

An infrastructure for Mobile Information Systems based on a Fragmented Object Model

Thomas Kirste

Computer Graphics Center
Wilhelminenstr. 7
D 64283 Darmstadt, Germany
email: kirste@igd.fhg.de

January 26, 1995

Abstract

Global information management systems based on the vision of the “docuverse” – such as the World-Wide Web (WWW) – show that on-line access to vast amounts of distributed information is possible not only for the expert, but also for the end user. At the same time, the availability of affordable mid- and long-range wireless data communication services, based, *e.g.*, on cellular phone technology, has put the vision of ubiquitous mobile information access within the reach of viable projects. It is now an interesting challenge to *combine* both concepts into a system model granting everyone ubiquitous access to the global information repository. However, problems such as low bandwidth and limited resources make this a non-trivial task.

This paper describes MIS/O, an experimental mobile information system based on wireless data communication which has been developed at the Computer Graphics Center during 1994. MIS/O and its underlying concepts address the above mentioned problem areas by introducing a *fragmented object model*. This model provides the necessary mechanisms for a (partial) migration of object behavior to the mobile system – acting as an intelligent data terminal – at runtime. In addition to conventional caching and compression techniques, this is required for coping with intrinsic system limitations and low communication bandwidth in an environment with a virtually unlimited number of information types and services such as the docuverse.

Keywords. Mobile Computing, Global Information Systems, Distributed Systems, Object-oriented Programming, Visualization.

1 Introduction

When T. Nelson coined the term “docuverse”, he envisioned a global hypertext-like structure, containing and interlinking the entire human knowledge (cf. [19]). Global information management systems based on this vision – such as the World-Wide Web (WWW) [1] – show that on-line access to vast amounts of distributed information is possible not only for the expert, but also for the end user. The ultimate goal of these activities is the construction of a unified, globally accessible repository for arbitrary information and services¹ which is available to everyone. (The term “docuverse” will be used to denote this repository).

On the other hand, the availability of affordable mid- and long-range wireless data communication services, based, *e.g.*, on cellular phone technology, has put the vision of ubiquitous information access for all users within the reach of not only technically feasible but also commercially viable projects. Mobile information

¹Such as the WWW Pizza ordering service available for Santa Cruz residents (URL: <http://www.pizzahut.com>).

systems (MIS) are envisioned, consisting of a portable computing device – such as (sub)notebook, palmtop, or PDA-like machinery – which is equipped with suitable wireless data communication hardware for accessing information from various sources. A number of vertical and horizontal applications already exist for such systems or begin to emerge, *e.g.* traffic guidance resp. mail-enabled applications (for a general discussion cf. [8, 21]).

It is now an interesting challenge to *combine* both concepts into a system model granting any user ubiquitous access to the global information repository. The idea is to use portable computing devices as mobile “windows” into the docuverse. These *mobile data terminals* (MDT) use wireless data communication to access *stationary data servers* (SDS), which implement the docuverse.

A simple straightforward realization of this scenario is inhibited by the following contradictory properties of the different architectural components:

1. Limitations of the communication services such as low bandwidth and temporary disconnection.
2. Limited local resources (in terms of storage capacity, computing power, data entry and data output capabilities).
3. Complexity of the docuverse (in terms of available data formats and display/interaction requirements).

Although many more problem areas for mobile information systems can be identified besides Pts. 1 and 2 (cf. *e.g.* [5, 6, 8, 23, 30]), the issues identified above are the the most obvious and pressing ones when trying to build a MIS.

This paper describes MIS/O, an experimental mobile information system based on wireless data communication which has been developed at the Computer Graphics Center (ZGDV) during 1994. MIS/O and its underlying concepts address the above mentioned problem areas by introducing a *fragmented object model*. This model provides the necessary mechanisms for a (partial) migration of object behavior to the mobile system – acting as intelligent data terminal – at runtime. In addition to conventional caching and compression techniques, this is required for coping with intrinsic system limitations and low communication bandwidth in an environment with a virtually unlimited number of information types and services such as the docuverse.

The fragmented object approach is based on a *remote programming* facility, which can be interpreted as a technique for the *implementation of self describing data*.

In addition, an example docuverse built on MIS/O provides a metaphor-based user-interface, creating an intuitively usable system for non-expert users. The implementation of the metaphor is based on the system’s object migration facility and thus – from the model’s point-of-view – a specific *application* within the docuverse.

The remainder of this paper is organized as follows:

Section 2 gives an in-depth discussion of the application scenario (“A window to the docuverse”) together with its pertinent problems.

Section 3 introduces the central solution proposal of this paper – a fragmented object model implemented using remote programming – and embeds it into a system model.

Section 4 gives an overview of the MIS/O system implementation. The example docuverse created for MIS/O is outlined in Sec. 5.

Finally, a summary of the work described in this paper and an indication of further work is given in Sec. 6.

2 A window to the docuverse

2.1 The docuverse

Based on the scenario described in the previous section, the following assumptions hold for the docuverse:

1. The docuverse contains not only multimedia documents, but also services which may actively communicate with the user or other services and might even *change* the docuverse. (*E.g.*, a subscription service adding a user to an object representing the list of subscribers.)
2. The set of object and service types present in the docuverse is constantly changing and expanding.

Taking point 1 to its extreme, the docuverse appears as a set of uniquely identifiable *active* objects, which communicate through mechanisms such as messaging. In conjunction with 2, the notion of the docuverse thus implicitly contains a dynamically extensible class system which allows the definition of an object's message response behavior.

This concept informally describes the notion of a docuverse as it is understood in the course of this paper. It is this dynamic set of heterogeneous, communicating active objects, which ultimately will be accessed by a MDT (and, of course, from stationary terminals too).

