A presentation model for mobile information visualization

Thomas Kirste
Darmstadt Technical University, Dept. Computer Science, Interactive Graphics Systems Group,
Wilhelminenstr. 7, D-64283 Darmstadt, Germany,
Email: kirste@igd.fhg.de

Uwe Rauschenbach
Rostock University, Dept. Computer Science, Computer Graphics Research Group
D-18051 Rostock, Germany
Email: urausche@informatik.uni-rostock.de

Abstract. One of the visions of mobile computing is to put "all information at the user's fingertips" – to allow a user to operate on any data, any time, anywhere. The idea is to create an information environment providing the homogeneous access to all data and services available in the distributed, mobile computing infrastructure.

A fundamental requirement for the access to such an open, distributed information system is an intelligent selection of methods for information visualization based on user requirements and available display functionality.

In this paper, a flexible concept is proposed that allows to enrich the nodes of an information structure with information about which alternative display methods can be used for what parts of the node. These *facets* are then used by a recursive view generation process for selecting suitable display methods while creating a visualization of an information structure. Influence parameters such as user characteristics, display resources, and data properties can be used to guide the selection process in order to create a presentation that optimally meets the user's goals.

1 Introduction

One of the visions of mobile computing is to put "all information at the user's fingertips" – to allow a user to operate on any data, any time, anywhere. The idea is to create an information environment providing the homogeneous access to all data and services available in the distributed, mobile computing infrastructure. The term "Infoverse" will be used to denote this information environment. The Infoverse can be seen as an extension of the "Docuverse" concept defined by T. Nelson [1, 2], a distributed hypermedia structure containing and interlinking the entire human knowledge[1].

The typical infrastructure of a system for the mobile access to the Infoverse consists of a mobile end system (MES) equipped with a wireless data communication facility, and a stationary data server (SDS) that is attached to both the wireless communication link and stationary high speed networks (cf. Figure 1). A typical MES might be a PDA (personal digital assistant) using a GSM mobile phone for wireless data communication. The SDS gives the MES access to the distributed multimedia information services available in the stationary network infrastructure. Such infrastructures are used for field service applications [5] as well as for distributed personal information management scenarios [6].

The main task of the MES is to provide a suitable user interface for these services, which means both information visualization and the support for direct interaction with these presentations.

[1] Today, the World-Wide Web [3] provides some – but not all – of the linking functionality defined for the "Docuverse" using "tumblers" [4].

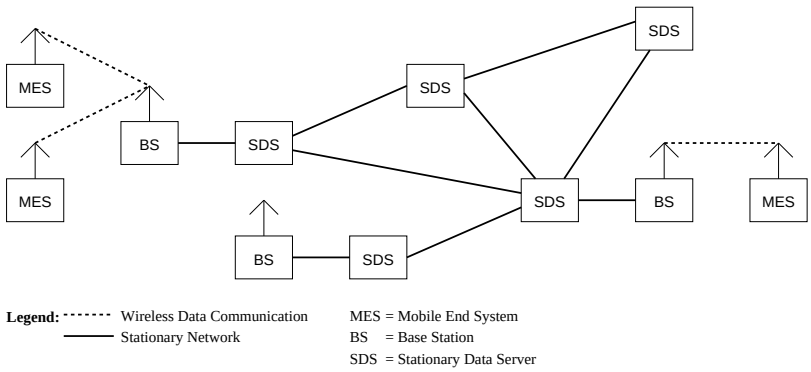


Figure 1: Typical wireless information infrastructure

The wireless infrastructure shown in Figure 1 presents some well known challenges for this task, which require special attention:

- Limited bandwidth of the wireless link (*e.g.*, 9.6 kBit/sec. in case of GSM); high bandwidth variability.
- Limited computation and storage resources of the mobile end system.
- Heterogeneous end system resources introduced by users using different end systems at different times (*e.g.*, a monochrome PDA on a business trip, a full color graphics workstation at the office.)

On the other hand, there is the challenge of the openness of the information environment – any time a new service (*i.e.*, a new data type with accompanying operations) can be added to this environment. Therefore, the MES must be prepared for situations, where the user dynamically wants to access (and operate on) data entities of a type unknown to the MES. The MES should be intelligent enough to upgrade itself with the functionality required for giving the user interactive access to this data. This requires a suitable data model for the information environment which provides a well defined way for adding new data types with their associated functionality for presentation and interaction. The MES uses this type information for determining what and how to upgrade.

With respect to the visualization functionality, it is important to support the flexible handling of alternative presentation methods. In the area of scientific visualization, it is a well known fact that different visualization methods show different aspects about the same data set [7]. Which visualization method to chose is a non-trivial process that depends on the user's informational needs, the characteristics of the data set to visualize, and the properties of the user's end system [8].

Especially for a mobile user, data access is expensive both in terms of time (due to the low bandwidth of wireless links) and money (due to the high cost of each transferred byte). Therefore, it is important to give a user the best value for his expenses – which means to let the user extract as much information as possible form a data set whose transfer he has paid for. This in turn means to support the selection of a visualization method best satisfying the user's interpretation goals. Also, the user should have the ability to switch between different alternative methods in order to extract the various informations buried in the data set.

Another point is the flexible handling of heterogeneous end systems: on a low-resolution monochrome display, different presentation techniques have to be used than on a high-resolution color display (*e.g.*, 2-D black & white iso-line drawings instead of a color coded 3-D height map – cf. [8]).

The point of this paper is to propose a concept for the intelligent handling of the visualization of multimedia information structures on heterogeneous end systems within the context of the Infoverse.

The central idea is to enrich the nodes of an information structure with information about which alternative display methods can be used for what parts of the node. These *facets* are then used by a recursive view generation process for selecting suitable display methods while creating a visualization of an information structure that is optimized with respect to user requirements, display characteristics and data properties.

The facet concept is embedded into a flexible, frame-based end-user data model. This model does not only support the browsing of distributed structured multimedia documents ("hypertext"), which have been deliberately created for this purpose by authors, but also the dynamic annotation, structuring, and composition of information. This is required to allow a user to embed his own working data into the global information environment.

The further structure of this paper is as follows:

Section 2 provides an in-depth analysis of the properties of a data model for the Infoverse, for whose instances a visualization system will have to provide suitable rendering facilities.

In Section 3, the concept of *facets* is developed and embedded into the data model identified in Section 2.

Section 4 covers additional issues of the facet concept relating to the detailed construction of the selection functionality.

