NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY, MATHEMATICS AND ELECTRICAL ENGINEERING
DEPARTMENT OF TELEMATICS

URD

An Agenda-system for mobile absentee marking and group scheduling.

Svein-Magnus Sørensen
December 2005

TTM4730
Specialization Project Assignment
Networked Services and Multimedia Systems

PREFACE

This report is written as a specialization project at the Norwegian University of Science and Technology, Department of Telematics, between August 2005 and December 2005.

Work on the project and the resulting report is the primary form of evaluation in the course TTM4730 Networked Services and Multimedia Systems, worth 23,5 ECTS-credits (75% of semester workload). The course is a specialization subject available in the final year of the Master of Science degrees in Telematics that are available at NTNU.

The project a part of the ongoing work at the "Program for Advanced Telecommunication Services" (PATS). PATS is a joint research program between NTNU, Telenor, Sintef and Ericsson aiming to establish platforms, frameworks and methods to support fast and secure development of advanced hybrid services in telecommunication networks.

I would like to thank both Professor Lill Kristiansen at NTNU and my supervisor Frode Flægstad from Telenor Research for their help and support during the work with this project. Also I would like to thank the other students working on projects closely related to mobile absentee marking alongside me, namely Mari Heibø, Jahn Arne Johnsen, Håkon Vestmoen and Ola Pedersen for their input.

Trondheim, December 2005.

Svein-Magnus Sørensen

TABLE OF CONTENTS

PREFACE .... **II**

TABLE OF CONTENTS .... **III**

TABLE OF CHARTS AND ILLUSTRATIONS .... **VI**

ABBREVIATIONS AND DEFINITIONS .... **VII**

ABSTRACT .... **IX**

1. INTRODUCTION **1**

1.1. BACKGROUND 1
1.2. LIMITATIONS IN THIS WORK 2
1.3. REPORT OUTLINE 2

2. DESCRIBING AN AGENDA-SYSTEM **4**

2.1. USER SCENARIOS 4
2.1.1. Meeting planner with extras *4*
2.1.2. Handling locations and missed flights *7*
2.1.3. New acquaintances *9*
2.1.4. Communications handling *11*
2.1.5. Status checking *13*
2.2. USER-INTERACTION 15
2.2.1. Base functionality in terminals *15*
2.2.2. Computer terminals *15*
2.2.3. Telephones and PDAs *16*
2.2.4. Other mobile smart-devices *17*
2.3. A UNIVERSAL SERVICE 18
2.3.1. Ubiquitous availability *18*
2.3.2. Unlimited connections *18*
2.3.3. Seamless integration *19*
2.3.4. Intelligent scheduling-help *19*

3. RELATED TECHNOLOGIES 20

3.1. EXISTING CALENDAR-SYSTEMS 20
3.1.1. Microsoft Outlook and Exchange Server 20
3.1.2. IBM Lotus Notes and Domino Express 20
3.1.3. Novell Groupwise 21
3.1.4. Apple iCal, iSync and .Mac 21
3.2. OTHER RELEVANT TECHNOLOGIES 21
3.2.1. vCalendar / iCalendar 21
3.2.2. SyncML 22
3.2.3. X.500 / LDAP 22
3.2.4. Parlay / OSA 22

4. REQUIREMENTS AND DESIGN OF URD 23

4.1. USER REQUIREMENTS 23
4.1.1. User groups 23
4.1.2. Functionality required by end-users 24
4.1.3. Functionality required by group administrators 25
4.1.4. Other non-functional requirements 25
4.2. THE NETWORK BASED AGENDA-SERVICE 26
4.2.1. Outline 26
4.2.2. Addressing and address lookup 27
4.2.3. Agenda-server functionality 29
4.2.4. Communication-protocols 37
4.2.5. Integration of third party services 39
4.2.6. Intelligent agents and agenda-updates 40
4.2.7. In the future: Mobile P2P servers 41
4.3. AGENDA-SYSTEM CLIENTS AND TERMINALS 42
4.3.1. Local computer-applications 42
4.3.2. Centralized server-side clients 42
4.3.3. Personal Digital Assistants and Mobile phones 43
4.3.4. Dial-in by regular phone or text-messages 44
4.3.5. In the future: Autonomous smart devices and P2P URD. 45
4.4. SECURITY CONSIDERATIONS 46

5. BUSINESS MODELS 47

5.1. SWOT ANALYSIS OF THE URD SYSTEM 47
5.1.1. Strengths 47
5.1.2. Weaknesses 49
5.1.3. Opportunities 49
5.1.4. Threats 50
5.2. OPERATORS 53
5.2.1. URD Corporation 53
5.2.2. Independent groups 55
5.2.3. Network-providers 56
5.3. REVENUE STREAMS 57
5.3.1. Pay-per-use 58
5.3.2. Subscription-models 59
5.3.3. One-time fee 59
5.3.4. Software licenses 60
5.3.5. Directed advertising 61
5.3.6. Combinations of various models 62
5.4. CUSTOMER BASE AND EARNINGS POTENTIAL 64

6. FINDINGS AND CONCLUSIONS 66

6.1. SYSTEM DESIGN AND TECHNOLOGY 66
6.1.1. The URD service 66
6.1.2. URD client terminals and interfaces 68
6.2. BUSINESS MODELS 69

7. FUTURE WORK 70

8. REFERENCES 72

TABLE OF CHARTS AND ILLUSTRATIONS

TABLE 2.1: ROLES AND ENTITIES INVOLVED IN SCENARIO 1 6

TABLE 2.2: ROLES AND ENTITIES INVOLVED IN SCENARIO 2 8

TABLE 2.3: ROLES AND ENTITIES INVOLVED IN SCENARIO 3 10

TABLE 2.4: ROLES AND ENTITIES INVOLVED IN SCENARIO 4 12

TABLE 2.5: ROLES AND ENTITIES INVOLVED IN SCENARIO 5 14

TABLE 4.1: FUNCTIONAL REQUIREMENTS FOR USERS 24

TABLE 4.2: FUNCTIONAL REQUIREMENTS FOR ADMINISTRATORS 25

TABLE 4.3: NON-FUNCTIONAL SYSTEM REQUIREMENTS 25

TABLE 5.1: COMPARISON OF REVENUE STREAM MODELS 57

FIGURE 4.1: COMPONENT LAYOUT OF AN URD SERVER INSTALLATION *30*

FIGURE 4.2: SAMPLE MESSAGE SEQUENCE CHART FOR THE URD CORE-SYSTEM AND SUBSYSTEMS *34*

FIGURE 4.3 CLIENT SYSTEM FOR URD *44*

FIGURE 5.1: TREE MODEL OF A DISTRIBUTED URD SYSTEM WITH SERVERS, PLANS AND USERS *52*

FIGURE 5.2: VALUE CHAIN ANALYSIS OF THE COMBINED REVENUE STREAM MODEL *63*

ABBREVIATIONS AND DEFINITIONS

3GPP	Third Generation Partnership Project
ACL	Access Control Lists
Agenda-service	The server side parts of an Agenda-system
Agenda-system	A universal system for both personal scheduling and calendar-sharing between any numbers of individuals belonging to any existing groups or none at all.
AJAX	Asynchronous JavaScript and XML
Associates	The group of people using the same server as you. When a server is installed for internal use in a company then common group policies for all employees on the server can be specified.
CAI	Command Access Interface
Calendar-system	A system using individual but shared calendars between a single group of people for cooperation purposes. Examples of such systems are Microsoft Exchange and Lotus Notes.
CIP	Command Interface Protocol
DAP	Directory Access Protocol, part of the X.500 suite.
DBMS	Database Management System
DNS	Domain Name System
DoS	Denial of Service (Usually performed by swamping a service with false requests)
DoT	Department of Telematics (NTNU)
ECTS	European Credit Transfer System
ETSI	European Telecommunications Standards Institute
GUI	Graphical User Interface
Host-servers	The URD-service operated by the URD Corporation. These are the default servers to use if there are no third-party servers available for your address.
IANA	Internet Assigned Numbers Authority
iCal	Short for of iCalendar, the Internet calendar specification, version 2 of vCal.
ID	Identification
IETF	The Internet Engineering Task Force
IM	Instant Messaging
IMC	Internet Mail Consortium
IMEI	International Mobile Equipment Identity
ISP	Internet Service Provider
ITU-T	International Telecommunications Union – Standards for Telecommunication
JAIN	Java API's for Intelligent Networks
LDAP	Lightweight Directory Access Protocol, inspired by and compatible with X.500
MAM	Mobile Absentee Marking
MMS	Multimedia Message Service
MSC	Message Sequence Chart
NTNU	Norwegian University of Science and Technology

OMA	Open Mobile Alliance
OSA	Open Service Access
P2P	“Peer to Peer” meaning all connected parties are equals in capabilities and power.
PATS	Program for Advanced Telecommunication Services
PDI	Personal Data Interchange
Personal scheduler	A simple calendar intended for use by a single person, like a normal almanac or the calendars available in mobile phones and PDA’s.
PGP	Pretty Good Privacy, an encryption standard originally by Phil Zimmermann
PIM	Personal Information Manager
PKI	Public Key Infrastructure
Posse	The group of people that you have been in contact with and are part of your group policies.
POTS	Plain Old Telephone Service
SIP	Session Initiation Protocol
SMS	Short Message Service (Used primarily within the GSM mobile phone networks)
SWOT	Strengths Weaknesses Opportunities Threats (Common form of business analysis)
SyncML	Synchronization Markup Language
TCP/IP	Transmission Control Protocol / Internet Protocol
URD	(1) One possible design of an Agenda-system.
	(2) One of the three goddesses of fate who are spinning the web of life in Nordic mythology.
URD Corporation	The group or company that will host the original URD-system.
VC	Venture Capitalist
vCal	Short for vCalendar, a standard format for PDI by the IMC
VoIP	Voice over IP (Internet Protocol)
WML	Wireless Markup Languge
WWW	World Wide Web
X.500	Networking standards for directory services, developed by ITU-T as ISO 9594
X.509	PKI and digital certificate authentication framework by the IETF for the X.500 suite.
XML	Extensible Markup Language

ABSTRACT

URD is the idea of a location-aware calendar system with both interactive and automatic modification of expected presence based on personal preferences and actual status.

The idea consists primarily of a calendar-service that may be accessed and updated through a variety of terminals and channels. These include the current mobile phone-network as well as advanced PDA's and computer terminals, but new technologies are also considered and planned for.

The general idea is that people shall be able to review and register planned locations and activities in the system so that others can check their planned status and use it to organize common meetings or to check their current availability and location. The system is company-independent to allow access to the calendars of colleagues working for other companies than your own, and this report also takes into account privacy and access control issues while retaining simplicity of use.

In addition to the information registered manually by the users, the system can monitor and register everyone's current status through secondary channels like IM-services, Outlook-servers and GPS-modules so that it can modify the calendar or notify the affected persons and others if their actual status doesn't match the planned one and a change is needed. This data can also be used to provide dynamic replies to communication attempts where various degrees of information about a person's status and availability are returned based on their mutual relationship or personal preferences as defined using group policies.

This project details how a service like the one described above can be realized using current technology based on the design principles behind the existing email system, as well as how new technology can provide added advantages and new features to the service. Also the project looks into some of the commercial aspects related to the service, like viable business models, competition, possible revenue streams and potential operators.

1. INTRODUCTION

1.1. Background

During the last few decades we have seen the usage and power of computers and the Internet becoming more and more ubiquitous. The obvious advantages they provide in administration and coordination of work and pleasure have secured the place of computers and communications-technology in many aspects of everyday life. However there are still much potential for improvement in many existing services, and the ongoing development and miniaturization of technology are constantly opening up new possibilities for the future.

In companies, families and most other groups of people there are usually a need to coordinate common activities between multiple persons so that the timing and location of an activity is well suited to as many as possible of the relevant people. For small groups such planning is usually a simple task, but if the groups are large there is a greater chance that some of the members cannot keep track of their activities so they get double booked, or that one or more of the essential persons are unavailable for the planning-session. This may lead to difficulties in deciding when to schedule an event and could result in there being important changes performed at the last-minute to accommodate the insufficient planning, or in the worst case the entire event might be cancelled if some essential people couldn't make it.
Currently there are a host of different calendar-systems available to help people organize their lives, from the small personal ones in cell phones and personal organizers used to plan and remember schedules for a single person, to the large corporation-wide network-based applications like Microsoft Outlook where one can plan activities to fit the schedule of every participant through interactively accessing the actual personal schedule of each participant, where all changes and invitations to new events are updated live in a common database. However there are still many shortcomings in the available systems, especially related to updating the calendar and sharing the functionality with anybody. This report will try to outline what an improved universal Agenda-system could be like, how it could be implemented and what sorts of business models could be used for its deployment.

1.2. Limitations in this work

Due to the limited scope and time available for this assignment it would be near impossible to complete all the potential work associated with the development of an Agenda-system.
For this reason I have decided to only research what an Agenda-system is and how it can be realized technologically, as well as look into potential business models and revenue-schemes for the realization of the system by a commercial entity. I have also made some comments about security issues, but full research into this as well as detailed design, protocol development and implementation of the Agenda-system described are left for others to complete at a later time.

1.3. Report outline

This report will look into the current status of technology being used to coordinate personal schedules, and detail how a new kind of scheduling-system can be created to become an omnipresent and somewhat intelligent tool available through communications-networks everywhere. The report will also look into the commercial aspects of an Agenda-system.

Chapter 2 provides a description of what an Agenda-system should be capable of doing, and how this may be perceived by users of the system in their daily life. This is detailed primarily through a series of scenarios and some analysis of them, but some more information will also be detailed on the functionality of servers and clients in an Agenda-system.

Chapter 3 contains descriptions of existing calendar-systems and technologies that can be seen as competitors for an Agenda-system and for technologies that may be used as building blocks when implementing such a system.

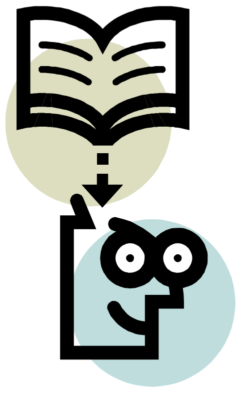
Chapter 4 looks into what technological requirements must be met for an Agenda-system system to be created, and which features users and administrators require. This chapter also contains a high-level design of an Agenda-system implementation named URD, with details on servers, directories, plug-ins and the clients that may be used. The chapter ends with a take on important security considerations related to the deployment of an Agenda-system.

Chapter 5 looks at the business side of things, discussing what kinds of business models and revenue streams that could be viable if one were to realize this kind of system on a large scale through a commercial entity. This chapter also considers potential competition from other types of scheduling-systems.

Chapter 6 is a summary of the lessons that can be learned from this report and makes some conclusions that future work in this field can be based on.

Chapter 7 provides some ideas and pointers for future work on the subject of Agenda-systems that will complement this report and pave the way towards the realization of a functioning system that can be deployed sometime in the future.

And finally in **Chapter 8** there are a list of the references and resources that I have used for researching and writing this report.



2. DESCRIBING AN AGENDA-SYSTEM

To create an Agenda-system one must first know what an Agenda-system is. This chapter contains a set of user scenarios and other descriptions of what I envision that an Agenda-system should be like. This section is partly based on the original MAM-description for this project assignment as provided by Frode Flægstad at Telenor Research.
The analytical method used on the scenarios is based on the contents of [Telenor].

2.1. User Scenarios

Here I have provided a series of user stories to show how URD can potentially improve coordination among people through new services with better availability and intelligence. I have used Microsoft Exchange as a baseline of the current state of affairs in shared-calendar applications for these scenarios as it is a widely used program in many large and small businesses. Despite many of these scenarios involving businesses and working professionals, most of the features can equally well be used within a group of friends or families.