2.2 The task of a MDT

On the abstract level, the functionality required for the MDT is easy to describe: it must be able to receive data from the docuverse, it has to display this data in an appropriate way so the user is able to understand its semantics, and it has to monitor user interaction, which eventually is translated into requests for the docuverse.

The central points of interest are the information objects in the docuverse, not the tools ("Browsers", "Clients") required on the MDT for accessing these objects (such as the various Mosaic-executables in the case of the WWW). It therefore makes sense to adopt a data-centric view and regard data visualization and interaction management as aspects of an object's behavior. Any object in the docuverse – regardless, whether it represents a simple static data item or a complex service – may therefore conceptually define its own specific user interface consisting of visualization (output) and interaction (input) behavior. These specific aspects will be called the object's *interface behavior*. The task of a MDT is then to realize this interface behavior.

2.3 Realizing interface behavior

The main difficulty in accessing the docuverse is the heterogeneity of the visualization and interaction requirements imposed by the different objects it contains. The docuverse does not simply consist of one or two selected information services with well defined interface behaviors: It contains the complete information universe with the full bandwidth of today's (and tomorrow's) structured and multimedia information. Therefore, every access tool must be prepared to realize any interface behavior, to satisfy any visualization and interaction requirement requested by an object present in the universe. An even greater challenge is the docuverse's dynamic nature with respect to the interface behaviors it contains (cf. Sec. 2.1): At any time, new objects with an up-to-now unknown interface behavior may be created – *e.g.*, by the introduction of a new service or visualization mechanism. So any access tool needs to be prepared to cope with unknown visualization and interaction requirements.

This results in the following – not really surprising – observation:

The number of visualization and interaction mechanisms required for accessing data and services in the docuverse is not fixed in advance and may be substantial (cf. Sec. 1, Pt. 3).

On the other hand, when looking at the idea of mobile windows to the docuverse, one has to cope with the limitations on the MDT side identified in Sec. 1 (Pt. 1 & 2). They inhibit the following simple solutions:

Dumb Server: One alternative possible for high-bandwidth networks is to define a front-end server (a la X [25]), which provides the necessary primitives for realizing an object's interface behavior. The behavior itself is realized by the object at an SDS site, where it can exploit all resources of a stationary computer. Only primitive requests are sent to the MDT. Within the expressive power of the front-end server, any possible interface behavior could thus be realized by a MDT, without exceeding the available resources.

However, since only primitive requests or event notifications are transmitted between MDT and SDS, a substantial amount of bandwidth is required in order to reach acceptable response times for interactive applications. Anyone who has tried to run the X protocol over a slow serial line has experienced this (cf. *e.g.* [9]). Furthermore, in the case of temporary disconnection (= very low bandwidth), interactivity is not given at all.

Therefore, this approach is generally not feasible considering the properties of wireless data communication. The MDT must be able to realize object behavior with more autonomy from the SDS².

Downloading: Using this approach, an object's complete interface behavior is migrated to the MDT as program executable on the MDT's operating system. (For today's global information systems, this is typically done manually via `ftp`.) This strategy clearly avoids the disadvantage of the high communication bandwidth required for the dumb server scenario and guarantees very high autonomy. But here too, severe disadvantages prohibit this approach:

- If heterogeneous MDTs are used (which is to be expected), a separate executable version of the interface behavior for every possible MDT needs to be available.
- Because MDT storage capacity is limited, the number of executables to be downloaded at the same time is limited (note that even in the age of shared libraries, executable programs are quite substantial in size). Assuming a large number of different object types (*i.e.*, different object behaviors) in the docuverse, this requires a caching scheme with preemption. This, however, causes problems in the case of a cache miss: because of the large size of the executables, transfer time will be at least several minutes, making information access unpleasant at best. Exploratory information access – which may cause a large number of executables to be loaded – will be avoided by the user.

Clearly, for a mobile window to the docuverse a concept is required which provides a kind of downloading of interface behavior, while at the same avoiding the problems of heterogeneity and size.

2.4 Requirements for an object model

Resuming the discussion in this section, an object model (OM) for the docuverse is required which provides the following characteristics:

Universality: The OM shall enable the MDT to cope with any object behavior that may exist in the docuverse.

Expressiveness: The OM shall allow the description of a rich set of different object behaviors.

Autonomy: The OM shall enable the MDT to autonomously realize object behavior, so that temporary disconnection and low bandwidth can be handled.

Parsimony: The OM shall allow a realization of Universality, Expressiveness and Autonomy without exceeding available MDT resources.

In the next section, a short sketch of an object model fulfilling these requirements is given.

²Because sending data from MDT to SDS via wireless is a power consuming job, the limited battery capacity of mobile systems is another reason for requiring more autonomy of the MDT.

3 Aspects of an object model for the docuverse

3.1 Location transparency and object fragmentation

The docuverse has been introduced as an abstract pool of objects. From this abstract view, no concrete hints for mapping objects to computing devices such as SDSs or MDTs should be inferred. It is likewise clear that only a distributed implementation platform yields a realistic opportunity for creating an effectively usable docuverse. The next question is then how to map objects to the available server sites.

Ideally, the mapping of docuverse objects to the available computing machinery should be transparent to any user, be it information consumer or service provider.

Once this *location transparency* is given, the opportunity to change the docuverse-mapping at runtime exists. This includes aspects such as migration and replication of objects. Even more interesting is the chance of *object fragmentation*: parts of an object may reside on different machines, and objects may be migrated or replicated *partially*. By distributing one object across different machines, one can exploit the fact that for certain operations the amount of communication between object parts may be much lower than between a given part and services located at specific computers³.

