19

Object oriented system architecture and strategies for the exchange of structured multimedia data with mobile hosts

J. Bönigk, U. von Lukas
Computer Graphics Center Rostock (ZGDV)
Joachim-Jungius-Str. 9, D-18059 Rostock, Germany
Phone.:+49 381 4024 150 Fax:+49 381 446088
E-mail: {joerge,uvl}@rostock.zgdv.de
URL: http://www.rostock.zgdv.de

Abstract

In this paper an architecture is presented which enables applications on mobile computers to transparently exchange multimedia objects with applications on stationary servers via the *Object Bus*. The components of our architecture face the common problems of mobile computing like limited bandwidth, end systems with limited resources and frequent disconnections. Therefore they are designed to efficiently use the available resources and to minimize these problems. This is done, e.g. by using object specific methods of data reduction and level-of-detail, by adaptation of transfer to given Quality-of-Service parameters, and by data compression. All these methods are influenced by contexts like local resources, available communication channels, and user preferences. Finally the abstract architecture is mapped to OMG CORBA. The architecture and a selection of the exchange strategies are used to build a prototype of a mobile information system.

Keywords

mobile computing, mobile communication, detail-on-demand, data reduction, mobile system architecture, distributed systems, object orientation, CORBA

1 INTRODUCTION

The main task of the underlying system architecture and communication model for the aim of mobile visualization within the MoVi project* (Heuer et al., 1995) is to enable the use of complex visualization applications and large sets of multimedia data in a mobile environment.

* The work of the project „Mobile Visualization" (MoVi) is supported by the German Research Association (DFG) under contract Schu 887/3-1.

J. L. Encarnação et al. (eds.), *Mobile Communications*
© Springer Science+Business Media Dordrecht 1996

One of our typical scenarios is the access to public or private data stored at several stationary data servers (SDS) for visualization on a mobile end system (MES) using GSM[1].

However, there are the well-known problems of mobile computing (Forman and Zahorjan, 1994) (Imielinski and Badrinath, 1994) (Weiser 1993) which should be considered when designing an architecture: limited bandwidth, resource poor mobile terminals, variations in quality of transfer and possible disconnections. Our architecture addresses the first two problems by reducing the amount of data to be transferred by compression and by filtering of relevant data according to the resources of the mobile system. The transfer of data is adapted to varying transfer characteristics and our communication model is designed to cope with frequent disconnections in all stages of the transfer.

The remainder of this paper is structured as follows: In the next section the components of our architecture will be introduced as well as the exchange strategies they make use of. In section 3 a mapping of the architecture and the communication model to OMG CORBA is presented. A first prototype of a mobile information system, which is based on the architecture and uses a selection of the exchange strategies is explained in section 4. After an overview of related work in section 5 a conclusion of our work and an outlook on future plans regarding the architecture and the prototype are given in section 6.

2 SYSTEM ARCHITECTURE

The system architecture has to serve as a flexible platform for the efficient exchange of user data and applications between stationary data servers and mobile end systems. Our main concept is an object oriented approach with the *Object Bus* (OBus) as one central feature. This Object Bus serves as a transparent layer for mobile communication and is responsible for the delivery of messages. The second main feature is the introduction of *Message Handlers* (MH) that act in place of the communicating processes when exchanging structured objects. They notify each other about transfer procedures and transfer the objects (see Figure 1).

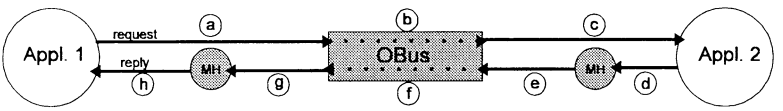

Figure 1: Exchange of messages via the Object Bus.

