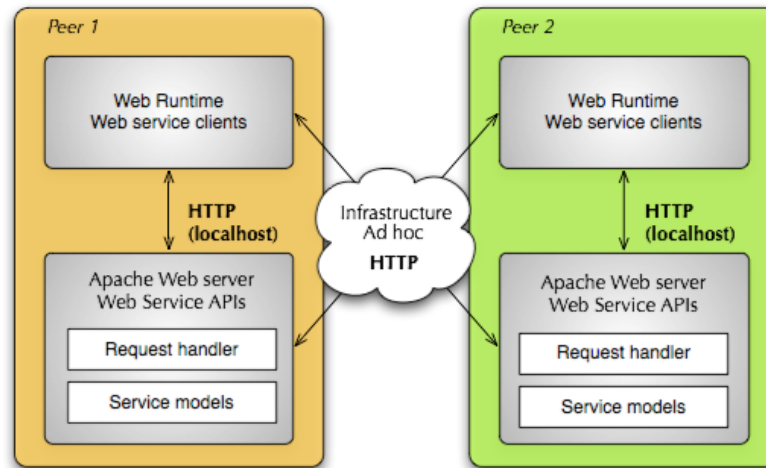


Figure 11. Architecture overview – each user (peer) can both provide and consume the collaborative information

4.6.1. The resource model

This subsection discusses the implementation of a resource oriented API. The reasons for this approach have been discussed in Subsection 4.3.1. In order to implement the server-side API in a resource oriented way, the needed information has to be modelled as resources. This basically means a transformation from information model acquired from the use cases into a resource model, describing the structure of the interface. This can be done by Transformation-Assisted Design [Siikarla et al., 2008].

Basically the resources serve as the abstractions of any relevant information provided by the service interface. The relevant information is derived from the requirements of the application. When transforming this information model into a resource model several key aspects should be taken into account. Adapted from the work by Siikarla et al. [2008], these key aspects can be listed as following:

- *Requirement covering* – the model has to cover all the requirements
- *Simplicity and intuitiveness* – the model should be simple enough, so that the resource structure does not get too granular or the hierarchy too deep
- *Constraint compliance* – the model and the resulting interface have to comply with REST constraints [Fielding, 2000]

- *Precondition coverage* – the aspects not expressible by resources should be taken into account as well, such as the semantics of different parts of the model
- *Aesthetics for human users* – the resources should be expressed by the terms of the particular model instance, using its vocabulary, for making a “human understandable structure”.

The transformation from the information model into the resource model is described above. That is an essential part of implementing a RESTful service interface. To put in a simple way, the RESTful service interface means an API of resources identified as URIs [Fielding, 2000]. Although the main concept is very straightforward, there are many different possibilities for the implementation of such server interface. An example of a RESTful interface for a mobile micro blog application is given in Table 2. The details for the implementation of a RESTful interface are discussed in the next subsection.

Resources/Actions	GET	POST	DELETE
Entries	http://<host>/<app>/entries/ => return the list	...entries/ => return entry id	N/A
Single entry	...entries/x/ => return the entry	N/A	...entries/x/ => return success
Comments	...x/comments/ => return the list	...x/comments/ => return comment id	N/A
Single comment	...entries/x/comments/y/ => return comment	N/A	...x/comments/y/ => return success
Video	... entries/x/video/ => return video	... entries/x/video/ => return success	... entries/x/video/ => return success
Photo	... entries/x/photo/ => return photo	... entries/x/photo/ => return success	... entries/x/photo/ => return success
Additional data	... entries/x/data/ => return data	... entries/x/data/ => return success	...x/data/ => return success

Table 2. Example of a RESTful API for a mobile micro blog

4.6.2. Server-side architecture

In this subsection the implementation details of the server-side architecture is discussed. First of all the server-side architecture must implement the RESTful interface as it is mentioned above. In addition, it must provide the needed resources and to respond to the resource requests using a compliant HTTP protocol.

From the functionality point of view the server is responsible for delivering the requested resources to the client applications. The server acts as a true Web server, but it is only used to access and save the resources. The user interface presentation and logic is mainly on the client-side reducing the need for communication significantly. The architecture consists of a *request handler* that recognizes the resource requests and the method to be executed and forwards it to the *service model*. The *service model* executes the action and sets the response accordingly. The *service model* uses the native device databases to store the content and device capabilities to access or capture the content.

One of the key aspects of the server-side functionality is the request handling. Since the Web architecture is originally stateless it relies merely on the request-response cycles. This improves scalability and keeps the server implementation lightweight since it does not have to recall client states. This is important in a mobile environment.

The *request handler* is responsible of handling the RESTful interface, basically meaning the resource requests. The most straightforward solution is to divide this handling by the method of action (*GET*, *POST*, *PUT* or *DELETE*). After the action is selected, the resource must be identified and the resource representation acquired from the model instance. The interface methods and actions are given in Table 3. This kind of approach is still feasible on a small scale interface, but when developing a large Web service this might result in a cluttered design. On the other hand, the possibility for using a RESTful architecture is already provided in many of the Web server backend frameworks such as Ruby on Rails [Ruby on Rails, 2009]. Thus “self-made” controllers are rarely needed in the traditional Web development. In this case it is reasonable to apply the concept on a very hands-on basis, so that it is possible to spot any kind of bottlenecks that the design may have for future improvements.

Controller method	Action	URI	HTTP Method
get(request)	show item 1	<host>/item/1	GET
post(request)	create an item	<host>/item/	POST
put(request)	update item 1	<host>/item/1	PUT
delete(request)	destroy item 1	<host>/item/1	DELETE

Table 3. RESTful interface methods

4.6.3. Client-side architecture

In this subsection the main emphasis is on the client-side architecture developed using Web technologies. The key issues on the client-side include

the user interface and interaction as well as the asynchronous communication with the server-side interface. The client-side architecture reflects these key aspects with the use of existing JavaScript toolkits to ease and enhance the development of the client-side logic.

The platform for the client-side architecture is the Nokia S60 Web runtime [S60, 2009c] which has been developed based on the open source WebKit [The WebKit Project, 2009], thus it enables the use of Web technologies such as HTML, CSS and JavaScript for developing the Widgets. The communication with the Web server is done asynchronously using XML or JSON as the data serialization format. This approach eases the amount of data transfer traffic and keeps the user interface as simple as possible. One of the important aspects on the development of interactive Web applications is the use of JavaScript as the engine for responding to the user inputs as well as for timing and communicating with the server-side API.

Several JavaScript toolkits are available for use when developing Web applications. These toolkits wrap the common procedures and functionalities and provide an extensive set of features. The most important functionality that is needed for a JavaScript toolkit is to provide the means to manipulate and traverse the DOM (Document Object Model), to bind event handlers to events, both programmatic and user generated, as well as to provide a wrap-up of common Ajax requests. The most commonly used JavaScript toolkits at the time of writing are: jQuery [jQuery, 2009], Prototype [Prototype (JS), 2009], Yahoo UI (YUI) [Yahoo! Inc., 2009], MooTools [Newton, 2008] and Dojo [Dojo Toolkit, 2009], although there is an extensive amount of other libraries as well as plugins to extend these core libraries.

It seems now that the most extensive and popular toolkits are gaining momentum and that the diversity is diminishing slightly, but there are still various different toolkits available and it seems to continue to be that way in the future as well. One big step for this “battle” among the different toolkits has been the announcement that Microsoft and Nokia are embedding jQuery as a part of their products and development environments as well as starting to contribute to the development of jQuery toolkit [jQuery, 2009].

The different JavaScript toolkits have some differences as to what comes to the performance [Pervilä, 2008]. These are mainly about the interoperability issues between different browsers and how well they support JavaScript, but in some cases it has an important affect on the overall usability of the application. Most of the performance differences are due to the different JavaScript engines integrated with the browsers and thus there are no real differences between the performances of different JavaScript toolkits. The other key aspect of the

JavaScript toolkits is the size of the library, since it is loaded along with the webpage while using an ordinary web browser. This aspect is not an issue though when talking about Widgets, since all of the code is already stored on the client-side.

4.6.4. Communication

Communication and interaction in a distributed system for collaboration has to be fluent and robust. The distributed nature of the content on the system relies on the asynchronous communication between the application instances. The communication is very often transparent to the user, and therefore, the state of the service can be unknown if the interaction and user interface has not been carefully designed. Since the system relies on the content distributed from the peer users, the whole value of the service is dependent on the content distribution and communication.

Communication capabilities or collaboration modes enable the communication via different communication channels. In the present concept system the available communication channels are all the available channels that support HTTP connection. These include for example the use of 3G and Wi-fi connectivity if present. Reif et al. [2001] list preferable collaboration modes for mobile collaborative systems. These include ad hoc, connected and disconnected modes. In the presented concept all of these modes are supported. The ad hoc connectivity can be enabled by an ad hoc Wifi connection, while the connection to the Web can be done via 3G packet data with a suitable gateway solution. There exist though several difficulties when using an ad hoc communication between mobile devices. These will be discussed more in Chapter 7.

4.6.4.1 Infrastructure

When an infrastructure connection like 3G packet data is available it should be used for the communication. For infrastructure connectivity, a gateway solution is needed to overcome the problems encountered by the changing IP address in a wireless network. While there are network layer solutions to overcome this problem, such as the Mobile IP, they are not available yet on commercially available wireless networks. In this work the solution includes a gateway and a DNS resolver provided by Nokia Research Center for their Mobile Web Server connectivity [Ilkka & Vainio, 2007]. This solution can be used already because of its availability as an end-user product.

4.6.4.2 Ad hoc

When an infrastructure connection is not available the communication should be done via an ad hoc connectivity. As the name implies ad hoc connectivity works without a network infrastructure. Thus the address resolving must happen in an ad hoc manner. To overcome the addressing problem the architecture can be used via a denoted *super peer*, which serves as proxy between the peers. This is a very simple solution, but it scales well to the proposed architecture as well as provides the needed functionality. Some problems were encountered though with the reliability of the connection in an environment with several other Wifi networks around. The architectural comparison between the ad hoc and infrastructure connectivity solutions is illustrated in Figure 12.

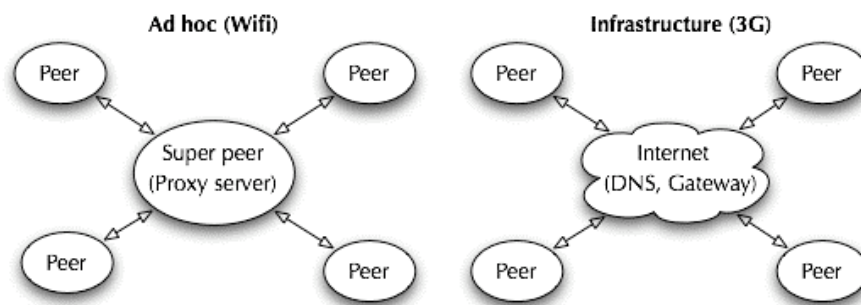


Figure 12. Ad hoc and infrastructure communication comparison

A possible group communication scenario is illustrated in Figure 13, including both personal group communication as well as communication outside the group. The communication and messaging between the application instances is described more in the following subsections. These include both the server-side and the client-side technologies.



Figure 13. Group communication – collaboration is done in small groups but the information can be shared to a wider context

4.6.4.3 *Ajax*

Ajax [Garret, 2005] is a technique using the asynchronous communication provided by the XMLHttpRequest –object with some data interchange format, mainly XML [Bray et al., 2000] to communicate with a Web server. By communicating asynchronously it helps to avoid page reloads and unnecessary data exchange, thus it makes the user interaction seem smoother and the communication more efficient. It also enables the development of Widgets, stand-alone Web applications which are installed on the client-side and only retrieve the content from the server-side. This content can be formatted using for example XML or JSON [JSON.org, 2009]. The comparison of these data serialization formats is given in the next subsection.

4.6.4.4 *XML versus JSON*

XML has been the de facto language to serialize data [Bray et al., 2000] for quite a long time, both in Web related applications as well as in traditional software. XML is used in a variety of interface solutions and data representation tasks because of its extendable and adaptable structure. It can represent very complex hierarchies and structures as a tree-type form. One disadvantage of XML though is the inefficiency when parsed to an object format for easier access. The extensive hierarchy makes it cumbersome and resource needy when parsing large XML-documents. In other hand, the advantage of XML is its uniform representation and format which makes it a very domain independent interface language.