2.1.1. Meeting planner with extras

Present day:

Alex is working as a district manager in a large company that uses a shared electronic calendar. He wants to have a meeting with some of the employees spread across several departments and reviews the schedules of everyone to pick a suitable time for the meeting. After he finds a good time he sends a digital invitation in Exchange to each of the participants with information on the meeting. Those that accept the invitation will have the timeslot automatically reserved for this meeting in the central Exchange server. So far so good, but since the departments are spread out geographically some of the employees need to travel a fair bit to get to the meeting, and must also register the time required as a separate meeting manually so that they aren't invited to something else during that timeslot. This could lead to problems if the meeting is cancelled or moved as the travel-entries would need to be updated manually once again. Another inefficiency is that a person can very well be available for phone-calls or other work-related tasks depending on the mode of travel, but this is will not be immediately obvious from looking at that persons schedule.

The result is that several people wasn't able to attend the meeting because they were already busy just before or after the meeting and didn't have any space left for traveling. Also several others lost many hours of time that could have been spent working if others had known they were partly available while traveling.

The future with URD:

Again we have Alex inviting everybody to the meeting electronically through the shared calendar-application on his computer, but this time while he is reviewing everybody's schedule he will notice that a few people have travel-requirements that needs to be fit into the schedule, so he pushes the meeting back a few days to accommodate everyone. The system knows the travel-times based on previous data that users have entered or by looking at the distance between locations if the meeting is held somewhere new. When people get the invitations and accept, the URD-system also asks what mode of travel they will be using. The required travel-time is then automatically reserved in the calendar and the level of availability during traveling is set automatically based on the mode of travel, but can be changed manually through an advanced terminal if required. This travel-time is linked to the meeting itself in case it should be cancelled or moved, so it will not be necessary to update it manually.

Capabilities of the system:

URD must be able to know or be told the location of any event, and it must calculate the required travel-time between these locations. The travel times for all participants must be shown to a user while planning an event time so it can be taken into consideration. Users that are adding invitations or personal items to their schedule must be told of common travel-times between their previously scheduled location and the current event and allowed to change these to match the preferred mode of travel. Any double bookings due to travel-times should result in a warning to the user so he can modify his schedule accordingly. Travel-times are linked to their event and are recalculated in case of rescheduling or cancellations. Recalculations must naturally result in a warning to affected users if the new time crashes with other plans.

Table 2.1: Roles and entities involved in scenario 1

Actors	Alex (Originator of invitation)
	Unspecified group of friends (Recipients)
Entities	URD-applications (User interfaces)
	URD-server (Schedule repository and directory gateway)
	User agents (Performing automatic operations)
	Directory service (User-account location lookup)
Information	Personal location-aware schedules
	Information regarding the meeting
	Data regarding travel between locations
	Changes regarding the meeting

Chain of events:

- Alex accesses URD application on his desktop computer.
- Alex adds new event and chooses who should be invited from his local cache.
- The URD-application connects to the default (local) URD-server over the company network and requests the home location of the selected people.
- The URD-server checks a directory service for the home-server of the selected people, and returns the found locations to the URD-application.
- The URD-application synchronizes the local schedule-cache with the home-server for each person, which in this case is the local company server for everyone.
- The URD-application calculates travel-times and availabilities on the fly for reviewing by Alex along with the times he have available for the meeting.
- When a time has been selected and information provided an invitation to the meeting is sent to the home-server of all invitees.
- Each invitee are notified of the invitation through his or her preferred means of communication by their user agent, and given the choice to accept or decline.
- Accepted meetings are registered in a users schedule after checking the originators server for any changes, and the originators server is informed of the acceptance.
- Later changes made to the event by the originator are broadcasted to all invitees by the server-agent receiving the modifications.

2.1.2. Handling locations and missed flights

Present day:

Melissa is working three days of the week in Oslo and the last two days and weekends at her home in Bergen. However there are frequently problems when she makes changes to the schedule or when people who don't know her schedule invite her to meetings in one city while she is in the other. There have been plenty of times when Melissa has missed out on important meetings because of this.

Even more frustrating however are the days in winter when extreme weather-conditions cause long delays or cancellations to the flights she rely on to get around. While her schedule for the entire day may be booked with items she can't attend in Oslo, she is sitting idle in Bergen waiting for the next flight without getting any work done at all. During the really bad storms she might lose out on days of work that could have been fully booked with meetings at the Bergen-office if just someone had known she was there.

The result is that Melissa has to spend lots of time on the phone telling people where she is and what has happened, instead of being able to work.

The future with URD:

Now when Melissa is commuting between Oslo and Bergen her planned location is at all time registered in the URD-calendar. This way anyone who wants to invite her to a meeting can see when and where she is available and pick dates accordingly, so she will never again miss a meeting because someone where unaware of her commuting schedule.

Another added advantage is the intelligent updating in URD. When Melissa misses a flight her actual whereabouts will soon divert from her planned schedule. But as she is running an URD-application on a GPS-enabled PDA the system can pick up on this mismatch and provide Melissa with the option of clearing all or part of her schedule in Oslo and perhaps make her available for new meetings in Bergen. On Melissa's request the system can also send notifications to the organizers of any cancelled activities with a message that Melissa will not be able to attend, and previously received invitations in Bergen can be presented to her again since she is now able to attend.

Capabilities of the system:

There must be an option to register your expected location for large periods of time without the location being linked to a specific event. Default general locations (like a city) should be specified at all times, with more detailed or modified information provided in connection to events and traveling.

Any kind of smart-devices can provide information to the system. Intelligent software-agents in the system can examine this information and make modifications to the schedule themselves or notify the user of inconsistencies if the information does not match the scheduled plan or expected values. The agents should also be able to notify other users that are registered at the affected events about the changes.

Table 2.2: Roles and entities involved in scenario 2

Actors	Melissa (Originator) Co participants at events in Oslo (Recipients)
Entities	URD-enabled PDA (User interface) URD-server (Schedule repository) User agent (Performing automatic operations)
Information	Personal schedule for Melissa with locations Current location-information / Presence

Chain of events:

- Melissa has previously registered her expected locations with URD, and she has accepted a range of meetings in Oslo and synchronized the schedule to her PDA.
- A location-enabled URD-PDA monitors Melissa's location and compares it to the planned location from the URD-schedule. If a mismatch occurs the PDA alerts Melissa to this fact and prompts her to modify her schedule accordingly.
- Melissa chooses to modify her schedule by wiping the schedule for Oslo and changing the expected location for the day to Bergen on her PDA.
- When the updates are synchronized with Melissa's home-server, her user agent will notify the organizers of the cancelled events that she will not be participating.
- The agent can also at this time notify Melissa of any previously declined invitations to meetings in Bergen on this day since she is now able to attend them.

2.1.3. New acquaintances

Present day:

Carl and Catherine are both working in large companies, and both are using current calendar-applications for time-scheduling in their daily work. They are also working within the same field and after being introduced at a party they figure it would be good to pool the resources of their respective teams to get some brainstorming going on, but planning a meeting is more hassle than usual. Despite both companies using the same software, they cannot easily invite participants from the other team directly in their usual system. It would all have to be handled manually twice, once for each company involved.

At another occasion Carl gets in touch with Peter, who is working independently but working together would be beneficial to both of them. However since Peter isn't connected to a big company he doesn't have access to a centralized calendar-application and can't be included into Carl's usual planning either. In this case all meetings and other cooperation would have to be handled manually between these two as well, with the downsides this may bring.

The future with URD:

Using a common Agenda-system like URD, Carl and Catherine don't have to struggle with updating the calendars separately in each of their companies anymore. Carl can now send invitations directly to Catherine who can forward them throughout the system to anyone on her team.

When Carl also wants to invite Peter, he can simply invite him as usual only needing to provide a few extra pieces of contact information for Peter, and the system will notify Peter that he has received an invitation and give some basic information on how he can customize the system to his needs. The system can also provide reminders for any events to people who are registered participants and provide a simple interface for group communications where each person can receive information through his or hers preferred channel.

Capabilities of the system:

Any users can invite any other user to schedule an event they are planning. Invitations are sent directly to each user's URD-account, which may be on any server operated by any group or company. Forwarding invitations to more users can be allowed by the original organizer of an event. When people accept an event they can be provided with the most current information available on the even. Further updates to an event will be issued to all currently registered participants. Users can be notified of updates through their preferred way of communication as registered on their URD-account.

People that are not currently users of URD can also be invited to the system by entering some kind of contact information for them, and the system will then create an account on a public server and inform the new user of how it can be managed.

Table 2.3: Roles and entities involved in scenario 3

Actors	Carl, Catherine (team managers in large companies) Catherine's team Peter (independent contractor) URD Corporation
Entities	URD-application (User interface) URD-servers (Schedule repository) User agents (Performing automatic operations)
Information	Contact information for the involved parties Information regarding the meeting Account management information for new users

Chain of events:

- Carl and Catherine meets and exchanges contact information.
- Carl sends an invitation to a meeting to Catherine through URD, which follows the procedure described in chapter 2.1.1 to locate Catherine. The invitation also includes a request to be linked with each other since there as been no prior contact.
- Catherine accepts, and forwards the invitation to the rest of her team. Based on the new link she can also define group policies for Carl.
- Carl meets up with Peter, and exchanges contact information with him.
- Carl sends an invitation to Peter through URD, but the local URD-server is unable to locate an existing account for the contact information provided. Because of this the invitation is forwarded to the root-servers maintained by the URD Corporation.
- The URD Corporation's server then creates an account for Peter with the information available, and informs Peter how he can use the system for future events.
- Peter learns how to use the system and accepts the invitation from Carl.

2.1.4. Communications handling

Present day:

Fredrick often needs to get in touch with his friend Frank immediately, but Frank is very busy and is usually never available when Fredrick tries to call. This can be very frustrating as the only information Fredrick gets is the generic voicemail telling him to leave a message. Similar problems can be imagined for most real-time communications channels and even for some other channels where you expect a timely reply as well.

The future with URD:

With an Agenda-system integrated with the voicemail functionality of various services, one can instead of having a generic reply use customized replies detailing where you are, what you are doing and when you will be available for calls. This information can then be further revised depending on who is calling, by modifying the voicemail-message depending on group policies so that your wife might just get a message saying that you are busy with work while a close colleague can be informed on exactly which task you are working on so he can decide whether or not to bother you himself.

Some of this functionality can be embedded in smart terminals so that some people are routed automatically to your voicemail while important calls will go through. They can even have various sound types and levels depending on the same group policies as mentioned above, so that you have more information to help you decide to accept it or not.

Capabilities of the system:

URD must implement a group-policy system to decide who can get what information. URD must also provide an interface through which voice-mail systems and other reply-systems can retrieve information to present as a reply to a communications attempt. The interface should require identification of the party initiating the communication so that the group-policies of URD can be used to decide what information should be disclosed.

URD-enabled terminals must also be able to use such policies to make decisions independently of the human user if they are programmed to do this.

Table 2.4: Roles and entities involved in scenario 4

Actors	Fredrick (Needs to contact Frank) Frank (Busy person)
Entities	URD-server User agent (Performing automatic operations) Telephone system Voicemail system Text-to-speech translator
Information	Current presence information on Frank Franks group policy information for Fredrick Fredrick's caller ID

Chain of events:

- Fredrick is having an emergency and tries to call Frank
- Frank is not available for a phone call so the call is routed to his voicemail.
- The voicemail-system makes an inquiry to Franks URD-server to get his current status with the caller ID of Fredrick.
- Franks User agent at the server looks up the best presence information available, and runs this by the group policy entry matching Fredrick to remove sensitive information that Fredrick should not have access to. The remainder of the information is returned to the voicemail system.
- The voicemail system runs the provided information through a Text-to-speech translator and plays the result to Fredrick to give him some clue of what Frank is up to and when he might be available.
- Fredrick can now use this information to either get a hold of Frank by some other means, or to decide that Frank's current affairs are more important than his own problems.

2.1.5. Status checking

Present day:

Michael is tired of studying for his exams, so he wants to hang out with his friend Nick, but he doesn't know if Nick is available. He could always give nick a call or send him a text message, but Nick said that he might be going to the movies so it would be a disruption to call him, and he probably wouldn't answer anyway. However Nick also said he might be staying at home to play online games, and Michael knows that when Nick is playing he doesn't answer the phone either, but he would be happy to have friends drop by.
The result is that Michael can't get in touch with Nick, even though Nick might very well be available. This means that Michael's only option is to drop by Nick's house to see if he is available, which would be a waste of good study-time if Nick has gone to the movies.

The future with URD:

If both Nick and Michael had been using URD all of the troubles above could have been easily avoided. All Michael would have to do is send a message to the URD-system requesting the current status of Michael. Since they are buddies Nick has allowed Michael to access all the information on his whereabouts, so the system replies with the best information available, which just happened to be that Nick's phone was turned off and the last reported location was outside the movie theater downtown.

Michael takes this information to mean that Nick went to the movies, so instead of spending a lot of time getting down to his place to check he can just stay at home to study some more or see if any of his other friends are available. Either way it has saved him a lot of hassle and perhaps an unnecessary trip to Nick's house and back.

Capabilities of the system:

URD should retrieve information from any and all sources possible to complement the scheduling-data so it can build an accurate profile of each users status. This information can then be provided to other users upon request based on the relevant group policies that define who can have access to what information.

Table 2.5: Roles and entities involved in scenario 5

Actors	Michael (Got some free time) Nick (Might be busy)
Entities	Any URD-client URD server Directory service Nick's user agent (Performing automatic operations)
Information	Information request from Michael Account information for Nick Presence information for Nick Nick's access control lists for Michael

Chain of events:

- Michael wants to know if Nick is available to hang out, and uses an URD-client to request information on Nicks whereabouts from the closest URD server.
- The URD server forwards the request to Nick's home-server after locating it from the directory service as described in chapter 2.1.1.
- Nick's user agent locates the requested information and runs it by the access control lists to ensure that Michael should be allowed the information.
- Michael receives the information from the user agent and decides based on it that Nick is likely to be at the movies so he should keep studying.

2.2. *User-interaction*

2.2.1. Base functionality in terminals

To be able to use the system efficiently so that it can provide an improvement over current systems, all terminals need to be able to handle a certain level of base functionality.
This functionality must include, but are not limited to:

- Adding and updating items in your own schedule
- Getting notifications from your own schedule
- Retrieval of current status for people you are connected to.

More advanced functionality that terminals should have if possible may include:

- Modifying default settings in your profile
- Reviewing calendars of multiple people concurrently.
- Sending event-invitations to other people.

These requirements mean that very simple terminals can become hard to use because of the multitude of choices available, but this can be mediated by using pseudo-intelligent algorithms to provide relevant choices that does much work automatically.

2.2.2. Computer terminals

The obvious way to interact with an Agenda-system is through a regular computer terminal. This system can be implemented in two possible ways, either as a web-service that you access through a regular web-browser or as a standalone application with its own communications protocols. Both approaches can be used successfully together as well since they have their own particular strengths and weaknesses. All of the ordinary functionality mentioned above can be implemented either way, and one could have a standalone application installed on your regular computers while the option to use a web-interface can be available if you are using another computer somewhere else.
The strength of the computer-based systems is that there are virtually no limitations on the functionality that can be implemented, so these can often provide a very advanced service and interface largely independent of the Agenda-service.

2.2.3. Telephones and PDAs

As computer terminals are still quite heavy and bulky they are not suitable for carrying around everywhere you go. Also the limited battery-life means that if you spend much time away from a fixed location you will likely need to access the Agenda-system at times when you cannot use your computer, so a natural step is to be able to access the system through Personal Digital Assistants or other handheld computers, but also through a mobile phone or even from a regular land-line.

For handheld computers and advanced mobile phones and PDA's this can be achieved by using a scaled down version of the same applications that are made for regular computers, as long as they have similar communications-capabilities. However for less advanced terminals the challenge is a bit more complicated.

An option for simple mobile phones is to use the regular SMS text messaging system for two-way communication with the Agenda-service. This could even be extended with a simple front-end if small applications are supported by the phone, or by using parsers that will allow the system to understand commands in a natural written language on the server-side.
On the other hand if a phone does not have SMS-support, as is the case with regular land-lines, a possible solution could still be developed by using a completely voice-controlled system or a combination of voice-feedback and touch-tone signals to let a user access the basic functionality of the system. This however would in most cases be quite cumbersome for the user and should only be provided as a backup solution for those that have no other options available. In the future when voice-recognition technology becomes mature it might be possible to simply tell the system your desires just like you do with other people, but then again it might be so common with advanced hand-held terminals by then that the support for ordinary telephones are no longer required.

