A Universal Messaging Agent Integrating Device Modalities For Individualised Mobile Communication

Markus Lauff, Albrecht Schmidt and Hans-W. Gellersen

TecO - Telecooperation Office - University of Karlsruhe
Vincenz-Prießnitz-Str. 1 - 76131 Karlsruhe - GERMANY
Phone +49 (0)721 6902-69, Fax +49 (0)721 6902-16
e-mail: {markus, albrecht, hwg}@teco.uni-karlsruhe.de

Abstract. A universal messaging agent (UMA) for more flexible communication with multimedia teleservices is introduced. The approach is general and application-independent; the UMA supports a variety of device modalities that can be flexibly configured for two-way communication. In particular different device modalities can be chosen for either direction. This paper focuses on the request/reply model for interaction with services underlying UMA. In this model, the *requester* and the *receiver* are separated clearly: requester and receiver can be located in different devices, utilising different *device modalities* such as SMS, Fax or Handheld Web browser. Further, requester and receiver can be separated in time, that is both requests and replies can be scheduled independently.

The UMA accepts connections from different devices; these request are using protocols that are device-optimised, for instance optimised for usage with a GSM phone. The first processing step is to translate the request into a number of commands that have to be processed to compute the reply. This request analysis and translation is based on user profiles and a service profile database. In the next step the information is gathered, either immediately or according to a requested schedule, and then filtered, structured, and bundled. The generated reply finally is optimised for the receiver device that may differ from the requester device.

The UMA enhances communication for heterogeneous and mobile environments through integration of device modalities. The new request/reply model underlying the UMA supports individualisation of mobile communication through support of highly configurable request/reply schedules.

Keywords. Multimedia-specific intelligent agents; Mobile multimedia systems; Device Modalities; Ubiquitous Computing; Media Conversion; Profiling

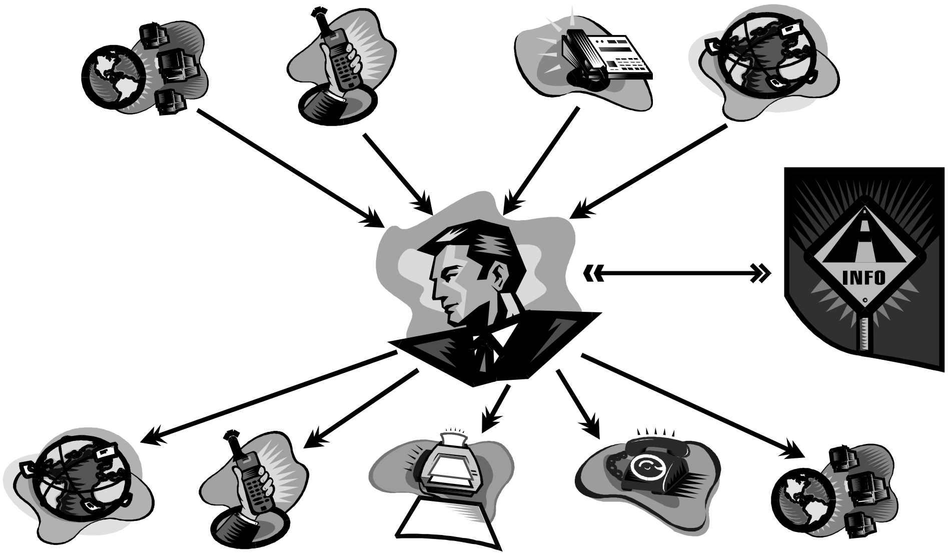
Introduction

Mobile people have an increasing number of different communication devices available in different surroundings. Ideally, these devices would be integrated to support flexible communication with any kind of teleservice (interactive services, distribution services, mail and conversation services) [2][3]. Currently, though, access to teleservices is everything but flexible:

- Services imply certain modalities for service access; these modalities are related to specific types of device, and henceforth referred to as *device modalitie*s. Example for implied device modalities are e.g. touch tone for phone-based information services, and desktop browser for web-based services. In order to provide for service access from other device modalities, gateways are used, which have to be chosen explictly though.
- Customisation is controlled by the server side; clients have no general means for customisation of service replies, such as for information filtering or media conversion. A notable exception is email with tools in abundance for individualisation. Still, even in this restricted domain, individualisation does not go as far as covering different device modalities.
- For request and reply the same device modalities are assumed. A characteristic of interactive multimedia services is, though, asymmetric communication with small requests and large replies. Therefore, it would

often be preferable to utilise different device modalities for request and reply in adaptation to the respective requirements. For example a request may be best made from a personal digital assistant to exploit personal profiles, but the reply may be better received on a (less personal) desktop.

- It is assumed that replies should be delivered as soon as possible to the requesting client. This may often be the straightforward mode for processing interactive services but in many cases it will also be less suitable. In particular in mobile environments, there will often be a demand for information delivery to a certain place (device) at a certain time.



The Universal Messaging Agent introduced in this paper addresses these issues based on a new request/reply model. The UMA is a mediator between an open range of device modalities and any kind of teleservice. Request and replies can be scheduled independently. Device modalities and request/reply schedules are determined based on request analysis, user profiling and service profile data. Subsequently, first the UMA model is introduced followed by an introduction of the optimised deveice-dependent protocol (ODDP) used for communication with the UMA. The section on ODDP aims at further motivating the UMA case with a range of examples. The following section will then describe the system architecture and key components. The paper concludes with a summary.

UMA Model

Using the suggested UMA architecture the request/reply communication is modelled in a more general way.

- The limitation that the requester has also to act as the receiver is eliminated; the requester can specify, explicitly or implicitly, the device to which the reply should be sent.
- Furthermore the type of media (prerequisite: there exists a conversion module) can be chosen, which is especially important when using devices with very limited displays, such as PDAs or GSM phones.

The tight temporal coupling of request and response is but one option in the UMA; the general case is that the requester is free to schedule information delivery. Looking at the range of devices that are considered as potential requesters and receivers one single standard protocol can not suit the needs. The price for a single and general protocol for all devices would be an enormous communication and protocol overhead. This would created a number of problems especially for devices like GSM phones with limited programming capabilities. Furthermore the user/protocol requirements of different devices (a fax machine, a Workstation, or a GSM phone) are very diverse.

In the suggested architecture the problem is solved by introducing optimised device depended protocols (ODDP) (up- and downstream from UMA to the device) and standard internet protocols from UMA to the Internet (UMA to information provider).

User requests are sent in the ODDP from the requester to the UMA’s request analyser. Dependent on the type of the request one or more tasks are scheduled for execution. A simple GET-request from a web-browser maps to a single task (getting the document) and would be scheduled for immediate execution. A request from an GSM to send the main stock figures from 1pm to a fax at 3pm though would result in a number of tasks scheduled at different times (at 1pm getting the stock table, before 3pm prepare table as fax document, 3pm send fax).

ODDP

ODDP stands for Optimised Device Dependent Protocol. This means, that the structure of a request is optimised for the capabilities of the requesting device and the data send within the reply is structured according to target device requirements.

ODDP request

The user can enter his request in a convenient manner. For example in the case of a SMS message, the user does not have to enter too many characters because the structure of the request is optimised to consist of abbreviations and code numbers. At first it seems awkward to use abbreviations and code numbers in cases where it is possible to enter the complete text, but in practice this is the often the more comfortable way to use the system.

The following section shows a summary of the optimisations used by the different devices.

SMS requests using a GSM phone:

Many GSM phones are able to send SMS messages. In general a SMS message can contain a maximum of 160 characters and digits. But because a common GSM phone only has about 12 buttons to enter the characters and digits it is very inconvenient to enter long messages. In addition to the inconvenient way to enter messages the usual display only has about 3 rows with each having 15 columns and so it can only show up to 45 characters at the same time.

Because of these limitations an SMS ODDP request sent from a GSM phone is abbreviated wherever it is possible.