That means, that the whole communication between two processes A and B (a) is divided into the following main steps (see Figure 1):

(1) Applications generate requests that are handed over to the local Object Bus for transportation (a).
(2) The following transfer inside the Object Bus is carried out by several components (Network Scheduler, Transfer Manager) which reside between application and network layer on every computer (b).
(3) The receiver gets the message (c) and generates reply objects.
(4) The reply objects are handed over to appropriate Message Handler processes (d). The Message Handlers use knowledge about the context to enrich the reply with additional information, that is necessary for an effective scheduling. Additionally they perform

[1] Global System for Mobile Communication

operations on the reply objects like data reduction that are influenced by the resources of the mobile host. These operations are necessary in order to save the limited bandwidth of wireless communication channels.

(5) The Message Handlers on both sides transfer the objects with suitable methods via the Object Bus (e/f/g).

(6) The application receives the reply objects (h).

The third main component is a *Context Manager* (see Figure 2), which covers all aspects connected with contexts; e.g., context sensitive trading, context based modification of requests, and the management of contexts. It intervenes in the communication flow (a),(c),(d), and (h) between the components in Figure 1. In addition, it provides Message Handlers and the Object Bus with context information that are necessary for their work.

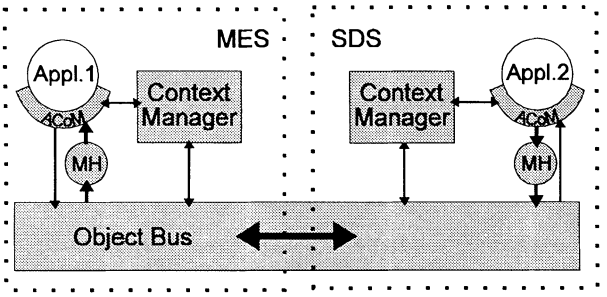

Figure 2: Overview: Components of the System Architecture on MES and SDS.

The communication between applications and Object Bus and Message handlers respectively is done via the Application Communication Manager (ACoM). The named components will be explained in the next sections in more detail.

2.1 Object Bus

In the context of mobile computing with slow, unreliable, and possible expensive wireless links, asynchronous calls (messages) are preferred to synchronous calls. The aim of the *Object Bus* (OBus) is to manage and efficiently transfer these messages on all kinds of networks. Components of the Object Bus are a *Network Scheduler*, a *Request/Reply Cache*, a *Transfer Manager* as well as a *user interface* on mobile hosts. The OBus is linked with the local Context Manager where it can access information that is necessary for its work, e.g. the current context information or the address of a requested service.

The *Network Scheduler* (NS) is the main component of the Object Bus. It is responsible for the efficient transfer of messages. To achieve this aim it manages a list of all requests that it has received and of all replies that have to be sent to other NSs together with their context information. Based on this list, the NS determines which request or reply should be send next in an existing connection with the best use of the channel. This message is then handed over to the Transfer Manager for transportation. The decision is not only based on information describing the request, like size, type, priority and requested Quality of Service (QoS) parameters, but also on information from the Transfer Manager about the state of connections and general context information obtained from the Context Manager; e.g., information about

available channels and their parameters. It depends on the intelligence of the Network Scheduler to what extend all this information is used to make the decision.

For incoming new messages the NS decides whether to start an appropriate MH that has to handle the message or to deliver the message directly to the receiver's ACoM. The latter case is true for messages that do not need a special processing; e.g., error messages, acknowledgements, or messages where their content is negligibly small or does not support special techniques (see section 2.3). Typical exchange strategies that are performed by this component are scheduling with priorities and with the influence of Quality-of-service parameters as well as data compression initiated by the NS.

The *Request/Reply Cache (Bus cache)* stores requests and replies at the local host to facilitate the immediate delivery of replies to the application when receiving an identical request later on. Since local applications also communicate via the Object Bus its cache can also be employed for acceleration of the local communication. Due to changing resources it is necessary to store also additional information about the current contexts. As a base for minimizing inconsistencies and for decisions which cached items have to be deleted serves a combination of several methods (Cate, 1992)(Chankhunthod et al., 1995). In our approach the time since the last modification as well as information about least recently accessed cached items and information about the type of the items are used for cache management functions in case that the cached items do not have timestamps that explicitly define their validation time.