A concrete application of the facet model is described in Section 5.

Finally, a summary and plans for future work are given in Section 6.

2 The underlying data model

Before developing a visualization concept for presenting the data in an information system, it makes sense to analyze the system's data model. This especially holds, if visualization information needs to be embedded into this data model.

The visualization model proposed in Section 3 is embedded into a data model that uses *frames* as the basic construct fore representing structured multimedia information. Frames are essentially unordered collections of *slots*, name and value pairs. (A more detailed discussion is given in Section 2.2.)

2.1 Why Frames?

One reason for choosing a frame-based model as starting point is clearly pragmatic: The concepts outlined in this paper have been implemented on Apple's "MessagePad" series hand-held computing devices. These machines use a software architecture entirely relying on frames for representing data and applications [9]. Delivering data to a MES using a structuring mechanism that directly corresponds to its internal data model obviously reduces the amount of work that has to be performed by the MES for the mapping of external data into a representation that can be efficiently handled by its internal data management facilities.

However, while this pragmatic reason justifies the approach taken for a specific MES software architecture, there are also considerations of a more general kind that seem to justify the use of a frame-based data model as a *general* data model for the Infoverse.

Originally, frames have been developed as a means for knowledge representation in artificial intelligence applications [10]. Powerful frame representation languages have been developed as early as 1977 (*e.g.*, FRL [11]). From this application area, frames have inherited the ability to cope with structured, fast changing information – as it also exists in the Infoverse.

Incidentally, other application areas for frame models, besides knowledge representation, have been Hypermedia-based information management systems[2] (*e.g.*, [12] and Oval [13]) and personal information management systems, such as the MessagePad.

Therefore, it seems appropriate to assume a frame-based model as underlying data model for the Infoverse rather than starting with the less expressive data model provided by the current incarnation of the Infoverse, the World-Wide Web. (See also [14] for a more detailed discussion of this point.)

Following, we give a brief outline of the essential properties of the frame model used in our work.

2.2 The frame model

The specific frame model described here is basically the one provided by the MessagePad's object system [9], which in some aspects is a simplified version of the object system built into the language *Self* [15].

[2] The whole concept of Hypermedia is quite close to a simplified frames model. Just replace the notion "Slot" by "Link".

As far as this paper is concerned, a frame is – quite similar to an object – an entity with a unique identity that contains a set of name/value pairs ("slots"). Among conventional data types, frame references may be slot values, so that frame structures can be created. Also, slot values can be *functions*, which may be invoked by message passing. Finally, frames can use other frames as *prototypes*, from which they inherit.

As an example for working with frames, consider the frames `p` and `q`, defined as follows[3]:

```
p := {a: 1, f: func() a+b};
q := {_proto: p, b: 2}
```

`p` is a frame with two slots, `a`, which contains an integer, and `f`, which contains a function. `q` is a frame with two slots, `_proto`, containing a frame reference (`q`'s prototype), and `b`, containing an integer. So `q` inherits from `p`. As one would expect, `q.b` gives the value 2, `q.a` the value 1. The message invocation `q:f()` gives 3. Here, the values of `f` (and `a`, which is accessed in `f`) are inherited from `p`. The assignment `q.c := 3` creates a new slot `c` with the value 3 in `q` (so, `q.c` now gives 3). The assignment `q.a := 4` overrides the slot `a` inherited from `p`, so now `q.a` $= 4$, and `q:f()` $= 6$, but still `p.a` $= 1$.

It is also possible to define a frame

```
r := {f: func() a*b)}
```

and change `q`'s prototype by assigning `q._proto := r`. So now `q:f()` $= 8$, because this time `f` is inherited from `r`.

So, frames can be basically be regarded as "objects with dynamic instance variables and dynamic inheritance". They provide a dynamic environment that allows the user to flexibly create, augment, and modify information structures to suit his personal needs.

As final remark, note that the frame model is nothing "new". It is not the claim or goal of this section to introduce a new data model. The interesting point made here is rather that in order to integrate personal data management into the Infoverse, one needs a data model that is substantially stronger than what is currently anticipated in, *e.g.*, the World-Wide Web. The frame model has been picked from a range of existing data models merely as an especially promising candidate.

In the next section, the central proposal of this paper, a model for supporting the flexible visualization of frame structures, will be introduced.

3 Frame Visualization

3.1 Basic frame visualization

Because frames inherently provide runtime information about their slots and the value types contained therein, it is in principle possible to create a mechanism for the automatic display of arbitrary frame structures – very similar to the "Pretty-Printing" facilities of LISP [17], such as described in [18]. The availability of such a mechanism is very important in an environment, where anytime the user may encounter unknown frame structures.

However, this basic concept assumes the visualization methods for each value type to be fixed. Within the Infoverse, this is unacceptable. As outlined in the introduction, alternative display methods have to be supported at least for the following two reasons:

- Depending on the available presentation resources, different display methods have to be selected for the same data.
- User preferences for different visualization methods should be observed. Also, the user should be allowed to switch between visualization methods in order to see different aspects of the same data.

[3] The frame language syntax used throughout this document is based on NewtonScript [16], using the following conventions: The notation "$\{s_i$: v_i,...$\}$" denotes a frame with slots s_i, which have values v_i. "[v_i,...]" is a list of values v_i. "f.s" denotes the access of f's slot s and "f.s := v" is the assignment of the value v to f's slot s. "func(x_i,...) $body$" denotes function with parameters x_i. Functions are legal slot values. "f:m(a_i,...)" denotes the invocation of the function stored in $f.m$ with parameters a_i (*i.e.*, a method invocation). For slots within the lexical or inheritance scope of the current expression, the "f." prefix can be omitted.

A possible solution is, for example, provided by systems such as Oval and HyperPicture [19]. Here, the runtime system ("Session Manager") knows a *number* of alternative visualization methods for each data object. Upon object display, an appropriate method is selected based on user preferences and resources. The user may afterwards switch between different methods.

Unfortunately, this solution is not sufficient within the scope of a global information system: The display methods known to the creator of a frame structure may not necessarily be known to a later user of this structure. In addition, it is usually not known which slots contain "interesting" information and which ones are merely used for internal purposes.

Therefore, each frame should be able to provide information about the display methods useful for visualizing frame information. This approach is used in principle by NewtonScript: The display method to be used for a frame can be defined by the frame's `_proto`-slot.