SMS ODDP request examples:

```
TI A67 A5
```

This request asks for the traffic information concerning motorways A67 and A5. All other information required to complete this request are provided from the UMA’s user profile agent (UPA, cf. below).

In this example the UPA determines the user using the SMS sender address and the device to use for the reply from the users profile.

```
PT 3 5 9.30
```

This is a request to send a public transport timetable for a connection from stop code number 3 to number 5 at 9.30 am. People usually travel among very few local places (e.g. home, work, central station) so that codes to be held in the UMA’s users profile are very suitable. In addition to the information specified within the request the UPA determines the requesting person and the preferred device for the reply.

```
ST DTE 10.00 14.00 S 15.00 F
```

This request asks for the stock quotes from the German Telekom (stock symbol DTE at 10am, 2pm send at 3pm via fax to the requester.

Requests entered through general phone dialling:

A telephone is a rather primitive device to be supported by the UMA . The big advantage of this device is its is ubiquitous availability. The telephone may be used in two ways, the first described in this section is to dial a sequence of numbers, the second way described in the following section is to use natural speaking.

After dialling a service number the user can submit a request dialling a sequence of code numbers.

Tone-Dialling ODDP request examples:

```
6#67*5#
```

This request produces the same result as the SMS ODDP request “TI A67 A5“ described above. The number 6 is the code for the traffic information service and the following numbers separated by an asterisk are the motorway numbers.

Requests via email

This way of submitting requests can be considered as a superset of the methods described above. In addition to these methods a more complex parser can add more functionality to the system.

This request method can also be used to set up a general gateway functionality on an email based system (see second example)

Email request examples:

```
Always send me the DTE stock quotes after a value change of 1.5 percent
```

This request will be scheduled to periodically request the DTE stock quotes and send a message to the user if the change exceeds the 1.5 percent.

```
SMS FWD +491726236837
```

With this command in the subject line of the email, the attached email will be forwarded to the SMS phone. Prior to sending the email as one or more SMS messages, the email is filtered and unwanted parts such as uuencoded files are removed.

Natural spoken requests

This is often the most natural way of requesting information. After calling a service phone number the request is spoken to a voice recognition system. This kind of requesting information requires a complex and expensive hardware for speech pattern recognition. Depending on the used hardware it is possible to speak complete sentences or to choose from a list of words.

Natural speaking ODDP request examples:

```
“Send me Traffic Information for A67 and A5”
“Send me the 4 day weather forecast via SMS”
```

HTTP requests

HTTP is the protocol used in WWW environments. The UMA is able to analyse the request identify the requesting user and machine and optimise the result of the request for this user and machine. This optimisation can be any kind of gathering, filtering, conversion or bundeling of information accessible for the UMA.

HTTP request examples:

```
GET http://www.berlin.de/index.html HTTP/1.0
User-Agent: PocketWeb2/MP2000
```

This is a request of a standard HTML page from a PocketWeb WWW browser [5] running on a Newton MessagePad 2000. Because the browser can only display GIF images the UMA has to convert all other type of images to the GIF format.

ODDP reply

After the complete processing of a command the UMA generates an ODDP reply. This sections shows some considerations contributing to the structure of the final reply.

SMS reply

A SMS message can only contain up to 160 characters or digits, so replies with more characters have to be split into small SMS messages. In addition to this the reading of a SMS messages with a GSM phone is very inconvenient because the display often only has 4 rows with 15 columns (only 60 characters!) and the navigation through the messages stored on the phone is limited to a simple “next message” command.

For example in the case of an email forwarding ODDP request, the structuring module (SM) removes all unnecessary information from the original message before splitting it into small parts. Which information is unnecessary is determined by the target device and the users profile. An attachment of a file is always considered to be unnecessary on a GSM phone whereas the importance of an email footer is defined in the users profile. The users profile can also define a maximum number of SMS messages that should be created on a single request and the structure of these messages.

Fax reply

This is a subset of considerations described in the previous section about SMS replies. There are no strange limitations concerning the length of the messages but unnecessary parts are filtered as described before. Because fax machines are sometimes shared by different people the SM can create a cover page depending on the users profile.