2.2.4. Other mobile smart-devices

The future will most likely not be limited to the computers and mobile phones that we are familiar with today. There is constantly being developed new kinds of smart devices with a range of new sensors and capabilities that isn't currently available.

For instance are GPS-units being installed in everything from wristwatches to toasters today, and coupled with the connectivity of a mobile phone it could be used to let the Agenda-system track you and inform you if your current location falls out of synch with where you planned to be at any given time

Various kinds of body sensors are another possible technology that could allow the system to screen calls based on your current state of mind. For instance could you program the system to divert calls to your voicemail if you are sleeping, running, or otherwise physically unable to attend to them. This is something that also could have obvious uses in the health-care sector and elsewhere to monitor people or alert emergency services if there is an accident.

The possibilities here are endless, as anything that can provide information about you to the system will allow it to personalize its service even more. The more artificial intelligence that is implemented to make use of this information and make decisions on the users behalf, the less work it will be to keep on top of the daily flood of information and communication that the information society brings.

2.3. A universal service

2.3.1. Ubiquitous availability

By using anything from ordinary phones to regular computers, and especially mobile technology, an Agenda-system should be available from virtually anywhere at any time.
To fulfill this there must at least be a service that you can connect to over the Internet, preferably both with a server-side web service and the option of standalone clients. There must also be one or more phone numbers available that you can dial to access basic functionality from an ordinary phone, and there should be an SMS-gateway to support text-commands from simple mobile phones and similar terminals.

As new communications standards are continuously developed and some of these get deployed into widespread use, the system should be able to be extended to support both new protocols and equipment as time passes.
All of these interconnects are also required since the system should be able to alert its users of upcoming events and of any mismatches between the schedule and reality, and for the alerts to be effective they must support the kinds of communications that a user has available.

2.3.2. Unlimited connections

Since you can never know who you will meet tomorrow, and even less if they can be an asset to you or your company the Agenda-system should have no limits on who can be linked.
Even if a person doesn't have an existing account with the system an invitation from someone that is already using it will automatically create an account and inform the person on how to use the system. This means you will never have to consider if a person you send an invitation to is using the agenda-system or not, you just need to enter some form of contact information and the rest will be handled automatically.
When you send invitations to someone you will be registered as linked with those people. All the people you are linked to make up your "Posse", and they can be handled through group-policies as individuals or as a group. You can also define subgroups of your Posse to further create advanced group policies to handle information. This allows you to give those you choose access to more information about you than what is usually available to anyone in the world (Which by default would be very little or nothing at all).

2.3.3. Seamless integration

If you have existing accounts within Exchange or Notes based calendar-systems, or if you use other types of calendars then these can be synchronized with your Agenda-system account.
By providing the necessary information the Agenda-system can automatically keep in synch with your other systems so that other coworkers still are able to contact you through the old system as usual.
This will ensure a hassle free transition if you are using one or more schedulers already and would like to start using an Agenda-system instead. The Agenda-system itself should also have an open interconnect architecture allowing other systems to get information from it if they can provide the right credentials. While this could be a disadvantage for the Agenda-system it will allow for closer cooperation with those using other system that are not interested in switching, for instance when several users in a large organization prefer to use an Agenda-system instead of the corporate standard.

2.3.4. Intelligent scheduling-help

Scheduling in itself can be a hassle for many people, but updating your entire schedule when something unexpected comes up can be even more work.
In cases where parts of the Agenda-system detects that you do not follow your schedule it will automatically contact you and provide some choices to adjust your plan accordingly. How the plan will be adjusted depends on the available information the system has. For instance can the entire schedule for a day or a week be cancelled if you find yourself unable to reach the physical location of your scheduled appointments, and your base location should be updated to represent the place that you will actually be available. These things will of course not happen automatically but should be among several options for simple schedule-updating that the system can present to you if discrepancies are detected.
Another thing in this section is automatic travel-time calculations depending on your planned locations so that the travel-time will be automatically reserved and linked to each appointment. This way you cannot be double booked due to location-confusion and the system can not only notify you about a scheduled event, but also provide alerts for when you must leave to reach an appointment on time.
These are only some examples and only imagination sets limits for what can be performed by semi-intelligent agents in the servers or locally on clients.

3. RELATED TECHNOLOGIES

The Agenda-system as described is naturally not the first attempt at creating aids for scheduling, and there are also many other valuable technologies that can be used with great success as building blocks in Agenda-systems. Here I will present some of the major existing calendar-systems and a range of other technologies that are relevant for this report.

3.1. Existing calendar-systems

3.1.1. Microsoft Outlook and Exchange Server

Microsoft Outlook is a communications manager that can handle e-mail, Usenet, contacts, calendars and tasks. It is a complete PIM-system with receiving and sending invitations to events and notifications, built around the basic email functionality. The calendar supports viewing of multiple calendars concurrently to aid scheduling of events.
Outlook also supports plug-ins of various kinds that can add special functionality that you need, for instance is there a SMS-plugin available for Outlook from Telenor enabling you to send short text-messages directly from Outlook to any mobile phones. [TelenorSMS]
Outlook can also be extended by using Microsoft Exchange Server, a collaboration tool combining e-mail, file-sharing and calendar-sharing in a single application. This allows you to review the calendars of other people in your organization for scheduling purposes, and you get the possibility of mobility with everything available from Outlook Web Access and through synchronization with mobile smart devices at remote locations.
More information can be found at [Outlook] and [Exchange].

3.1.2. IBM Lotus Notes and Domino Express

A service very similar to Microsoft's offering, with Notes being the client side application providing access to e-mail, calendar, instant messaging and collaboration tools for small and medium sized organizations. Through Domino Express you get most of the same advantages that Microsoft Exchange Server provides, to the point where you can use Microsoft Outlook as a client side application for Domino instead of Notes.
More information can be found at [Notes] and [Domino].

3.1.3. Novell Groupwise

Novell Groupwise is also a similar service to Microsoft Exchange Server, providing integrated e-mail, instant messaging, scheduling, and task, contact and document management for clients within a business through a server based system. It also supports the use of Microsoft Outlook as a client-side application in addition to most of the other common features like schedule-reviewing and notifications. More information can be found at the [Novell] website.

3.1.4. Apple iCal, iSync and .Mac

iCal is a PIM-system for Apple OS X where you can create various personal calendars to help organize your own and your family's life. These calendars can then be uploaded to a .Mac account online for sharing with friends and coworkers, and you can also subscribe to their calendars so you can keep updated on their doings or check when everyone is available for an event. You can also subscribe to public online calendars and you can send invitations to events to other people. If you are a person on the go, you can use iSync to keep your calendar updated between multiple computers, as well as on your mobile phone or iPod so it is available wherever you go. More information can be found at [iCal] and [iSync].

3.2. Other relevant technologies

3.2.1. vCalendar / iCalendar

vCalendar is a standard for Personal Data Interchange (PDI) originally created by the Versit Consortium and later transferred to the Internet Mail Consortium for further development and promotion. It is a platform independent format for automated interchange of scheduling-information between different types of personal information managers (PIM) and schedulers. It can be used for exchanging any kind of calendar information through any channel, for example email or from websites, and to be imported or exported between any applications. This means that you can exchange such information between people using any number of completely different systems as long as all support the vCalendar-format.
Since version 1.0 of vCalendar development has been performed by the IETF and the name of the standard has been changed to iCalendar (iCal).
More information can be found at [vCal] and [iCal]

3.2.2. SyncML

The SyncML Initiative is a joint working group including Ericsson, Nokia, Palm Inc., Psion, Starfish Software, IBM, Lotus and Motorola founded in February 2000, with the goal to build a common industry-wide synchronization protocol. The Initiative has evolved to become the Open Mobile Alliance (OMA). [SyncML]
The work performed has resulted in two protocols; the Synchronization Protocol that describes the packets required to communicate changes between hosts [Sync], and the Representation Protocol that describes how the representation of data that is part of a synchronization should be performed as an XML document [SyncRep].

3.2.3. X.500 / LDAP

X.500 is a series of standards regarding electronic directory services created by ITU-T to support name lookups in email. X.500 itself provides an overview of concepts models and services, and amongst them the X.519 DAP, Directory Access Protocol. However well defined and powerful it did not become very popular because it demanded a full OSI-stack implementation on the host computer. This caused the development of LDAP, a lightweight version of the X.500 DAP using TCP/IP directly and simplified the protocol greatly by leaving out many rarely used and duplicate features.
Since it is a directory service, LDAP is optimized for reading and does not provide complete data consistency like most relational databases. It is therefore best used when there is a high read:write ratio and temporary consistency glitches are not essential to the operation of the service. Read more about this at [X.500] and [LDAP].

3.2.4. Parlay / OSA

Parlay is a series of APIs for the telephone network created by ETSI, 3GPP and The Parlay Group, a technical industry consortium founded in 1998, in cooperation with a number of JAINs. The APIs are specified in the CORBA Interface definition language and include services for; call control, conferencing, charging and user interaction by text and audio.
The interfaces are largely independent of the underlying telephony networks to make it possible for IT-developers to create services in addition to telephony experts. [Parlay-FAQ]
This work have also resulted a new specification called the Parlay X Web Services, a set of building blocks of telecom capabilities that developers in the IT-community can easily use to create new innovative applications for the next generation networks. [ParlayX] [Parlay-Spec]

4. REQUIREMENTS AND DESIGN OF URD

The goal of this document is to detail more closely how an Agenda-system can be realized sometime in the near future. Below I will describe how an implementation could potentially be realized and what technologies could be used to support it. I have chosen to name this specific kind of an Agenda-system for URD.

4.1. User requirements

4.1.1. User groups

To find a suitable design for the system itself it is first necessary to analyze what kinds of user groups will potentially be using the system. Following the KISS-design philosophy (Keep It Simple Stupid) I have chosen to only differentiate between just two kinds of users:

1. Single users or a group of single users that want the service provided by a third party. This includes most independent people, families and small groups.
2. Groups of users interested in running their own service. This can be a technically inclined family, a large organization or a company with strict security requirements.

It is upon this distinction that the URD-design and business analysis below are based.

The distinction presented immediately lets me identify two ways to use the URD-service.

1. As a single user you establish an URD-account with an existing URD-system. The accessibility and capabilities of your account is decided by the policies and equipment of the third party provider of the service.
2. A group of users can choose to deploy their own copy of the URD software at a server of their choice. They may modify the capabilities and accessibility of accounts on their server as they see fit, and also specify group policies for communication with other URD-servers.

4.1.2. Functionality required by end-users

End users require a certain amount of functionality for the system to be useable. Here is an overview of some features that should be implemented as based on analysis of the scenarios in chapter 2.1. This is not an exhaustive list, but only the basics that either must be in place for the system to work and some that give URD an edge on competitors.

No. #	Requirements	Scenario	Importance
1	Registration and modification of account details and defaults.	2.1.1	*High*
2	Modifying items in personal schedule.	2.1.1	*High*
3	Automatic location based travel-time calculations and warnings.	2.1.1	*Low*
4	Scheduling conflict detection.	2.1.1	*Medium*
5	Review own and others schedule and plan events based on the review.	2.1.1	*Medium*
6	Warnings of status mismatch with option to update schedule.	2.1.2	*Medium*
7	Location awareness independently of scheduled events.	2.1.2	*Medium*
8	Sending event-invitations to users and groups at any server (or none at all)	2.1.3	*High*
9	Forwarding received invitations to others based on DRM in invitation	2.1.3	*Low*
10	Defining group access policies	2.1.4	*High*
11	Get notifications for scheduled events and for updates to events.	2.1.4	*Low*
12	Automatic-responses with status-information (e.g. voicemail).	2.1.4	*Low*
13	Request current status for others.	2.1.5	*Medium*
14	Secure authentication for users and secure accounts	All	*High*
15	Automatic time zone adjustments	All	*Medium*
16	Ease of use	All	*Medium*

Table 4.1: Functional requirements for users

4.1.3. Functionality required by group administrators

Administrators of an URD server have some more functionality demands than regular users since they need to control usage and be able to fix errors and problems. Such features are usually only available through an administrator-interface on the server itself, not from the normal range of potential URD-enabled clients.

No. #	Requirements	Importance
17	Administrator access to the entire system with preferences and configuration, including logging, surveillance and billing.	*High*
18	Modifying user and group policies for the entire server.	*High*
19	Full access to user accounts for support and control.	*Medium*
20	Raw data access for statistics and auditing.	*Low*
21	Secure authentication for users	*High*
22	Easy migration of users from other servers.	*Low*
23	Accounting features for statistics and billing.	*Medium*
24	Content management for the directory service	*Medium*

Table 4.2: Functional requirements for administrators

4.1.4. Other non-functional requirements

In addition to the functionality required by users and administrators there are also a range of non-functional demands that must be met if the service are to be seen as a trustworthy service that is run by professionals, two very important things in gaining the trust of customers.

No. #	Requirements	Importance
25	Scalability: The system must be able to support an arbitrary number of users.	*High*
26	Stability: URD should avoid crashes that cause downtime.	*High*
27	Trustworthiness: Ordered services (e.g. notifications) must be delivered.	*Medium*
28	Integrity: Stored information cannot be corrupted or disappear.	*High*
29	Low cost of deployment: The third-party cost of installation should be low.	*Medium*
30	Adaptability: Extensions to support new technology must be easily deployable.	*Medium*

Table 4.3: Non-functional system requirements

4.2. The network based Agenda-service

4.2.1. Outline

To provide a scalable solution suitable able to serve both groups of users described above, the Agenda-service must be implemented by using a distributed system of servers. I propose to model this system upon the well-known and highly adaptable system of email communication as this will have several advantages. Firstly it is a proven design that scales very well and can be adapted to any necessary performance and security measures required. This is proven both by the [Email] system and the [Jabber] IM system that also uses this architecture. Furthermore the system allows a reasonable way to exchange messages between the nodes for updates and requests, and in addition the basic design and operation of the system is well known among potential server-operators within the IT-industry.

In addition to the servers themselves there also needs to be a way to uniquely look up the addresses of calendars of various persons worldwide. This problem has been solved for the regular Internet by the invention of the Domain Name Service [DNS] and the [X.500] Directory Service, so I propose using a similar solution for URD. This will allow groups to be assigned to administer their own domain namespaces and usernames internally without the need to communicate these changes to the root host continually.

To exemplify the how the URD-system will be operating I will be referring to the URD Corporation in this text. This is simply the organization or company that will develop and deploy the URD service. They will be running the original URD-servers and lookup-services required, as well as provide interfaces for advanced functionality to third-parties running their own URD-servers but that do not have the desire or equipment to implement the interfaces themselves. This can for instance be a voice-recognition system connected to the regular telephone-network for oral control of an URD-account, which may be too expensive to implement for small groups, as it might require advanced hardware in addition to a sufficient number of lines and interfaces to the telephone-network.

4.2.2. Addressing and address lookup

There is a range of requirements to consider when designing an addressing system for URD:

- Addresses must be unique worldwide
- Addresses must allow a lookup system to locate the server it resides on
- Addresses should identify a single person, but a person may have several addresses.
- Addresses should not be dependant on the location of the server
- The namespace must be divisible into a range of separate administrative domains.

An address type that qualifies all of these requirements can instantly be recognized to be the addresses used for the common email service. This means that an end-user address could be any regular email address registered in the system, as this ensures three of the above requirements. The final requirement, which is that of server-lookup by address, can be performed by using a kind of DNS-system where the different addresses are registered.