The solution concept proposed in this section is essentially a unification of these two techniques, based on the concept of *facets*.

3.2 Facets and views

The extended visualization model consists of the following elements:

Facets. Facets have two tasks:

- They define groups of slots that can be presented together in a meaningful way.
- They can map these slot groups to display methods that are appropriate for visualizing the slot values.

The idea is to use a frame's facets for determining *which* information of the frame to display and *how* to do the rendering.

Display methods. A display method is the definition of a function which is able to visualize a set of values with a given structure (*e.g.*, a string and a list of integers). A display method is *applicable* to a frame, if the frame matches the *structural constraints* of the display method. These constraints are defined by the set of slots the frame must contain and the types of the values contained in these slots.

Facet views. Facet views display a facet of a frame using a specific display method. Also, they allow the user to switch between the different display methods applicable to the presented facet.

Frame views. Frame views are responsible for orchestrating the display of the various facets of a frame. In addition, specialized frame views can be provided for presenting *sets* of frames. For this case, they may support layout techniques such as tables or graphical overviews ("structure maps") and the consistent selection of facets across the whole frame set.

3.3 Facet declarations

Every frame may include facet declarations which the visualization system expects to find in a slot named `facets`. The value of a frame's `facet` slot is a *facet declaration frame*. Each slot in the facet declaration frame contains a *facet declaration*, the slot name of a facet declaration is the *facet name*. A facet declaration is itself a frame consisting of an (optional) *template definition* (a slot named `template`) and an (optional) *display method definition* (a slot named `display`).

The content of a template definition is a frame whose slots contain *value expressions*. These value expressions may be constants, functions with a single parameter, and "path expressions" such as `'a.b.c`. Path expressions allow to "pull" a slot buried deeper in the frame structure upward into the template frame.

When the view system creates the visualization of a frame's facet, it does not apply the facet's display method directly to the frame. Rather, it instantiates the facet's template from the frame and applies the display method to the instantiated template frame. This allows to define (quite arbitrarily) derived frame structures, whose contents is better suited for a facet's visualization problem than the original data itself (see Section 3.5 for a detailed example).

The `display` slot of a facet declaration defines the set of display methods that are – from the frame creator's point-of-view – applicable for presenting the instantiated template frame. The value of this slot can be:

- The name of a single display method.
- A frame whose slots contain alternative display methods (the slot names are the alternative's names). The value of an alternative method slot may be a display method or again a frame that contains a `method` slot, whose value is the desired method, and other slots that will be passed as parameters to this method (see Figure 3 for an example).

Consider the following frame:

```
f := {
  facets: {facet1: {template: {string: 'stringSlot},
                    display: {simple: 'simpleFrameDisplay,
                              fancy: 'fancyFrameDisplay}}
           facet2: {template: {number: 'numberSlot}
                    display: 'numberFrameDisplay}},
  stringSlot: "Some Text",
  numberSlot: 1234}
```

This frame defines two facets, `facet1` and `facet2`. The `template`-slot of `facet1` defines a single-slot template frame that will contain the value of `f.stringSlot` upon instantiation, and which should be displayed using either the display method `simpleFrameDisplay` or `fancyFrameDisplay`. `facet2` will display the value of `f.numberSlot`, using the display method `numberFrameDisplay`.

3.4 The view generation process

The view generation process is defined by the three core functions `buildView`, `buildFrameView`, `buildFacetView`, and the respective display method, which call each other recursively. A rudimentary definition of the core functions and a sample display method is given in Figure 2, using NewtonScript inspired pseudo-code. Based on these functions, the visualization system proceeds as follows when called through `buildView` with a value (*e.g.*, a frame structure) to visualize:

- `buildView` determines if the value to visualize is structured (*i.e.*, a frame) or simple and calls the respective specific view creation function. (The `parentView` parameter contains the view into which to embed the view created for `value`.)
- For frames, the function `buildFrameView` is called. It first creates a child view that will contain the frame's different facet views. Then it selects the first facet to visualize, instantiates its template, and calls the function `buildFacetView` with the instantiated template frame.
- `buildFacetView` selects an appropriate display method using the facet declaration's `display` slot and possibly information about additional local display methods that may be applicable. It then calls the selected display method with the instantiated template frame. (Facet and display method selection are further discussed in Section 4.)
- The display method is responsible for displaying the instantiated template frame in a suitable way. The sample method `simpleFrameDisplayMethod` simply iterates over all slots in the template, creating a label and a value view for each one. The value view is created by recursively calling `buildView`, thereby closing the circle.

3.5 An example

This section gives a small example illustrating the use of the facet model. In Figure 3, a sample frame structure including facet declarations is given. The different renderings of this frame structure generated by choosing different facets and display methods are shown in Figure 4 to 6.

This frame structure illustrates the following features of the facet model:

```
buildView: func(parentView,value)
  if IsFrame(value)
    then buildFrameView(parentView,value)
    else buildImmediateView(parentView,value)

buildFrameView: func(parentView,frame) begin
  childView := buildFrameContainer(parentView);
  facet := selectFacet(frame);
  template := facet.template;
  instTplFrame := instantiateTemplate(template,frame);
  buildFacetView(childView,instTplFrame,facet);
end

buildFacetView: func (parentView,instTplFrame,facet) begin
  displayMethod := selectDisplayMethod(facet);
  call displayMethod with (parentView,instTplFrame)
end

simpleFrameDisplayMethod: func(parentView,instTplFrame)
  foreach slot,value in instTplFrame do begin
    buildSlotLabelView(parentView,slot);
    buildView(parentView,value)
  end
```

Figure 2: Core functions of the visualization system

- The declaration of multiple facets for a frame (*e.g.*, the `sales` and `oview` facets for the top level frame).
- The declaration of multiple display methods for a facet (*e.g.*, the `fancy` and `simple` display methods in the `sales` facet of the frames `fstruct.tbl` and `fstruct.y.c`).
- The use of deep-structure value access in the definition of a facet's template (*e.g.*, the `m` slot of the top-level frame's `sales` facet template). Note how this is used to create a display that presents the message `fstruct.y.c` and the table `fstruct.tbl` at the same level in Figure 4. Compare this with the direct rendering of the frame structure in Figure 6.
- The use of on-the-fly computations in the definition of a facet's template (*e.g.*, the `v` slot of `fstruct.y`'s `facets.oview` template, which sums up the individual unit numbers in the frame's `a` slot, displaying a grand total to the user).