Typical examples are interface behavior and valuation (the process of determining the current object state) of an object. The interface behavior uses the MDT functionality to communicate with the user, while the valuation may require powerful stationary computing machinery to give results within acceptable time constraints (*e.g.*, gas distribution simulation for on-site planning of evacuation routes in mobile disaster management applications for the chemical industry). Fig. 1 sketches such a situation and indicates the various abstraction levels involved.

The fragmented object model (FO-model) supports partial migration by exploiting location transparency. The next question to solve is: how can location transparency for active objects be provided in distributed architectures?

3.2 Representing self-describing objects

As far as the current definition of the docuverse is concerned, an object is completely described by its messaging behavior. Thus objects in a docuverse could mathematically be simply defined as functions mapping streams of requests to streams of replies⁴. Given suitable alphabets for requests and replies, this abstract definition is powerful enough to model any possible object behavior.

As long as an object resides on a specific server site, the computing machinery available there is responsible for implementing the object's messaging behavior. But as soon as the mapping from objects to server sites can be changed dynamically – by replication or migration – an object has to be moved from one site to the other. Because real communication channels can only transport bit streams and not abstract message behavior functions (which describe the object), these functions have to be converted into a bit-stream representation which the receiving site then has to interpret to recreate the messaging behavior, *i.e.*, the object. Therefore, a well-defined *representation language* (RL) is required for active objects (as well as for static data such as lists etc.), if non-trivial distribution strategies are to be considered.

Of course, such topics are investigated in the research on distributed operating systems, where location transparency for active objects (read: processes) is a fundamental goal. The interesting results for the docuverse are:

³Refer to [18] for an application of object fragmentation to the realization of group models in distributed operating systems.

⁴This is identical to the definition of a process in functional programming languages using the model of stream-based I/O [7]. (In Haskell, the corresponding type is written as: `type Process = [Request]->[Reply]`.)

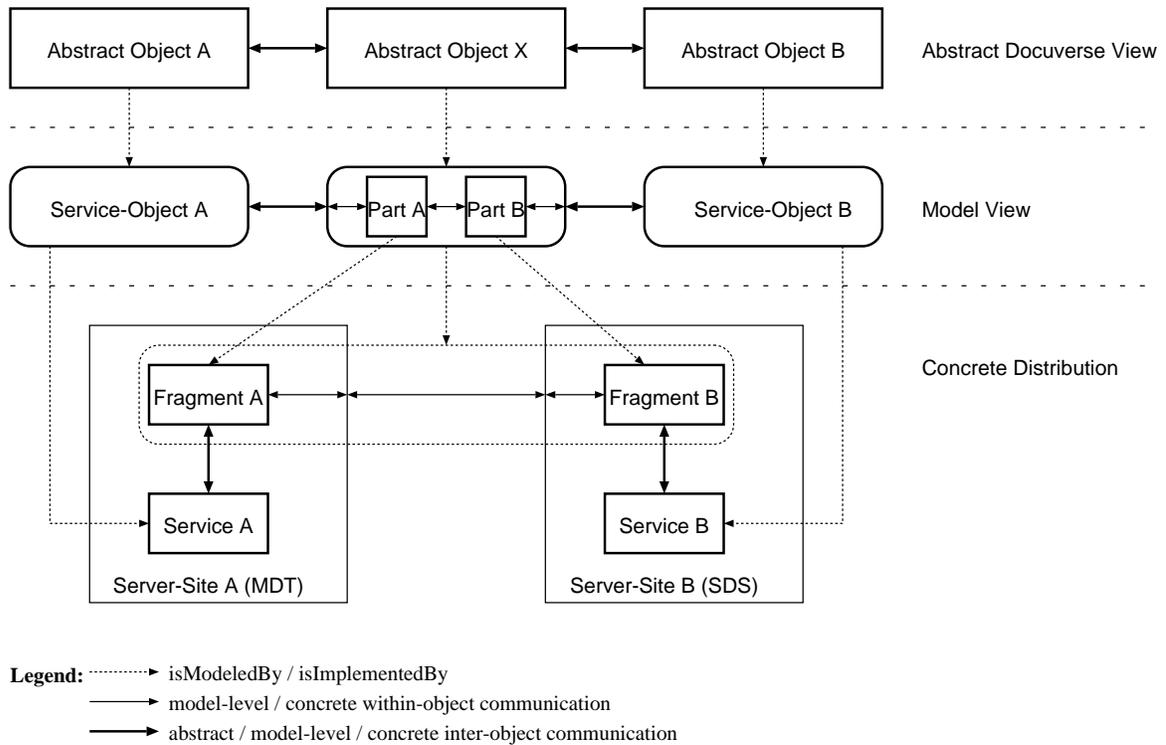


Figure 1: An object fragmentation example

- Implementing the docuverse is similar to constructing a distributed operating system.
- As the docuverse will encompass a wide variety of different servers, this operating system will have to make different host architectures and host operating systems transparent to the “docuverse object code”, the RL.

Another conclusion can be drawn for the RL: Because all kinds of information services will probably be within the docuverse, potentially any imaginable computation might be eventually performed by an object in the docuverse. This means the representation language of objects must be able to represent arbitrary computations. It needs to be *computational complete*.

In addition to computational completeness, the language must provide a complete set of expressions for requesting primitive input and output operations in order to describe the interface behavior of an object and the communication with other objects. (This functionality may also be called *resource completeness*. It can be modeled by sending appropriate messages to a globally and always accessible object representing the whole docuverse and its facilities. As this object will never have to move because it is everywhere, it needs no external representation.)