Providing server lookup based on an address can be performed in several ways. A simple way would be to implement a separate lookup system run by the URD Corporation where one can register e-mail-addresses and the associated server. This has the obvious advantages of being tailored for the URD-system, but it may take a lot of work to create and might not scale very well depending on the design.
Another way is to add a new record-type to the existing DNS-system and use it in a way similar to how email does it. This is a proven system that scales well and it is a widely used standard for address lookups and load balancing. For these reasons I think this would be an excellent way to enable URD as a truly distributed system. The record-type could for instance be "UC" for URD Calendar, which is what I will use for the remainder of this report when addressing the record-type in question. To use this form of lookups however it will be necessary to get the new entry-type ("UC") accepted by the IANA, and have it implemented into existing DNS-applications like BIND. It will require a lot of administrative work bordering on being impossible to get this accepted and make use of this solution, but it should not require any development on the part of the URD-team if it does happen.

One drawback with using a DNS-based solution with email is that people will not be able to register their existing email addresses unless domain-owners add the relevant UC record-entries in the DNS-record for all domains with email-addresses that will be used with an URD

account. This limitation could be a major hurdle to the adoption of URD, but can be overcome by implementing a default route in all URD-applications to the host servers of the URD Corporation, meaning that email addresses that do not have a local URD server registered are assumed to be registered by a set of root servers or not have an URD account at all.
To avoid having to specify a static number of host servers in applications the identification of a host-server could be implemented through an anycast address. This way any changes in the set of host servers at the URD Corporation only need to be updated at a single location. The problem with this solution is that it breaks the inherent simplicity in using DNS by requiring extra hacks for special cases in all URD-client applications.

A second way to get around the drawbacks of DNS could be to assign a new class of URD-addresses to all accounts that are created instead of using current email addresses. This way the system design would be even more like that of email and the DNS service could be used according to its intentions. However this would require all users to get and remember one more type of address for all of their contacts, and a side effect is that possession of someone's email address no longer ensures that you can request their calendar-information. To retain this functionality one would need to locate anyone else's URD-address manually, or one could implement some kind of email<->URD lookup system within the URD-system.

This lookup-system could for instance be the X.500 Directory service, an ITU-T and ISO standard for distributed directory services. The service is actually so full-featured that it can be used exclusively to provide all of the lookup requirements of URD by itself without having any need for running DNS as well. This can be done while still retaining most of the advantages of DNS like distribution of servers and local domain and directory management at each distributed site. This will also give you a single point of contact for any lookups, which may be performed by searching for the email, phone number, name or URD-address of any user in the system and have it answer with the necessary information to make contact.
Another service that could do the trick is LDAP, which is a lightweight version of the X.500 Directory Access Protocol developed at the University of Michigan, which implements most of the same services as X.500 but through a much simpler interface that uses TCP directly. This allows for a much smaller footprint when implementing the service, and makes it a lot more suitable for mobile computers with limited resources.

4.2.3. Agenda-server functionality

To be able to create a service like the one described there is a number of pieces that must fit together and work together. Some of those pieces form the distributed agenda-servers that will be run by the URD Corporation.

Since the URD servers must be able to handle a range of different clients and tasks, I believe it would be good to use a modular design for the server. This will enable the possibility of scaling the footprint and functionality of the server to match the requirements of its environment and usage. Other advantages of modular design is that each piece can be developed and tested separately and with well defined and documented interfaces new modules can even be designed by third-parties to match their requirements. [Modular]

Here I will present the primary modules that make up the URD-server software:

The system core

This is primarily an interface providing a uniform means to access the calendar-database and the directory service of URD. The major part of the system core is related to enforcing access control for the system data and user information based on the client authentication subsystem, and to provide other services for the database, like synchronization and accounting.

Command Access Interface (CAI)

To hide the peculiarities of the particular core-implementation from external plug-ins and services, a generic command access interface is required. This interface must implement all commands that the system can accept and provide a common response format for them as well. This will be like the programmers API for any external plug-ins to the system, allowing the entire structure and internal protocols of the core to be changed without having to modify any of the existing plug-ins for the system.
This module also makes provisions for security in that all communications must come through the CAI. All incoming communications from the CAI are then verified using the authentication and policy-subsystems in the core before any commands are executed and the results committed to the calendar-database.

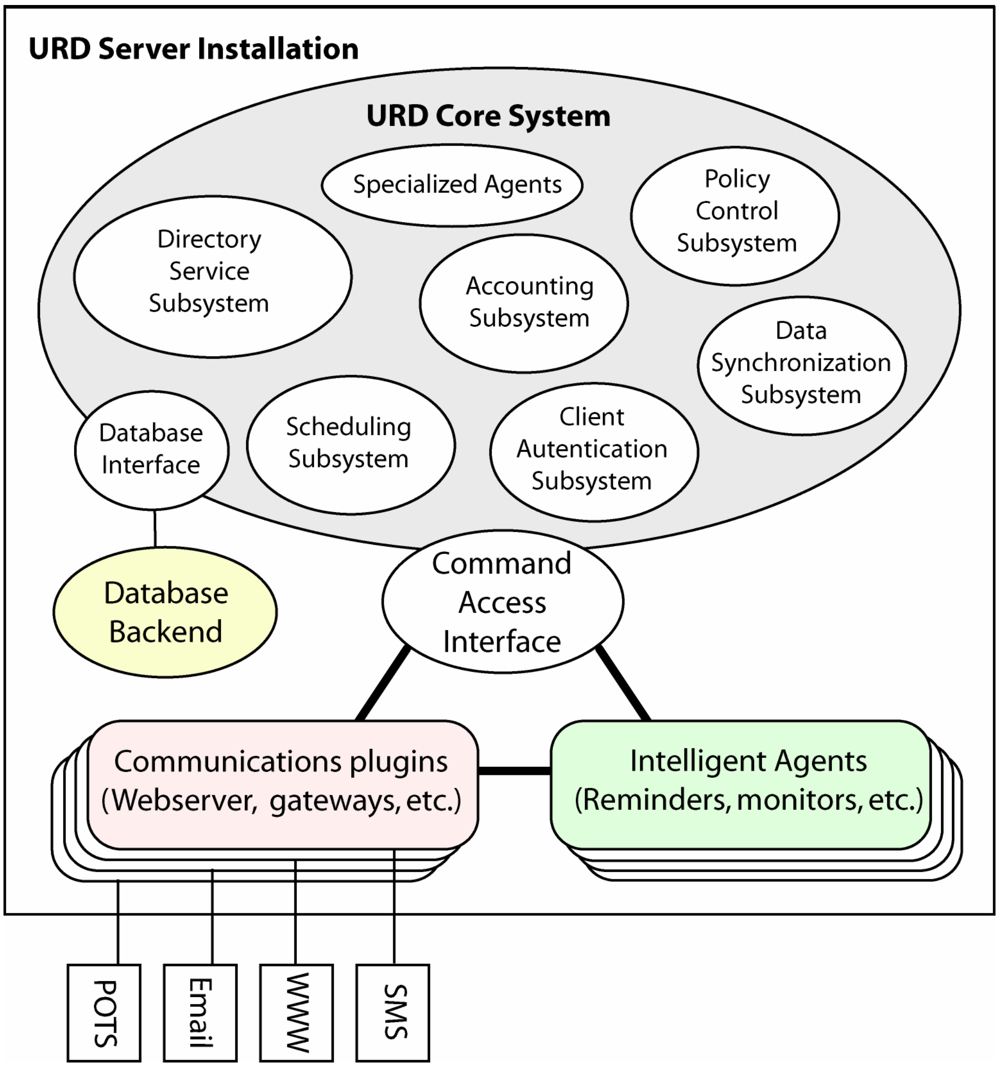


Figure 4.1: Component layout of an URD Server Installation

This graphic details how the components of an URD Server Installation relate to each other.
Inside the URD Core System are all the subsystems detailed in this chapter. These subsystems exchange information through the regular programming interfaces in the programming language used for implementation. There are two interfaces between the URD Core system and the outside. The first is the Database Interface that only connects to the backend database system for data-storage facilities. The second is the generic Command Access Interface through which all the regular requests, commands and updates are passed. This interface can be accessed by any object, but is most commonly accessed either by Intelligent Agents on the server or through one of the Communications plug-ins that provide a set of common interfaces to various communications channels that can be used by any person, client or application that wants to access the URD services.

Data synchronization subsystem

Since users of some interfaces, like computer application clients, would like to download their calendar-information to view offline, there needs to be a standard for synchronization of calendar-data with external entities.
This system is also important when users are migrating to a new server when for instance an organization starts running their own URD-system a while after several of the employees have started using it on their own, and all their data must be transferred to the new server while retaining full service for the user.
[Skogstad] has written a report on how SyncML and a Sync4J SyncServer can be used for synchronization between clients and a server based repository of information. The solution proposed is not directly applicable for URD, but it is a good starting point for the design along with the report by [Vo] on calendar synchronization with a P900.

Policy control subsystem

Based on the identity of a user as confirmed by the authentication subsystem, the policy control system defines which operations are allowed to be performed by the user. There can be a range of different policies, for instance applicable to:

- Any single specified person
- Any specified group of people
- Members of any generic group, for instance:
    - All Posse members (registered contacts)
    - All Associates (users on the same server)
    - The General Public.

The policies can limit the information about your person or group that will be provided to other users, and as mentioned above there it should be possible to create group-wide and server-wide policies in addition to those regarding a single user.
This system should incorporate the required data-hiding, as some users or groups of users can be limited to access only part of the information in another user's or group's calendar. This means information should be ordered in a set of levels that the group policies can easily address.
A basic layout of such a level structure may for instance be:

Level 0 - No schedule information provided.
Level 1 - Busy/Available status and contact information provided.
Level 2 - Generic location provided (city/country).

Level 3 - Personal/professional status specified for busy-states.

Level 4 - Specific location provided for busy-states.

Level 5 - Event topics provided for busy-states.

Level 6 - All information provided.

The database interface

This is the link to the database system in which the system data and schedule-data are stored for all users. It should be possible to implement the backend database itself using any kind of database, which is why this interface is provided as a separate subsystem to hide the specifics of the used database so ach site can select their preferred backend database to be used freely.

This subsystem should as a minimum provide an interface to use regular SQL-supporting databases, but could also implement any database-specific improvements for common systems like mySQL and Oracle DBMS's.

Scheduling subsystem

This subsystem keeps track of the personal schedules of every user. All additions and modifications to someone's schedule goes through this subsystem to verify the data integrity of the incoming request and to check for duplicates and other errors that could be introduced with the update. Some faults should produce warnings (like double-booking) while others cause errors and are not committed (like faulty requests).

Accounting subsystem

Since administrators may find great value in statistics of usage and load for system optimalization and debugging, and there will be a requirement for the ability to charge for usage by commercial operations there must be a Accounting-subsystem in the core. This subsystem will register all transactions that take place, both incoming and outgoing as well as details on what was going on. All of this must be stored in a database so that billing-systems and other tools may analyze the data and extract the information that is required to perform their tasks.

This is especially important if there will be interconnects to the telephone system as phone-operators will be charging the URD Corporation for its use.

The directory service subsystem

The base-system must implement a directory-service to be used for server-lookups and user-identification. This can for instance be X.500 or DNS, but it must be the same for all instances of URD. One root instance will be operated by the URD Corporation, and parts of the namespace will be distributed for management of each respective instance of URD. This system has to be part of all URD-systems as they need to be able to control of their own domain, and it allows a user to request a lookup to any given URD-server which will process the lookup request locally instead of the lookup service being dependant on the root host at all times.
In principle this subsystem could be just a stub like the database-interface allowing the use of any kind of directory service, but since this would lead to a range of difficulties when the namespace is being split up into separate administrative domains it makes for a far better solution to stick with one specific kind of directory for all servers.

Client authentication subsystem

The authentication subsystem will confirm the identity of a user to the command access interface. This can be quite a large task since users can authenticate through a large number of ways, including synchronous logon, SMS-sender number, PGP-signed emails and Caller ID, but to allow rapid deployment of new modules the authentication system should only accept one or a few standardized forms of valid authentication. Any interfaces that cannot authenticate natively using this format must implement some form of authentication translation in its User Interface-module. The native authentication standard can for instance be based on X.509 certificates or PGP signatures.

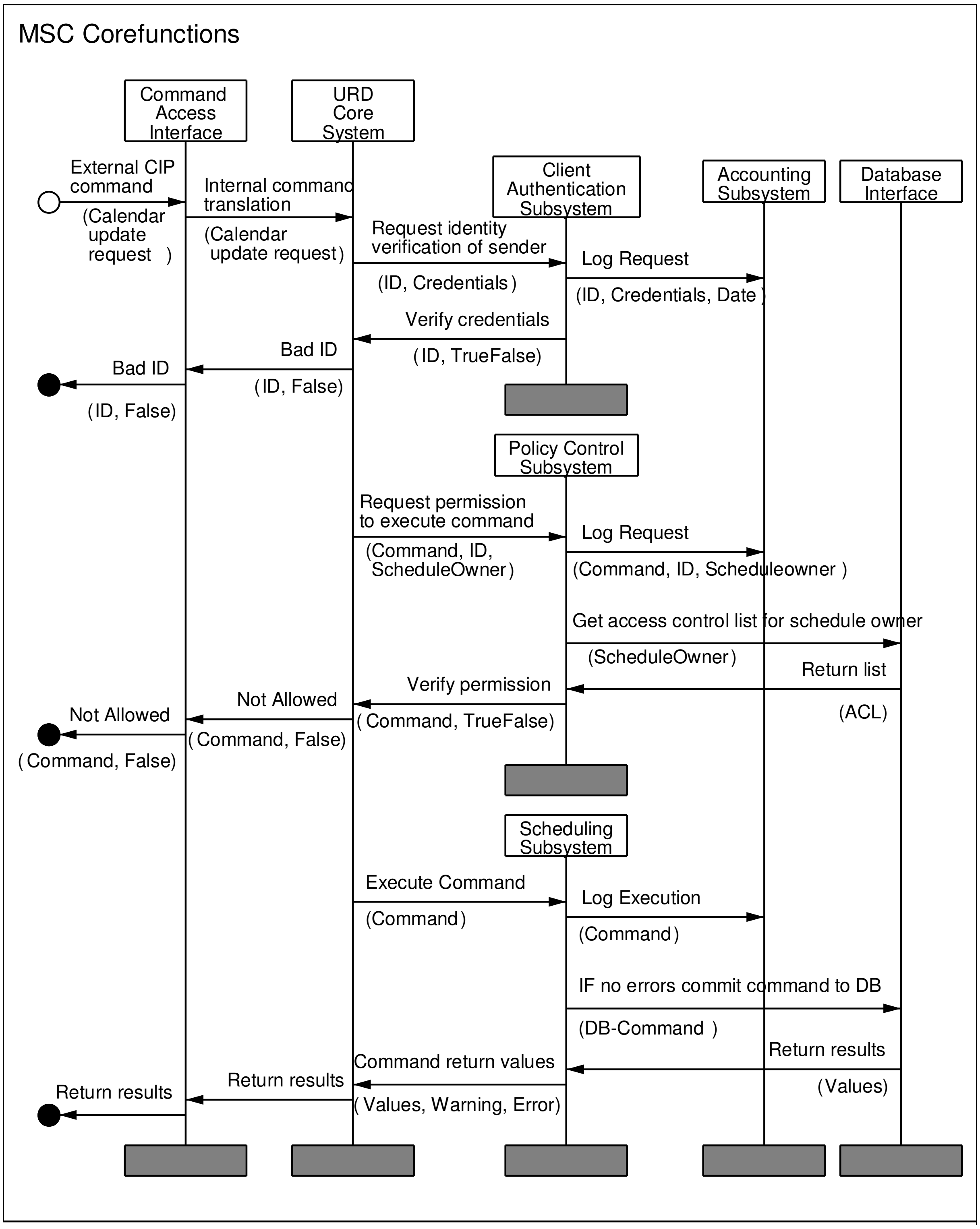


Figure 4.2: Sample Message Sequence Chart for the URD Core-system and subsystems

Here is a sample of the internal communication that are performed in the URD Core when a command is recived through the Command Access Interface. This MSC relates specifically to when an calendar update request is recieved, but the sequences for other commands will usually be very similar to what is shown above.
To limit the size of the diagram I have included multiple cases concurrently. The ”Bad ID” and ”Not Allowed” commands are only sent if the previous return value is ”False”, else the sequence continiues.