Once displayed, a facet view allows a user to dynamically switch between different display methods for the same facet, as shown in Figure 4. Note that the visualization layout may change in response to switching a display method.

Figure 6 shows a rendering generated by choosing *no* facets and display methods. In this case, the original frames are directly used as instantiated templates, whose slots and values are displayed using standard methods. Note that this is also done with frames lacking facet or display-method definitions, such as `fstruct.z`. This defaulting mechanism allows to handle the display of arbitrary frame structures without requiring the author to spend extensive thoughts on useful facet definitions. These can easily be added at a later time.

4 Display method selection and execution

In this section, we discuss how display methods are selected, and how an optimal execution of a display method's presentation pipeline can be planned.

```
fstruct :=
{facets: {sales: {template: {m: 'y.c, t: 'tbl}},
          oview: {template: {a: 'y, b: 'z}}},
 x: 1, /* some internal value of no interest to the user */
 y: {facets: {oview: {template: {m: "Company total sales (units)",
                                 v: func(f) begin
                                      local s:=0;
                                      foreach v in f.a do s:=s+v;
                                      s
                                    end}}},
     a: [400,280,460,380],
     c: {facets: {sales: {template: {text: 'bla},
                          display: {fancy:  {method: 'directTemplateView,
                                             .../*fancy text style params*/},
                                    simple: {method: 'directTemplateView,
                                             .../*simple text style params*/}}}},
         bla: "Sales Report:\nHere is the recent ... "}},
 tbl: {facets: {sales: {template: {value: 'dta},
                 display: {fancy: 'graphTableView, simple: 'textTableView}}},
       dta: [["Sales", "1993", "1994", "1995", "1996"],
             ["Prod. A:", 100,110, ...], ...]},
  z: {m: "Company sales area", p: ... /* Image Reference */}}
```

Figure 3: Sample frame structure

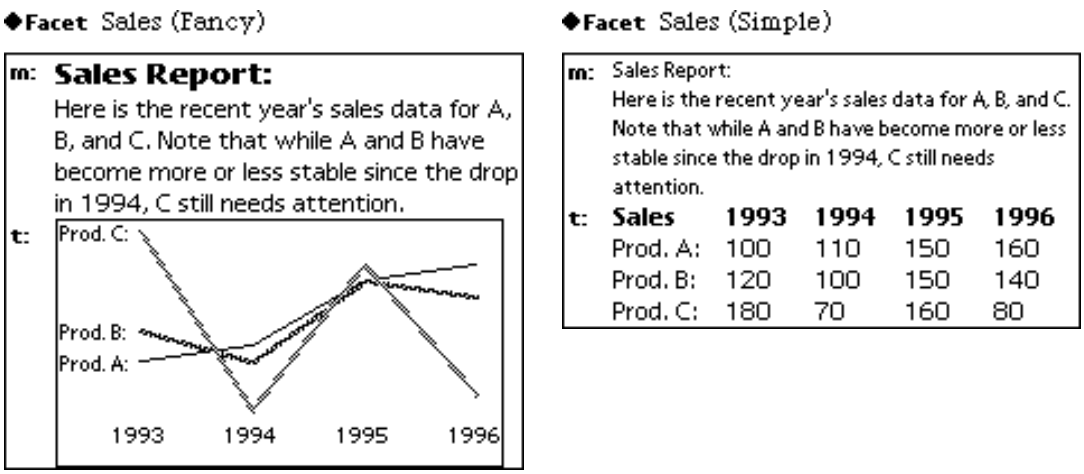


Figure 4: Display of facet `sales` using display method `fancy` (left) and `simple` (right)

4.1 Determining the effective method set

The first step in selecting a display method for a facet is to determine the facet's *effective method set.*

Because of the Infoverse's openness, a user's end system may contain display methods that are not contained in the frame's facets (and vice versa). Therefore, the set of display methods applicable to a frame's facet have to be computed dynamically. With respect to computing the *effective set* of display methods for a facet, the simplest choice is to intersect the facet's method set with the set of applicable methods that are

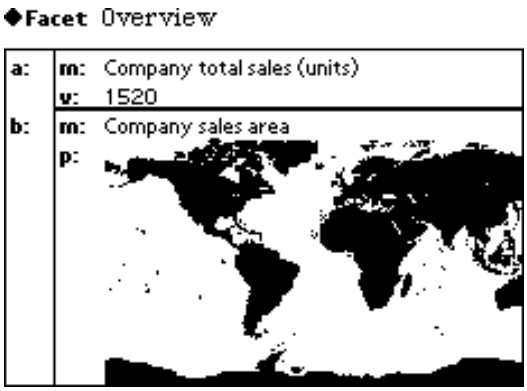


Figure 5: The `oview` facet

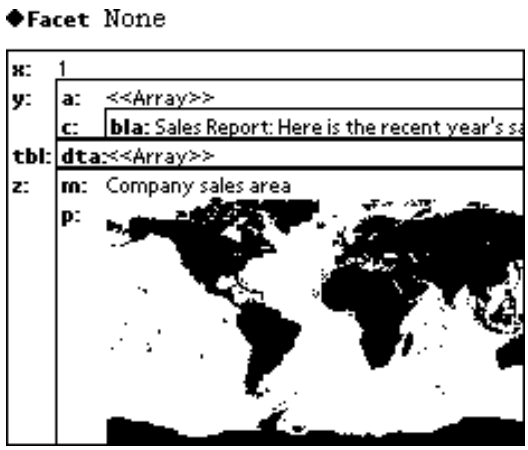


Figure 6: Using standard visualization

available locally. (A display method is applicable to a facet, if its structural constraints are matched by the facet's instantiated template frame.)

If the intersection is empty, the system may decide to dynamically down-load one or more of the unknown display methods named in the frame's facets[4].

The visualization system may also choose to offer local applicable methods that are not contained in the facet declaration's method set as additional alternatives.

4.2 Selecting a method

After computing the effective method set, the optimal display method has to be selected from this set. This selection depends on various *influence factors* such as goals, preferences, and capabilities of the user, time and quality demands of the user, resources of the mobile environment, and rendering properties of the display method.

In scientific visualization, several different techniques for the selection of suitable visualization methods have been proposed. Possible approaches are *rule inference systems, sets of functions* (see, *e.g.*, [8]) or a *comparison of cost and benefit* (see, *e.g.*, [21]).