HTML reply

Especially on small devices such as personal digital assistants (PDAs) many HTML browsers do not support all HTML features specified in the latest HTML specification and therefore used within HTML documents.

With the help of the UMA architecture it is possible identify the capabilities of the requesting device and to replace unsupported HTML elements through an alternate structure. For example an unsupported table construct can be replaced by a table send as preformatted text, or a JPEG image can be converted to a supported GIF format. For more information about these conversions see the chapter “Adaptive Proxy Architecture” in [5].

Phone reply

In this case the SM creates a set of messages that is spoken to the user using a text to speech module (T2SM). One important task for SM is to generate an individual set of rules for T2SM depending on the message to enable the navigating through the message. For example the question of the T2SM “Press 1 to skip the footer of the message or press 2 to continue” must be defined by the SM and integrated into the set of rules belonging to the reply.

System Architecture

In this section the main components of the UMA are described. The architecture is depicted in Figure 1 and contains three active components: the request analyser, the scheduler and the distribution module. The other components are invoked by these active components.

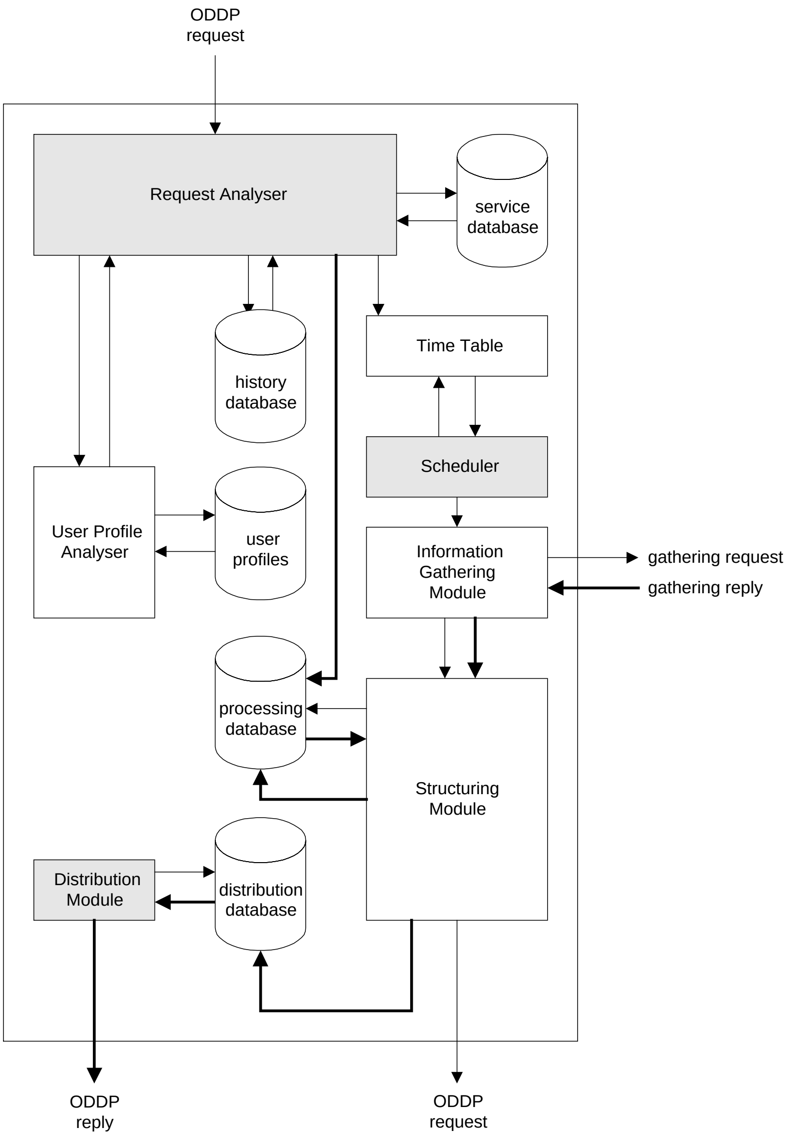


Figure 1: UMA System Architecture

Request Analyser

The concept of an ODDP depends on a module that has the ability to translate requests from different ODDPs into one system understandable request language. In the described system this is the task of the request analyser.