Communications plug-ins

In addition there are a number of plug-ins that should be available that add more or less important functionality to the system. Each of these interfaces can be run independently of a core-system, so the interfaces run by the URD Corporation for instance can in theory be used to access any available URD-server. If a given interface is available to connect to a given URD-server can be decided by the owner of the interface, so the URD Corporation could for instance limit usage of the WWW User Interface to their own users, but sell access to the SMS and POTS-interfaces to any interested parties. The basic interfaces are:

WWW User Interface

The WWW user interface is listed here as it is a server-side interface, but it could just as well have been amongst the client interfaces below. The interface is providing web-applications to a user's web-browser, through which the user should have access to all the same functionality as from an ordinary application-based client.
For authentication the WWW-UI must implement authentication translation from any kind of web-authentication unless the client supports the native authentication of URD and can provide such credentials directly.

SMS User Interface

The SMS user interface exists to allow fast changes on the move by sending commands to the system with SMS messages. This could either be done manually by typing in any required commands and sending them, or another kind of user interface could be implemented on a SMS-enabled device while simply using SMS as the backend transport protocol for commands. The SMS user interface must implement authentication translation from the verified SenderID to the authentication format accepted by the core.

Email User Interface and Server

Commands can also be sent and recieved using email or any other kind of messaging system. Again this could be used as the transport layer of a higher-level interface, or a user could type commands directly and send them to the system. The system could also use email to send invitations and other information to third-parties on a standardized format like vCal.
Authentication can be done by using email-standards like X.509 or PGP, and might require authentication-translation if it is not the native format of the core being used.

POTS User Interface

To provide backwards functionality for users that need access to URD but don't have any recent aids available it should be possible to dial in to URD using a regular telephone. The authentication like in the SMS-UI must be handled by translation of the CallerID to the relevant credentials. Such a POTS interface can for instance be realized by using the Parlay/OSA framework and can have several sub-interfaces to handle different kinds of telephone based communication.

Touchtone subsystem

At the very least the ordinary touch-tone telephones should be supported. The system can be accessed by entering selections on the touch-tone dial based on alternatives provided vocally by the automated response-system. The system will most likely be quite cumbersome to use due to its limitations, but for limited important tasks it may be important.

Voice-recognition subsystem

To make things a bit simpler than using the touch-tone system a voice recognition interface should be implemented when technology permits. This way a user can simply tell the system the commands desired and have the system reply with results. This is however something that is still a while into the future as voice-recognition systems are not yet capable of understanding arbitrary commands in a way that will make the system substantially easier to use than the touch-tone system.

4.2.4. Communication-protocols

As the URD-system is distributed and modular it requires a series of standardized protocols between each of the communicating layers to work. Here I will list and comment on the protocols needed and suggest if existing solutions can be of use.

Data synchronization protocol

Since some clients might need the ability to download all or part of the calendar for one or more users, there need to be a synchronization protocol available in URD so that multiple downloads and uploads can be performed without having to update all the information every time. Also this will allow a user to update his or her local copy of a calendar even when the device is offline, and then be able to synchronize the changes into the master database whenever there is a chance to go online again. This functionality can also be useful when users are being migrated between servers and when updating the calendars of the people in your Posse to your server. One potential candidate for this protocol is SyncML, a standard for synchronization between devices developed by the Open Mobile Alliance.

Server and user lookup protocols

Depending on which addressing scheme is followed it will be best to use the lookup protocol provided for finding the server of a specific user and other things. This means that if an LDAP-compatible directory is selected to handle user-addressing, then it will be best to also use LDAP for any lookups that are required. The reason for this is to streamline and simplify the system design as much as possible to make it easier to understand and extend.

Command Interface Protocol (CIP)

Since there may be a wide range of URD-enabled devices and terminals with varying capabilities there need to be a standard way to communicate commands between the various server-interfaces for each device and to the URD-core system. This protocol is very dependant on the specification of URD itself and will therefore need to be developed especially for URD, but as most of the information to be communicated is calendar-related, and this calendar-information should be communicated as standards-compliant iCal-data, it would make sense to base this protocol on an open standard like XML that are capable of integrating the iCal-data in a natural way. It is also important that the protocol can integrate other kinds of data as both synchronization and lookups are best performed using specialized protocols for these tasks.

The protocol must be able to communicate all kinds of operations and requests to the URD-core from any source, and return any responses to the original sender. This is important as the protocol should allow messages to be forwarded between servers, a feature that will allow users to send their commands to any available URD-server and have the URD-system locate where the relevant user accounts are located.

Due to firewall-traversal considerations and other traffic-limitations that might be imposed on private networks the protocol should be able to backtrack any responses until it reaches the original sender, but only after attempting direct communication with the sender.

4.2.5. Integration of third party services

Since there are very many existing solutions for calendar sharing and scheduling it is important that URD can integrate with these systems to provide a seamless transition for users, even in the cases where the transition happens in steps or where only some users switch to URD. Most important is it that URD can read calendar and presence information from sources like Exchange and Lotus systems to integrate this data into its own calendar. The reason is that URD-users who are invited to events or have other entries within these systems should get everything available inside URD so they don't have multiple systems to check. It would be even better if URD could export relevant schedule-information into these system as well and keep them up to date so other users of these other systems still can access the calendars of those that have migrated to URD, but this might not be possible for all systems since many are proprietary with closed protocols and no interest in cooperation.

There are also a range of other common applications that can be useful for URD when it is determining a users presence, amongst these are Instant Messaging systems the most obvious since many of them already have simple presence information available. This information combined with what URD already has access to can provide improved accuracy in determining presence and will therefore allow URD to make better decisions regarding call-screening and other tasks to improve the user experience of URD and provide a better service than it would otherwise have been able to. More details on gathering presence information from a multitude of sources is described by [Foerster].

The functionality of integrating the data from third-party systems into URD can probably be best solved by using separate Intelligent Agents to perform the tasks in question. This will allow rapid deployment of agents for any required third-party systems without stressing installations that do not require such functionality by the extra load required.

4.2.6. Intelligent agents and agenda-updates

In addition to the regular calendar functionality with sharing and synchronization there is one thing that can make URD stand out from the crowd and starting a revolution in usability and improved schedule-handling. This thing is intelligent agents with much wider capabilities than what is normally available. An intelligent agent in this setting is a piece of software emulating intelligent behavior by reacting to certain events based on some specified variables.

The agents can run anywhere, either on the server or at one or more terminals, and they will be keeping an eye on your agenda to help out if anything needs to be done. Amongst the things they will help you with are:

- Provide reminders for upcoming meetings and travel-arrangements which can be delivered through any means that you select. Also the agent can keep notifying you using various means like email, sms, phone or IM until you acknowledge receipt of the reminder. This way you will not forget the meeting or avoid getting the reminder out of forgetfulness. If you do not acknowledge the reminder in time the agent can notify organizers or participants at the scheduled event that you are prevented from participating.
- Agents can also be monitoring your status, presence and location and notify you if a mismatch occurs. Depending on the mismatch the notification can provide you with some options to automatically modify your schedule, for instance by deleting the next event or clearing the entire day if something has come up, and organizers of cleared entries can then be notified of your cancellation. This way you won't just be missing if something comes up but can notify those affected without much work. Like with reminders a failure to acknowledge the reminder can cause notification of affected people, or maybe even emergency-services if status shows that something is wrong.
- Another service that can be provided by intelligent agents is aiding with scheduling common events. For instance if a department shall have a common meeting once during the first week of each month an agent can monitor all the participants calendars and dynamically find the best matching timeslots to maximize availability of people. This timeslot does not have to stay fixed but can be modified along with the schedules of the participants until a deadline before the meeting is passed and everyone notified of the final time and location as decided by the agent.

Agents like client-software can be programmed to perform any imaginable task, and can therefore be very valuable for users of the system as they can take on much of the workload with scheduling-related tasks. This will hopefully result in more time to focus on your core activities and as a result give both you and any managers you have more value in return.

The agents on the server-side are usually separate from the URD-core and will perform their required tasks through the CAI just like communications plug-ins, and they can use the communications plug-ins too when necessary. There can also be some specialized agents running as subsystems of the URD-core if they are required to perform tasks not available from the outside, but this should generally be avoided since weaknesses in such agents can allow illegitimate access to internal functions of the core by bypassing the security-mechanisms that protect the system.
Client-side agents can take any shape and form depending on the tasks they are to perform. Small agents in embedded chips will often be stand-alone applications that communicate with other clients or the URD-servers directly through a plug-in interface or the CIP, while agents in more powerful mobile devices like enhanced PDA's could look more like those on the server-side functioning as plug-ins for the URD-client available on the terminal.

4.2.7. In the future: Mobile P2P servers

Sometime in the near future there will be mobile devices that have both the processing power and connectivity required to run a full-featured URD-server locally. After such devices become commonplace the hierarchical network of URD-servers accessed by client-terminals that I have described here might disappear, as everyone can run their own server on a device in their hand. This will bring new challenges, but can also mean major improvements of how things must be handled today. Availability of a server's complete dataset in addition to being within arms reach of your person will allow for faster and better agents to serve you, and with more data available from new sensors and smart devices the agents could help you with other tasks as well. Since everyone can carry their own server the system will become a direct P2P-system and the scalability issues of today will be forgotten, but other problems might become even more pressing and the system might require a redesign to function as intended in the new era and to address the issues that are at hand.

4.3. Agenda-system clients and terminals

4.3.1. Local computer-applications

Local computer application can take any shape and form, and their functionality are only limited to the imagination of the developer. It is primarily through applications such as these that access to the complete set of advanced features of URD will be available. This includes all features mentioned both for users and for administrators in chapter 4.1.
The applications can implement the URD Command Access Protocol and communicate with an URD server directly, or they can work as a local front end for secondary interfaces like web and email if the application is very specialized to specific tasks where this is better.

4.3.2. Centralized server-side clients

One of the primary ways to contact the URD-system for information retrieval or updates are through web-based and other kinds of server-side interfaces that can be scaled to fit any kind of terminal, independent of its own processing capabilities as most of the work is performed server-side.
This includes everything from ordinary web-based interfaces to scaled-down WML-pages for mobile phones and handhelds, and scaled-up AJAX web-applications that can be very similar to ordinary local applications. How these should be implemented has already been covered in some detail in the Plug-in-section of chapter 4.2.3.

Another kind of centralized client can be an email-system where commands and authentication is received through email and performed. The email solution is especially well suited for sending messages to external parties which may or may not be using URD. Implementation details of email have also been covered in chapter 4.2.3.

4.3.3. Personal Digital Assistants and Mobile phones

Like on regular computers there can be URD clients running on various handheld systems like advanced mobile phones and personal digital assistants. Due to the limitations inherent in many of these terminals all of them might not be able to run the full suite of URD-functionality. However as these terminals are mobile they can offer some advantages and potential uses that are not possible with ordinary computers.

For instance are many such terminals equipped with location-aware equipment, either using GPS or by receiving information from mobile or other wireless networks. This equipment can be used by the URD-client to monitor if you are following your planned schedule, and perhaps alert you or others to the changes if your actual location diverts from the planned one. This information can also be used to determine your presence status, for instance if someone requests it to see if you are available or to use when the system must determine whether you should be notified of incoming calls and messages as per your settings.

For advanced mobile phones running an URD-client another kind of functionality could be automatic call-screening based on your group-policy lists. The screening could also be performed based on your current presence information, giving you a range of options for automatic treatment of various callers depending on both who they are and what you are doing at the time. This functionality could of course also be used for any other kind of real-time communication such as VoIP or instant-messaging as well. Details of how an automatic call-screening system may be realized for SIP-calls have been written about in more detail by [Osthus]. Another relevant functionality that could find good use here is the personalized no-answer options described by [Bolstad]. This means that instead of just giving a voice-message or busy-signal to callers you do not wish to accept at the moment, you can send them alternative venues of contact like email or IM, or simply direct them to a webpage if they are using compatible terminals.

4.3.4. Dial-in by regular phone or text-messages

To support users that are out of bounds and do not have access to other kinds of communication, it should be possible to dial into the system through a normal phone, authenticate by a pin-code or perhaps voice-recognition, and issue commands to the system either verbally or by using the touch-tone keypad of the phone. This would naturally have to be very broad and simple commands since the interface ordinarily can't support advanced features very well, but it might still be very useful for some people and could for instance replace many current systems where office-workers indicate their availability by using dial-codes on their office-phone.

Like with email it should also be possible to send text-messages to URD with short commands contained in regular SMS-messages. Authentication is simple since the messages always have a sender-ID associated with them, so one can for instance allow a user to register the phones from which SMS-commands will be allowed.
A limitation with phone-network interfaces is that many smaller companies and groups will not have the resources, desire or knowledge to install a phone-interface to URD as this requires phone-lines or other access to the network to be available for the computer. To alleviate the need for every server to install such interfaces locally it should be possible to forward commands via the phone or SMS-gateways of other public URD-servers like those operated by the URD Corporation, who then in turn can forward the message to the correct server.

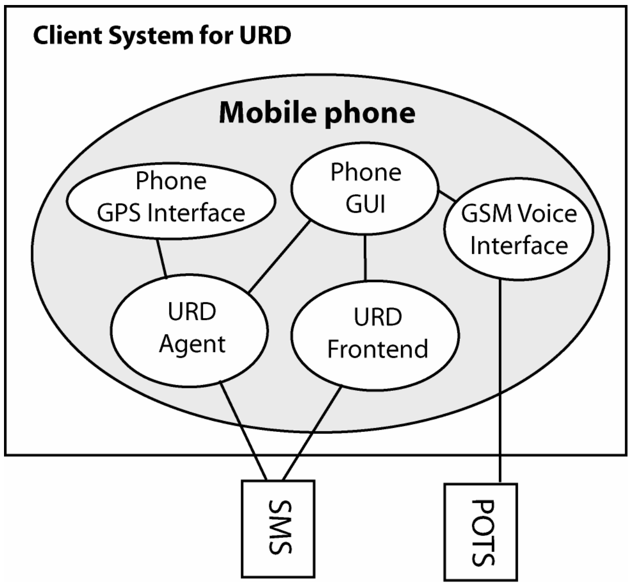


Figure 4.3 Client System for URD

This figure illustrates how URD components may interact with the interfaces of a mobile phone.
Both the URD Agent and Front-end make use of the SMS services in the phone, and both can be controlled through the phone GUI.
The URD Agent also uses the GPS interface in the phone to register the current location.
In addition the user may access the URD Service through the regular telephone network by making use of the GSM Voice Interface of the phone.

4.3.5. In the future: Autonomous smart devices and P2P URD.

In addition to the possible clients and terminals that are analyzed above, the future may hold many new technologies that can make URD even better at aiding in your scheduling.
One of these technologies are autonomous smart devices, this means small devices that often are created for a specific task and can act when defined triggers are met. This can for instance be health-monitoring equipment that can send information to the hospital or contact the nearest doctor if you suddenly get ill. Such equipment can also help with other tasks, especially in determining presence and other status information. If data from such a device could be monitored by one of the URD-enabled PDA's mentioned above the PDA could not only screen calls or modify your schedule based on your location, but also use your heartbeat and breathing to evaluate if you are available for a call. This could help you filter out the world while you are jogging or doing other kinds of exercise while transparently letting you receive calls while you are resting in between exercise-sessions.
Such devices could also be located inside other kinds of equipment such as cars or heavy machinery, or it could be located along with RFID-readers in doorways enabling some rooms to be quiet-zones where all URD-enabled equipment will block communications-attempts.

As with computer software the possibilities are only limited by the imagination of the engineer who designs the systems, so there are no limits to how this service could be used as new technologies emerge and the capabilities of handheld devices grow larger. Eventually it may even be possible that every person will have their own personal full-featured URD-server running on their smart phone so that the server directly can monitor any kinds of presence and status-information. This will make the system fully P2P where every user is also a server, and with each server there can be a general client with a range of user-interfaces suitable for any kind of screen-size and device capabilities.