JSON is a new approach deriving from the popularity of Ajax and JavaScript. JSON is a lightweight computer data interchange format. It is a text-based, human readable format that represents simple data structures as serialized objects or associative arrays [JSON.org, 2009]. The main difference to XML is that since the JSON is described as an associative array the name-value pairs have to be grouped by unique names. This means that the elements occurring more than once are grouped in an array under the name of that corresponding element [XML.com, 2009].

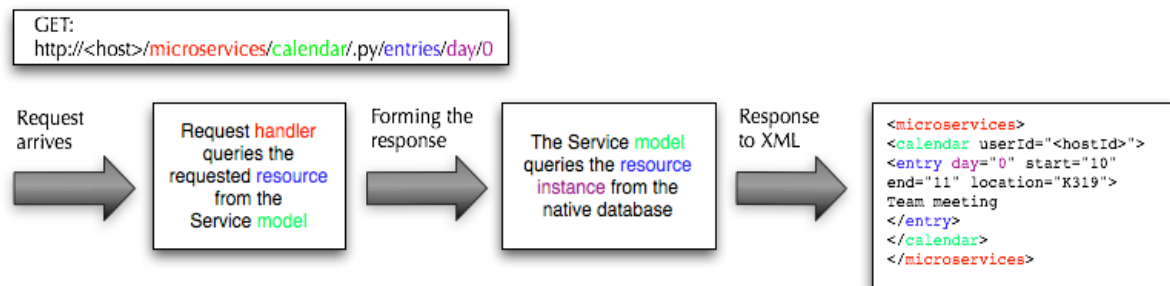


Figure 14. Communication procedure

The overall communication procedure is depicted in Figure 14. The procedure is started by a request from a client to the service to access a certain resource. Then that request is handled and a response is formed accordingly. In the end that response is sent back to the requestor, which will parse and make use of that resource.

4.7. Summary

This chapter has given the basis for the concept of a mobile peer-to-peer collaboration tool using Web service architecture. This concept includes the key characteristics relevant to the concept as well as the requirements and main building blocks for the architecture. As a summary, the approach uses lightweight Web service architecture for sharing native information resources in a scalable and interoperable way. The goal is to use existing technologies when possible and to design the solution to be portable for different platforms.

The architecture consists of a resource-centric Web service interface providing the native information resources in both ontological and machine-readable format for the client applications to consume. Due to the limited processing power on a mobile device, the architectural decision was made to use a resource-centric Web service architecture. This approach is most commonly described as RESTful Web service architecture [Richardson & Ruby, 2007]. The reason for the decision is that the resource oriented model is more

simplistic and tactical than a SOA approach, thus it is more suitable for an ad hoc Web integration (e.g. mash-ups) and lightweight Web service implementations [Pautasso et al., 2008]. It also fits well to the intended application domain.

Moreover, in order to make use of the provided information on the go, *task-specific mobile Web service clients*, such as Mobile Widgets [Caceres, 2007] can be used as the user interface for these collaborative services while used on a mobile device. In the following chapter this concept is taken into practice by implementing a prototype system using the concept architectural approach.

5. Concept prototype – Mobile collaborative scheduling

In order to evaluate the concept described in the previous chapter, an example service with two client applications was created. The goal was to implement a prototype by complying with the described characteristics. The main idea is to share the native, already existing resources collaboratively in a distributed manner. The task in this case is collaborative scheduling, which means a process of scheduling a meeting or appointment with a group of users gathering the required information automatically from the participants, in this case from the calendars of their mobile devices. To put it simply, this service is used to share the native calendaring events stored on the mobile device itself. The user scenarios, requirements and implementation of this prototype service are described later in this chapter.

5.1. Collaborative scheduling

The collaborative scheduling service enables the sharing of calendar events to be used by the client applications. The calendar events are accessed from the devices of the individual users and then combined to a single view. The calendar events are exposed to the client applications through a RESTful server interface. The server interface offers several different ways of accessing the information. Mainly there are two kinds of information available through the API: the *event* information and *availability* information. Therefore, it is possible to retrieve the events of a certain day or ask if the user is available at a certain time. For both information there are different use cases, thus two different client applications were implemented. Since there are several ways to access the same information, it is often important to model the server interface in such a way that it covers most of these possible use cases. The modelling of the server interface was discussed in Subsection 4.6.1. That approach has been used to model and remodel the server interface for this prototype service.

5.2. Requirements for the prototype

The procedure of gathering the requirements for the mobile collaborative scheduling service is described in this section. At first, a user scenario is presented as the basis for the requirements gathering. Then, the important use cases are derived from the scenario depicting the most important requirements. At last the functional requirements for the application are acquired from the set

of use cases to be the basis of implementation and evaluation of the prototype application.

5.2.1. User scenario

In this subsection, a user scenario is presented in order to capture the important use cases and requirements for the system. The user scenario is an example of the possible usage situation of a collaborative scheduling task. It has been formed from a discovered need for group communication from a research and development team in a research organization. The discovery was made by discussions with the team. This user scenario represents a subset of the findings from that discovery.

Scenario:

“Persons A, B, C and D are having a meeting. They do not want to have their laptops with them, because it would distract the meeting. On the other hand they have their personal mobile devices with them almost all the time. After the meeting the group needs to schedule their next meeting. Since the people in the group use their mobile devices for personal information management (i.e. synchronizing calendar events from/to their PC’s), the easiest way to do this is to take a look at their calendars on the mobile devices. To make this task even easier they all have a collaborative service on their devices for sharing their work calendars with each other. The director of the meeting has a client application that collects the events from the corresponding participants and displays them to him. Now he can see the available times for the next meeting and can post this new event to everybody in the group, straight to the personal calendars on their mobile devices.”

Other possible and interesting scenarios are in a home environment as well as in a group of friends. The personal information can be shared outside these groups, even publicly to everyone in the world.

5.2.2. Use cases

The use cases try to capture the behaviour of the system by extracting details from the user scenarios [Jacobson et al., 1992]. The use cases are one way of gathering the requirements for the system. The basic use cases gathered for the calendar application interactions are very simple and straightforward. The new approach that has to be taken into account is that the information sources are distributed and thus the communication between the peers is essential.

The basic use cases for the scheduling application are:

- Select a day or a week for the initial view,

- Select an appropriate time for an event from the basis of availabilities,
- Enter the details for the event, and
- Submit the event.

These use cases form the basic functionality for the collaborative scheduling application. To use these as the basis of the development, these use cases are transformed into requirements.

5.2.3. Functional requirements

The functional requirements specify the functionality of the application and act as reference points for the evaluation of the application in the end. These functional requirements specify what the prototype service should do as well as the behaviour of the client application. Functional requirements describe the behavior of the system, the inputs and outputs and how it works [Kotonya & Sommerville, 1998]. Functional requirements are gathered from the use cases, each use case implementing one or more functional requirement.

The following list of functional requirements must be fulfilled by the prototype service:

- *FR1*: The user must be able to select a desired day for viewing.
- *FR2*: The user must be able to select an appropriate time for the event.
- *FR3*: The system must provide a way to enter the event details.
- *FR4*: The user must be able to submit the event to the participants.
- *FR5*: The user must be able to see the system state at any given time.

The assessment of the prototype is done towards these requirements in the evaluation chapter of this thesis.

5.3. Implementation

There are different ways of describing the system architecture. A common approach for software design is to divide the architecture into logical components. This architecture could be characterized as three tier architecture [Eckerson, 1995], thus the application can be divided into three logical components: the presentation, logic and storage. These components are illustrated in Figure 14.

In the case of Web applications this division is often hard to define and not always even possible to make. The basic physical parts are the server and client, which are responsible for the whole application implementation. In this prototype it can be said that the server is only responsible of providing the content, thus it only spans the storage part of these three parts. Although it can

be acclaimed that the server-side implementation also includes some logical functionality, most of the application logic is still on the client-side. Thus for the sake of simplicity it can be said that the client will span the logic as well as the presentation tiers of the application.

The basis of the implementation:

- The user interaction logic is completely on the client-side.
- The user interface presentation is completely on the client-side.
- The content is shared from the server-side through and resource oriented service API.
- The content is saved by the server-side, accessing the native data sources

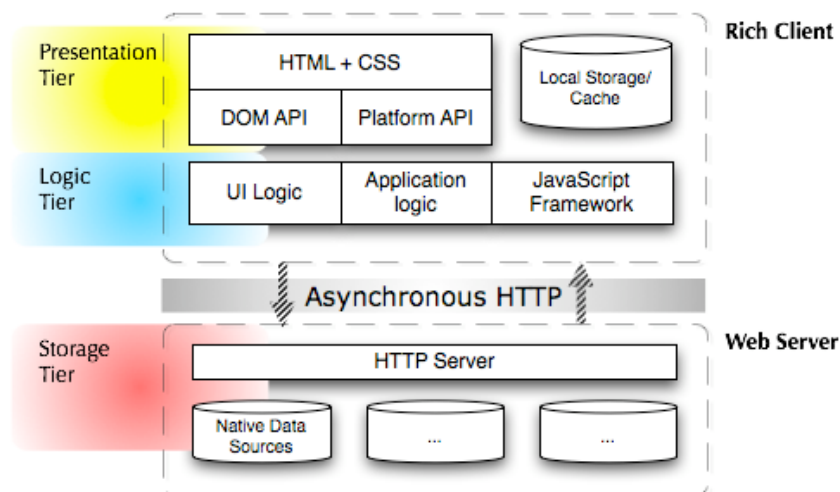


Figure 15. Implementation architecture – illustrates the logical components of the implemented architecture

As can be seen from Figure 15, the implementation can be divided into two parts: the server-side and the client-side implementation. In the following subsection these implementation details are discussed in more detail.

5.3.1. Server-side implementation

From the technical point of view the server-side implementation resembles a traditional server-side scripting approach. It consists of an overwritten request handler as well as a model class for the needed resources, in this case the calendar events. The server-side implementation was done using Python programming language, with the *mod_python* module [Mod_python Project, 2009] for Apache Web server to enable the scripting of an own request handling procedure.

The main classes of the server-side implementation are the *Request handler* and *Service model* classes. The *Request handler* represents the REST –controller

described in Subsection 4.6.2 and is responsible for handling the request by querying the needed resources from the Service model class. The *Service model* is responsible for the specific functionality in the application domain. In the following list, these activities are listed in more detail. The class diagram for the server-side implementation is depicted in Figure 15.

Class structure:

- *Request handler*
 - First directs the request according to the method (*GET*, *POST*, *DELETE*)
 - Then queries the response from the Calendar model according to the requested resource
- *Service model (Calendar)*
 - Models the Calendar resources of the device itself with the applicable methods for interaction.
- *Interaction Logger*
 - Logs the interaction for evaluation and debugging.

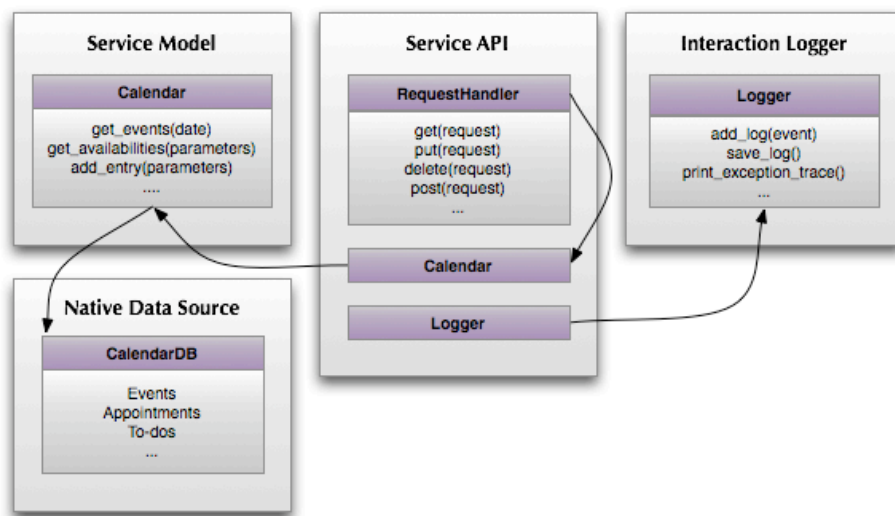


Figure 16. Class diagram for the server-side implementation

5.3.1.1 *Request handler*

The *Request handler* class overwrites the *mod_python* module's own request handling procedure. The reason for this is to provide a flexible interface for the abstract resources stored in the native databases accessible by simple URIs. This means that the resource structure used to represent the API structure is a virtual representation of the actual resources. These resources are

programmatically gathered when the request takes place by calling the corresponding method from the model representation of the calendar interface.