The basic approach used in UMA is that an incoming ODDP request is analysed and together with the knowledge about the requester (from the user profile analyser) and the information about the service that is requested (from the service database) a command is generated. This command consists of the main send command together with subcommands (information gathering, filtering, etc). Each request is stored in a history database with a unique ID; this keeps the information for the case that the request can not be processed.

The following two small example are given to show in more detail how this process works:

A simple HTTP request send from an PDA (Apple Newton):

Using a browser like PocketWeb [6] on a PDA a usual http get-request is sent to the UMA. In this case the ODDP is http and the request looks like this:

```
GET http://www.berlin.de/index.html HTTP/1.0
User-Agent: PocketWeb2/MP2000
```

The request analyser translates this request in sequence of commands that are schedule and entered in the timetable; in this case in the time slot 'now'.

```
HTTP_REPLY(Connection_handle):now
  {GET http://www.berlin.de/index.html HTTP/1.0;Filter=html4pda:now}
```

In a next step the scheduler is taking out this command from the timetable. Calling the IGM (Information Gathering Module, cf. below) with following command:

```
GET http://www.berlin.de/index.html HTTP/1.0;Filter=html4pda:ID
```

Reschedule the reduced command:

```
HTTP_REPLY(Connection_handle):now
  {ID}
```

Then the scheduler is taking out the command again; it is a single send command. This command is given to the IGM (there is nothing to do) then it is handed further to the packaging module. Here a protocol dependent package for the information stored under ID is created a put into the distribution database. Then the distribution module is sending the information back using the connection handle.

For this scenario the scheduling is not really necessary. In the next example it becomes more evident why this architecture is needed for a UMA.

Considering the SMS request explained above:

```
ST DTE 10.00 14.00 S 15.00 F
```

The RA gets the request and with the request the user identified by the phone number. Using the phone number the user profile can be acquired. In the user profile the information about the fax number of the user is stored (user_fax_number). The information services (http://www.stockmaster.com/sm/g/D/DT.html) relevant for getting information about the stock market are stored in the service database. To select a particular value there exits a filter module, which also described in the service database. The whole command is in the timetable at the slot a 10.00, at the time of the earliest action.

```
FAX_REPLY(user_fax_number):15.00
  { GET http://www.stockmaster.com/sm/g/D/DT.html;Filter=stock(DTE):10.00 -
    GET http://www.stockmaster.com/sm/g/D/DT.html;Filter=stock(DTE):14.00}
```

At 10.00 the scheduler is taking out this command from the timetable. Calling the IGM with following command:

GET http://www.stockmaster.com/sm/g/D/DT.html;Filter=stock(DTE):ID1

Reschedule the reduced command at 14.00:

```
FAX_REPLY(user_fax_number):15.00
  { ID1 -
    GET http://www.stockmaster.com/sm/g/D/DT.html;Filter=stock(DTE):14.00}
```

At 14.00 the scheduler is taking out this command from the timetable. Calling the IGM with following command:

```
GET http://www.stockmaster.com/sm/g/D/DT.html;Filter=stock(DTE):ID2
```

Reschedule the reduced command at 15.00:

```
FAX_REPLY(user_fax_number):15.00
  {ID1 - ID2}
```

Then the scheduler is taking out the command again at 15.00; it is a single send command. This command is given to the IGM (there is nothing to do) then it is handed further to the structuring module. Here fax report using the information form ID1 and ID2 is created and put into the distribution database. Then the distribution module is sending the information back using the connection handle.

Scheduler

The scheduler is the main active component. It controls the information gathering, filtering, extraction, structuring, bundling, and indirectly the deliverance of information on time.