In the facet model, the set of display methods varies dynamically with the selected facets and their effective method sets. Therefore, a selection mechanism is required that allows the dynamic composition of method specific selection functionality. Rule inference systems do not meet this criterion, because the effect of composing partial rule systems is difficult to predict.

Using a cost-benefit-approach, both cost and benefit of a display method are separately described by functions. A display method is identified as suitable if its cost-benefit-ratio is high enough. The problem with this approach is to find a suitable common measure for relating cost and benefit (see [22] for a detailed discussion).

Therefore, we are using a function-set approach, which describes the suitability of display methods with respect to the influence factors by independent, method-specific *suitability functions*. The following influence factors are used as input parameters for these functions:

- *User characteristics and goals.* This includes information about the user (*e.g.*, is he able to perceive color) as well as his interpretation goals (*e.g.*, discovery of correlations in the data set[5]).
- *Data characteristics.* This contains meta-data about the information to visualize. For multivariate data, this includes information such as the number of variables, the number of observations, the average number of different values per variable, and the maximum number of correlated variables.

[4] This down-loading is supported by the mobile frame model outlined in [20].

[5] For a more detailed description of interpretation goals and their influence on the selection process see, e.g, [23] or [24].

- *Mobile client display resources.* This includes properties such as the display hardware characteristics, *e.g.*, the number of pixels and number of colors.

The suitability functions of the various display methods map these parameters to *suitability values*, which are then used to select the appropriate method. Formally, this selection process can be described as follows:

Let U be the set of user characteristics, R the set of resource characteristics, and D the set of data characteristics. The set of characteristic values C is then defined as

$$C = \mathbb{P}\,U \times \mathbb{P}\,R \times \mathbb{P}\,D,$$

where $\mathbb{P}\,X$ denotes the set of all subsets of the set X. Furthermore, let M be the effective method set, G the set of interpretation goals, and S an ordered set of suitability values. For each method $m \in M$, we can then introduce a *suitability function*

$$f_m : C \times G \to S,$$

which assigns suitability values to pairs of characteristics $c \in C$ and goals $g \in G$. So $f_m(c, g)$ describes how well method m is suited for fulfilling goal g under the characteristics c.

The user can specify his interpretation goals by assigning *satisfactory suitabilities* – elements of S – to the goals in G, giving a weighting function

$$w : G \to S.$$

Given a characteristics $c \in C$ and a weighting function w, the suitability functions define a partial order $\geq_{(c,w)}$ on the method set M. This partial order is described by first introducing a method's *specific suitability* as, *e.g.*,

$$s_m(c, w) = \textstyle\sum_{g \in G} \min(f_m(c, g) - w(g), 0).$$

This function assigns 0 to methods reaching the satisfactory suitabilities specified by w and negative values to less suitable methods. The order is then defined by:

$$\forall\, m, m' \in M : m \geq_{(c,w)} m' \Leftrightarrow s_m(c, w) > s_{m'}(c, w) \vee m = m'.$$

(In other words, methods are ordered by their specific suitability, as one would expect.) The order is partial, because different methods may have the same suitability value. For automatic method selection, the best method (i.e. the largest with respect to $\geq_{(c,w)}$) is chosen. (If more than one method qualifies as best, the method execution costs described in Section 4.3 are used as additional ordering criterion.) For user controlled method selection, those methods are provided whose specific suitability is larger than a certain threshold.

New display methods can easily be added to this selection process, as it is only required to define new suitability functions f_m. The design of the suitability functions is a non-trivial task, however. At the moment, this is done heuristically by the developer of the technique or by a visualization expert and then evaluated by user testing.

4.3 Display method execution

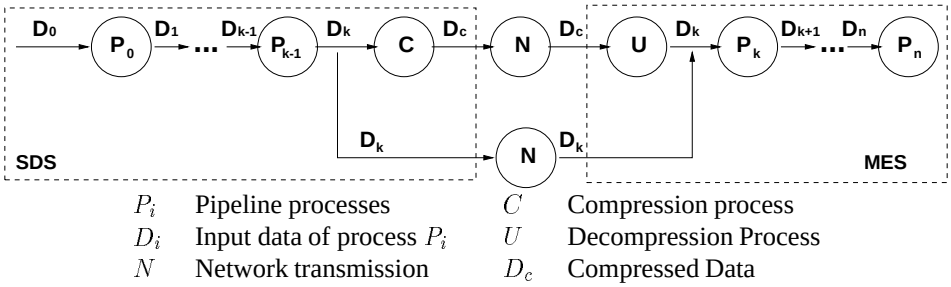
After the selection of the display method, its execution on the network infrastructure has to be planned.

For complex rendering tasks, the display method is often not a single process but a chain of processes, which is referred to as *visualization pipeline* (cf., *e.g.*[25]) or *presentation pipeline* (cf. [22]). It transforms the abstract information stored in the frame's slots into a concrete visual representation. In order to do that, it usually includes pipeline processes such as "data subset selection", "mapping of abstract to geometry data", "rendering of geometry data", and "presentation of the rendered data".

With respect to the constrained computation resources of mobile clients and the low bandwidth of wireless networks, it is now an interesting idea to *partition* this pipeline between MES and SDS in order to make best

use of the available resources, as outlined in [26]. (This is a special case of the general concept of *application partitioning*, which is exploited in mobile computing. See, *e.g.*, [27, 6].)

We call the allocation of the presentation pipeline processes to MES (client) and SDS (server) *presentation pipeline partitioning*. Figure 7 illustrates the idea: The presentation pipeline is partitioned between MES and SDS by inserting a network transmission process between pipeline processes P_{k-1} and P_k. The problem is now to determine the optimal partitioning point k and whether or not to use data compression.



P_i	Pipeline processes	C	Compression process
D_i	Input data of process P_i	U	Decompression Process
N	Network transmission	D_c	Compressed Data

Figure 7: The presentation pipeline partitioning problem

This decision is based on suitable cost functions modeling the processing and communication costs for the various pipeline elements with respect to the machine they are executed on. For a simple pipeline execution model – i.e. non-parallel processing and zero cost for local transport – the total costs can formally be described as follows:

Let $t(d, p, h)$ denote the time process p requires for processing data d on host h. Then the time spent on the server (SDS) for a pipeline split at k is given by

$$T_s(k) = \sum_{i=0}^{k-1} t(D_i, P_i, Server).$$

The time spent on the client (MES) is

$$T_c(k) = \sum_{i=k}^{n} t(D_i, P_i, Client).$$

The time spent for network transport depends on whether compression is chosen or not. Let $m(d)$ be the cost for moving data item d across the network. Then the communication cost are given by

$$T_n(k, \alpha) = (1 - \alpha) * m(D_k) + \alpha * (t(D_k, C, Server) + m(D_c) + t(D_c, U, Client)),$$

where $\alpha \in \{0, 1\}$ chooses between compressed or uncompressed transfer.