The request handling procedure includes reading the request, processing the headers and gathering the content for the response. When processing the headers, the request is directed first according to the applicable method (*GET*, *POST* or *DELETE*) to add, get or delete content. Then the request is directed by the resource identification (*URI*) to the applicable method on the Calendar model. After adding the content from the model instance the right HTTP return code is returned and the response is sent back to the client. A code example of a *request handler* is given in Listing 1.

```
# Request handler class, implements the REST interface
# Directs the request according to the requested action
from mod_python import apache
class RequestHandler(object):
    def handler(self, request):
        if requ.method == "GET":
            return self.get(request)
        # else PUT, POST, DELETE or return apache.HTTP_NOT_ACCEPTABLE

    def get(self, request):
        path = request.uri
        if(path == RESOURCE_TODAY):
            request.write("<calendar  userId=\"%s\">%s</calendar>" %
(host, self.cal.get_today()))
            return apache.OK
```

Listing 1. Request handler example using *mod_python*

5.3.1.2 *Service model*

The *Service model* class represents the interface for accessing the resources programmatically. While the request handler represents the API to the “outside world” and can handle several different resources for several data models, the model class represents an internal interface to a specific set of resources, such as the calendar events. Therefore, the *Service model* class represents the domain model of the specific application and could be used for several service interfaces. On the other hand the *Request handler* can wrap several service models, since they are merely represented as a tree of identifiable resources.

The main use of the *Service model* in this case is to retrieve and save calendar events. The calendar events are stored in the native database for calendaring information. In order to access this database from the Python implementation, a

module is needed to wrap the native calendaring interface for the Python environment. This is done by the Python for S60 environment [Nokia Research Center, 2009b], which enables the running of Python scripts on the Symbian S60 platform [S60, 2007]. It also provides several modules to access the native data storages, such as the calendar events. A code example of a *service model* is given in Listing 2.

```

# Service model class for the Calendar instance
# Provides the Calendar events for the Calendar requests
import calendar
class Calendar(object):
    def get_this_week(self):
        monday = self.now-(self.week_index*self.day)
        friday = self.now+((5-self.week_index)*self.day)
        this_week = self.cal.find_instances(monday, friday)
        return self.create_busy_entries_XML(this_week)

    def get_this_month(self):
        this_month = self.cal.monthly_instances(self.now)
        return self.create_busy_entries_XML(this_month)

    def get_today(self):
        today_entries = self.cal.daily_instances(self.now)
        return self.create_busy_entries_XML(today_entries)

```

Listing 2. Service model example using *mod_python*

5.3.2. Client-side implementation

The client-side implementations are task specific Web applications. They access, interpret and manipulate the content delivered from the server. They are built as stand alone Web applications or Widgets that are installed on the client-side, and are rendered by the platforms own HTML rendering engine. On the Symbian S60 platform used for the prototype development these client-side Web applications are rendered by the Web runtime environment [S60, 2007].

The client-side architecture consists mainly of the presentation of user interface and the logic for handling the user interaction as well as the communication to the servers. The overview on the architecture is described in Subsection 6.3.2.1. In addition, the user interface structure and its dynamic manipulation are described in Subsection 6.3.2.2. The logic for handling the user interaction and the server communication are described in Subsections

6.3.2.3 and 6.3.2.4 respectively, and finally the S60 Web runtime specific APIs are described in Subsection 6.3.2.4.

5.3.2.1 *Client-side components*

As it was briefly mentioned in the beginning of this section, the client-side implementation can be divided into two logical components: the presentation and the logic (cf. Figure 16). The presentation includes the user interface which presents the content acquired from the server. The logic is responsible for handling the user interactions and the communication with the server. In addition, there is a third logical component, the application content. In this case the content is provided from the server and then presented to the user via the user interface, thus it is not considered here to be the part of the client-side components.

Another division can be made by the physical parts of the client application package. These are the main techniques of any Web application, the user interface structure, described using HTML, the presentational styles described using CSS and the business and interaction logic implemented using JavaScript. These physical partitions can be described as the structure, style and the logic. The client-side architecture is depicted in more detail in Figure 17.

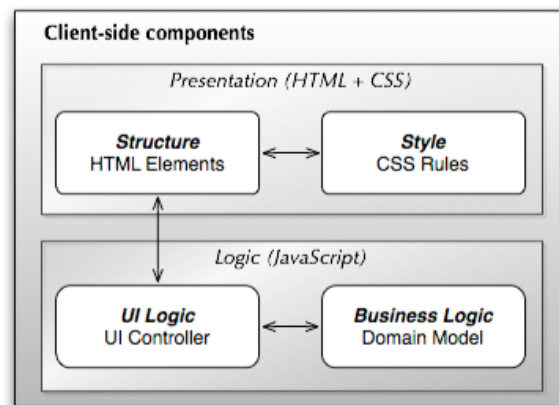


Figure 17. Client-side components

5.3.2.2 *DOM manipulation*

An important part of the user interface development using HTML, CSS and JavaScript is the manipulation of the Document Object Model (DOM) [Le Hors et al., 2000]. The DOM is basically a tree of HTML elements and to be able to modify that tree, the needed node has to be found. Therefore, many of the JavaScript toolkits available have a specific way to access the nodes in the DOM tree. The same goes for CSS (Cascading Style Sheets) [Bos et al., 1998], in order to apply the style rules to an element, the element has to be explicitly selected.

These CSS selectors have also been used in jQuery –toolkit as a model of selecting the desired element from the DOM. Nevertheless, whatever the convention is the basic approach is to build the HTML structure with some “anchors”, such as the id’s for the elements to enable the selecting of these elements from the logic. Often this is done to bind an event handler for a certain interface object.

5.3.2.3 *Event handling*

The event handling is the most important part for the user interaction logic. Event handling basically means the handling of user inputs as well as the system inputs, such as the communication events. The binding of event handlers to the user interface can be done in several ways. Since the user interface structure consists of the visible HTML elements, it is possible to bind the event handling logic straight to the individual user interface elements. This convention is not the best possible solution since it leads to a cluttered design mixing the event handling logic with the presentation elements. A better approach is to bind the objects within the logic partition by selecting an element and binding it with an event handler. The latter approach has been used in the prototype client applications. Listing 3 shows how the event binding can be done using the jQuery –toolkit [jQuery, 2009]. There are slightly different approaches and conventions in different toolkits.

```

$("a#back").bind("click", function(e){
    $("div#eventDetailView").hide();
    $("div#availabilityView").show();
});

```

Listing 3. Event binding using jQuery –toolkit for JavaScript

5.3.2.4 *Ajax communication*

The client-side Web applications differ from browser based applications due to the fact that most of the code and mark-up is already stored on the client-side. This has the implication that the view cannot be refreshed in the same way as in the browser based Web applications or Web pages in general. Thus the use of Ajax communication is essential to enable smooth integration between the client interface and the server-side content. Most of the JavaScript toolkits provide extensive support for Ajax communication and make it easier to develop client-side Web applications. They wrap the underlying, browser-specific implementations of the XMLHttpRequest –object and thus enable the use of clean and compact code for Ajax –functionality. A code example of an

Ajax call using the jQuery toolkit is provided in Listing 4. In the prototype client applications Ajax is used to retrieve the content from the server.

```
// Loads the XML -content from the provided URL
function loadXML(url) {
    $.ajax({
        type: "GET",
        url: url,
        dataType: "xml",
        success: parseDOM,
        error: function(msg) {
            alert("error: " + msg);
        }
    });
}
```

Listing 4. Ajax call using jQuery –toolkit for JavaScript

5.3.2.5 *Web Runtime specific features*

The Web Runtime on Symbian S60 platform [Nokia Research Center, 2009a] has its own specific features that provide richer interaction with the mobile device using the Web techniques. This is important since these techniques were originally meant for static Web content accessed via a Web browser from a desktop computer; the difference to interactive content accessed from a mobile device is tremendous. These specific features include for example the ability to set the navigation paradigm from cursor to tab, rotating the display between portrait and landscape, as well as launching native applications. Moreover, in the later version, Web Runtime 1.1 the features have been extended to include access to some of the device capabilities such as the Calendar, Contacts and Location [S60, 2009b].

In this prototype system, these Web Runtime specific features have been taken into account to enable richer and more intuitive interaction as well as to extend the use of the client application from a mere Web service client into an interactive, rich Web application with access to device capabilities for content and context information. More information about these features and implementation details for the prototype clients are described in the following subsections.

5.4. Design alternatives

Since there exists at least two different ways of accessing and interacting with the calendar data, these were taken into account when designing the server interface. Due to this it was also natural to design and implement two different client applications. Although these two approaches make use of the same data, the use cases and interactions are quite different in reality. The design has tried to capture the specific user tasks, thus two different client-side interaction styles have been developed to evaluate the best approach for the development. Moreover, by developing two different client applications for the same data source and server interface the basic idea of the concept described in the earlier chapter can be evaluated.

5.4.1. *Shared Calendar* user interface approach

In the *Shared Calendar* client the user interface is designed as an ordinary calendar layout that shows the user the set of pre-existing events. The user can select to view the events from a single day or a week. The loading of calendar events from other participants in the group is done asynchronously on the background once desired. The addresses of the other participants can be saved on the native address book database and retrieved from there to be used on the *Shared Calendar* client tasks.

In order to add a group meeting, the user selects a suitable time from the calendar interface. The application selects that time as the preset for the event information view. When the needed information has been entered, the user can send the meeting information from his device to all the participant's terminals. The event details will be saved to the native calendar database of all the participating people's terminals.

The interface design has been developed from the point-of-view of ordinary calendar interfaces with the emphasis on the restrictions made by the small screen size of a mobile device. The interface is designed by using HTML elements as the structure of the interface and CSS for applying the style attributes on those structural elements. The user interaction is handled by JavaScript event handling, without the need for refreshing the interface while navigating. In the Figure 18, the user interface and interaction is described by a wireframe approach.

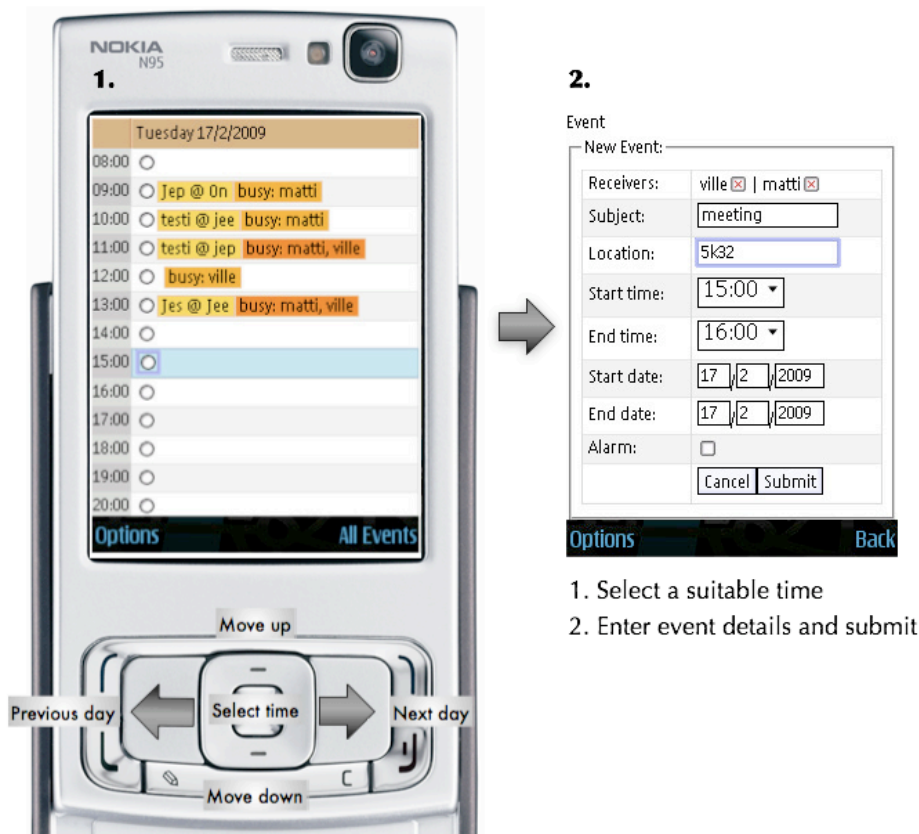


Figure 18. *Shared Calendar* user interface approach

5.4.2. *Group Scheduler* user interface approach

The *Group Scheduler* client application has a different approach for taking advantage of the existing calendaring data than the *Shared Calendar* client. The *Group Scheduler* client only queries the data source for availabilities on a certain set of times. For example if the user has a set of times already in mind he can query the availabilities of the other user's using those times as the inputs.