On mobile end systems the Object Bus includes an *user interface* as a separate application independent component. In addition to functionality that is offered by applications and their respective ACoMs the user interface of the Object Bus provides the user with information about the current state of all active requests and pending replies and allows the interactive control of exchange parameters. The user has for example the possibility to modify request priorities, to cancel transfers, and to configure the handling of special message types.

The *Transfer Manager* carries out the lower level network functions like opening and closing of connections as well as the actual send and receive functions. In order to have the possibility to evaluate QoS parameters this component monitors the traffic of current connections. These traffic data are handed over to the Context Manager. So the other components like the Network Scheduler can make use of information about the last connections to a special host as a base for planning their work.

2.2 Context Manager

Another basic component of our abstract architecture is the *Context Manager*. The *context* is our main concept for describing all dynamic and static characteristics of the mobile and stationary entities (user, resources, information, location and time). Since the contexts represent the parameters of the mobile environment, they have an effect on all architectural components. This includes, how to locate, to access, to transfer, and to present information. Main tasks of the Context Manager include:

Context sensitive trading:

The Context Manager serves to deliver the addresses of desired information and services. This process is influenced by contexts. If no address information is locally available by the use of this technique, a request will be sent to the next Context Manager on a remote site that can in turn involve further Context Managers to obtain the required information.

Modification of requests and replies according to context criteria:
The Context Manager is in addition responsible for the adaptation of requests to the mobile environment by evaluation of contexts in order to save the mobile resources. That includes substitution of environment variables like location and time with their current values and the modification of requests, for example including information about resources (e.g. free disk capacity, display properties) and user preferences (e.g. maximum costs for a transfer, priorities for certain data types).
Management of all relevant context information:
This point includes the management of all current context parameters including statistics as well as context information from some other frequently contacted SDS and MES. Context information on the one hand will be inquired not only by the ACoM but also by all other components of the architecture. On the other hand, contexts will be inserted from all components of the architecture. Therefore, the Context Manager allows all components of the architecture to access the contexts in a specified manner. The Context Manager stores the contexts in its context database or it uses built-in functionality to inquire them directly on demand. For example, resource information with a high dynamic is always requested directly, since their storage in a database would consume too many resources for its continuos updating.

2.3 Message Handler

Message Handlers are built as processes for the optimized exchange of structured information. They can be launched by the Application Communication Manager of an application that has generated a reply as well as by the Network Scheduler that receives a suited reply. Due to their *object specific* design and their knowledge about type and structure of the data to be transferred they are able to use type and structure dependent methods for minimizing network traffic and response times. One typical group of methods are level-of-detail concepts; e.g., *Successive Refinement* and *Detail-on-demand* (see Figure 3).

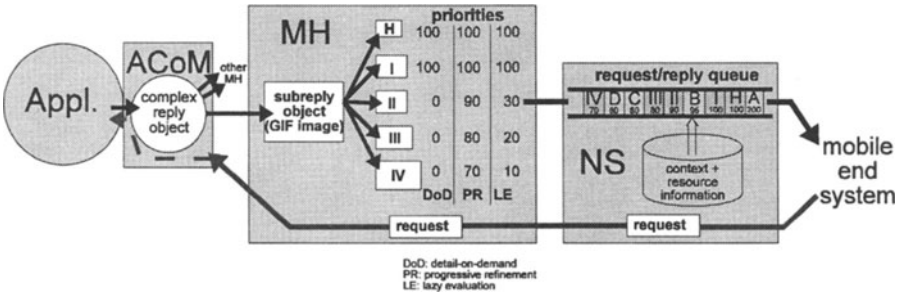

Figure 3: Level-of-detail methods: cooperation of Message Handler and Network Scheduler on a SDS.

Another method is *Data Reduction* which can be achieved by the modification of requests (see Heuer and Lubinski, 1996) and by reducing reply objects according to available resources, for example reduction of color images to the color depth of the mobile display, no transfer of objects where there are no viewers on the MES configured for, no transfer to a MES without sufficient resources, and adaptation of data to user defined constraints regarding the network connection.