4.4. Security considerations

Due to the large amount of personal information that can be found in the URD-system it is very important that the system is secure from both accidents and malicious attacks.
Because of this the authentication mechanisms chosen must be highly secure so that no user can gain illegitimate access to someone else's account or to initiate commands meant for administrators. Another matter is that the authentication mechanisms are required to ensure the correct operation of the group-policy system, since it would not be able to guarantee to limit information to authorized users if it cannot securely verify which users are authorized.

This is the reason that I have proposed using secure schemes like X.509 certificates or PGP-signatures to authenticate the user of a communications plug-in or agent to the system-core. However if the security of the system are to be of any value at all then the authentication performed by the plug-ins must be equally secure as no chain is stronger than its weakest link. Only through such means can the URD-system be accepted world wide since there are many businesses and individuals that are very concerned with the confinement of information.
Some of the plug-ins, like those for email and direct access to the CAI, can implement the chosen standard natively, but other will need translation of the credentials form another system of authentication. For instance may phone users be authenticated properly as those networks themselves provide a method of authentication through the use of caller-ID, which may then be translated to the native system by the plug-in. Despite this simple solution being apparently secure, the matter should not be taken lightly and it would still be a good idea to use PIN-codes or other secondary means of authenticating as well since a phone may easily be used by other parties than the owner through simple physical theft.

Another matter is how the various URD-servers should be able to authenticate each other as being valid originators for calendar updates and other commands, but this is a far to complex problem to be discussed in detail within the scope of this report. This along with the distribution of secure tokens and exactly how the authentication translation in a plug-in will be performed should be analyzed in a lot more detail than is possible here and is therefore left to be examined in a separate report by an information security specialist.

The considerations described are based on knowledge from [Infosec].

5. Business Models

To create a successful system it is not enough to simply create the system itself. If it is to be established as a widely used system it is a major advantage to build upon solid business models and creating a solid company to run the service so that potential users can be ensured that they will receive high-quality service and good support. This chapter will therefore do a systematic analysis of how such a company can be run and examine which business models that is most likely to succeed in the long run.

5.1. SWOT Analysis of the URD system

A SWOT analysis is usually one of the first things to do when analyzing a business system with the intention of changing it or introducing something new. SWOT is an abbreviation for Strengths, Weaknesses, Opportunities and Threats, the four primary attributes that you use in this method to classify yourself in relation to your competition and the market. The analysis may uncover important limitations in your system and allow you to address these early on, and it can also give you a detailed view of the strengths and opportunities that will allow you to succeed so you can harness these in the best way possible.

5.1.1. Strengths

What make URD stand out from the crowd? On what areas are URD better than potential competitors and which functionality is a must-have for users.

- *Interoperable with other calendar systems.*
  If you are currently using other calendar-systems you may transition gradually using both systems for a period of time, while at the same time having everything available in URD.
- *Highly scalable design.*
  Since any group of users can operate their own server, the system as a whole will not have any performance limitations even with a massive number of users.
- *Near universally extendible.*
  Because of the modular design and focus on plug-ins for changing tasks the system can adapt to meet any changes in the business that may appear in the future.

- *No limits on who can join.*

  Since the URD Corporation can offer accounts on general servers to anyone, there are nothing that will keep anyone from using URD if they can afford it.
- *Can connect to any user.*

  Since the distributed servers can communicate between each other and there is a complete directory service with all the users of the system you can look up and connect to any user no matter which server the user resides on.
- *Near ubiquitous accessibility.*

  With a large number of different devices capable of connecting to the URD-system, including ordinary telephones, it can be accessible from practically anywhere on the planet if you need it to be.
- *Self-operated or hosted service.*

  For users with low budgets or no computer-departments the service can be hosted commercially by the URD Corporation, but if there are security or other considerations that require in-house control this is possible with self-operated servers.
- *Data hiding based on group-policy lists.*

  Every user or operator can define their own, or use predefined, access control lists to limit the amount of information provided to others from the same server or elsewhere in the world. This will keep confidential information secure while allowing the dissemination of harmless details in the calendar to external contacts.

5.1.2. Weaknesses

What does competitors do better, and what can cause problems with development and deployment of URD. Which vital parts are missing from the system?

- *Complex series of interfaces required*
  Many of the required interfaces are complex and some require cooperation from other companies to be of value. These can be a challenge to implement and if they cannot be deployed it can limit the user-value of the system considerably.
- *High cost for some interfaces*
  Interfaces like those for phone and SMS might be very costly to implement and will therefore require high charges for usage which might make the system less popular among users. Also these interfaces will not be available on many servers and might therefore cause even less popularity.
- *Messaging between users*
  Various kinds of messaging could be useful and a killer-app that really makes URD the ultimate personal client, but this is not something yet part of the URD design. However this can become an opportunity as described below.

5.1.3. Opportunities

Which possibilities may put URD on top of the game and what features make URD unique and provides a competitive edge in the market?

- *Network effects.*
  With a large enough user-base URD will be self-propagating until most of the potential market is URD-users, and competitors will not have a chance to enter.
- *New technology*
  This can make it easier to use URD and allow advanced applications to be available on more platforms. Also this can allow creation of new functionality for URD.
- *No similar competitors*
  While there are many competitors with calendar-systems there are no direct competitors offering the complete package and universality provided by URD. This can make URD stand out and give it the boost required to start the network effects.

- *Bundling with others*

  A partnership with a large company and bundling URD-clients and user-accounts with already deployed software with a large user group, like for instance the MSN or AOL Instant Messengers, could be a great advantage almost guaranteeing success.
- *Highly planned and busy lives in the general population*

  Many will find a use for URD in their lives to help coordinate and plan work and pleasure to be as smooth and efficient as possible.
- *First mover advantage*

  The radical move to universal applicability as provided by URD can give it the advantage of entering the market very early, despite that similar but inferior attempts from Microsoft have failed earlier.

5.1.4. Threats

Having success is business is never easy, and there are many obstacles that may block the path. Which problems and challenges are URD likely to face?

- *Sabotage and unfair competition*

  URD can be vulnerable to sabotage by other companies. Spam, false users and DoS-attacks may cause problems at servers, and users will blame URD for not working.
- *Loss of communication*

  Firewalls from Telcos or ISP's or others can block communication with URD. ¨
  Since URD is usually communicating over internet or telephone-links that belong to others these may be blocked on purpose by their owners to disable the use of URD. In addition to this, loss of communication for URD can be caused by accidents that disable the system. This will be especially harmful in the early stages after deployment before high distribution is reached.
- *Privacy abuse*

  The system might be prone to abuse of privacy issues through software bugs. An exploit giving a malicious user access to more information or functionality than has been granted can cause major problems, especially with security conscious companies or through identity-theft.
- *Limited interconnections with other applications*

Interfaces can be blocked or sabotaged by makers of the other calendar-systems. Microsoft and others may change the interfaces to their own calendar-software so that the URD-plug-ins no longer functions and cause users to stick with what they have.

- *Network effects*

  If URD does not gain a critical mass of users to enable the network effects to work within a short enough time, it may never get a large base of users and sustainability, which again leaves the road open to bankruptcy due to a lack of revenues.

- *Competing products*

  Similar functionality might be available in competitor's products soon. Competitors can and will copy the ideas they can and quickly have these available in their own offerings to avoid loosing their own base of users to URD. Since many of the competitors have strong financial and organizational backing they can out-compete URD on mere size.

- *Buy-outs and lay-offs*

  Competitors might try to buy out URD Corporation to kill URD network. This is a well-known tactic used by large companies to prevent competition, but some URD functionality might make it over to the competitor's product-line.

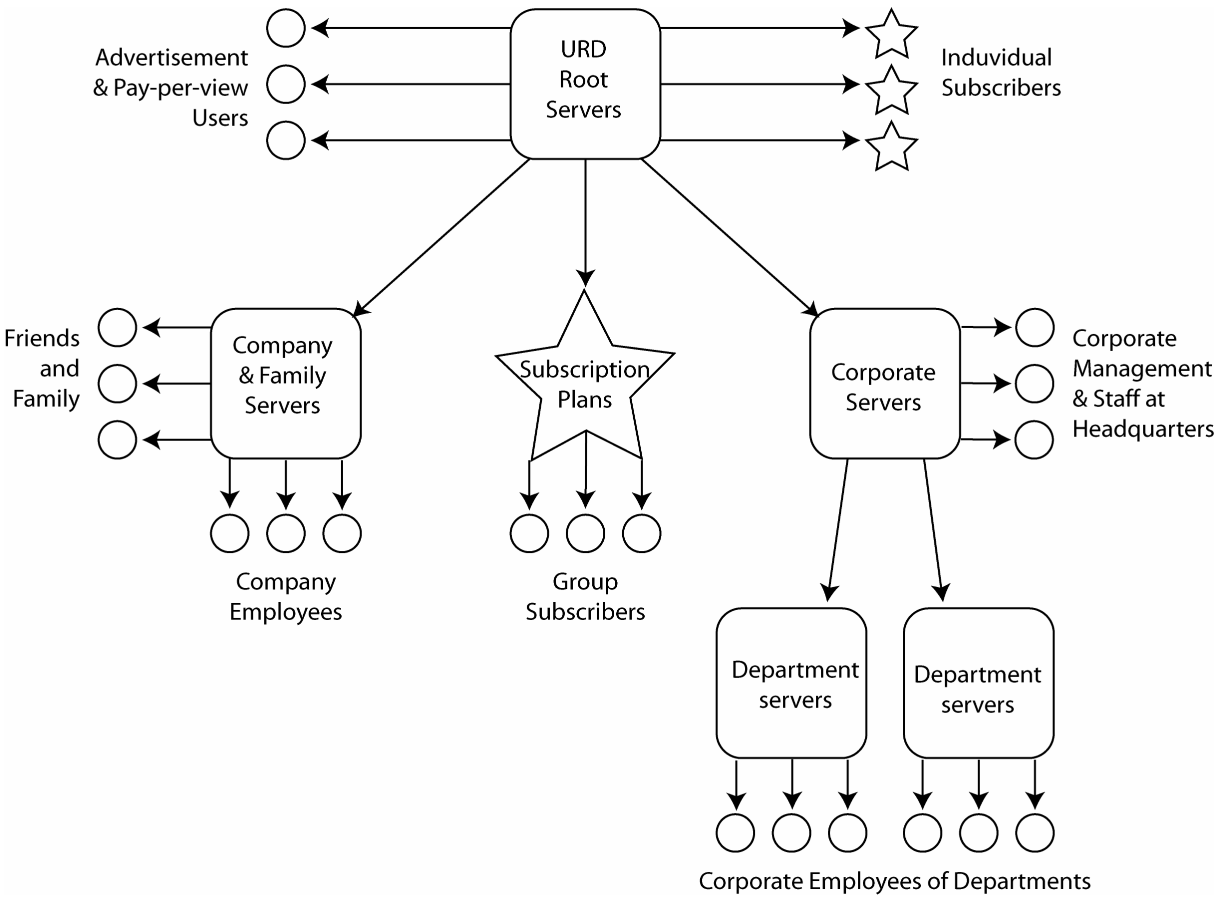


Figure 5.1: Tree model of a distributed URD system with servers, plans and users.

This figure shows the tree-structure of a distributed URD system with a range of various users, servers and subscription plans in place and operated by various entities.

- The rounded-squares symbolize a set of URD servers owned and operated by the entity marked on the server. The set of servers can either be linked to provide load-distribution or they can be separate entities at different locations with different user-groups.
- Stars symbolize subscriptions, either as individual subscribed users or a subscription plan to which a range of users can be assigned, for instance one used by a company.
- Circles are ordinary users connected to their local server or subscription plan

The figure also represents the distribution of the directory service, where the rounded-squares each have management rights to their own assigned namespace. The rights are inherited down the tree, so the Corporate servers can re-assign its rights to a sub-set of their namespace to one or more Department servers. This allows for better load-distribution in large corporations with very many employees.

5.2. Operators

For a service to become successful there must be someone working to develop and deploy it and to provide advertising and customer support. Without this the service is most likely doomed to fail, especially since there need to be a base operator in charge of the root directory and to make the server accessible for single users. Because URD is designed as a distributed service there will be a range of cooperating organizations that will act as operators for it.

5.2.1. URD Corporation

I have chosen to use the term URD Corporation to address the company or organization that does the original development and deployment of the URD system. There are many potential candidates that may become the URD Corporation, and the differences between them will greatly affect which business models will be chosen for URD.
In addition to performing development and the original deployment this URD Corporation will be the entity in charge of the root-servers available for independent users, and it will also control the root of the URD directory service. In addition it will perform registration and provide software for independent URD-servers, and it will operate any required support services for the third-party administrators.

A new start-up

The most likely event is that URD is developed by a start-up company funded with venture-capital and the spirit of entrepreneurship. While this ensures the will to start the social-revolution that URD may bring, it will become challenging to deploy URD as there are no proven track-record or any existing user-base to build on. However this may lead to more innovation in the project and if it succeeds there is a chance for new solutions to take hold.
The major problem with this option is that there will most likely be a strict requirement to get successful and profitable quickly to satisfy the VC's, and this is something that might not be possible to achieve unless a very large market share is gained quickly.
But this could be corrected if the VC exit-strategy involves selling the system or usage rights to another larger company to handle the deployment and long-term operations.

An existing company

A good bet is that a large existing company will find the idea of URD intriguing and might invest in development and deployment to supplement its current product line or try to enter new markets with the innovative solutions provided by URD.

How this will play out depends very much on the goals and strategies of the company in question and on their internal culture and management, so it is very hard to predict how such an event might turn out. What is fairly certain however is that a company doing this will be prepared to invest heavily to ensure success, and will therefore provide the amount of money and manpower that is required to ensure that URD becomes a reality and that it secures most of the market with them.

To achieve the full potential of change that URD can bring to society this is most likely the solution that will be most successful, but there are some inherent risks involved as well. For instance can the commercial aspect become too dominant within the company, which in turn may affect the quality and services offered by the system in a negative way.

A non-profit organization

Another possibility is that a non-profit organization interested in building the foundations for the information society will take it upon themselves to develop URD. This can for instance be the Apache Foundation or a completely different entity.

Usually such an organization will not run the system commercially, but this may cause several modules to be unusable due to limited options for negotiating access with Telco's and other commercial companies that parts of URD depend on as third-party services. Still the services that are implemented can be expected to be of high-quality, but the limitations above and limited interest in commercially pursuing more users and partners can prevent network effects to appear and hence leave the system as a good idea that didn't catch on.

A random coalition of developers on an open-source project

If enough skilled people find the concept of URD intriguing a random coalition can spontaneously gather around some enterprising person(s) to develop URD on a voluntary basis as an open-source project, perhaps organized on SourceForge [www.sourceforge.org].

While this is a noble way to create the system it will likely not get much impact then as there will not likely be anyone capable of deploying the primary servers in a stable and trustworthy fashion, without which the system will likely never gain the large share of independent users that are required to cause the network effects to take place. There are also all of the downsides of a non-profit organization unless one of these chooses to adopt the project. This means that there won't even be a defined entity to negotiate deals for the root servers, and even less one to provide support and marketing for the system.

5.2.2. Independent groups

Many large companies and organizations will likely be interested in running their own URD-server for members and employees, in the same way that they are running Exchange or Lotus today. The reasons that they do this will vary from security concerns to being able to offer a better and/or cheaper service for their members, but the URD-systems that they run will be mostly equal. However there are some differences between the groups that need to be addressed, and this can be done through providing different packages of URD.

The primary groups that need to be addressed are:

Technological families and organized groups of friends

They will most likely want to run their own server to limit dependability on a commercial third-party, and to save on communications costs by avoiding potential fees for usage. It will probably be handled through a single or a few skilled people that deploy and maintain the service and provide it to a group of friends and family-members. The groups will most likely never consist of more than 50 individuals and the services provided can be limited.
This group could also include small sized companies with high technological competency.
This means that a stripped down and limited functionality version of the basic URD-system could be provided for these groups at very low cost reflecting the actual potential of lost users.

Large companies and organizations

The primary group expected to run their own servers are large companies whom because of security and other reasons prefer to keep systems like this in-house. Here there can be almost any number of users from less than a hundred to several thousands and there will most likely be a dedicated in-house or outsourced IT-department taking care of operations. These companies will require the same or better functionality than what the URD Corporation provides for independent users, so they should run an URD-system very similar to what is on the root servers and is able to pay well for the required licenses.