The interaction follows a “*query approach*”, where the information is gathered and shown by a sequence of queries and selections. This kind of step-by-step approach is well suited for the interaction with a device that has limited screen-size and input abilities. The visualization of the available times are given as a matrix of times and persons, which gives an instant visual feedback of the best suitable times. This visualization approach is also scalable for much larger groups than the *Shared Calendar* user interface approach. In Figure 19, the user interface and interaction is described by a wireframe approach.

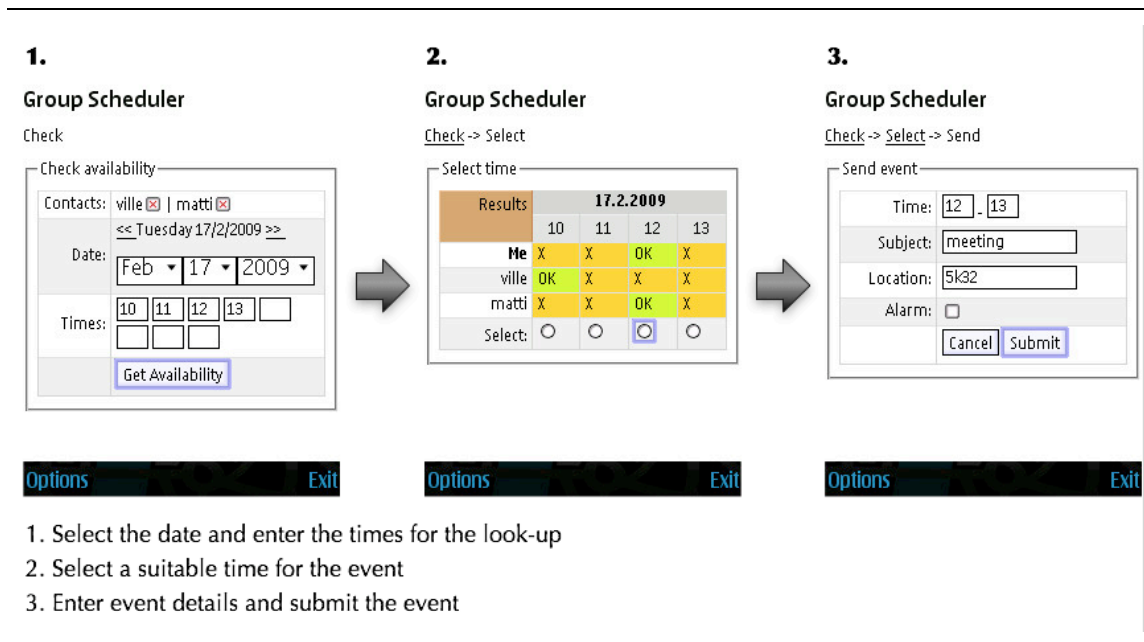


Figure 19. *Group Scheduler* user interface approach

5.5. Summary

This chapter has discussed the development of a prototype for the concept described in Chapter 4. Both the Web server and the client side implementation has been described as well as introduced the reference user interfaces for the prototype service for sharing calendar events in a group of people. In the following chapter both the architecture and the prototype implementation will be evaluated from the basis of the requirements as well as by conducting a user study.

6. Evaluation

“If I would have asked people what they wanted, they would have said a faster horse.” - Henry Ford

This chapter will provide the critical, evaluation and analysis part of this thesis. The goal is to get an overview on the user acceptance and expectations of this kind of system and to evaluate whether the prototype system fulfils these requirements. Another goal is to evaluate whether the functionality of the system provides a viable solution to the given research problem. The methods and measurements for realizing these overall goals are described in the following subsection in more detail.

6.1. Methods

The evaluation was carried out with an initial survey, a requirements analysis and a user study. The initial survey was issued to acquire background information from a group of people about their expectations towards the use of mobile devices and shared calendars. The survey was divided into two parts. One part was more general, considering the whole concept and the other one was more specific, considering the initial prototype evaluation. The concept architecture was evaluated against the requirements for *mobile collaborative systems* and the requirements for *Web services architecture*. These requirements were listed in Subsection 4.5.1. The functionality of the prototype implementation is evaluated against the functional requirements listed in Subsection 5.2.3. A user evaluation was also conducted to evaluate the usability of the user interface approaches as well as to compare these solutions in the context of the given tasks. These evaluations are given in the following sections in the introduced order.

6.2. Background analysis

An analysis about the background for the work was issued due to the novelty of the approach. The first step was to conduct a survey to gather some background information from the users. The goal of this was to get an initial direction about the important factors in the use of mobile devices and applications in general, as well as the use of mobile devices to access web content and services. Furthermore, some information was gathered about the

use of mobile devices for personal information management, such as the calendar, and the attitudes towards the sharing of that information in a group of users. The main focus of this was to define some important factors for the design that could be used in the evaluation of the concept and the prototype implementation.

The survey was conducted as an online questionnaire, which consisted of two parts. The first part was directed to the overall design issues, and the second part to the use of shared calendars, which was the topic for the prototype implementation. Since the questionnaires were independent from each other, they were issued separately. 34 persons answered the first one and 31 persons answered the second one. The respondents represented 5 different nationalities. Overall, since many of the respondents answered both, 36 individual respondents answered the questionnaires. Thus it can be acclaimed that the amount of respondents gives a significant value for the results. The results of the questionnaires are discussed in the following subsections.

6.2.1. General aspects

The survey featured questions about the importance of performance, robustness and usability when using mobile devices. These answers can be used to rank the requirements for the architecture as well as for the individual applications. For example, it was found that the usability of a mobile device and its applications is ranked more important than the efficiency and robustness of the applications. Of course this does not mean that the latter ones would be less important, since the efficiency affects the perceived usability as well. What it does reveal is that people perceive and rank the user interface and interaction of applications quite important in mobile devices, which is hardly a surprise. It can be said that the mobile devices are in fact even more affected by the perceived usability and user experience than personal computers, because they are lacking in capabilities and input mechanisms, but are often used for similar tasks.

6.2.2. The use of Web content on mobile devices

To analyze the background for the architectural decisions made, it is important to know if people use the mobile devices for accessing Web content. This is because the architecture and the proposed client applications imply the use of the Web as the platform for collaboration and communication. Therefore it is useful to know how much and for what kind of purposes people use the Web access on a mobile device.

Overall, it seems that people have started to use the Web through mobile devices as well. As it is shown on Chart 1, 73% of participants have used a

mobile device to access Web content or services. In addition, 52.1% of participants ranked the importance of a mobile device for accessing the Web as very important or essential. It seems that the most common uses of Web from a mobile device are the information services like bus and train timetables. Also Web-based email services were mentioned a few times. In addition, some participants announced they use applications like Facebook and Twitter from the mobile device to enable quick access for updates. Calendar services like Google Calendar were also mentioned, although they did not seem to be that popular accessed from a mobile device. This might be due to the use of the mobile device's own calendar. Nevertheless it seems that the mobile device is already an important medium to access Web content, which gives this work a solid background.

6.2.3. The use of a mobile device for PIM

The important question towards the use of mobile devices for collaboration is how much people use them for storing important information that could be useful to share. In the survey it became quite clear that people use mobile devices for Personal Information Management (PIM) purposes extensively, as 91% of the participants answered that they use a mobile device for PIM (cf. Chart 1). In addition, 56.7% of participants ranked the use of a mobile device for personal information management as very important or essential. This gives sound ground for the use of a mobile device for collaboration.

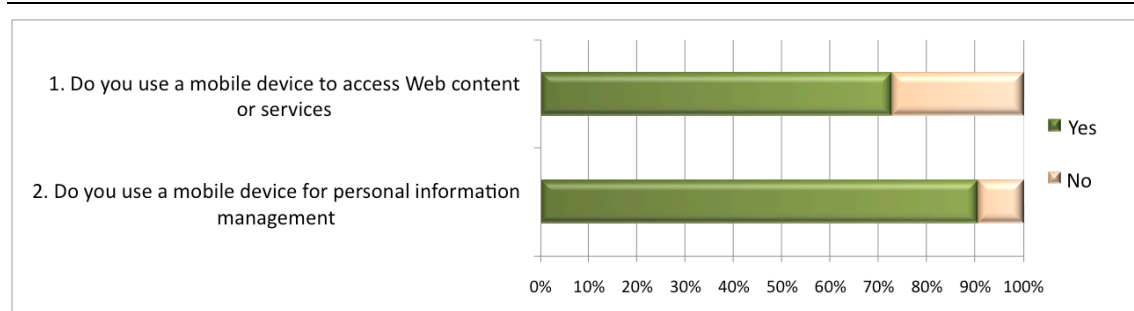


Chart 1. The use of mobile device for Web content and PIM

6.2.4. The use of a mobile device for calendar purposes

The use of calendar on the mobile device is very common. 80% of the participants use the mobile device to store their calendar information. It is also more commonly used than desktop or Web-based calendar software which were used by 57,89% and 31,58% of participants, respectively. In addition, over 68% of the participants ranked the use of the mobile device for calendar information as very important or essential. The extensive use of the mobile device for storing calendar information is important for the collaborative use as

well. It is very common that people do not like or want to save the same information twice, therefore they usually use the one that is available most often, like the mobile device.

Another important aspect regarding the collaborative scheduling task is the use of shared calendars among the participants. 56% of the participants have used a shared calendar, but the usefulness of sharing calendar events from the mobile device within a certain group was not found to be significantly useful. About half of the participants found it useful whilst the other half did not. This might be due to the specific use cases for this kind of capability. It is needed only in rare use cases, thus it would not be valued in an ordinary calendar application. Nor would it seem to bring extra value if it would require extra effort from the users. The same goes for overall importance of sharing personal information from the mobile device. It is only valued by some participants, this might indicate that it is suitable only in some use cases, in other words it would be suitable as a specific functionality or as a separate, dedicated application.

6.2.5. Summary of the analysis

The use of mobile devices to save important information such as calendar events and to-do lists as well as contact information is popular because it is almost always available. In addition, it is almost always on, which makes it a suitable device for alarming and calendaring purposes. This also gives the possibility to use the device for providing information as a server for local, person-specific content. This information gives the use of a mobile collaboration platform some justification, but also some requirements and challenges as to what come to the usability and efficiency of use. These challenges have to be answered when designing a platform for collaboration, which is usable with mobile devices in a mobile user context.

6.3. Concept evaluation

In addition to the background information survey, a requirements analysis was completed. The requirements analysis was divided into two parts, the first part was dealing with the architecture for the concept and the second part was dealing with the prototype system. The concept architecture can be evaluated from two directions, these being the two components of the proposed solution: *the mobile collaboration platform based on Web service architecture*. The requirements for these both “components” were listed in Chapter 4 in Subsection 4.5.1, and they are used as the points of reference in the evaluation of the concept. In some cases these requirements are overlapping, but overall they try to cover the requirements from both directions of the approach. The

assessment was done by a simple *Yes* or *No* whether the requirement is fulfilled. The reasoning for these assessments is written on the *Arguments* column in the table.