Because an external object representation does not only contain state variables, but also the object’s complete behavior description, there is no additional information required for using the object and its methods. Therefore, such objects can be called *self-describing* (cf. [12]). Once computational and resource completeness can be granted, any possible information object or information service has a RL representation and can thus be migrated and fragmented across SDSs and MDTs.

3.3 Remote Programming

Once the RL is computationally complete, it is more or less equivalent to any other programming language. So the description of object behavior can be regarded as a *program*. If programs are moved from one computer to another, where they are automatically executed, one can speak of *remote programming*. So remote programming can be thought of as a technique for implementing object migration in heterogeneous environments.

Languages such as Telescript [31], applications like Enabled Mail [2] (based on the language TCL), and the concepts in [29] are based on the same idea⁵. However, the point of this paper is not the idea of moving programs between heterogeneous computers (= implementation level), but the definition of a generally applicable object model for realizing ubiquitous, mobile access to the docuverse, which is based on partial object migration.

Note especially, that the idea of mobile agents (= migratable processes) described in [31] is just *one* way of exploiting the general idea of location transparency. The FO-model outlined in this section is another way (which contains unfragmented migration as special case).

3.4 Meeting the requirements

By providing a fragmented object model with complete external representation, the requirements of Sec. 2.4 can be met:

- Universality is granted by the well-defined external representation.
- Expressiveness is given through resource and computational completeness.
- Autonomy is available through (partial) object migration to the MDT.
- Parsimony is available through migration and fragmentation (only the currently required aspects of object behavior need to be available on the MDT).

It can not be denied that the description of the object model as given in this section is rather abstract and conceptual. However, it clearly identifies the relevant solution concepts, allowing a “straightforward” in-depth investigation of concrete definition possibilities.

In the next section, the MIS/O is described, which realizes an experimental mobile information access to the docuverse based on a simple implementation of a the fragmented object model.

4 The MIS/O

4.1 Overview

MIS/O is an experimental prototype of a MIS, based on the concepts discussed in Sec. 2 and 3. In addition to creating a platform allowing an evaluation of the viability of these concepts, the following objectives have been pursued by the MIS/O implementation:

- Enable the demonstration of the potential of ubiquitous mobile information access to expert and non-expert users (information consumers as well as information providers).
- Demonstrate the “docuverse” concept of a unified global information and service management and the idea of a mobile windows to the docuverse.

⁵The Datex-J extension “KIT” (Window-based Kernel for Intelligent Communication Terminals) [4] too contains some aspects of remote programming

For this purpose, an example “micro” docuverse (DV/O) has been created for MIS/O, demonstrating the following facilities:

- Access of private data.
- Access to public data (a simple electronic newspaper).
- Transparent migration of specialized object behavior.

The initial user-interface to this DV/O is created by the interface behavior of an object in the docuverse itself. It is highly interactive and metaphor-based (therefore more or less “intuitive”, aiming at non-expert users), demonstrating the wide range of interface behavior available for objects in the docuverse.

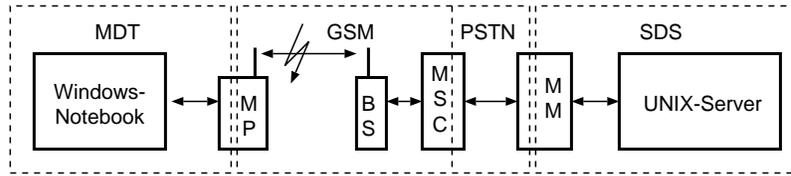
A more detailed discussion of this application is given in Sec. 5.

4.2 Fundamental design concepts for MIS/O

4.2.1 Hardware and communication

MIS/O consists of multiple MDTs (Notebooks running MS-Windows) and a single SDS (a stationary UNIX-Server). The Notebooks are equipped with GSM (= Global System for Mobile communications) mobile phones for wireless data communication; they use the 9.6 kBaud data transmission service provided by the GSM standard. The SDS is equipped with a standard analog PSTN (Public Switched Telephone Network) modem. The transition from GSM to PSTN is the responsibility of the GSM service provider. (Cf. Fig. 2.)

High-level communication between SDS and MDT is based on stream sockets relying on SLIP (Serial Line Internet Protocol) and TCP. This reduces available net bandwidth, but substantially simplifies the implementation of data communication – e.g. through providing multiple logical channels across one physical channel, access within the complete Internet, and error recovery.



- Legend:**
- MP = GSM Mobile Phone with data transmission capability.
 - BS = GSM Base Station.
 - MSC = GSM Mobile Switching Center with signal conversion to analog PSTN modem transmission.
 - MM = Stationary analog PSTN modem.

Figure 2: Communication architecture of MIS/O

4.2.2 Object representation and object migration

Within MIS/O, the language HCL is used as RL for representing object behavior. HCL⁶ [11] is a LISP-like dialect with small-footprint interpreters available for UNIX and Windows. It contains a run-time extensible object system (HCLOS), socket-based communication mechanisms, and functions for creating graphical user-interfaces (using X/Motif resp. MS-Windows). So, resource and computational completeness for object representations can be granted.

Because HCL (and LISP in general) has the capability to create an external representation of (almost) anything which may be denoted by a HCL term, it is well suited for serving as RL for the migration of active

⁶HCL is an acronym for “HyperPicture Command Language”; it has initially been developed for the description of active objects in the computational hypermedia system “HyperPicture” [16, 10].

objects. In MIS/0, the behavior of an object is defined by a HCLOS class, so that migration of object behavior can be implemented by migrating classes. The latter is supported by the dynamic extensibility of HCLOS and HCL's ability to create external class- and method-representations from classes and methods.