The total cost function is then given by $T(k, \alpha) = T_s(k) + T_n(k, \alpha) + T_c(k)$ and the goal is to find the pair (k_0, α_0) where $T(k_0, \alpha_0)$ is minimal.

The main problem here is of course the definition of $t(d, p, h)$, as this task has no analytic solution in general ([21]). It furthermore depends on a detailed resource description as well as on the concrete implementation of pipeline processes. We plan to use suitable heuristics for this purpose; ongoing work in the MOVI project is expected to provide these.

5 A sample application

In this section, we describe a concrete application that is being developed in cooperation with the forestry commission of Mecklenburg-Vorpommern, a Land of the Federal Republic of Germany.

The forestry commission maintains a relational database containing information about the various forestry districts in its area. This information needs to be available indoors, on the stationary computers of the various forestry offices, as well as outdoors to forest wardens when examining the state of a district.

Within MoVi, we have built a prototype application for accessing this data using stationary PCs indoors and "MessagePad" PDAs when outdoors. The facet model is used to embed the various visualization options for this heterogeneous environment into the data.

5.1 Data set and user access requirements

Organizational, each forestry district is divided into *departments*, which themselves are subdivided into *areas*. Areas areas consist of a number of *sections*, the atomic unit of a forest district.

The forestry commission's database (FCDB) contains information at section resolution. A number of aspects of a section, relating to section state and section utilization, are stored in the FCDB. With respect to section state, aspects such as growth class, pollution damage, tree variety, and section size are stored. Typical section utilization aspects are production goal, access frequency, and national park zone.

A record of the FCDB basically consists of a section key and a value for each of the various aspects. For a single district, there are typically about a thousand sections (*i.e.*, records).

The goal of the prototype was to provide the following features:

- Allow information access at department level granularity, providing either an overview of the whole department state or a detail presentation showing the state of each area and section in this department.
- Allow the user to access department information by simply clicking on a display of the district map. Also, the user should be allowed to switch dynamically between the detail levels and between the aspects displayed.
- Allow information access either through stationary PCs or through mobile PDAs.
- Support the specific display characteristics of the different end systems (*e.g.*, color for the stationary PC, monochrome in case of the PDA)

The next section discusses the techniques chosen for implementing these visualization requirements on the different end-systems.

5.2 Visualization strategy

Figure 8 and 9 show a typical display of the PDA-based front-end. The presentation of a detail view on the PC front-end is shown in Figure 10.

The department overview (Figure 9) is shown as standard pie chart. A detail view (Figure 8 and Figure 10) is presented as two-dimensional icon, using a kind of row/column layout. The major dimension encodes the area, the minor dimension encodes the section. In order to optimize the use of screen real-estate, the major dimension is chosen in direction of the larger screen dimension, because there are usually more areas in a department than sections in an area. So for the PDA, column major layout is used, while row-major layout is chosen for the PC.

For presenting the different aspect values, color coding is used for the PC and pattern-coding for the PDA's monochrome screen.

The department view's [E] ([O]) button allows to switch between overview and details view. The [Zeige] (= show) button allows to select the aspect to display. The current aspect (*e.g.*, "Zustand.Baumart" = state.tree-variety) is displayed in the area left of the button.

Also, in order to allow a user to change screen layout to better suit his needs, data view and the optional legend view are presented using draggable windows on the PDA. On the PC's larger color screen, the complete district map is visible so that the data view can be directly overlayed onto the selected department. Also, the legend can be displayed permanently in a fixed area beside the map.

Next, we outline the frame structure used for creating these visualizations.

5.3 Use of the facet model

The mapping of the FCDB data to a corresponding frame structure has been straightforward: Each section record is represented as a frame with a slot for each aspect. An area is represented by a frame with a slot

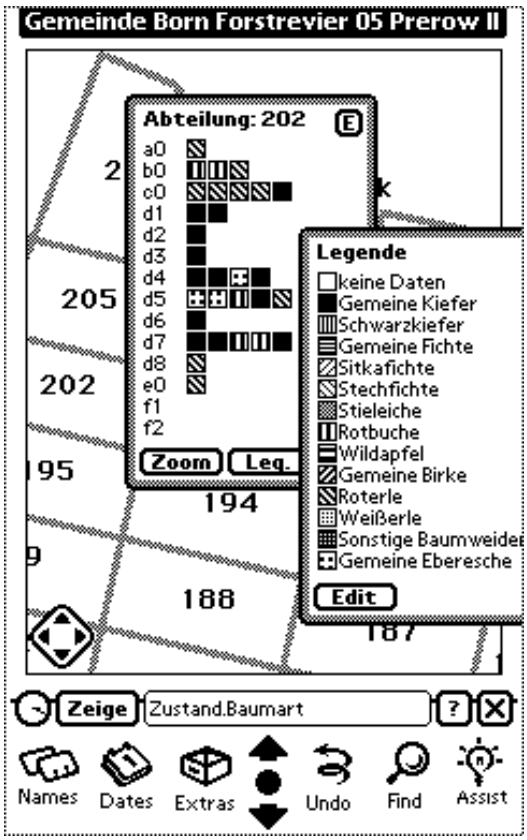


Figure 8: PDA detail display

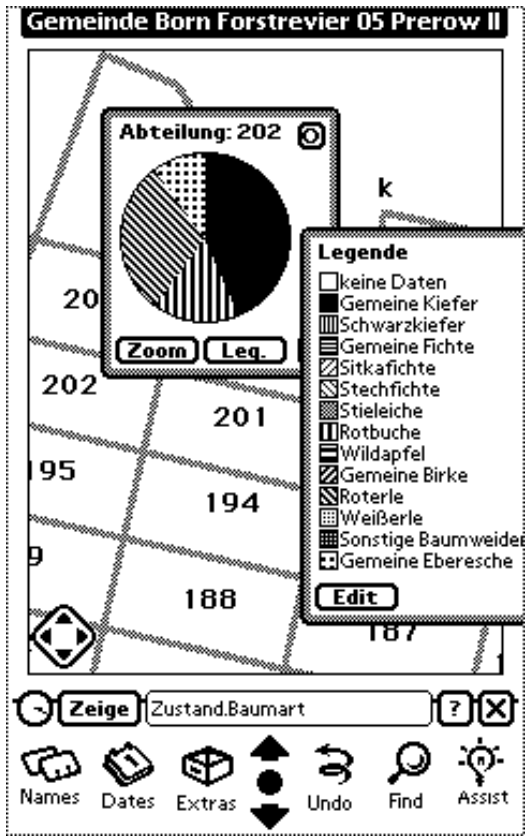


Figure 9: PDA overview display

containing an array of its section's frames, and a department is a frame containing an array of sections. In addition, the frames contain facet declarations (see Figure 11).