6.3.1. Mobile collaborative system requirements

The requirements for mobile collaborative systems act as the requirements for the collaborative functionalities of the system. These requirements should be fulfilled in order for the system to support mobile collaborative work and thus they are the basis of the assessment of the concept as a collaboration platform. These requirements are assessed in Table 4. It can be seen that the concept architecture meets the requirements for a mobile collaborative system.

Requirement	Fulfilled	Arguments
RMCS1. Knowledge sharing (both ways)	Yes	The given user can participate in both consuming and providing information.
RMCS2. Device-independence and interoperability	Yes	The system can be used by different devices and is interoperable among different platforms. (More in the next section)
RMCS3. Communication and collaboration	Yes	The system can be used fully whilst connected, locally when disconnected and have a possibility to enable ad hoc connectivity depending on the technology in case. (More in the next section and discussion)

Table 4. Assessment towards the mobile collaborative system requirements

6.3.2. Web services architecture requirements

The requirements for Web services architecture act as the requirements for the architectural aspects of the system. These requirements evaluate the concept architecture towards the requirements for Web services architectures and thus apply the basic elements that are essential for a design of Web services architecture to this architectural approach. The requirements are assessed in Table 5. Overall, all of the requirements for Web services architecture are fulfilled.

Requirement	Fulfilled	Arguments
RWSA1. Interoperability	Yes	The services architecture is interoperable by different platforms as well as with different client-side and server-side technologies.
RWSA2. Reliability	Yes	The architecture is reliable for group communication. (More in the next section)
RWSA3. Integration with the World Wide Web	Yes	The architecture is fully consistent with the evolution of the World Wide Web (uses a REST compliant architecture).
RWSA4. Security	Yes	The architecture can be used via a secure HTTP connection (SSL).

RWSA5. Scalability and Extensibility	Yes	The architecture is scalable to be used by different target devices and is extendable for serving several resources. (More in the next section)
---	-----	---

Table 5. Assessment towards the Web services architecture requirements

6.4. Prototype evaluation

The prototype system has been implemented based on the concept architecture. Although it does not provide all the possible functionalities, it is extendable and it fulfils the described requirements for the architecture. In addition to the previously described evaluation of the concept architecture, a more specific evaluation was done for the prototype system. The prototype evaluation was done by a requirements analysis and a user evaluation. The evaluation is based on the implemented system and its perceived features. These evaluations are described in more detail in the following subsections.

6.4.1. Functional evaluation

This subsection describes the evaluation of functional requirements for the collaboration tool. The functional requirements for the prototype implementation are listed in Subsection 5.2.3 and they are assessed in Table 6. Both client application approaches are evaluated towards the requirements for the prototype implementation. From the table it is visible that both applications fulfil the functional requirements for the collaborative scheduling application. This evaluation is just an initial analysis, which is completed with a user evaluation. The user evaluation results are given in the next section, the references to the corresponding evaluations are presented after the assessments (i.e. *assessment | reference to user evaluation charts*).

The functional requirements	Shared Calendar approach	Group Scheduler approach
FR1: The user must be able to select a desired day to be viewed.	Yes C1.3 & C1.4	Yes C1.3 & C1.4
FR2: The user must be able to select an appropriate time for the event.	Yes C1.5 & C1.6	Yes C1.5 & C1.6
FR3: The system must provide a way to enter the event details.	Yes C2.4 & C2.5	Yes C2.4 & C2.5
FR4: The user must be able to submit the event to the participants	Yes C2.6	Yes C2.6
FR5: The user must be able to see the system state at any given time	Yes C2.3	Yes C2.3

Table 6. Assessment of the functional requirements for the shared calendar - prototype

6.4.2. User evaluation

A user study was conducted for evaluating the developed scheduling applications as well as to gain some insight into the concept from a user's perspective. The test included a discovery and instantiation of a group meeting, done by the "responsible" person for that task. This evaluation did not consist of any group behaviour evaluation, or the evaluation of the outcome of this task for the other members of the group. Hence the usability test was conducted as a single user test, evaluating the use of shared calendar information from a mobile device using two different interface approaches. The test process is described in the following subsections.

6.4.2.1 Background information

There were two main purposes for the user tests. First of all the test was conducted to evaluate the usability of the implemented prototype applications and thus to test the usability challenges of the selected user interface technologies in a mobile usage. Secondly, the results from the test questionnaires were used to compare the two implemented approaches for group scheduling in order to gain a better understanding for future development directions.

In total, 14 test users were recruited from an organization conducting technical research. This was not considered as a problem due to the emphasis of the test on collaborative scheduling in a team use. Even though all of the participants were employees of the same organization they had different backgrounds and fields of work. The participant's job descriptions spanned from Research Trainees to a Technology Manager (the leader of an organizational unit), thus they also had different priorities and needs for communications and collaboration. The ages spanned from 25 to 40, with the median age being 26.5 years, thus the main emphasis on the test was on the younger employees.

The participants were asked some background information about their use of mobile devices as well as the use of different types of electronic calendars (mobile, desktop and web-based). 64.3% of the users stated that they have experience with smart phones, which was considered to be a plus. The lack of experience in using similar devices earlier can affect the performing and perceived usability but this was not considered a problem. With electronic calendar use, 57.1% stated that they use only desktop calendar while 42.9% stated that they use both a mobile device calendar and a desktop calendar. In the case of two or more calendars used, the users were asked which one they would consider to be the most important. One third of the users of several electronic calendars considered the mobile device calendar as the most

important while the rest stated that the desktop calendar was the most important. Web-based calendar software such as the Google Calendar [Google Inc., 2009a] was only used by one participant. This finding contradicts with the results from the background survey. It can be explained somewhat with different backgrounds of the participants. In the work place environment people tend to become used to using the most commonly used applications for their work, thus in an office use, the Web-based calendars are often not supported by the common communication systems. Nevertheless, overall the gathered users covered the selected user group focus well.

6.4.2.2 *Test setup*

The test was conducted in a work place environment, since the usage situations of the applications are most likely to happen in that kind of an environment. A mobile use was not considered in this evaluation since it was not the main design principle in this work, although it could have been interesting to apply the challenges accompanied by the mobile use in this evaluation as well. Nevertheless, the test was conducted as an ordinary usability test in a laboratory environment introducing a user scenario and a set of tasks for the users to carry out during the test. The user scenario was adapted from the initial user scenario, which was introduced in Subsection 5.2.1 in order to comply with the functional requirements for the prototype.

The scenario was described to the users as the following: *“You are part of a ‘mobile team’, a small team which is often located in several physical locations, but who do a lot of collaboration together. In the test scenario you have a couple of tasks, which involve scheduling a time for a meeting for a group of people. The team in question is an imaginary team with three persons”*.

The tasks were designed to be a set of different kinds of user situations when a scheduling task could happen. The users were asked to complete a set of five tasks with both applications. Two sets of similar tasks (the same tasks, but with different times and dates) were done in order to evaluate and compare the two developed approaches without the user being able to remember the result from the previous task. After each set, the users were asked to complete a questionnaire with a set of questions to evaluate that particular application.

The questions were given in a form of statements about the use of the application with multiple-choice answers from 1 to 5, stating how well they agreed with the statement. The statements were for example: *“exploring the group’s calendar events was easy”* or *“finding a suitable time for a group event was fast”*. The answer scale was put in a literal form, varying from *“totally disagree”* to *“totally agree”*. The questionnaire follows loosely the questionnaire style for user interface evaluation introduced by Lewis [1995]. At the end of the test,

after testing both applications, the users filled a final questionnaire to compare the two applications from the basis of the given tasks.

In order to avoid any biased results from learning of the calendar contents during the test, the participants were divided into two groups, *Group 1* and *Group 2*. *Group 1* used *Shared Calendar* as the first application to test and then the *Group Scheduler* as the second; where as *Group 2* used them in the opposite order. The participants were divided in these two groups randomly and equally.

6.4.2.3 *Pilot test*

At first, a pilot test was conducted in order to test the test setup and make sure that the questions were understandable as well as unambiguous. During the pilot test, some small refinements were made to the test setup and to the test tasks. Some usability issues were also obtained, which resulted into one change in the user interface of the *Group Scheduler* application right before the tests. The changed feature was the input of times the user wanted to check. It was changed from individual times into a time period. This change was mainly due to the set of selected tasks in order to have a fairer comparison between the two approaches. The change did not affect other parts of the application or the interface. After these changes the test was considered ready for the real users.

6.4.2.4 *Test results*

After the pilot test, the main user tests were conducted over three days. The main results, which are relevant for this thesis, relate to the comparison of the two evaluated approaches. Since the usability results are not that important for this thesis work, they are introduced only briefly, with emphasis on general level findings and the most interesting issues found.

In the test the users were asked to evaluate the applications after they had performed the set of tasks. In the first part of that questionnaire the users were asked to evaluate the usage of the application. The main focus was to evaluate the easiness and the fastness of the use of the applications. The same statements were made to state both the easiness and fastness of use. The results for the comparison between the two evaluated applications can be seen in Chart 2.

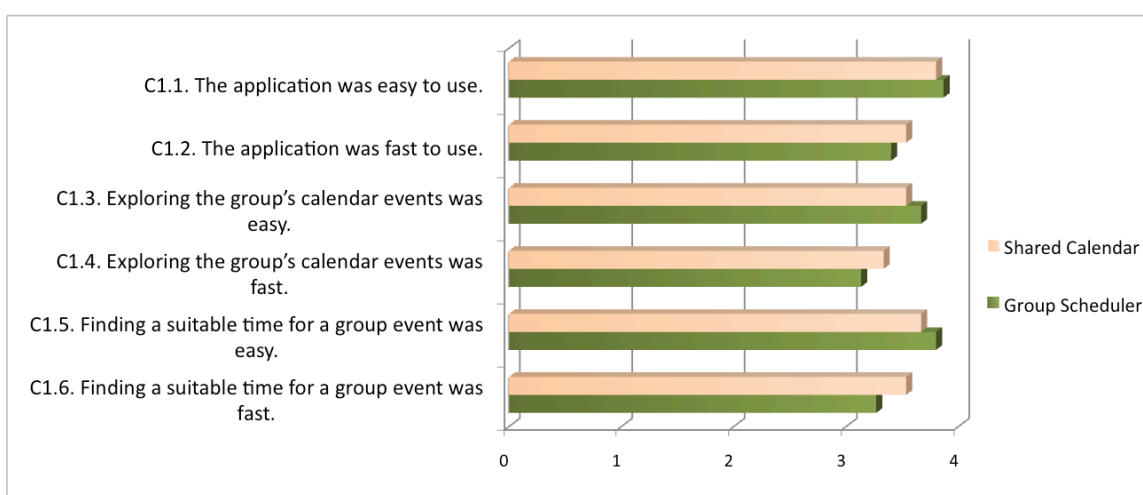


Chart 2. Evaluation of user interface and interaction – the bars represent the weighted average scores for each question dealing with the usage of the application.

The users were also asked to evaluate more generally the user interface and the interaction with the application. The results from this part of the questionnaire are depicted in Chart 3. In this part of the questionnaire, the emphasis was to evaluate the suitability of the interface and the needed user effort as well as the practicality of the user interface elements to be used with the mobile device. Also general level likings such as the easiness and the pleasantness of use were evaluated. The compound, weighted results from these parts of the questionnaires can be seen in Chart 3.