Both SDS and MDT employ an HCL-Interpreter to realize object behavior. With respect to behavior definitions, the HCL running on the SDS operates as master, containing all known object behaviors (*i.e.*, classes and methods). If the slave HCL on the MDT encounters objects with unknown behavior, it requests the necessary classes and methods from the master and augments its local object system.

MIS/0 provides a rather simple implementation of the FO-model (built on top of HCLOS) by allowing a HCLOS-method to be marked by the implementor as "migratable" or "non-migratable". At class migration, non-migratable methods are replaced on the MDT by stub-methods, which use communication functionality to invoke the real method functionality on the SDS⁷. An example for a necessarily non-migratable method is the function for fetching the content of an object from the SDS. (However, getting the content from the MDT cache *is* migratable. Cf. Sec. 4.4.1.)

4.3 The data model

Except for fragmentation and migration, the data model provided by MIS/0 is a simple persistent object model with the usual properties. Objects persist across sessions; they have a system-wide unique identification; their internal structure and behavior is determined by a hierarchical class system (provided by HCLOS). The following particularities are worth noting:

Identification: The object identifier scheme used by MIS/0 allows to compute the name of an object's class directly from its identifier. This is necessary because an object's class must be known by the MDT before its MDT-local parts can be created and accessed⁸.

Content: Many objects in the docuverse represent classical documents (text, image, graphics, . . .), with the content conceptually being stored in the SDS file system. However, because in MIS/0 content access – as every object access – is performed via method invocation⁹, arbitrary computations may be performed for a *dynamic* content generation. (A simple example is to use the standard output of a UNIX-command such as `rwho` as content of a text-object.)

Abstracts: In order to support a simple detail-on-demand feature, objects may define text and image abstracts. This enables information browsing by allowing the user of the MDT to get a quick overview of the object's content.

Composites: Simple composite objects allow to impose structure on a MIS/0 docuverse. The content of a composite object is a list of object identifiers, so that composites can be used for creating directory-like hierarchical structures, or for representing descriptors such as keywords.

4.4 System functionality outline

Overall system operation and architecture adheres to the concepts outlined in Sec. 3. However, several components exist in the MIS/0 implementation of MDT and SDS, which provide further facilities for optimizing and simplifying user access and data communication. Fig. 3 displays the system architecture, whose functionality will be outlined below.

⁷Automatic stub generation for methods is very simple in HCL. Assuming a function `remote-eval`, which is able to evaluate a term on the master HCL, a function for creating a stub definition for any given class and method selector can be defined as follows, using LISP's backquote:

```
(defun stubbify (class sel)
```

```
  `(defmethod (,class ,sel &rest args)
    (remote-eval `(send self ,,sel ,@args))))
```

⁸The other alternative would have been a costly inquiry at the SDS, which clearly should be avoided in low-bandwidth scenarios.

⁹The usual practice in OO programming.

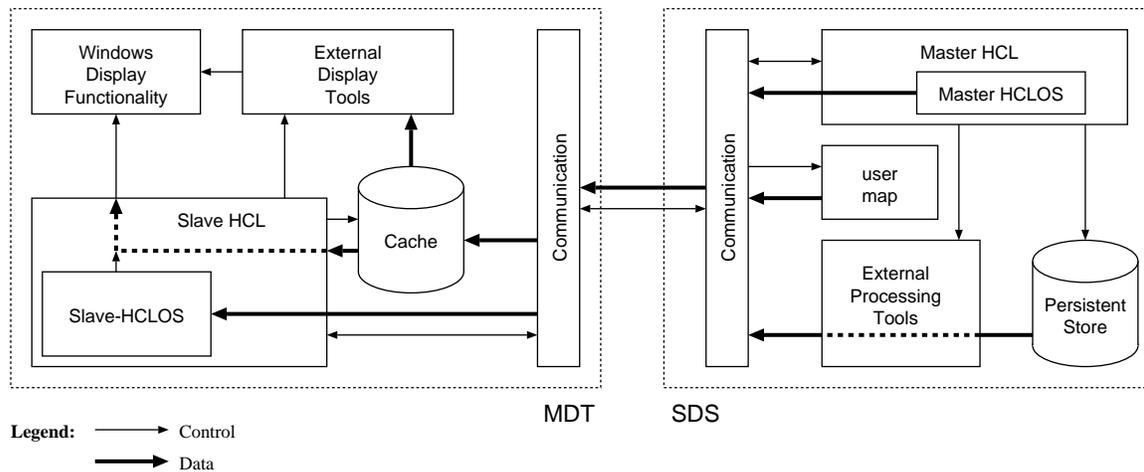


Figure 3: The system architecture of MIS/O

4.4.1 MDT functionality

The main component of the MDT is the slave HCL responsible for realizing migrated object behavior. This slave HCL provides functions for creating user interfaces based on MS-Windows. Furthermore, it can spawn external processes in the MS-Windows environment for creating the interface behavior of several standard object classes (e.g., WinMosaic or WinWord).

The MDT also provides a cache for storing object data (such as abstracts or content). This cache persists across sessions and is conceptually comparable (although functionally inferior) to caching systems for disconnected operations such as proposed in [24].

The MDT's slave HCLOS instance uses a two-stage load-on-demand mechanism for migrating object behavior: In the case that the slave HCLOS encounters an object with a yet unknown class with name c , it first consults the SDS's master HCLOS for the names of all of its superclasses, $sc(c)$. After removing all names of classes already known by the slave HCLOS from $sc(c)$, the remaining unknown classes and c are requested from the master HCLOS. The first access of an object with unknown class therefore requires three protocol transactions: two for fetching the interface behavior and one for getting the object component. Any further access can then be handled locally.