2.4 Application Communication Manager

The *Application Communication Manager* is an application specific interface to the components of our architecture. It can be realized as a built-in interface in especially designed mobile applications or as a separate component that uses a defined interface of an existing application (see Figure 2). Tasks of the ACoM include the management of a list of all requests and related replies of an application together with applied contexts and the decomposition/composition of complex compound replies. The sending ACoM divides compound replies into several subreplies of certain defined types and hands them over to MHs for type specific exchange. These MHs have to be started by the ACoM. They transfer the subreplies to their counterpart at the receiving site (started by its NS) and hand them over to the receiving ACoM that will reassemble the reply object.

2.5 Message Exchange

The protocol that we use for the communication over the Object Bus is a combination of an efficient protocol for coding header information and the possibility to express the contents of messages in an application dependent format, like SQL, HTTP, MIME, or KQML. Since the content of a message should not be evaluated by the Network Scheduler all data that are necessary for planning the transfer, e.g. priority, size, and QoS demands, have to be part of the header. Our protocol allows splitting of larger messages into smaller pieces and sending them separately. The headers of these pieces are provided with offset values so that a transfer can be continued at this offset without a full retransmission after a disconnection has occurred.

3 USE OF CORBA

The acceptance of a new approach is often measured by its integration in the "real world". Due to this, the use of widely known and accepted standards and the integration of legacy systems into the new environment is crucial. We propose the Common Object Request Broker Architecture (CORBA) of the Object Management Group (OMG, 1991) as a platform for the design of object oriented, distributed systems – even in the context of mobile visualization.

We show, how this standard can be used to support resource poor mobile hosts and minimize the amount of data by optimization of the data flow. To do this we map the abstract architecture of the OBus to CORBA and point out which work is necessary to gain interoperability between standard CORBA and the modified broker architecture implementing the OBus.

3.1 CORBA Motivation

The design of software systems in general and of software for special circumstances like mobile environments in particular is a non trivial and expensive challenge. The problems of today's software engineering can only be faced by the use of modern paradigms and powerful tools that support the user by hiding great parts of the complexity one has to cover. The role of CORBA in this context is a platform for integration and distribution: objects can be used in a comfortable way independent of their location in the network, their implementation language

and platform. The inter process communication and the marshalling of parameters are done by this middleware (the CORBA implementation).

All the functionality which should be available to local or remote clients resides at objects in the server. Servers are described by the signature of their interface. This interface is specified in the Interface Definition Language (IDL). An IDL compiler maps the description to a specific implementation language and generates marshalling code for client (stub/proxy) and server (Basic Object Adapter, BOA) side.

In a mobile environment, we have to deal with resource poor mobile terminals that are not able to host applications with high memory or processing demands. These parts should reside at a stationary server and be coupled with the mobile application. The location of external services and the communication among these components is managed by CORBA. The configuration, which services to provide local or remote is specified at runtime. Further more, CORBA accelerates the development of applications by using existing OMG defined services like persistence, transaction, or security.

But CORBA in its actual state is not prepared to solve the special problems in a mobile environment. It is not only the lack of special purpose CORBA implementations but also some fundamental insufficiencies of the architecture itself: CORBA hides a lot of details and prevents the application to deal with a variety of parameters. Buffer sizes, coding of parameters, location of objects, choice of protocols, marshalling etc. are not under control of the developer. This total information hiding is inadequate for mobile computing. Also, there is no way to deal with disconnections but to completely restart the transfer. Finally, CORBA does not have a stream concept at the moment and CORBA does not support any form of QoS.

3.2 Implementation of the OBus as a modified ORB

The concept of the Object Bus can be implemented on top of a request broker core. In this section we will map the abstract OBus based architecture presented above, to the more concrete architecture proposed by the OMG.