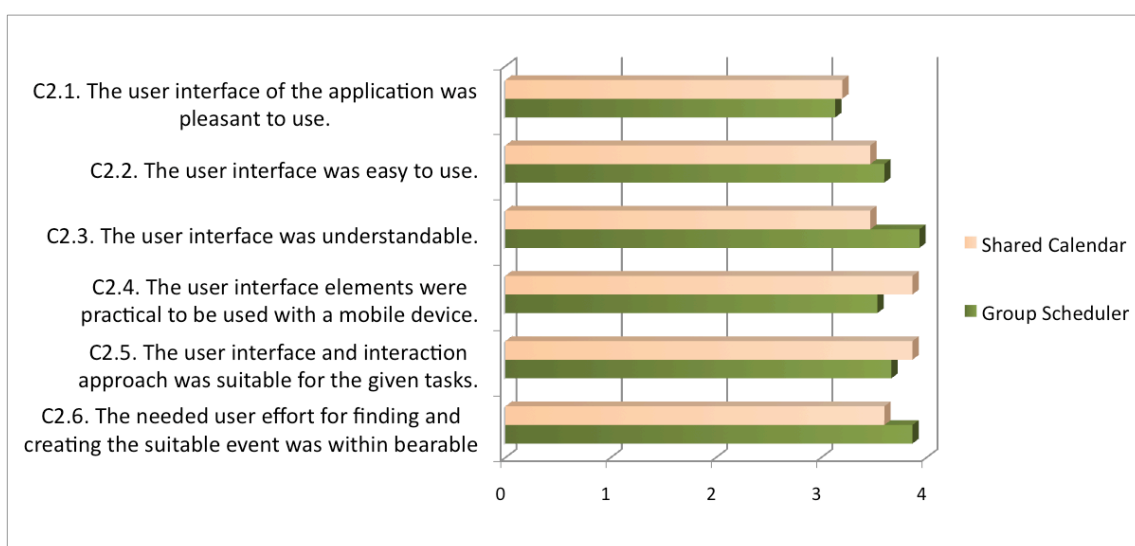


Chart 3. Evaluation of the usage of the application – the bars represent the weighted average scores for each question dealing with the user interface and interaction with the application.

Even though no great differences were found between the two evaluated approaches, there was a clear pattern visible. It was visible that the Group Scheduler application (introduced in subsection 5.4.2) was repeatedly stated as easier to use than the Shared Calendar application (introduced in subsection 5.4.1). These results can be seen from Chart 2, from the points *C1.1*, *C1.3* and *C1.5*. Moreover, the user interface of the Group Scheduler approach was stated as more understandable (in Chart 3, C2.3: 3.9333 vs. 3.4667), easier to use (in Chart 3, C2.2: 3.6 vs. 3.4667) and the needed user effort being less than with the Shared Calendar approach (in Chart 3, C2.6: 3.8667 vs. 3.6).

On the other hand, the Shared Calendar application was repeatedly stated as faster to use than the Group Scheduler. These results can be seen from Chart 2, from the points *C1.2*, *C1.4* and *C1.6*. In addition, the users ranked the user interface of the Shared Calendar approach as more practical to be used with a mobile device (in Chart 3, C2.4: 3.8667 vs. 3.5333) and more suitable for the given tasks (in Chart 3, C2.5: 3.8667 vs. 3.6667). It was also stated more pleasant to use (in Chart 3, C2.1: 3.2 vs. 3.1333), which does contradict slightly with it being stated as less easy to use. This might be due to it being perceived as faster to use, which in some cases affects the user experience more.

Some conclusions can be drawn from these results. It seems that the user interaction and the user interface were more understandable in the Group Scheduler application (cf. Chart 3, C2.5: 3.9333 vs. 3.4667). The interaction approach in the Group Scheduler application was a straight forward query of available times for a certain set of times, and it can be seen that this approach suited well with the given user tasks. On the other hand the Shared Calendar application was repeatedly stated as faster to use (cf. Chart 2, *C1.2*, *C1.4* and *C1.6*). This might be due to the tasks where the user was asked to browse to the next suitable time, which was easier with a “regular” type calendar interface, like the Shared Calendar. In conclusion it can be seen that the Group Scheduler was better within the *search* tasks, while the Shared Calendar was better in the *browse* tasks.

In the final questionnaire the users were asked to choose which of the applications they felt was easier to use, was more suitable for the given tasks and which one they liked more to use. The results summed up almost equal for each of the questions, stating that both applications had their own good and bad sides and thus no differences could be based on those answers. On the other hand, some of the comments did open up the differences between the evaluated applications. It was commented that the Shared Calendar was preferred because it was faster to use than the Group Scheduler. This is consistent with the results described earlier, derived from Chart 2. For the

Group Scheduler it was stated that it was clearer and would be more scalable approach with bigger user groups than the Shared Calendar application. This is consistent with the results derived from the Chart 3, about the user interface and interaction evaluation.

6.4.2.5 *Comments and improvement ideas*

In addition to the quantitative analysis of the results, some comments were gathered from an interview after the user test. The comments did clarify some distinctive differences and reasons for choosing the preferred approach. They also provide some interesting points about the whole concept.

It can be seen that the decision between the two approaches is dependent on the usage of the application. It is visible from the quantitative results that the *Group Scheduler* was considered easier to use, possibly due to a more straightforward interaction and better visualization. However the important difference is really the intended use. If the person already knows the times for which they need to schedule a meeting, then the *Group Scheduler* application would be an obvious choice. On the other hand it was clear that the users perceived the *Shared Calendar* application as faster to use in general. Therefore, if the intended use would be to browse for example, the nearby vacant times for an ad hoc meeting, the Shared Calendar approach would be an obvious choice for that scenario. One user confirmed this finding by commenting that:

“It depends on the purpose. If the time is more relevant, then Group Scheduler, if seeing other peoples vacant times then Shared Calendar”.

It was also shown that the integration with the phone calendar is a requirement for this kind of system to be usable for a real use. This was obviously one of the main points of the system in the first place, thus it can be considered a strong asset for the overall concept. The use of the devices native information sources for content should be developed even further. In addition, the use of the device’s contact information as the user identification in the application would integrate the system with the native information sources even further. One comment was made about the use of the contact information as unified user identification whilst using the application. This comment confirms that people often need only few contacts that they often meet:

“Quicker than with Outlook, it is good that your contacts are the ones you added, no use for adding the same persons for every meeting”.

The direction for future development could possibly grow out from both of the provided solutions. While there were no clear differences in which approach the users preferred, the main conclusion might be that they both provide a feasible solution to the tasks, but rather than competing with each

other they are completing each other. One user in her comment also brought this up by stating that:

“I would choose both if possible, Shared Calendar was more difficult to use but it provided the information more efficiently”.

6.5. Summary

To summarize the contents of this chapter the evaluation of the concept was achieved from different viewpoints. These viewpoints gave the needed empirical background for the analysis of the approach as well as some interesting issues to consider in the future development of this system. The methods used were common techniques of user-centred design process, a background survey and a usability test.

The background survey gave some important background information for the whole concept. It showed a wider perspective for the electronic calendar use as well as for the use of mobile devices for sharing information. These issues will be taken into account in the discussion chapter as well. The requirements analysis worked as a link to the related work as well as helped to formalize the key aspects of the architecture.

The user evaluation gave valuable input from the end-user perspective. It also worked as the final stage for the implementation task, and thus helped to formalize a complete version of the prototype that can be used for the further development. To summarize the contents of the user evaluation, it can be said that both of the applications had strong assets that came clear during the evaluation. In addition there was a clear distinction of strengths and weaknesses between the applications. The Group Scheduler was clearly better within the *search* tasks, while the Shared Calendar was better in the *browse* tasks. Overall it was found that the different approaches could complete each other rather than compete.

7. Discussion

In this chapter the main findings and contributions are discussed and related to the background of this work. This includes some general discussion about the concept and its implementation as well as some findings that were gathered at different stages of the work. One of the key aspects in this thesis is the use of *Web-based technologies* to enable *collaboration between mobile devices*. From the concept design as well as from the prototype implementation it seems that this kind of approach can be feasible in some cases, mainly to be used with *asynchronous communications and collaboration*, one form of the collaboration modes in Computer-Supported Collaborative Work [Baecker, 1995] described in Figure 1 in Chapter 2.

It seems that the sharing of personal information as well as user context from the mobile device using a Web service kind of interface would be feasible and useful in different user scenarios such as a teamwork scenario. From the background survey it can be concluded that the use of mobile devices for personal information management is almost self-evident thus the sharing functionality would extend the possible uses of this information.

From the technical viewpoint, the prototype system proved that the proposed architecture was feasible to implement. Also the usability of the developed system was on a functional level, as it can be concluded from the user evaluation. In addition to these overall remarks, discussion about the results and findings of this thesis is given in the next subsections.

7.1. Results

This section will give a short summary of the results gathered from the work. The results are reflected towards the hypothesis and the objectives of the work described in Sections 1.1 and 1.2 respectfully. These results try to wrap-up and summarize the most important outcomes of this work as well as to discuss them in the context of the related work introduced in Chapter 2.

First of all the hypothesis for the work was that:

“The available Web techniques can be applied to mobile environment to enhance the ability to interact in a group and to share content in a peer-to-peer manner”.

This hypothesis was proved to be correct from the technical perspective by implementing a feasible prototype system, as well as from the user’s

perspective by the results from the user study. As a summary, the available Web techniques used in this work were found feasible and fairly effective to use, of course a lot of development could be done in order to enhance the user experience and reliability of these technologies.

In addition to the hypothesis of the work, there were three main objectives for the work: *research*, *implementation* and *evaluation*. While the two first ones are quite self-evident objectives, the third objective, will need some further discussion. Therefore, the *evaluation* objective was broken down into more specific tasks. These consisted of the evaluation of:

1. Web service architectures in a mobile environment.
2. Mobile web services combined with task-specific clients as a collaboration platform.
3. The concept prototype service and clients against the requirements for the collaboration tool.
4. The feasibility of the prototype system by conducting a user study.

The first two objectives (1 and 2) were done implicitly during the work while the latter two objectives (3 and 4) were evaluated in Chapter 6. The goal here is to gather a quick summary of these evaluations for further reference. First of all, the use of Web service architecture in a mobile environment was found feasible and useful for collaborative work. The evaluation of the architecture proved it interoperable, scalable and extensible. Also the reliability was proved to be on a functional level during the user evaluation. The background for this was discussed in Section 2.4. The architectural decisions were also described in Chapter 4. These give the overall reasoning for the provided solution. The solution decision was a resource-centric approach for the architecture. The approach follows the RESTful Web service architecture [Richardson & Ruby, 2007].

The feasibility of this approach was proven by the prototype system and was evaluated by the user study. The conclusion is that the use of Restful Web service architecture is a feasible solution in a mobile environment. In the work by Srirama et al. [2006b] they provide a feasibility study for a similar case, a mobile Web service provisioning, but in their work a different architectural approach was chosen. The conclusions from their work support the work done for this thesis, on the parts that are applicable. On the other hand, they do not discuss the reason for their approach or the possibility of different types of Web service architectures in their study. This work contributes another feasible architecture to be considered for future developments.

The second objective was the evaluation of the mobile Web service provisioning coupled with task-specific clients as a collaboration platform. This evaluation was done implicitly by implementing the prototype system. The implementation task confirmed that the coupling is feasible, and in addition it was found considerably efficient as well. The latter two objectives were evaluated in Chapter 6, in Section 6.4 and the results for the evaluations can be summarized from Sections 6.4.1 and 6.4.2.4, these results are not discussed here further. As a summary for the results, the main research objectives for the work were met and the proposed solution was proven to be enough lightweight and efficient to enable a feasible provisioning of information collaboratively from a mobile device to a group of users.

7.2. Findings and considerations

In addition to the actual results, a lot of interesting findings for future development were found. These findings can be used as references or starting points when planning and developing similar studies. These findings and considerations are reported more in the following subsections.

7.2.1. Architectural considerations

In this subsection, the architecture is evaluated from the point-of-view of its obvious constrains and how these affect the overall capabilities and usefulness of the system. The architectural constrains are factors that exist independent from the implementation decisions. Therefore, these should be taken into account when evaluating the overall system.

The first consideration is the *extendibility*. Web services architecture is inherently an easily extendable architecture, thus the real task in this case is the modelling of the domain to gather a set of requirements covering the needed features. The architecture described in this thesis helps in this modelling task as well. It provides an architecture that enables the separation of the underlying information model and the Web service interface thus the task can be done by a model transformation. Siikarla et al. [2008] provide an approach for this kind of model transformation.

The second consideration is the *efficiency*. Networked applications are always affected by the latency and connection times. This effect cannot be discarded in total, but obviously enhancements for this can be made by different means. These means could be for example:

- Smart caching of information
- Development of peer-to-peer content delivery methods
- Intelligent prediction of user interaction to enhance the usability.

The overall efficiency is independent from a single point of failure, because there is no need for dedicated central server infrastructure for content storage and delivery. In the distributed environment the efficiency is very much related to the amount of users for the system in the given point of time. Thus this consideration should be taken into account when deciding the architectural approach for a particular application.