4.4.2 SDS functionality

The main component of the SDS is the master HCL, containing all known object classes of the docuverse in its HCLOS instance. It furthermore contains a persistent storage mechanism based on the file system for storing the bulk data of objects (such as image data for image objects etc.). For the dynamic computation of object contents, the master HCL's capabilities for process control and inter-process communication in the UNIX environment are used.

In addition to managing the classes and objects known to the docuverse, the SDS provides the "user map", which maps user-identities to docuverse objects. These objects are the entry-points into the docuverse for the respective users. The interface behavior of such an entry point determines the (initial) facilities available to the user for accessing the docuverse. The user map is consulted during the process of connecting the MDT to the SDS.

For DV/0, one entry-point object has been implemented which creates the metaphor-based user-interface to DV/0.

4.4.3 The MIS/0-Protocol

For communication between MDT and SDS in MIS/0, the following protocol primitives have been defined:

LogOn(u): Registers the MDT with user u at the SDS and returns the entry-point object for u .

ClassChain(c): Gives the list of all names of all superclasses of the class named c , $sc(c)$.

ClassDefns(c₁, ..., c_n): Fetches the classes corresponding to the class names given as argument. (This and the previous protocol element are used for implementing object behavior migration.)

GetComp(o, p₁, ..., p_n): Fetches the the components p_1, \dots, p_n for the object with the identifier o . By explicitly providing this protocol primitive, the MDT's low level communication is able to detect bulk data transfers. This allows some optimization such as shunting bulk data bytes directly into the cache without traveling through the slave HCL.

RemoteEval(t): Evaluates the HCL-term t on the SDS and returns the evaluation result. This protocol element is used by method stubs residing on the MDT for invoking the operations on the SDS (cf. Footnote 7).

5 The DV/0

The application scenario for DV/0 has been the "traveling manager" who needs an integrated on-line access to private and public data and services. As private data, a variety of standard document types for the Windows environment are supported. Public data is represented by a (simple) electronic newspaper written in HTML (HyperText Markup Language). The storage of private data in the docuverse is motivated by the idea of commercial services providing safe storage of private data (just as a bank storing the customers money is safer than keeping the same money under one's private mattress).

5.1 Supported object classes

Based on local viewers, the content types Image (GIF), Text (ASCII), Document (Word) and Hypertext (HTML) are currently used in DV/0¹⁰. There is also the class "Executable", whose instances have contents which are programs executable in the MS-Windows environment.

Because object classes may use the contents of Executable objects as local viewers, the "downloading" approach described in Sec. 2.3 is fully supported as "side effect". Despite its disadvantages, a careful employment of the downloading approach is very useful for making standard high-performance viewers available at the user's MDT (maybe using the temporary availability of high-speed communication services such as an ISDN-line during night at home).

Besides the classes using external viewers, the "Internal" class (and its subclasses) contain objects whose interface behavior is completely described by HCL. Currently, three kinds of objects based on this class are accessible in DV/0:

- The entry-point object.
- A simple on-line database residing on the SDS (using a form-based interface).
- A subscription service for electronic newspapers, which performs an on-line modification of the user's entry-point object by adding an access method for the subscribed newspaper.

¹⁰However, adding such classes, even dynamically from the MDT, is extremely simple – e.g.: `(remote-eval '(def-pview ExcelDoc "c:/excel/excel" "xls"))`.

5.2 A metaphor for accessing the docuverse: The entry-point

The entry-point to DV/O uses the concept of *metaphors*, analogies to real-world objects, for presenting an intuitively usable interface to DV/O and its services. (For a general discussion of metaphor-based user-interface construction see *e.g.*[27].) As an in-depth discussion of this metaphor is out of the scope of this paper (see [28] for a complete discussion), only a short overview is given here.

The entry-point object presents itself as two-story “shopping & service center” (the entrance hall of this center is shown in Fig. 4). The ground floor contains public services (*e.g.*, the newsstand providing access to newspapers), the first floor gives access to private services (*e.g.*, the user’s office with a cabinet containing folders and documents etc.) Based on this metaphor, a user may navigate through DV/O by “walking” around the building, “knocking” at doors, “entering” rooms etc. Besides this primary navigation mechanism, a selection of classical navigation support tools as can be found in hypertext systems is available: Various “information booths” provide orientation and descriptive access through overview maps and keyword search; a history-list and user-configurable hot-lists allow for backtracking and shortcuts.

Finally, “Persons” can be charged by the user to perform more complex tasks. One example is the vendor at the newsstand, who will perform the necessary modifications of the metaphor when subscribing to a newspaper.



Figure 4: The metaphor’s entrance hall

6 Conclusions

6.1 Summary

The aim of this paper has been to give an outline of a possible general solution concept for the creation of complex global information management systems accessible via low-bandwidth networks. The fragmented

object model seems to be a viable alternative by allowing the partial migration of object behavior, opening the full range of tradeoff alternatives between proximity to location-fixed services and object autonomy in a heterogeneous environment.

An example application based on this model, MIS/O, shows the fundamental realizability of this concept and illustrates the general aspects of accessing the docuverse from mobile hosts. Both MIS/O and DV/O are very simple and straightforward implementations; the conformance to the fragmented object model requires at some points rather liberal interpretations and abstractions. However, it has not been the goal to provide a “reference implementation” of this model, but to get a first impression of a possible system.