The "normal" way to invoke CORBA servers is by a synchronous call of a method of a remote object. This call can have in and out parameters and also an optional return value. This call blocks the client until the remote function returns. In our scenario where the transfer and the generation of the result may be very slow, this synchronous mode is not acceptable. We only support oneway calls that cannot have out parameters or return values. Synchronous calls must be transformed into oneway calls by the ACoM. This transformation is necessary to facilitate the use of all the existing and standardized services with their asynchronous interface.

First we have to decide, at which level mobility support should be inserted. The application level could be used for most of the concepts but is not very fast and would not be transparent for the users. The design for a mobile application should not differ significantly from a stationary solution.

Stubs and skeletons are generated by the compiler. This code is linked with also implementation specific client or server libraries for inter process communication. Existing CORBA implementations cannot deal with priorities and disconnections. They only have the functionality of a "dumb Transfer Manager". For optimal support we have to insert the functionality of the Network Scheduler at this level: It holds a list with submitted calls, extracts the context information and manages the choice of physical network. The priority driven transfer is done by calling functions of the chosen Transfer Manager. Figure 4 gives a sketch of the architecture with the modified/additional components. As we can see there, the broker also

must be adapted to make best use of the introduced features. The broker now includes the functionality of the Context Manager and uses dynamic as well as static context information to bind to an appropriate server. It also supports the negotiation of QoS parameters for stream connections between client and server.

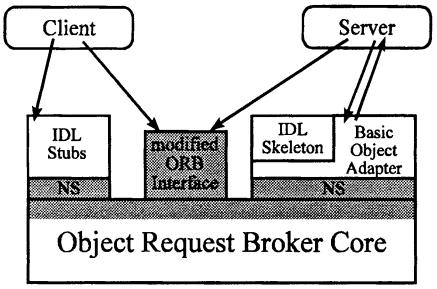

Figure 4: Modifications of the ORB for mobile applications

One concept that is not available in CORBA today is the concept of streams. They are not only useful for multimedia applications with realtime demands but also to implement successive refinement concepts. With streams we can think of an image transmission as a stream from the source (e.g. data base at SDS) to the sink (application at mobile terminal). This stream can be interrupted if the necessary level of detail is reached. But this is only possible when streams are introduced beside attributes and operations, as a new feature in CORBA. A stream is characterized by a frame type, a direction, some QoS characteristics, source and sink callbacks.

The modified communication bus has an additional interface. This is used for query and manipulation functionality at request level. The application or a special control application for the OBus can use it to facilitate a higher level of controllability for the user.

3.3 Access to standard CORBA services

Interoperability of the modified broker and standard CORBA is reached by two half bridges which translate their format to the standardized UNO protocol defined by CORBA 2.0 (OMG, 1995).

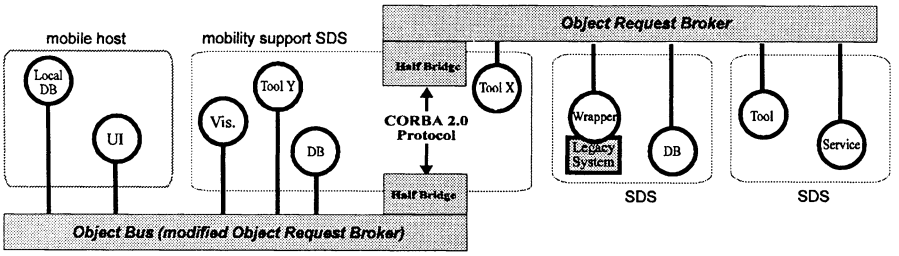

Figure 5: Bridging between modified and standard ORB

Using this approach we lose performance, but win flexibility and lots of existing and future services using CORBA. The optimized communication can only be gained for objects on the mobile host, contacting services at the mobility support SDS. Remote calls over the bridge are limited to standard features. The resulting structure with the mobile host and several stationary servers connected, is presented in Figure 5.