7.2.2. Communication considerations

In addition to the previously mentioned *efficiency*, the communication in a distributed environment can sometimes have more dramatic effects. In this subsection some of the experienced communicational difficulties and findings are discussed. In the preliminary plans, the scope of the work was covering an ad hoc type of communication between the users. This was done using an ad hoc Wifi connectivity between the peers and by using one of the collaborators as a *super peer* – a sort of a proxy between the peers. This kind of communication and connectivity was quite quickly found unreliable though, at least in the laboratory environment filled with Wireless connections. Nevertheless the architecture proved to be feasible for this kind of connectivity as well, if one would only discard the reliability problems between the peers.

7.2.3. User interface and interaction considerations

While the underlying architecture is not important or interesting for the end-user, it is important that the architecture provides a feasible platform to build a satisfying user experience for end-users. In general the user experience is built from several parts, the user interface, the user interaction as well as the meaningfulness of the information structure [Garrett, 2002]. The user interface development for mobile devices is always demanding, the limitation in screen space and input mechanisms cannot be avoided, although a huge development has been seen in the last couple of years. The findings and considerations here target mainly the meaningfulness of the application and the goal of a simple and efficient interaction. One of the main characteristics of the proposed concept is the use of task-specific client applications. These are introduced in subsection 4.3.2.

It became clear during the development and user evaluation that these task-specific client applications should target a “*simple as possible*” user interface and interaction. One reason is the limitations found on expressiveness when developing the user interfaces, but more important reason is the intended focus on specific user tasks. Therefore, these client applications should really be designed specifically for a *single* task in order to maximize the simplicity. This

simplicity will help to develop the best possible user experience for the application, since the focus will be on the most important tasks, less compromising needs to be done during the development.

7.3. Future work

In addition to the findings and considerations, some interesting development ideas have been found during the evaluation. These ideas and plans for the future development are described in this section divided into the *architectural advancements* and the *advancements for mobile collaborative scheduling*. In general the future work consists of formalizing the concept further as well as developing the prototype in order to expand the functionalities as well as to study the possibilities it may provide. The first step has been to introduce the concept approach and architecture as a research article [Antila & Mäntyjärvi, 2009] (currently under review), but further investigations need to be done in order to uncover the possibilities and pitfalls as well as the future use cases for this framework.

7.3.1. Architectural advancements

The described concept architecture was designed to be extendible. Therefore, the advancements here are directed towards the extension of this platform. The suggested architectural approach could be used to serve any information relevant to be shared with other people, for example location, context information, calendar events, photos and videos. The key aspect here is that the server-side architecture can be extended without having to upgrade the whole system. This is because the Web service architecture in general is inherently loosely coupled. Moreover, sharing information in the form of these kinds of mobile *micro services* can be seen as a part of the “*Internet of things*” development. The key aspect here is to use the Web as a platform for the interoperability and interconnectivity of different applications from different domains. The possibilities are only confined by imagination.

7.3.2. Advancements for mobile collaborative scheduling

From the basis of discussions and the user evaluation remarks some advancement features have emerged. These can be divided into *user interface advancements*, *system advancements* and *interaction advancements*.

First of all the user interface can always be developed further. One main direction for this would be to develop the user interface to be more flexible and powerful. This is mainly a task of coding and testing the capabilities faced with the mobile device screen size and input mechanisms. Another user interface

direction is the use of visualization techniques for the calendar events for a more clear distinction between own events and others'. Also the integration of the *Group Scheduling* –view as functionality within the *Shared Calendar* – application could be envisioned.

Another interesting advancement would be to further enhance the scheduling interaction by an automatic selection of the best suitable times for a desired time period. This automatic scheduling would be based on the availabilities of the group members as well as with a set of rules describing the desired decisions to take if no perfect solution were to be found. This kind of functionality could be visualized within the calendar interface or by a separate query interface that could visualize the best, second and third best solutions et cetera. This kind of functionality would enhance the intelligence of the solution further.

The system level advancements would mainly be enhancements for the acquirement of the calendar events as well as a more intelligent caching of calendar views on the client-side. Moreover, the future development of the used technologies further improves the possibilities, such as the access to the device capabilities directly from the Web runtime environment to acquire the local calendar view. In addition, the use of available contact and group information from the device native data sources could be used for the contact information when accessing group members' calendars. This would enhance the integration with native device resources and applications.

8. Conclusions

Much of the everyday work includes collaboration and co-operation. In addition, the mobility of the work is becoming increasingly important in the globalizing world. Therefore, the required collaboration tools, such as a group calendar should also be available when the users are moving around. In order to approach this challenge, a concept for *mobile peer-to-peer collaboration* platform was described in this thesis. By using the individual devices as the information sources in collaborative applications the information is always up-to-date and usable. In addition, storing this information locally and sharing it when needed will avoid the need for data synchronization to keep the information updated. Moreover, this way it is possible to benefit from the implicit use of mobile devices for personal information management in a collaborative manner.

To enable this concept, some emerging technologies were consulted during the work. It was discovered that the Web technologies provide a distributed, extendible and flexible architecture for loosely coupled systems, thus they give a suitable platform for building collaboration tools as well [Jerstad et al., 2005]. They also provide an interoperable platform that can be used via a variety of devices. Therefore the proposed approach is to use lightweight Web services for mobile collaboration. The collaboration is done directly between mobile devices in a distributed manner without the need for centralized information storage. Therefore the hypothesis formed to be that *the available Web technologies can be used to enhance mobile collaboration*.

After these background investigations, concept architecture was designed. The main goals for the design were to meet the requirements for a mobile collaborative system, as well as to comply with the requirements for Web services architecture. These requirements served as the basis when formalizing the proposed approach. To realize this, a set of novel requirements for *mobile peer-to-peer collaboration* platform was created. After the architectural design, the concept needed to be proven by implementation. This implementation linked the concept better with the current technological developments. It also worked as the testing ground for the limitations and constraints in the application domain. The prototype system also enabled the testing with end-user's in order to get the valuable feedback for future development. As an overall result the concept was proven feasible by an end-user study.

As a summary of the work, the main contribution in this thesis has been to design and implement a platform for mobile collaborative work using the

emerging Web technologies. With a combination of constructive and empirical research methods, the approach was considered feasible and the hypothesis was proved correct. Therefore, it can be claimed that this work has successfully introduced and evaluated a novel concept for enhancing mobile collaboration. Although it can be concluded that much of the work in this thesis was done on the implementation task for realizing the concept, most of the actual innovation was done on the formalization of the concept. The main challenge in the work has been to combine knowledge from the wide range of related work. Moreover, now when the technology is available, tested and ready to use, the challenge is to find the right direction where to use, extend and improve it.

Acknowledgements

I would like to thank my instructor Dr. Jani Mäntyjärvi for his advice and ideas during the work. I would also like to thank my thesis supervisor Prof. Roope Raisamo for his constructive comments when formulating this thesis. Special thanks go to VTT in Oulu for providing the possibility to work with interesting research and to devote the time to do my thesis. I am also grateful to the researchers at VTT who participated in the user study, which gave valuable input for this work. I would also like to acknowledge the very important feedback from Ms. Georgie Cash for proofreading and commenting on the grammar of this thesis.

Oulu, May 24th, 2009

Ville Antila

References

- Anderson, C. & Andersson, M. P., *The Long Tail*. Bonnier fakta, 2007.
- Antila, V. & Mäntyjärvi, J., Distributed RESTful Web Services for Mobile Person-to-Person Collaboration. *Submitted to Third International Conference and Exhibition on Next Generation Mobile Applications, Services and Technologies NGMAST'09*, 2009.
- Austin, D., Barbir, A., Ferris, C. & Garg, S., Web Services Architecture Requirements. *World Wide Web Consortium Recommendation 2004*, doi: <http://www.w3.org/TR/2004/NOTE-wsa-reqs-20040211/>, Vol. No. 1/26/2009, 2009.
- Baecker, R.M., *Readings in Human-Computer Interaction: Toward the Year 2000*. Morgan Kaufmann, 1995.
- Berners-Lee, T., Fielding, R. & Masinter, L., Uniform Resource Identifier (URI): Generic Syntax. *IETF Network Working Group RFC* doi: <http://tools.ietf.org/html/rfc3986>, 2005.
- Berson, A., *Client/server architecture*. McGraw-Hill, Inc. New York, NY, USA, 1996.
- Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. & Orchard, D., Web Services Architecture. *W3C Working Group Note* Vol. 11, pp. 2005-2001, 2004.
- Bos, B., Lie, H.W., Lilley, C. & Jacobs, I., Cascading Style Sheets, level 2 CSS2 Specification. *W3C Recommendations are available at <http://www.w3.org/TR>*, May, Vol. 12, pp. 80, 1998.
- Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H.F., Thatte, S. & Winer, D., Simple Object Access Protocol (SOAP) 1.1, W3C Note 08 May 2000. *Available via the World Wide Web at <http://www.w3.org/TR/SOAP>*, 2000.
- Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E. & Yergeau, F., Extensible Markup Language (XML) 1.0. *W3C Recommendation*, 2000.
- BSCW, BSCW, OrbiTeam Software GmbH [Publisher: OrbiTeam Software GmbH], [Online]. *Available: <http://public.bscw.de/>* [2009, 22/5/2009], 2009.
- Caceres, M., Widgets 1.0: Packaging and Configuration. *W3C Working Draft 14 April 2008* doi: <http://www.w3.org/TR/2008/WD-widgets-20080414/>, 2008.
- Caceres, M., Widgets 1.0 Requirements. *W3C Working Draft 05 July 2007* doi: <http://www.w3.org/TR/2007/WD-widgets-reqs-20070705/>, 2007.
- Caceres, M. & van Kesteren, A., Widgets 1.0. Specification. *Widgets 1.0. W3C working draft*, 2007.

- Carvalho, M.B., Karila, K., Pyykönen, M. & Hilska, J., *Calendaring Software in the Office: Better Than Paper? Groupware Coursework, University of Tampere*, 2008.
- Champion, M., Ferris, C., Newcomer, E. & Orchard, D., *Web Services Architecture. W3C working draft, Nov* doi: <http://www.w3.org/TR/2002/WD-ws-arch-20021114/>, 2002.
- Christensen, E., Curbera, F., Meredith, G. & Weerawarana, S., *Web Services Description Language (WSDL) 1.1. Web Services Description Language (WSDL) 1.1*, 2001.
- Coley Consulting, *What is a user requirement or specification* [Publisher: Coley Consulting], [Online]. Available: <http://www.coleyconsulting.co.uk/require.htm> [2009, 22/5/2009], 2009.
- Dojo Toolkit, *The Dojo Toolkit* [Publisher: Dojo Toolkit], [Online]. Available: <http://dojotoolkit.org/> [2009, 22/5/2009], 2009.
- Doodle AG, *Doodle AG* [Publisher: Doodle AG], [Online]. Available: <http://doodle.com/> [2009, 22/5/2009], 2009.
- Dourish, P. & Bellotti, V., *Awareness and Coordination in Shared Workspaces. CSCW '92: Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, doi: <http://doi.acm.org/10.1145/143457.143468>, 1992.
- Dustdar, S. & Gall, H., *Architectural concerns in distributed and mobile collaborative systems. Journal of Systems Architecture*, Vol. 49, No. 10-11, pp. 457-473, 2003.
- Dustdar, S., Gall, H. & Schmidt, R., *Web services for groupware in distributed and mobile collaboration. 2nd International Workshop on Distributed and Mobile Collaboration (DMC 2004)* Pp. 241. ISBN 1066-6192, 2004.
- Eckerson, W.W., *Three Tier Client/Server Architecture: Achieving Scalability, Performance and Efficiency in Client Server Applications. Open Information Systems*, Vol. 10, No. 1, pp. 3, 1995.
- Ellis, C.A., Gibbs, S.J. & Rein, G., *Groupware: some issues and experiences. Communication of ACM*, Vol. 34, No. 1, pp. 39-58. ISSN 0001-0782. <http://doi.acm.org/10.1145/99977.99987>, 1991.
- Erl, T., *Service-oriented architecture: concepts, technology, and design*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2005.
- Facebook, *Facebook Apps* [Publisher: Facebook], [Online]. Available: <http://www.facebook.com/apps/> [2009, 22/5/2009], 2009.
- Ferris, C., *What are Web services? Communications of the ACM* Vol. 46, No. 6, 2003.
- Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. & Berners-Lee, T., *Hypertext Transfer Protocol -- HTTP/1.1. IETF Request for Comments: 2616*, doi: <http://www.w3.org/Protocols/rfc2616/rfc2616.html>, 1999.