The commands, consisting of one main command and a number of subcommands, build by the request analyser as described above are stored in the timetable. A command is found in the timetable at that time when the first action (command or subcommand) within this command is scheduled.

The scheduler is scanning the time slot at the time 'now', if there is no command in this slot no action is carried out. If there is a command in the slot this command is taken out of the timetable. The scheduler then analysis whether the command is a single delivery command or a complex command, having subcommands.

If the command has subcommands than the scheduler identifies the subcommands that have to be processed 'now'. This URL-based subcommand is given together with a unique ID to the information-gathering module. Within the command the subcommand is substituted with the ID. This ID is used later for bundling information pieces belonging to one request. After the substitution is done the reduced command is put back into the timetable at that time when the next action is scheduled.

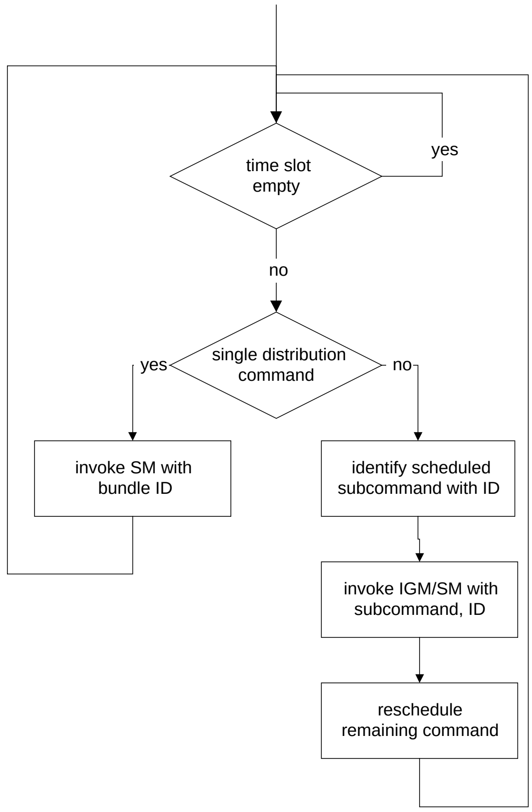


Figure 2: UMA Scheduler

If the command is a single delivery command the than the information gathering is void, in the structuring module all the information pieces belonging to this request are put together in the requested way (bundling) and stored in the distribution database. This causes the distribution module to deliver the information.

Information Gathering

The information gathering module (IGM) is the component used to collect information from different sources. The main source used in the project is the WWW, other sources are telephone services, like a weather forecast.

In the process of analysing the request as described above the information source is determined using the service database and the user profile. During this process also the protocol for the request is specified. When the request is scheduled the IGM receives the request together with an ID. This ID is used to store the retrieved information for further processing.

The information gathering request is rather easy; the module uses the specified protocol to get the information and stores it in the processing database using the given ID. The request processed by the IGM have the following form (The filter is used in the section "Structuring Module" and will not be discussed here further.):

```
GET protocol://information-location[;filter]:ID
```

This is oriented on request used on the Internet. In the context of the project this is extended more general as shown in the following example.

```
GET phone://011503:ID1
```

Using the information from this request the IGM dials the number 011503 (information on German TV-program) and stores the received audio data using the ID1.

Structuring Module

The structuring module (SM) is responsible for the processing of the available information. The kind of processing is defined in the filter part of the executed command.

```
GET protocol://information-location[;filter]:ID
```

The functionality of the SM can be grouped as follows:

- Converting
- Filtering
- Bundling

Converting

If the target device does not support the current data format used for the information, the SM converts the information to another data format (for example JPEG to GIF, ANSI character set to ASCII, text to speech).

In addition to the converting because of unsupported data types, converting is used to enhance the usability of target device. If the target device is connected through a slow link, the RA can use the user profile analyser (UPA) to determine the users quality of service (QOS) settings and convert the information according to this. While the choice of device modality often depends on pragmatics such as mere device availability the UMA will incorporate intelligence for device modality selection based on a sound theory similar to Bernsen's modality theory [1].