Very large international corporations

Some corporations are so large that a single set of URD-servers can not cover their requirements, often due to large departments located on different continents. The ordinary URD-system as provided for large companies could do the job, but it might require a lot of extra work by the URD Corporation since there could be a demand for frequent changes extending to the root databases. To remedy this situation one could allow those companies to run an extended version of the URD-system that will allow them to have more control over their URD-domain by allowing them to split it further into sub domains that can be managed locally. They could also be allowed to run backup root-servers for the entire URD-system to keep their branches online even if the URD Corporation should encounter major problems.

5.2.3. Network-providers

Since all of the communication to and from URD must pass over some kind of communications network the operators of these networks will naturally have some power over URD. Because of the desire to maximize usability there must be agreements negotiated with both some major Internet Connection Points (e.g. NIX in Norway) and with some or all of the major phone-companies (e.g. Telenor, Netcom).
It is required to have server farms connected nearby or directly to internet interconnects to ensure fast response times for users at any provider, and the phone-companies are compulsory partners to provide dial-in services and SMS-gateways for commands and notifications. Since such services will cost extra it might be required to charge extra for their actual usage independent of the models for revenue streams discussed below.

5.3. Revenue Streams

For any business to be successful and prosper there must be some kind of returns in exchange for the goods or services provided. While the open-source group and non-profit organizations mentioned above will most likely not require mandatory monetary contributions, they too will expect something in return, either as donations, voluntary work or applications from new members. Since such expectations may vary enormously between organizations I will not provide a detailed analysis of how the service can be managed by them, but instead focus on the monetary returns generated by the commercial business models that the up-start and existing companies will most likely choose. Below is an analysis of the most common models to generate revenue streams, as well as some others, to see how they can be leveraged to provide a solid financial foundation for an URD Corporation with happy customers. Knowledge on relevant models and markets are based on [Shapiro] and [WikBM].

Model (Score)	*Customers*	*Income*	*Stability*	*Adaptability*	*Scalability*	*Cost/user*
Pay-per-use (8)	Individuals	Low	Low	High	Low	High
Subscription (13)	Individuals Small groups	Medium	High	Low	Medium	Low
One-time-fee (14)	All	Low	Low	High	High	Low
Software licenses (11)	Large groups	Medium	Medium	Medium	Medium	Medium
Directed advertising (9)	Individuals	Medium	Low	High	Low	High
Combination (14)	All	High	Medium	Medium	Medium	Medium

Table 5.1: Comparison of revenue stream models

Based on this chart there is a tie between the one-time-fee and combination models in suitability, but as the latter give the higher income it is likely a better model for URD.

Definitions of Criteria

Score	Total sum of criteria. Each criterion is assigned 1-3 points depending on value.
Customers	The most suitable customer groups for the model. (More is better)
Income	The relative amount of income generated in the long term. (High is better)
Stability	The relative dependability that the revenue stream will continue. (High is better)
Adaptability	The relative effort required to change models or services (lock-in). (High is better)
Scalability	The option to increase number of users drastically without increasing costs equally. (High is better)
Cost/user	The relative administrative cost per user. (Low is better)

5.3.1. Pay-per-use

With an accounting system available in URD a natural revenue-stream is to charge users for their accesses to the system, potentially with different costs for different operations. This will be very much like the accounting systems of the phone-networks where you pay a set fee for every time you use some given service. Payments can be associated with each message, update, linked user or calendar entry, and with higher tolls for operations using costly third-party networks like phone and SMS.

However there are several downsides to this solution, especially the amount of work required to perform registration of users and to do correct personalized billing. If the billing cycles are short, like weekly or monthly the work involved can be quite demanding, especially for a start-up company without well established routines and a competent accounting department. Also, a pure pay-per-use model can not be trusted to bring a steady income since the amount of usage can vary a lot and thus cause problems for the URD Corporation. This can be especially risky during major holidays when usage may drop significantly while the fixed costs associated with running a server-park will keep going.
In addition while this can be a very suitable model for new users with low usage, many users and especially companies who expect to be using the service a lot might avoid it due to fears of attracting larger bills than expected without getting their moneys worth. This too can cause limitations in cash-flow and cause the demise of the service.

A more viable pay-per-use model is one using a pre-paid system, where a user can buy a number of credits in advance and perhaps be able to receive discounts for large amounts. This will allow users to limit themselves to a set cost while still providing a larger incentive to keep using the service, at least as long as there are credits remaining that cannot be converted back to hard currency, however many of the risks mentioned are still present.

5.3.2. Subscription-models

Opposite the complex pay-per-use systems you find the simple subscription systems. Here you can have a flat monthly or yearly fee for each user, independent of actual usage. This will give a high level of stability both for the revenue-streams to the URD Corporation and for the users who will not have to worry about a growing bill. While there must still be a billing-department managing the subscriptions the workload should be considerably less than for the pay-per-use model since all the amounts are standardized. Further advantages are the customer lock-in effect caused by a running subscription, which will increase a user's perceived cost of leaving the system.

A slight drawback with the subscription model is that it will give new users a very high-threshold to join the service, since very few would like to pay for an expensive subscription to a service they have not tested and might not even use more than once. A way to mitigate this problem somewhat is to find something that will increase a user's perceived value of joining the network, but this can be a daunting task and should not be depended upon.

However, as shown by the comparison chart the subscription model is a very suitable model for the URD system given its stable income stream and low cost per user.

5.3.3. One-time fee

On the other side of the subscription model there is the one-time fee, an even simpler model as each user will only pay once after which they can use the service freely. There are few requirements of the billing department when using this model, but the company as a whole will be a lot more pressed to constantly find new users. While this can be possible for a while the market will eventually be saturated and the income streams will dry up. Another problem is that the company will have no reason to focus on their existing customers, and hence has no incentive to make updates or fix errors to keep them happy. In the long run this will alienate their customer base and they will most likely stop using the service and spread bad publicity.

For the reasons stated above the one-time fee is not a suitable model at all for a venture like the URD Corporation described above, despite the high score in the comparison chart.

5.3.4. Software licenses

A subset of both the one-time fee and the subscription model is the use of software licenses. This can be a suitable model for those that will be operating their own server, as it will be hard to perform billing to them based on the regular models. Instead the groups running their own server will first buy a time-limited license to use the URD software for a one-time fee and deploy it as their own system. While there are no guarantees that the groups will buy another one-time license when their original one expires, there are strong incentives for them to do so based on the time and effort they have put into the service already.
Upon renewal the licensing can be modified to be more like a subscription model where future renewals are handled automatically without requiring the company to request each of them specifically, risking interruptions of service if it is not handled in a timely fashion.

Based on the three distinct independent groups that will be likely to operate their own URD-servers, there should be three different software licenses available with appropriate pricing for the amount of users and work required by the URD Corporation for each of them. Pricing for each of the subscription-types does not have to be a fixed value, but can be based on the number of users they register in the directory service or on another valuation method. This will provide a steady stream of income with prices suitable for each of the expected buyers, while at the same time retaining clearly separated markets without much risk of interference between them that could force a decline in prices.

5.3.5. Directed advertising

Because of the direct impact use of the URD system has in the lives of people, and the large amount of personal information that will be gathered over time, URD can easily become the wet dream of any marketing department. Because the system both can be used to create detailed profiles on habits and interests, and at the same time have a range of direct channels of communication directly to the user a combination of the two could be used to deliver accurate directed advertising personalized for every user. This can be the proverbial goldmine of advertising allowing for great revenues unless the concept is banned by the government for being too intrusive, or users boycott the system in disgust of the marketing tactics.

While such dangers are quite probable the directed advertisements model could still find a smaller yet viable target in those users willing to use the service with advertisements, but who would not use it if they would have to pay for the service personally. While the size of this user group is undetermined the successes of services like Eurobate who deliver free SMS services to the Scandinavian markets in exchange of permission to send directed advertising to registered users through SMS [Eurobate].

This would also present a good way to introduce the system for new users since they will not be required to invest anything other than time initially, and to discourage users from staying with the advertising-model the amount of advertisement could be matched to a users activity.

5.3.6. Combinations of various models

Of the models for revenue-streams introduced above there are major disadvantages related to each of them that may limit the deployment of URD severely. However most of the disadvantages in question are only related to single groups of users, meaning that a combination of several models can negate all the disadvantages and maximize both profits and user satisfaction in every market-section.

For the low-end market of new and occasional users the pay-per-view or advertisement models are suitable. The users do not have to commit themselves to large expenses and the perceived cost of trying out the system is therefore very small, and the cost will stay small as long as the users are not being very active, but it will make for an easy transition to subscription based system for those that use the system increasingly.
For those users that are initially more active and for small companies the idea of frequent advertisements or metered costs without bounds are usually not very attractive, so they should be given the chance to sign up for various forms of subscriptions through which the service will be provided at a periodic cost that can be used in financial planning as it varies predictably based on a few variables that the company are in control of, like for instance amount of users. This will also provide some lock-in effects due to the cost of learning the system and the uncertainty of changing from system with known costs and advantages.

In the high-end market of large companies and corporations with an amount of users that would make the subscriptions very expensive to maintain or with other requirements like security or stability, companies can choose to buy software licenses to deploy URD on their own premises. This will free them from potentially costly subscriptions and allow the group to decide and deploy the license that matches their requirements best, and through it only commit themselves to a fixed price for a set time, regardless of how much the system is used and without any requirements to renew the license. This will however cause a very powerful lock-in effect since the cost of deploying a new system is often very high, so it ensures long-customer relationships. This arrangement will give the URD Corporation a great deal of market power since even the larger clients will be reluctant to replace their systems.

For these reasons and the high score on the comparison chart, this business model is clearly the best suited to maximize income and customer satisfaction with URD.

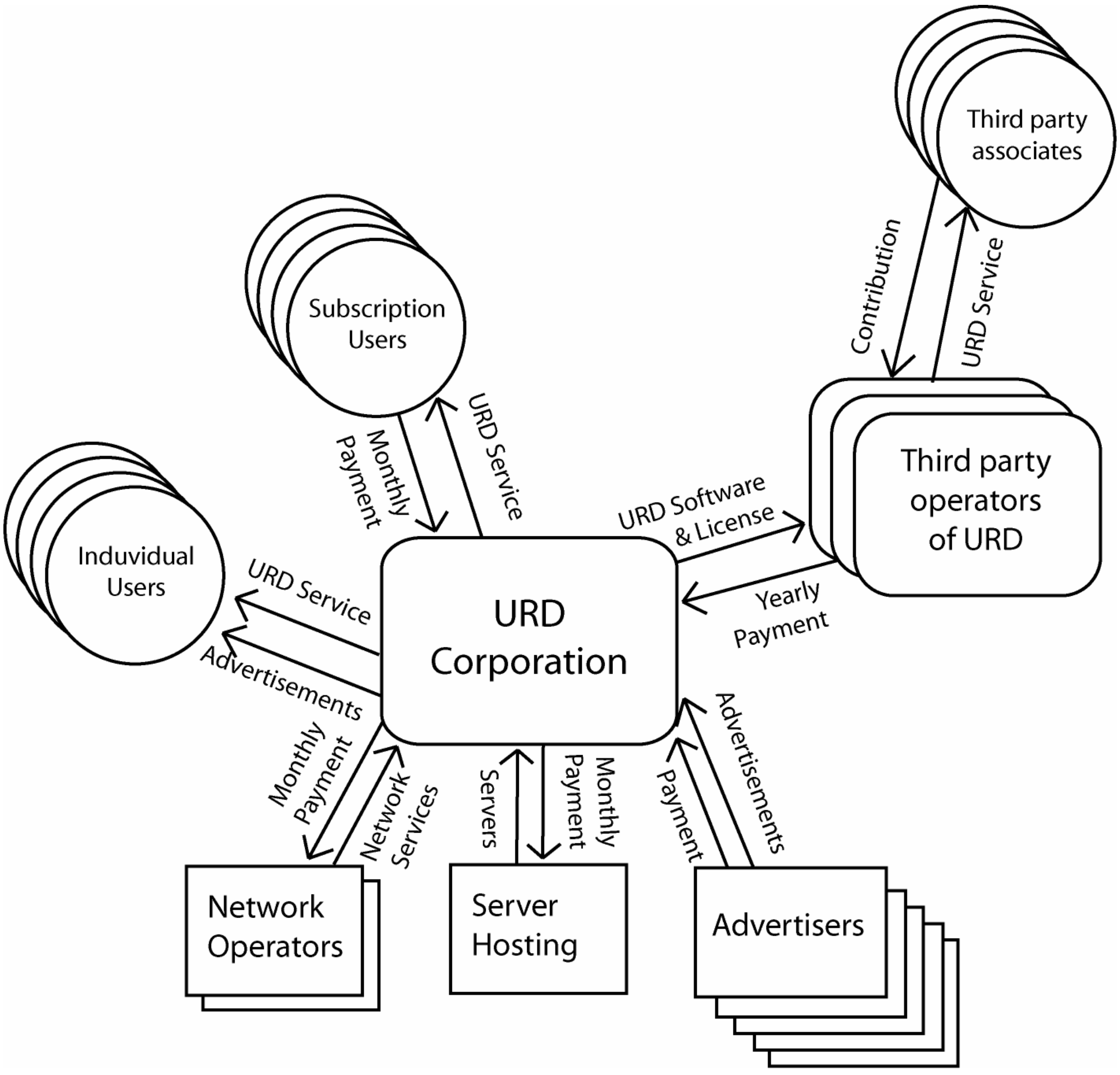


Figure 5.2: Value Chain Analysis of the combined revenue stream model

This figure shows how value items are transferred between the various factions involved in the URD Value chain. Three distinct business models can easily be identified:

1. Advertising model with advertising and payments from Advertisers going to the URD Corporation, who provides the advertisements and URD Service to Induvidual users.
2. Subscription model where Subscription users pay a monthly fee for the URD service.
3. License model where Third party operators pay a yearly license fee in exchange for the URD Software and License which is used to provide the URD Service to the associates of the Third party operators.

In addition we see that the URD Corporation exchanges montly payments for server hosting and network services to be able to provide the URD Service. This applies to Third party operators as well but is not shown in the figure as it is centered on the URD Corporation.

5.4. Customer base and earnings potential

Based on the business models and relevant customer groups described in chapters 5.2 and 5.3 above I believe that it would be a good idea to target the business-market first. This is because the revenue models that are most suitable for large companies will bring a steadier stream of income with longer lasting contracts than one is likely to get when targeting individual people. The earnings-potential in the business market is also greater for than it is in the private market because businesses are usually a lot less price-sensitive than private people for services, and especially if they perceive that using the service will cause improvements in the profitability or efficiency of the company.

Another advantage is that businesses running their own services will require a smaller support system in place at the URD Corporation than would be necessary to maintain the same number of private clients at in-house servers, since home users are usually less technically inclined than the administrators at large businesses. This means less money has to be spent initially by the URD Corporation to be able to support the growing installed user-base. However this approach also brings yet another advantage when the URD Corporation eventually will proceed to deliver a service to individuals because if the system become a success among businesses, then many of the employees might want to use the system to organize their private lives as well so large amounts of advertising costs can be saved as there will likely be a good amount of word-of-mouth publicity going around.

But let us start by focusing on the large business segment of the market, and lets at the moment limit us to the potential available in Norway to see if there is a viable market for URD at all. According to the Central Statistics Bureau [SSB] there were 1562 active registered businesses in Norway in 2005 with more than 100 employees, and these companies employ about 700.000 people. All of these should principally be interested in using the URD-service, but to be more realistic we assume that an estimated 25% of them would actually deploy URD in the near future if it was available as a service right now. This means that the potential market for URD is about 400 businesses with an average of 450 employees per company. Note that these are very low estimates and that the real figures could be much higher, especially since there is a lot of businesses with less than 100 employees that would most likely be interested in the URD-service as well.