- Fielding, R.T., *Architectural styles and the design of network-based software architectures*. University of California, Irvine, ISBN 0-599-87118-0, 2000.
- Garret, J. J., Ajax: A New Approach to Web Applications, doi:
<http://www.adaptivepath.com/ideas/essays/archives/000385.php>,
 2005.
- Garrett, J. J., *The elements of user experience*. New Riders Indianapolis, Ind, 2002.
- Google Inc., Google Calendar [Publisher: Google Inc.], [Online]. Available:
<http://www.google.com/calendar/> [2009, 22/5/2009], 2009a.
- Google Inc., Google Docs [Publisher: Google Inc.], [Online]. Available:
<http://docs.google.com> [2009, 22/5/2009], 2009b.
- Google Inc., iGoogle [Publisher: Google Inc.], [Online]. Available:
<http://www.google.com/ig/> [2009, 22/5/2009], 2009c.
- Grudin, J., Computer-supported cooperative work: history and focus. *Computer*,
 Vol. 27, No. 5, pp. 19-26, 1994.
- Haas, H. & Brown, A., Web Services Glossary. *World Wide Web Consortium*, doi:
<http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>, 2003.
- Harjula, E., Ylianttila, M., Ala-Kurikka, J., Riekkilä, J. & Sauvola, J., Plug-and-play
 application platform: towards mobile peer-to-peer. *Proceedings of the 3rd
 international conference on Mobile and ubiquitous multimedia*. ACM New
 York, NY, USA, 2004. Pp. 63, 2004.
- Haveri, M., Blom, J., Virtanen, J., Tarkiainen, M. & Häkkinen, J., mCell: platform
 independent communication for small groups. *MobileHCI '07: Proceedings
 of the 9th international conference on Human computer interaction with mobile
 devices and services*, ISSN 978-1-59593-862-6. doi:
<http://doi.acm.org/10.1145/1377999.1378008>, 2007.
- Ilkka, J. & Vainio, C., Mobile Web Server Handbook, [Online]. Available from:
http://mymobilesite.net/files/MobileWebServer_Book_en.pdf, 2007.
- Jacobs, I., The W3C Technology Stack. *World Wide Web Consortium*, doi:
<http://www.w3.org/Consortium/technology>, 2008.
- Jacobs, I. & Walsh, N., Architecture of the World Wide Web. W3C
Recommendation 15 December 2004, doi:
<http://www.w3.org/TR/2004/REC-webarch-20041215/>, 2004.
- Jacobson, I., Christerson, M., Jonsson, P. & Overgaard, G., *Object-oriented
 software engineering: a use case driven approach*. Addison-Wesley, 1992.
- Jerstad, I., Dustdar, S. & Thanh, D., A service oriented architecture framework
 for collaborative services. *14th IEEE International Workshops on Enabling
 Technologies: Infrastructure for Collaborative Enterprise*, 2005, 2005.
- Johansen, R., *Groupware: Computer support for business teams*. The Free Press
 New York, NY, USA, 1988.

- Jones, N., Nokia Widgets Will Encourage S60 Mobile Services. *Gartner Research*, doi:
http://www.gartner.com/resources/148000/148087/nokia_widgets_will_encourage_148087.pdf, 2007.
- jQuery, jQuery: The Write Less, Do More, JavaScript Library [Publisher: jQuery], [Online]. Available: <http://jquery.com/> [2009, 22/5/2009], 2009a.
- jQuery, jQuery, Microsoft, and Nokia [Publisher: jQuery], [Online]. Available: <http://blog.jquery.com/2008/09/28/jquery-microsoft-nokia/> [2009, 22/5/2009], 2009b.
- JSON.org, JavaScript Object Notation (JSON) [Publisher: JSON.org], [Online]. Available: <http://www.json.org/> [2009, 22/5/2009], 2009.
- Kaar, C., An introduction to Widgets with particular emphasis on Mobile Widgets., doi:
<http://www.symbianresources.com/tutorials/techreports/widgets/kaar07widgets.pdf>, 2007.
- Koskela, T., Kostamo, N., Kassinen, O., Ohtonen, J. & Ylianttila, M., Towards Context-Aware Mobile Web 2.0 Service Architecture. *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2007. UBICOMM '07*, pp. 41-4810.1109/UBICOMM, 2007.
- Kotonya, G. & Sommerville, I., *Requirements engineering: processes and techniques*. Wiley, 1998.
- Lämsä, A., Distributed Dynamic Threads. *Master's Thesis, University of Oulu*, 2008.
- Le Hors, A., Le Hegaret, P., Wood, L., Nicol, G., Robie, J., Champion, M. & Byrne, S., Document Object Model (DOM) Level 2 Core Specification. *W3C Recommendation*, 2000.
- Lewis, J. R., IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. *International Journal of Human Computer Interaction* Vol. 7, pp. 57-57, 1995.
- Mesbah, A. & Deursen, A., An Architectural Style for Ajax. *IEEE Computer Society*, 2007.
- Mobiforge, PAMP: Personal Apache MySQL PHP [Publisher: Mobiforge], [Online]. Available: <http://mobiforge.com/developing/story/pamp-personal-apache-mysql-php> [2009, 22/5/2009], 2009.
- Mod_python Project, Mod_python - Apache/Python Integration [Publisher: Mod_python Project], [Online]. Available: <http://www.modpython.org/> [2009, 22/5/2009], 2009.
- Newton, A., *MooTools Essentials: The Official MooTools Reference for JavaScript™ and Ajax Development (Firstpress)*. Apress, 300. ISBN 1430209836.
http://isbndb.com/d/book/mootools_essentials_the_official_mootools_reference_for_java, 2008.

- Neyem, A., Ochoa, S.F. & Pino, J.A., Supporting Mobile Collaboration with Service-Oriented Mobile Units. *Lecture Notes in Computer Science*, Vol. 4154, pp. 228, 2006.
- Nokia Oyj, Nokia Easy Meet | Nokia Beta Labs. Available: <http://betalabs.nokia.com/betas/view/nokia-easy-meet> [2009, 22/5/2009], 2009.
- Nokia Research Center, S60WebKit [Publisher: Nokia Research Center], [Online]. Available: <http://opensource.nokia.com/projects/S60browser/> [2009, 22/5/2009], 2009a.
- Nokia Research Center, Python for S60 [Publisher: Nokia Research Center], [Online]. Available: <http://opensource.nokia.com/projects/pythonfors60/> [2009, 22/5/2009], 2009b.
- Oram, A., *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. Vol. 32. 57, 2003.
- O'Reilly, T., What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software. *Communications & Strategies*, No. 1, p. 17, First Quarter 2007, doi: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1008839, 2005.
- Oulasvirta, A., Tamminen, S., Roto, V. & Kuorelahti, J., Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile HCI. *Portland, Oregon, USA: ACM*, 2005.
- Pautasso, C., Zimmermann, O. & Leymann, F., Restful web services vs. 'big' web services: making the right architectural decision. *WWW '08: Proceeding of the 17th international conference on World Wide Web*. Beijing, China, New York, NY, USA: ACM. Pp. 805. ISBN 978-1-60558-085-2. <http://doi.acm.org/10.1145/1367497.1367606>, 2008.
- Pemberton, S., Althaim, M., Austin, D., Boumphrey, F., Burger, J., Donoho, A.W., Dooley, S., Hofrichter, K., Hoschka, P. & Ishikawa, M., XHTML 1.0: The Extensible HyperText Markup Language. *World Wide Web Consortium Recommendation xhtml1*, January, 2000.
- Pervilä, M., Performance of Ajax applications on mobile devices. *Master's Thesis, University of Helsinki*, 2008.
- Preece, J., Rogers, Y. & Sharp, H., *Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons, Inc. New York, NY, USA, 2002.
- Prototype (JS), Prototype JavaScript framework: Easy Ajax and DOM manipulation for dynamic web applications [Publisher: Prototype (JS)], [Online]. Available: <http://www.prototypejs.org/> [2009, 22/5/2009], 2009.
- Reif, G., Kirda, E., Gall, H., Fenkam, P., Picco, G.P. & Cugola, G., A Web-Based Peer-to-Peer Architecture for Collaborative Nomadic Working. *WETICE '01: Proceedings of the 10th IEEE International Workshops on Enabling Technologies*. Washington, DC, USA: IEEE Computer Society, Pp. 334. ISBN 0-7695-1269-0, 2001.

- Richardson, L. & Ruby, S., *Restful web services*. O'Reilly, ISBN 9780596529260, 2007.
- Ruby on Rails, Ruby on Rails [Publisher: Ruby on Rails], [Online]. Available: <http://rubyonrails.org/> [2009, 22/5/2009], 2009.
- S60, S60 platform 3rd Edition Overview [Publisher: S60], [Online]. Available: http://www.s60.com/pics/pdf/S60_3rd_Ed_2007.pdf [2009, 22/5/2009], 2009a.
- S60, S60 platform 5th Edition Overview [Publisher: S60], [Online]. Available: <http://www.s60.com/life/thisiss60/s60indetail/softwareversions/5thedition> [2009, 22/5/2009], 2009b.
- S60, S60 Info [Publisher: S60], [Online]. Available: <http://www.s60.com/life/thisiss60/s60indetail/technologiesandfeatures/webruntime> [2009, 22/5/2009], 2009c.
- Saroiu, S., Gummadi, P.K. & Gribble, S.D., A measurement study of peer-to-peer file sharing systems. *Proceedings of Multimedia Computing and Networking*. Vol, 2002.
- Siikarla, M., Laitkorpi, M., Selonen, P. & Systs, T., Transformations Have to be Developed ReST Assured. *Theory and Practice of Model Transformations*, doi: <http://www.springerlink.com/content/2kn4412184w46243>, 2008.
- Srirama, S., Jarke, M. & Prinz, W., Mobile Host: A feasibility analysis of mobile Web Service provisioning. *4th International Workshop on Ubiquitous Mobile Information and Collaboration Systems, UMICS*. Pp. 942–953, 2006a.
- Srirama, S., Jarke, M. & Prinz, W., Mobile web service provisioning. *Telecommunications, 2006. AICT-ICIW'06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*. Pp. 120, 2006b.
- Taivalsaari, A., Mikkonen, T., Ingalls, D. & Palacz, K., Web Browser as an Application Platform: The Lively Kernel Experience. *Sun Microsystems Laboratories Technical Report SMLI-TR-2008-175, January*, 2008.
- The WebKit Project, The WebKit Open Source Project [Publisher: The WebKit Project], [Online]. Available: <http://webkit.org/> [2009, 22/5/2009], 2009.
- Wikipedia, Blog [Publisher: Wikipedia], [Online]. Available: <http://en.wikipedia.org/wiki/Blog> [2009, 22/5/2009], 2009a.
- Wikipedia, Internet forum [Publisher: Wikipedia], [Online]. Available: http://en.wikipedia.org/wiki/Internet_forum [2009, 22/5/2009], 2009b.
- Wikipedia, Collaborative blog [Publisher: Wikipedia], [Online]. Available: http://en.wikipedia.org/wiki/Collaborative_blog [2009, 22/5/2009], 2009c.
- Wikipedia, Mashup (web application hybrid) [Publisher: Wikipedia], [Online]. Available: [http://en.wikipedia.org/wiki/Mashup_\(web_application_hybrid\)](http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)) [2009, 22/5/2009], 2009d.

Wikman, J., Nurminen, J.K., Kokkinen, H., Muilu, P. & Heikela, M., Mobile Web Application Development Stack. *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*. Pp. 1246, 2008.

XML.com, XML.com: Converting Between XML and JSON [Publisher: XML.com], [Online]. Available: <http://www.xml.com/lpt/a/1658> [2009, 22/5/2009], 2009.

Yahoo! Inc., The Yahoo! User Interface Library (YUI) [Publisher: Yahoo! Inc.], [Online]. Available: <http://developer.yahoo.com/yui/> [2009, 22/5/2009], 2009.