The idea of location transparency in a heterogeneous environment is a very powerful concept addressing a large number of different issues in globally distributed systems, once the inherent problems (*e.g.*, synchronization, consistency . . .) are solved:

- The idea of “intelligent agents” or “knowbots” roaming on behalf of the user through the network and looking for information is equivalent to object migration, which itself is a special case of partial object migration.
- One of the concepts of ubiquitous computing [30] is the ability to switch one application to different displays (with different capabilities). In the FO-model, this is achieved by migrating (or even replicating) the display portion between several MDTs.
- Given sufficiently fast synchronization mechanisms, CSCW may be enabled by replicating the interface part of one object to several MDTs.

All these points can be seen as aspects of an efficient and flexible *implementation* of location transparency. They do not affect the model of the docuverse itself.

6.2 Project state & experiences

Both MIS/O and DV/O are “up and running”. However, the MS-Windows side requires some fleshing out and optimization. Especially the low-level communication libraries used by the MIS/O implementation seem to cause problems: the system is remarkably less reliable when using the SLIP-driver than when communicating via the Ethernet-Driver.

A negative – although expected – experience has been that downgrading from UNIX to MS-Windows causes a lot of headaches because of the much lower security of the operating system. Although this allows one to answer the ubiquitous question “does it run on my PC too?” with *Yes!* (accessing a potentially large market), it is much better to employ a high-level operating system (such as Mach and Unix used for the Dataman project [8, 3]) for research purposes.

6.3 Future Work

The FO-“model” itself as described in this paper is much too conceptual to serve as a specification. Although it identifies the general direction into which to move, it requires substantial detailing and clarification of the more subtle matters such as consistency, synchronization, and manipulation of objects. Additional questions concern architectural matters such as communication facilities and the primitive operations to be provided by the MDT. Some items for further work on the FO-model are the following points:

Display primitives: Besides agreeing on the fundamental architecture of a docuverse implementation (*i.e.*, where are the procedures for implementing complex interface behavior based on primitive operations executed), the set of required primitives itself has to be determined.

In a dynamic environment such as the docuverse, this is a quite difficult task, because ideally every possible future input or output primitive should be available – without knowing what the future might

bring. There is a clearly defined value space for any possible output primitive – the capabilities of the human sensory system –, so that in principle one could identify a set of orthogonal primitives covering the complete area. However, the complexity of this value domain clearly inhibits any approach in this direction. At the same time, there must be a mechanism for coping with the widely different output and input capabilities of the various devices used for accessing the docuverse. Even today, these potentially range from high-resolution stereoscopic rendering down to 64×128 monochrome display. Some solution alternatives to these problems specifically for the X-protocol are discussed in [9]. However, for input and output requirements going beyond 2D, a new set of primitives is required (or at least additional ones).

Transport and exchange functionality: Not only the migration of object behavior, but also the transport of multimedia object content data is a difficult problem in low-bandwidth scenarios. While caching and compression are typical technologies for optimizing the access of multimedia object contents as such, additional concepts can be based on exploiting the user's content access behavior for beginning content rendering before finishing the content transfer. Here, the access of individual contents or the browsing of multiple contents can be supported by techniques such as prefetching, detail-on-demand, and successive refinement. The challenge is to provide a content-type *independent* model for embedding such concepts into the transport layer.

Also, there needs to be a tight integration between transport layer and display functionality – *e.g.*, for the immediate rendering of arriving detail data without further application involvement. Furthermore, the transport layer has to support the simultaneous transfer of multiple contents – possibly with a guaranteed Quality-of-Service (QoS), as well as the management of simultaneous successive refinements.

When looking at formats for exchanging *structured multimedia documents*, concepts are required which make structure information available before bulk data and allow for the dynamic interleaving of different bulk data parts. It is clear that the applicability of exchange formats such as MHEG [17] and HyTime [20] needs to be reviewed under these assumptions.

Detailing of object model: The most obvious points currently missing from MIS/0 are the persistent manipulation of objects (note that the “Data” arrows in Fig. 3 are unidirectional!) and the *replication* of object parts. A limited replication is currently supported by behavior migration to multiple MDTs, but this mechanism produces inconsistencies once persistent object manipulation is allowed. Replication of other parts of object behavior – *e.g.*, valuation – would allow for a parallel execution of this behavior once suitable synchronization primitives have been defined. An object mining for information in the global network may therefore “fork” itself to all sites possibly maintaining interesting information, so that search on all these sites executes in parallel. The necessary basic solution concepts for these functions can be borrowed from results in distributed operating systems, the main work to be done here is the selection of the best solutions and the integration into the docuverse object model.

Besides sheer expressiveness another aspect of an object model is the syntactic structure it imposes on an object representation. Typical object models only define “slots” and “methods” as structural elements of an object representation. While this is quite sufficient from the viewpoint of expressiveness (one can even live without slots, once the model allows for mutable methods on a per-object basis), *extended object models* such as the ones described in [22, 14] provide for a much finer structured object representation. In addition to supporting object implementation by an information provider, extended object models simplify automatic object modification and fragmentation: because all aspects describing, for example, interface behavior are syntactically separated from the ones describing, for example, valuation, the deliberate modification or migration of one of the parts is possible using a simple syntactic rather than a semantic analysis of the object representation.

As visualization and interaction are the enabling technologies for mobile information systems [5], the main focus will be put on the facilities for defining and identifying an object's interface behavior. The MOVI project

[15, 13] funded by the German Science Foundation (DFG) will investigate these aspects on a larger scale (the author of this paper is responsible for the scientific coordination of this project).

Acknowledgments

Kaisa Väänänen has designed and implemented the interface behavior of the DV/0 entry-point. The metaphor's 3D scenes have been rendered by Ulrike Spierling.

Olaf Potzelt has supported the implementation of the SDS and wrote the electronic newspaper of DV/0. Jian Zhang succeeded in talking the obstinate Windows socket communication into doing their job.