A department frame contains facets for presenting overview and detail views. This facet is selected with the [E]/[O] button as explained above. Note that there is no template for the `details` facet, meaning the whole department frame is used as its own template.

An area frame contains only a single facet for presenting the area's sections. When visualizing an area frame, this facet is always selected. A section frame contains a facet for each aspect slot with a template referencing only this aspect slot. The facet to display for all section frames is selected by the [Zeige] button.

The display method selection is fixed by the type of end system: for the PDA, the `mp` methods are used and the `pc` methods for a PC.

5.4 Implementation state

Both PC and PDA front-end are fully functional stand-alone applications, where the FCDB data is stored in native format "on-board" of the end-system. The next step will be to provide them with suitable (*e.g.*, wireless in case of the PDA) communication mechanisms for retrieving "live data" from the FCDB. The end-system's on-board store then will be used as a cache.

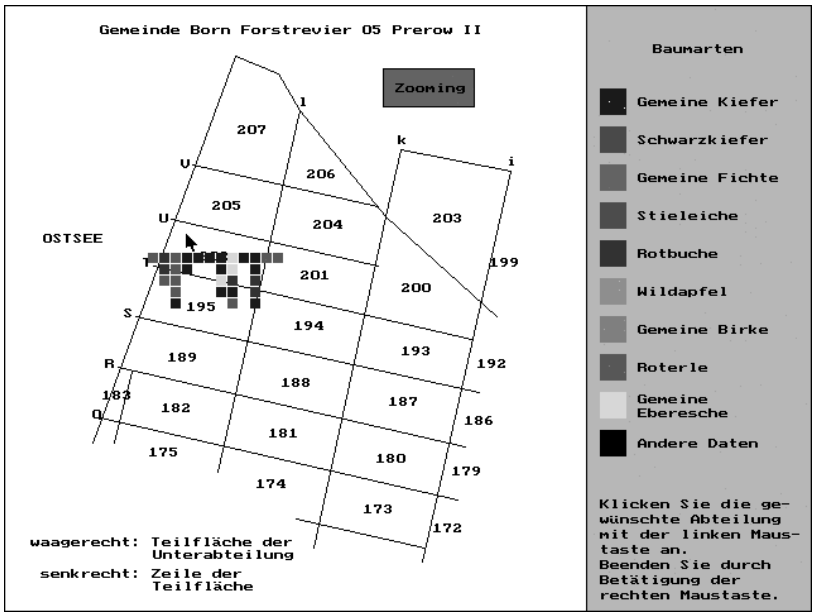


Figure 10: PC detail display

```
department := {
  facets: {oview: {display: {pc: 'colorPiechartView,
                             mp: 'monochromePiechartView}.
                   template: {totals: func(f) begin
                                      /* compute an array of totals
                                         for the district's sections
                                         and current section facet */
                                      end}},
           details: {display: {pc: 'areaRowMajorView,
                               mp: 'areaColumnMajorView}}}
  areas: [ ... ] /* Array of area frames */}
```

Figure 11: Part of application frame structure

6 Conclusion

6.1 Summary

The purpose of this paper has been to introduce a flexible presentation model for open, distributed information systems. This model should allow to embed presentation information into distributed data structures, while at the same time supporting the flexible and dynamic choice between alternative presentation methods, based, *e.g.*, on user preferences and display characteristics.

In the first part of the paper, the essential aspects of an open, distributed information system, especially from the viewpoint of a mobile user, have been analyzed. These considerations have been based on the integrative concept of the Infoverse. An important observation has been that in order to support manipulation functionality in the Infoverse – which is required at least for embedding personal information management – a suitably expressive data model is required. It has then been argued that "frames" could be a possible candidate.

The second part of the paper has concentrated on the definition of a suitable visualization model and its

embedding into the basic frame model. The central idea of the visualization model has been to enrich frames by the concept of *facets*. A facet maps parts of a frame to alternative presentation methods, providing the information required for choosing which parts of a frame can be displayed meaningful and how to do the rendering. This choice may be influenced by user requirements, data characteristics, and display capabilities.

6.2 Future work

The visualization process of the facet model introduced in Section 3 has been implemented on a MessagePad. The screen-shots in this paper have been generated by applying this implementation to the frame structure shown in Figure 3.

However, while the fundamental viability of the concepts introduced in this paper is proven by this implementation, there still remain numerous open questions. Some of them are outlined below.

- Obviously, the concrete definitions of the selection functions and their parameters, as outlined in Section 4 is an important part that is yet missing. (Currently, preferences for facets and display methods are selected by hand.) Ongoing work in MOVI is expected to provide a solution here.
- Another important aspect is geometry management and the parameterization of display methods: it depends on the user's entry point into a frame structure, at what level in the display hierarchy a frame will be displayed. This influences the amount of space available for displaying facets of this frame. So there must be a mechanism for describing and handling the dynamic layout of display methods over a broad range of available "screen real estate". Because switching facets may change the space required for a view, this geometry computation must also be carried out for views while they are displayed.
- A related aspect is the definition of layout constraints between different component frames in a composite frame. Such a constraint may read, for example, "component frames a and b must always be displayed side by side using the same facets". The definition of such constraints is especially required for the description of the coherent rendering of frame structures representing compound multimedia documents. In the terminology of MHEG, a suitable constraint mechanism effectively allows to overlay (and merge) multiple final form presentations over the same data structure.
- Finally, besides the plain selection of a visualization method, another important aspect is to decide at what time to present which components of a structured data set. This decision depends on the user's shifting focus of interest and directly influences the scheduling of requests for the data set's components.