Filtering

The information either obtained from the IGM or stored in the processing database can contain unwanted elements. The filtering removes these unnecessary parts and stores the result in the processing database.

For example if the user wants to have the latest value for a specific stock. The IGM retrieves a page with a set of stock values and the SM filters the requested stock

Bundling

The bundling is the last step of the information processing and combines the different parts belonging to the current command and stored in the processing database to the final message.

Depending on the output device this message may be further processed by the SM to integrate text to speech rules or to split the message into multiple smaller ones (examples can be found in section ODDP reply)

Distribution Module

After a request has been fully processed the distribution module (DM) is sending the information to the client.

The structuring module stores the final ODDP reply together with the information about the receiver (phone number, email address, TCP-handle, etc.) and the protocol to use in the distribution database. For each package the ID of the original request is stored in the database, too.

The DM periodically checks if a package is ready for delivery. The information package is taken out of the database and the corresponding ODDP-reply is sent to the specified receiver.

If the receiver is not available (e.g. the phone is busy or the computer is down) the DM notifies the UMA using the ODDP and sending the original request ID together with a problem report.

Conclusion

Today's communication devices, such as GSM phones, email clients or fax machines are mostly used in homogenous environments. To enable communication between different types of devices special gateways are built that enable the exchange of information from one device to another one (SMS-email gateway).

Disadvantages of this approach are that the user has to explicitly select a gateway and that the gateway is not able to adapt the information to the modalities of the target device.

The described Universal Messaging Agent enables flexible information exchange between any type of communication devices. The target device can be selected from the universal messaging agent using the user profile analyser. The communication is optimised on the modalities of the used devices and through this the operational area is extended.

The clear separation between the request and reply allows the architecture to handle synchronous and asynchronous action in the same way. Due to the open concept of the request analyser and the structuring module the system can be easily adopted to the user requirements and new modalities or devices.

In addition to the new services the architecture can be used to model existing services (for example FAX or SMS gateways) in a more flexible way to enhance the overall usability.

In this paper the general request/reply model and the notion of device-dependent protocol has been highlighted, reflecting the current status of our work. A running prototype integrates phone, SMS, PDA and desktop modalities. The UMA project in addition focuses in particular on user profiling and issues related to device modality selection and media conversion.

References

[1] N. O. Bernsen: Modality Theory: *Supporting Multimodal Interface Design* Multimodal Human-Computer Interaction, ERCIM Workshop Report ERCIM-94-W003, Nancy, 2.-4. Nov. 1993, p. 13-23

[2] L. Cappelletti. *Mobile Computing - Beyond Laptops*. The 15th Annual International Conference on Computer Documentation. SIGDOC 97. Conference Proceedings. ACM 1997.

[3] L. Francis. *Mobile Computing - A Fact in your Future*. The 15th Annual International Conference on Computer Documentation. SIGDOC 97. Conference Proceedings. ACM 1997.

[4] S. Gessler, A. Kotulla., *PDAs as mobile WWW browsers*. Proc. of 2nd International WWW Conference, Chicago, Oct. 1994.

[5] M. Lauff and H-W. Gellersen: *Multimedia Client Implementation for Personal Digital Assistants*. Proc. of European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'97), Darmstadt, Sept. 1997, Lecture Notes in Computer Science, Springer-Verlag.

[6] Pattie Maes on Software Agents: *Humanizing the Global Computer*. Interview with Pattie Maes, IEEE Internet Computing. July-August 1997

[7]M. Mühlhäuser, H-W. Gellersen, S. Gessler: *WWW Distributed Client Architectures*. Interactive and Distributed Multimedia Systems on Highspeed Networks. WWW-3 Workshop, Darmstadt, April 1995.

[8] A. Schmidt, A. Specker, G. Partsch, M. Weber, S. Höck. *An Agent-based Telecooperation Framework*. First International Workshop on Cooperative Buildings. Darmstadt 1998.

[9] MOBYDICK. http://wwwspa.cs.utwente.nl/~havinga/mobydick.html. 1998.