The work on MIS/0 and DV/0 has been supported by De·Te·Mobil.

References

- [1] Berners-Lee, T., Cailliau, R., Groff, J., Pollerman, B. WorldWideWeb: The Information Universe. *Electronic Networking: Research, Applications and Policy*, 1(2):52–58, Spring 1992.
- [2] Borenstein, N.S. Email With A Mind of Its Own: The Safe-Tcl Language for Enabled Mail. Internet Draft, Proc. ULPA'94, 1994.
- [3] DATAMAN Lab Configuration. Available via WWW (URL: <http://paul.rutgers.edu/~acharya/lab.html>).
- [4] Deutsche Bundespost Telekom et al. KIT – Window Based Kernel for Intelligent Communication Terminals, version 1.0. Deutsche Bundespost Telekom, Germany, November 1994.
- [5] Encarnação, J.L., Frühauf, M. Visualization and Interaction: The Enabling Technologies for Applications and Services of Mobile Computing Systems. In *TENCON'94*, 1994. (Keynote Address).
- [6] Forman, G.H., Zahorjan, J. The Challenges of Mobile Computing. *IEEE Computer*, 27(4), April 1993.
- [7] Hudak, P., Sundaresh, R.S. On the Expressiveness of Purely Functional I/O Systems. Technical Report YALEU/DCS/RR665, Department of Computer Science, Yale University, March 10 1989.
- [8] Imielinski, T., Badrinath, B.R. Mobile Wireless Computing: Challenges in Data Management. *Communications of the ACM*, 37(10):18–28, October 1994.
- [9] Kantarjiev, C.K., Demers, A., Frederick, R., Krivacic, R.T., Weiser, M. Experiences with X in a Wireless Environment. In USENIX [26], pages 117–128.
- [10] Kirste, T. forthcoming Ph.D. Thesis.
- [11] Kirste, T. HCL Language Reference Manual, Version 1.0. ZGDV-Report 68/93, Computer Graphics Center, 1993.
- [12] Kirste, T. The Concept of Self-Describing Data and its Relevance for Open Hypermedia Systems in Network Environments. In *Proc. Workshop Open Hypertext Systems*. Universität Konstanz, May 26–27 1994.
- [13] Kirste, T. Mobile Visualization – Project Home Page. Available via WWW (URL: <http://www.igd.fhg.de/www/zgdv-mmivs/~movi/>), January 1995.
- [14] Kirste, T. Some issues of defining a user interface with general purpose hypermedia toolkits. In Schuler, W., Hannemann, J., editors, *Proc. Workshop on Methodological Issues on the Design of Hypertext-based User Interfaces (July 3–4 1993, Darmstadt, Germany)*. Springer, 1995.

- [15] Kirste, T., Heuer, A., Kehrer, B., Schumann, H., Urban, B. Concepts for Mobile Information Visualization – The MoVi-Project. Submitted to Eurographics Workshop on Scientific Visualization 1995, January 1995.
- [16] Kirste, T., Hübner, W. An Open Hypermedia System for Multimedia Applications. In Kjelldahl, L., editor, *Multimedia: systems, interaction and applications (Proc. 1st Eurographics Workshop on Multimedia, April 18–19 1991, Stockholm, Sweden)*, pages 225–243. Springer, 1992.
- [17] Kretz, F., Colaitis, F. Standardizing Hypermedia Information Objects. *IEEE Communications Magazine*, pages 60–70, May 1992.
- [18] Makpangou, M., Gourhant, Y., Le Marzul, J.-P., Shapiro, M. Fragmented Objects for Distributed Abstractions. Technical report, INRIA, France, October 1991.
- [19] Nelson, T.H. All for One and One for All. In *Proc. Hypertext '87 (November 13–15 1987, Chappel Hill, North Carolina)*, pages v–vii. The Association for Computing Machinery, 1987.
- [20] Newcomb, S.R., Kipp, N.A., Newcomb, V.T. The “HyTime” Hypermedia/Time-based Document Structuring Language. *Communications of the ACM*, 34(11):67–83, November 1991.
- [21] Parkin-White, A. Market Development in Mobile Data. *Mobile Communications International*, 14:56–59, Summer 1993.
- [22] Rettig, M., Simons, G., Thomson, J. Extended Objects. *Communications of the ACM*, 36(8):19–24, August 1993.
- [23] Satyanarayanan, M. Mobile Computing. *IEEE Computer*, 26(9):81–82, September 1993.
- [24] Satyanarayanan, M., Kistler, J.J., Mummert, L.B., Ebling, M.R., Kumar, P., Lu, Q. Experience with Disconnected Operation in a Mobile Computing Environment. In USENIX [26], pages 11–28.
- [25] Scheifler, R.W., Gettys, J. *The X Window System*. Digital Press, 1992.
- [26] *Proc. USENIX Symposium on Mobile & Location-Independent Computing (August 2–3 1993, Cambridge, MA)*. USENIX Association, 1993.
- [27] Väänänen, K. Interfaces to Hypermedia: Communicating the Structure and Interaction Possibilities to the Users. *Computers & Graphics*, 17(3):219–228, 1993.
- [28] Väänänen, K. Hypermedia im Mobilfunk: Spezifikation der Anwendersicht mit der Metapher “Dienstleistungszentrum”. Project Deliverable R.1, Computer Graphics Center, July 1994.
- [29] Watson, T., Bershad, B.N. Local Area Mobile Computing on Stock Hardware and Mostly Stock Software. In USENIX [26], pages 109–118.
- [30] Weiser, M. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–85, July 1993.
- [31] White, J.E. Mobile agents make a network an open platform for third-party developers. *IEEE Computer*, pages 89–90, 1994. November.