As detailed in chapter 5.3 the business models that appear to be best suited for businesses is the sale of time-limited software licenses for those running their own servers or some kind of subscription model if the URD Corporation are to operate the servers. For simplicity I will assume that both models will have comparable costs (and income) in this example.

The only unknown factor left to calculate the potential-market size now, is the actual price that will be charged for the URD service. This is rather difficult to calculate accurately at this point since it depends greatly on both the development costs that will need to be reclaimed, and on the ongoing costs for operating the servers, customer support and administrative tasks in the URD Corporation. However there are somewhat similar services available on the market that can be used to get a price-point for some estimation.

Using the prices for the Microsoft Exchange hosting service at [Localweb] as a baseline, the 450 users per company would cost about 4000 USD per month on a set of dedicated servers. If all 400 companies ordered this solution, the estimated yearly market value of providing the service to large businesses in Norway is 19.2 million US dollars by a very cautious estimate. Using Microsoft's own prices for Microsoft Exchange Server licensing as provided on [MSLicense] the estimated cost varies between 30.000 USD and 63.000 USD depending on the products chosen; however this cost is a one-time fee instead of a recurring one. Despite being a one-time fee this makes the total market in Norway worth up to 25.2 million dollars by the sizes stated above, and while the companies could keep using the existing version of Exchange for a long time it is likely that most will upgrade every few years incurring additional costs, just as a can be the case with URD if the URD Corporation chooses so.

Most likely the full market value will be several times the figures stated above, and in addition there is the private market for individuals and smaller companies as well. Based on this the annual market value in Norway alone is likely to be worth at least 50 million US dollars per year, and since the URD-system is designed to scale well it can potentially get a large piece of the entire world market for scheduling applications, which could be worth billions of dollars just in licensing revenues.
On a different note it should be mentioned that the pricing and licensing of Microsoft Exchange may vary considerably from what will be viable for URD. Because of this the previous calculations

6. FINDINGS AND CONCLUSIONS

6.1. System design and technology

In chapter 4.2 I have presented a sample high-level design of an Agenda-system based on the requirements of chapter 4.1 and the Agenda-system descriptions in chapter 2. The system proposed consists of three primary parts; namely the main Agenda-service itself running on a set of distributed servers, a Directory service which usually will be combined with the main agenda service to allow identification of the home-server for any given user, and last a wide range of terminals from which users will send and receive commands to retrieve and modify information from the Agenda-service. For the Agenda-system with this design I have chosen to use the name URD from Nordic mythology.
The working structure of the full system of distributed servers are loosely based on the architecture of the common email-system, where an unlimited amount of servers provide facilities for the reception of digital messages for their own users with the help of DNS, with each server retaining full control of their own domains.

6.1.1. The URD service

Out of the available options that I have researched for the Directory service it appears to me that the use of X.500 DAP or LDAP will be best suited to complement the Agenda-service part of URD. This decision is because a DAP system will both allow a distributed directory to be created and deployed without requiring any formal approvals from standards-organizations. The directory will also be capable of storing any kinds of identifying information about a person which means that identifying the home-server for someone will be very simplified since it will only require that any one of the identifiers for a person are known instead of requiring that users get one specific identifier unique to URD.

The main Agenda-system will be best implemented as the combination of two separate parts, namely the base system itself and a series of plug-ins to provide the communications interfaces. This distinction is important because it will allow any given installation to choose which interfaces will be available by only installing the required plug-ins while at the same time allowing fast and easy implementation of plug-ins to support new communications protocols that will be developed in the future. All these plug-ins will communicate with the

base system through a standardized interface called the Command Access Interface (CAI) using the Command Interface Protocol (CIP)
This modular architecture is also retained inside the base system where there is a main system core that controls the operations of the system. In addition there is a series of subsystems to handle the various specialized tasks required to act as an Agenda-system, amongst these are subsystems for client authentication, the directory service, usage accounting and billing, policy-control mechanisms, data synchronization and of course an interface for the schedule-database itself. Each of these subsystems will perform their designated tasks to ensure the correct operation of the system core when given commands through the CAI.

In addition to these three main parts of the URD-service there are also Intelligent Agents operating both on the server and client-sides to provide automatic operations and better customization of user interfaces based on presence-information. The intelligent agents can also be used for automatic notifications and modifications by users when this is required. Usually these agents act outside the system using the CAI just like the communications plug-ins, but in a few specialized cases they can run like a core-subsystem.
The agents can also be available directly on URD-enabled terminals or even as stand-alone applications in embedded systems to provide data or services to an URD-client, and they can also be used for integrating data from third-party systems like Microsoft's Exchange into URD to smooth transitions for groups that are already using an established system.

In the future the increased capabilities of mobile systems and networks may allow every device to run its own copy of the URD-server, which will make the system into a true Peer-to-Peer network where every participant is equal, but this might require some changes to the system, and especially to the directory service due to its hierarchical design.

6.1.2. URD client terminals and interfaces

This brings us to the client-side of the URD-system that can be deployed using current technology. The primary terminal for advanced use of the URD-system will most likely be stand-alone applications for personal computers combined with web-applications that can be accessed from anywhere. This will make implementation of such clients, agents and other functionality quite simple since the development of applications has become routine.

However since personal-computer based terminals are not very mobile there are also a requirement to be able to access the URD-service through PDA's and mobile phones as well, and perhaps also through the regular POTS-network.
Most such applications will have to be developed specifically for every terminal, but by using JAVA and .NET technologies it will be possible to create cross-platform clients for a range of different terminals. For some high-end PDA's it might even be sufficient to just downgrade the regular applications to match the more limited environment.
Despite this some client-interfaces like those to the POTS-network and SMS-services cannot be implemented on the client side at all, but must have a server-side component to function. This can be solved through the communications plug-ins for the base-system mentioned above, but the interfaces will most likely be very primitive and limited to just the most basic functionality until better voice-recognition systems and natural-language parsers are developed.

As mentioned in the previous chapter the future might also bring a host of autonomous smart devices acting as intelligent agents and performing many of the client-tasks. The emergence of a true P2P based URD-system where every device will run a full copy of the URD-server will also diminish the requirements for specialized interfaces since each device can have a general client running along with the server, and this client can include interfaces to fit most of the possible variations of screen-sizes and other capabilities.

6.2. Business Models

The SWOT analysis in chapter 5.1 shows that while there are many threats to overcome during the development and deployment of an URD-system, the strengths and opportunities it provides can allow it to all but monopolize the market for scheduling services if success is achieved. However for URD to become a major player on the market for personal scheduling systems it will most likely require a commercial company to back it up.
This will be the URD Corporation and it can either be a start-up company with URD as its primary business model funded by Venture Capitalists, or it could be an existing company looking to enter new markets or gain a competitive advantage. Also with the URD Corporation as a commercial entity there is a much greater chance to negotiate good deals and perhaps even cooperative partnerships with other large companies to attain both monetary and competitive advantages over the competition, like for instance a bundling of the clients with some other popular software like MSN Messenger or the AOL Instant Messenger.

The combination of various business models as described in chapter 5.3.6 can fuel such an URD Corporation to become a very powerful player on the world market with a large market-share and considerable influence both with individual persons and large corporations.
The separation of markets described means that each market will be approached to maximize both profits and customer satisfaction, while still retaining the difficult low-end customers that provide recruiting for the higher levels.
In addition the valuable customers in the medium and high-end markets will be inclined to continue the arrangement with URD indefinitely due to the lock-in effect and high switching costs involved in choosing another operator at a later time. These things combined with the powerful network effects that can be realized from URD will ensure a solid grasp of the market within the foreseeable future, unless a new revolutionary competitor should emerge.

That the market of scheduling-applications is a goal worthy to pursue should be without a doubt based on the calculations in chapter 5.4. While being based on very rough estimates these calculations show that the Norwegian market alone is probably worth more than 50 million US dollars in annual licensing fees. And as Norway is a fairly small country this would most likely make the world market potentially worth billions of dollars.

7. FUTURE WORK

In this report I have described an Agenda-service called URD, detailed how it may be implemented and evaluated some business aspects related to deploying it. However this report is only a pilot study on the topic of Agenda-systems, and it merely establishes a foundation that may lead to years of further research and development before URD can become a reality.

Here are some pointers to future work, ordered by category:

Security

If the system is to become truly secure it is important that it is designed for security from the bottom up. Therefore the security essentials should be decided first so the rest of the system can be designed around a secure base.

The things that need to be addressed first and foremost are:

- User authentication
- Mutual server authentication
- Authentication translation in plugins
- Secure communication with users

It would be a great advantage if these points could be addressed in a uniform and consistent way to promote simplicity in the design and reuse of components.

Design and Developement

Before undertaking the work of implementing URD, it is neccessary to create detailed specifications and design of every component to be used in the system. The high-level outline provided in this report is the start of this work, but some modifications may be required based on limitations or choices that are made when designing the core.

Here are the most important pieces that need to be completed:

- *Protocol design of the CIP*

  The Command Interface Protocol must be designed in detail since this is the most important interworking part of the URD system. It is important that this design is well thought out since it must be able to support all the planned funcionality of the URD-system and it should be easily extendible to support new functionality as well.

- *Inter-core communications and detailed core-design*
  When the CIP is in place it is necessary to plan out the system core in detail, including how the core-modules will communicate with each other. This includes deciding exactly which functionality will be located in the various core modules, and which parts would be better located in agents either at the server or at the terminals.
- *Design of various plug-ins, agents and terminal interfaces*
  For the system to be of any real use it is neccesary to design the interfaces and functionality of the communications plug-ins and terminals through which users will access the system. The specifications for the most useful of the intelligent agents (e.g. reminders and status monitoring) should also be created as these abilities are important to give users an advantage over their current systems.
- *Implementation*
  Nothing but a daunting task, as implementation of the abovementioned components will require considerable time and effort. This is probably best performed through a commercial entity that can assign large amounts of resources to the task, but it could also be completed as a collaborate work in the spirit of Open Source.

Business

The chances of URD becoming a universally used system will be much higher if its deployment is backed by a commercial entity. A large existing company with a solid financial position will be best equipped to handle this, but since it will not be hindered by existing company-politics or bureaucracy an upstart company might very well be a better choice if it uses sound planning and competent people.

Regardless of which entity deploys URD the following tasks will need to be performed:

- Wide market research must be performed where amongst other things the strength of competitors, price elasticity and customer interest must be polled.
- A sound business plan must be written based on the market research.
- The expected costs of development and operations must be determined as a base for working out pricing schemes and profitability.
- A marketing strategy and deployment road-map should be created.

And then there will hopefully be both profits and a better world with URD! ☺

8. REFERENCES

[Bolstad] Bolstad, K., "*Personalized options* on *no answer conditions*", Cand.Scient. thesis, Department of Informatics, UiO, May 2001.

[DNS] IETF, "DNS Resources Directory", Information website, http://www.dns.net/dnsrd/, Accessed 01.12.2005

[Domino] IBM, "*IBM Domino Express*", Information website, http://www-142.ibm.com/software/sw-lotus/products/product4.nsf/wdocs/dominoexpress, Accessed 12.11.2005

[Email] GBdirect, "*An overview of Internet Email*", Information website, http://ebusiness.gbdirect.co.uk/howtos/mail-system.html, Accessed 01.12.2005.

[Eurobate] Eurobate ASA, "*Vilkår for Gratis SMS-tjenester*", 2005, http://www2.eurobate.com/index.php?option=content&task=view&id=39&Itemid=149, Accessed 4.12.2005

[Exchange] Microsoft Corp., "*Exchange Server 2003 Product Overview*", Information website, http://www.microsoft.com/exchange/evaluation/overview/, Accessed 12.11.2005

[Foerster] Foerster, E., "*Kontekst I heterogene nett med fokus på brukerkontekst*", Project Assignement, ITEM, NTNU, June 2004

[iCal] Wikipedia, *"iCalendar"*, Online Encyclopedia, http://en.wikipedia.org/wiki/ICalendar, Accessed 24.11.2005

[iCal] Apple Computer Inc., "*iCal*", Information website, http://www.apple.com/macosx/features/ical/, Accessed 12.11.2005.

[Infosec] Stallings, W., "*Network Security Essentials 2.ed*", Prentice Hall, 2003

[iSync] Apple Computer Inc., "*iSync*", Information website, http://www.apple.com/macosx/features/isync/, Accessed 12.11.2005.

[Jabber] Jabber, "Jabber :: About :: Overview", Information website, http://www.jabber.org/about/overview.shtml, Accessed 01.12.2005

[LDAP] Howes, Timothy, "*The Lightweight Directory Access Protocol: X.500 Lite*", CITI Technical Report 95-8, University of Michigan, 1995, http://www.kingsmountain.com/directory/doc/ldap/ldap.html

[Localweb] LocalWeb.com, "*Leverage the power of Microsoft Exchange Server*", Information Website, http://www.localweb.com/adti/hosting_exchange.html, Accessed 06.12.2005.

[MSLicense] Microsoft, "Pricing and Licensing for Medium Organizations", Information website, http://www.microsoft.com/exchange/howtobuy/medium.mspx, Accessed 06.12.2005

[Notes] IBM, "*IBM Lotus Notes*", Information website, http://www-142.ibm.com/software/sw-lotus/products/product4.nsf/wdocs/noteshomepage, Accessed 12.11.2005

[Novell] Novell, "*Novell Groupwise 7*", Information website, http://www.novell.com/products/groupwise/overview.html, Accessed 12.11.2005

[Osthus] Østhus, E.C, "*Presence and call screening in VoIP*", Project Assignment, ITEM, NTNU, December 2004

[Outlook] Microsoft Corp., "*Outlook 2003 Product Overview*", Information website, http://www.microsoft.com/office/outlook/prodinfo/, Accessed 12.11.2005

[Parlay-FAQ] The Parlay Group, "*FAQ*", Information website, http://www.parlay.org/en/about/faq.asp, Accessed 15.11.2005

[Parlay-Spec] The Parlay Group, "*Specifications*", Information website, http://www.parlay.org/en/specifications/index.asp, Accessed 15.11.2005

[ParlayX] Telcordia Technologies, "*Parlay X*", http://www.argreenhouse.com/parlayx/, Accessed 15.11.2005

[Shapiro] Shapiro, C. & Varian, H.R., "*Information Rules: A strategic guide to the network economy*", Harvard Business School Press, 1998

[Skogstad] Skogstad, B.E., "*Mobile Repository System: Net-centric data synchronization and storage",* Masters Thesis, ITEM, NTNU, June 2005.

[SSB] SSB, "Bedrifter og føretak", http://www.ssb.no/emner/10/01/naeringsliv/, Accessed 06.12.2005

[Sync] The SyncML Initiative, "*SyncML Sync Protocol, version 1.0.1*", 2001, http://www.openmobilealliance.org/tech/affiliates/syncml/syncml_protocol_v101_20010615.pdf

[SyncML] Open Mobile Alliance, "*SyncML Whitepaper v1.0*", http://www.openmobilealliance.org/tech/affiliates/syncml/whitepaper.pdf

[SyncRep] The SyncML Initiative, "*SyncML Representation Protocol, version 1.0.1",* 2001, http://www.openmobilealliance.org/tech/affiliates/syncml/syncml_represent_v101_20010615.pdf

[Telenor] Telenor R&D, "*Appendices and related / used reference material*", Telenor R&D R 4 / 2005.

[TelenorSMS] Telenor Mobile, "*SMS fra Outlook*", Information website, http://telenormobil.no/tjenester/smsfraoutlook.do, Accessed 12.11.2005

[vCal] Internet Mail Consortium, "*Personal Data Interchange: vCard and vCalendar*", Information Website, http://www.imc.org/pdi/, Accessed 12.11.2005.

[Vo] Vo, Hong N.T., *"Calendar and contact synchronisation with P900"*, Project Assignment, ITEM, NTNU, December 2004

[WikBM] Wikipedia, "*Business Model*" and related articles, Online Encyclopedia, http://en.wikipedia.org/wiki/Business_model, Accessed 1.12.2005

[X.500] Shuh, Barbara, "*Directories and X.500: An Introduction*", 1997, http://www.lac-bac.gc.ca/9/1/p1-244-e.html, Accessed 25.11.2005