4 MOBILE INFORMATION SYSTEM

The presented Object Bus architecture and a selection of the exchange strategies are used to build a first prototype of a mobile information system (MIS). This MIS is intended to serve as integrating framework to provide a user with necessary information while he/she is mobile. First data types that are supported by Message Handlers include textual information and GIF images. They are used when a mobile user makes a query to a remote database to get the brand-new weather information regarding his/her current location. That means that database queries are modified to include the current location as an additional selection criterion. Information about the resources of the mobile end system and user preferences are used to achieve a further reduction of data; e.g., the user can specify that images should match the display capabilities of the mobile system.

The application consists of several modules: the user interface, a huge database holding actual weather data and visualization tools that produce still images. All these modules are implemented as CORBA clients and servers that communicate via the OBus. The complete database is stored at the stationary mobility support server at the office or can be accessed via this server. Due to the poor resources of the mobile terminal, we split the application: Only the user interface and the main control reside on the terminal. The computing intensive visualization and the data are located at the workstation. A typical request would be generated by the user, who is asking for a forecast for the actual location six hours later. The ACoM of the database decomposes the compound reply into several subreplies of individual types; e.g., textual descriptions and satellite images. The subreplies are handed over to Message Handlers. There the image and the text objects are treated separately for optimization issues; for example, the image is reduced in resolution and color depth adequate for the display capabilities of the notebook. At the client side, the objects are decompressed if necessary and are handed over to the UI to present it to the user. In case of an interlaced GIF image, streams can be used to implement a successive refinement that can be interrupted if the level of detail is acceptable.

5 RELATED WORK

There are some other projects that focus on some problems that arise, when mobile end systems access globally distributed information on stationary servers. Mobisaic (Voelker and Bershad, 1994) is an information system for a mobile wireless computing environment, which provides users with documents that are dependent on dynamic environment variables such as location. Mowser (Joshi et al., 1995), a smart Web browsing application, performs transactions based on the user's available resources like display resolution and sound capabilities. Other projects try to find solutions for context-aware applications (Adams et al., 1994) (Satyanarayanan et al., 1994) and for the message exchange with mobile hosts; e.g., the agent-mediated message passing (Athan and Duchamp, 1993).

Two groups also use standardized distribution platforms in their projetcs. The MOST project (Friday et al., 1996) relies on ANSAware. A group at the Dresden University of Technology use OSF DCE as the platform for distribution (Schill et al., 1995). A research group at BBN deals with Quality of Service for object communication in a more general manner (Zinky et al., 1995).

6 SUMMARY

We proposed an architecture, where applications transparently exchange objects via the Object Bus. It enables the development of mobility aware applications with state of the art support for disconnections and bandwidth minimization. Message Handler processes and a Network Scheduler, the main components of this architecture, were described as well as the methods that they perform to make the best use of slow and unreliable communication channels when acting in mobile environments. These methods include level-of-detail techniques, data reduction, caching, data compression and scheduling with priorities and influence of Quality-of-Service parameters. The given abstract architecture was finally mapped to OMG CORBA. The architecture and a selection of the exchange strategies were used to build a first prototype of a mobile information system.

The following table sums up the presented components of the Object Bus, the objects/data types treated by them with typical operations and gives the equivalent component of the CORBA mapping.

Table 1: Logical components of the architecture and their CORBA equivalents

component	operating on	typical operations	CORBA equivalent
ACoM	request, reply (complex objects)	decomposition, reassembly	application level
Context Manager	request, context data	address resolving, enrich request	broker, trader
NS	request/ reply messages	compression, fragmentation, QoS-handling	core
Message Handler	homog. data (simple objects)	disconnection-handling, reduce color, drop frames	BOA + stub (proxy)
TM	package	send, multiplex	BOA + stub (proxy)

7 ACKNOWLEDGEMENT

We would like to thank Thomas Kirste and Astrid Lubinski for fruitful and valuable discussions regarding the presented object bus architecture and exchange strategies. Additionally, we would like to thank Steffen Nowacki for cooperation regarding the integration with CORBA.

The described work is part of the workpackages "Exchange strategies" and "Distributed functionality" of the MoVi project funded by the German Research Association (DFG).

8 REFERENCES

Adams, N.; Gold, R.; Schilit,B.; Tso,M.; Want, R.(1994) An Infrared Network for Mobile Computers. *Proc. of the 1st Usenix Symposium on Mobile & Location Independent Computing*, August 1994.

Athan,A.; Duchamp,D.(1993) Agent-Mediated Message Passing for Constraint Environments *Proc. Mobile and location-Independent Computing, USENIX,* Cambridge, August 1993.

Cate, V.: Alex - a Global Filesystem. *Proceedings of the USENIX File Systems Workshop, pp. 1-11, May 1992.* `ftp://alex.sp.cs.cmu.edu/doc/intro.ps`

Chankhunthod,A., Danzig,P.B., Neerdaels,C., Schwartz,M.F., and Worrell,K.F.(1995) A Hierarchical Internet Object Cache. *University of Southern California, University of Colorado*, Boulder, 1995.

Forman, G.H. and Zahorjan, J.(1994) The Challenges of Mobile Computing. *IEEE Computer*, 27(4), April 1994.

Friday, A.J., Blair, G.S., Cheverst, K.W.J., Davies,N. (1996) Extensions to ANSAware for advanced mobile applications. *Proceedings of the IFIP/IEEE International Conference on Distributed Platforms,* Dresden, 1996.

Heuer, A. and Lubinski, A. (1996) Mobile Information Access - Challenges and possible Solutions. *Workshop "Information Visualization and Mobile Compuing" (IMC '96),* Rostock, February 1996.

Heuer, A., Kehrer, B., Kirste, T., Schumann, H., and Urban, B. (1995) Concepts for Mobile Information Visualization - The MoVi-Project. *Proc. Eurographics Workshop on Scientific Visualization 1995,* May 1995.

Imielinski, T. and Badrinath, B.R (1994) Mobile wireless computing. *Communication of the ACM*, Oct. 1994, Vol.37, No.10.

Joshi,A., Weerasinghe,R., McDermott,S.P., Tan, B.K., Bernhardt,G., and Weernawarna,S. (1995) Mowser: Mobile Platforms and Web Browsers. *Purdue University, 1995.*

Object Management Group (1995) CORBA 2.0/Interoperability - Universal Networked Objects. *OMG TC Document 95.3,* March 20, 1995.

Object Management Group (1991) The Common Object Request Broker: Architecture and Specification. *OMG TC Document 91.12.1 Revision 1.1,* 1991.

Satyanarayanan, M., Noble, B., Kumar, P., Price, M. (1995) Application-aware adaption for mobile computing. *Operating Systems Review*, 29(1) January 1995.

Schill,A., Bellmann,B., Böhmak,W., Kümmel,S. (1995) System Support for Mobile Distributed Applications. *IEEE Workshop on Services in Distributed and Networked Environments (SDNE),* Vancouver, June 1995.

Voelker, G.M., Bershad, B.N. (1994) Mobisaic: An Information System for a Mobile Wireless Computing Environment. *University of Washington*, Sept. 1994.

Weiser,M. (1993) Some computer science issues in ubiquitous computing. *Communications of the ACM,* 36(7), July 1993.

Zinky,J, Bakken, D.E., Schantz, R. (1995) Overview of Quality of Service for Objects. *Proceedings of the Fifth IEEE Dual Use Conference* , May 1995.

9 BIOGRAPHY

Jörg Bönigk studied computer science at University of Rostock. 1993 he received his diploma and then became a member of the scientific staff of the Computer Graphics Center Rostock. His research topics are scientific visualization and mobile computing. He is currently involved in the project MoVi (Mobile Visualization).

Uwe von Lukas graduated in computer science from Darmstadt University of. He became a research assistant at Fraunhofer Institute for Computer Graphics in 1994 and is now a member of the scientific staff of the Computer Graphics Center Rostock. His research interests are in the area of distributed platforms, CSCW and the application fields of CAD.