  It is possible to exploit the specific processing sequence of frame visualization and the frame structure for a high level optimization of networked frame access. Specifically, the following techniques are easy to support in a network scheduler:

  Prefetch: Because the frame structure can be analyzed automatically, it is possible to support prefetch based on such a structural analysis: While a frame is being displayed and the system is otherwise idle, pointers to other frames may be extracted from the frame's slots and followed, prefetching the destination frames to the user's terminal system.

  Structural detail-on-demand: For component frames containing less important information, the access and visualization of these frames can be delayed until the user actually requests the display (*e.g.*, by clicking on a place-holder, similar to the "Delay Image Loading" option of the "Mosaic" WWW Browser [28]). The visualization is then dynamically extended by adding the component frames.

  The development of such a network scheduler is part of ongoing research in MOVI. See also the paper by Bönigk and Lubinski in this issue.

As concluding remark, it should be noted that through a slight extension of the basic frame model, *application partitioning*, the dynamic distribution of application functionality across the available network nodes, can be supported. See [20] for a brief outline of this *mobile frame model*.

Acknowledgments

The work presented in this paper has been funded by the German Research Society (Deutsche Forschungsgemeinschaft, DFG) as part of the MOVI researcher group.

I would like to thank all my colleagues in the MOVI project – Jörge Bönigk, Andreas Heuer, Bernd Kehrer, Susanne Lange, Astrid Lubinski, Uwe Rauschenbach, Anke Rieck, Andreas Sandberg, Jürgen Schirmer, Heidrun Schumann, Bodo Urban – for encouragement, motivation, and critical questions regarding the concepts outlined in this paper.

References

[1] T.H. Nelson. Replacing the printed word: A complete literary system. In S.H. Lavington, editor, *Proc. IFIP Congress 1980*, pages 1013–1023. North-Holland, 1980.

[2] T.H. Nelson. 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.

[3] T. Berners-Lee, R. Cailliau, J. Groff, and B. Pollerman. WorldWideWeb: The Information Universe. *Electronic Networking: Research, Applications and Policy*, 1(2):52–58, Spring 1992.

[4] T.H. Nelson. Immense storage management. *Byte*, 1988.

[5] N. Davies, G. Blair, K. Cheverst, and A. Friday. Supporting adaptive services in a heterogeneous mobile environment. In MCSA'94 [29].

[6] T. Kirste. An infrastructure for mobile information systems based on a fragmented object model. *Distributed Systems Engineering Journal*, 2(3):161–170, 1995.

[7] S. Arndt, K. Lukoschek, and H. Schumann. Design of a visualization support tool for the representation of multi-dimensional data sets. In *Proc. 5th Eurographics Workshop on Visualization in Scientific Computing*, Rostock, Germany, 1994.

[8] S. Lange, U. Rauschenbach, and H. Schumann. Alternatives for the presentation of information in a mobile environment. In IMC'96 [30].

[9] W.R. Smith. The Newton Application Architecture. In *Proc. IEEE Computer Conference*, San Francisco, 1994.

[10] M. Minsky. A Framework for Representing Knowledge. In P.H. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, New-York, 1975.

[11] R.B. Roberts and I.P. Goldstein. The FRL Primer. AI Memo No. 408, Artificial Intelligence Laboratory, MIT, Cambridge, Mass., 1977.

[12] Y. Shibata and M. Katsumoto. Dynamic Hypertext and Knowledga Agent Systems for Multimedia Information Networks. In *Proc. Hypertext'93 (November 14–18 1993, Seattle, Washington)*, pages 83–93. The Association for Computing Machinery, 1993.

[13] T.W. Malone, K.-Y. Lai, and Ch. Frey. Experiments with Oval: A radically tailorable tool for cooperative work. *ACM TOIS*, 13(2):177–205, April 1995.

[14] T. Kirste. A flexible presentation model for distributed information systems. In *Proc. 4th Eurographics Workshop on Multimedia (EGMM'96)*, Rostock, Germany, May 28–30 1996. accepted for presentation.

[15] D. Ungar and B. Randall. Self: the power of simplicity. In *Proc. OOPSLA'87 Conference*, pages 227–241, Orlando, Florida, 1987. Published as *SIGPLAN Notices 22*, Dec. 1987.

[16] J. McKeehan and N. Rhodes. *Programming for the Newton*. AP Professional, 1994.

[17] G.L. Steele Jr. *Common Lisp: The Language*. Digital Press, second edition, 1990.

[18] R.C. Waters. GPRINT: A LISP Pretty Printer Providing Extensive User Format-Control Mechanisms. AI Memo No. 611, Artificial Intelligence Laboratory, MIT, Cambridge, Mass., 1981.

[19] T. Kirste. INCH – *Ein Konzept für* Interactive Computational Hypermedia *auf der Basis einer funktionalen Objektinterpretation*. Shaker Verlag, Aachen, Germany, 1995. ISBN 3-8265-0748-7.

[20] T. Kirste. The MOVI Project – Introduction and Modeling Concepts. In IMC'96 [30].

[21] T.A. Funkenhouser and H. Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proc. COMPUTER GRAPHICS Annual Conference*, 1993.

[22] S Lange. Visualisierungsentscheidungen unter Berücksichtigung verfügbarer Ressourcen. *Rostocker Informatik-Berichte*, 17:19–34, 1996.

[23] H. Theisel. Automatische Auswahl geeigneter Visualisierungstechniken für allgemeine wissenschaftliche Datensätze. Master's thesis, Rostock University, Dept. Computer Science, 1994.

[24] S Wehrend and C. Lewis. A problem-oriented classification of visualization techniques. In *Proc. Visualization '90*, San Francisco, 1990.

[25] G.M. Nielsen et al., editors. *Visualization in Scientific Computing*. IEEE Computer Society Press, Los Angeles, 1990.

[26] J.L. Encarnação, M. Frühauf, and T. Kirste. Mobile Visualization: Challenges and Solution Concepts. In *Proc. Fifth International Conference on Computer Applications in Production and Engineering (CAPE'95)*, Beijing, China, May 16–18 1995.

[27] T. Watson. Application design for wireless computing. In MCSA'94 [29].

[28] NCSA. Mosaic. http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html.

[29] *Proc. Workshop on Mobile Computing Systems and Applications (MCSA'94)*, Santa Cruz, CA, December 8–9 1994. IEEE Computer Society.

[30] *Proc. Workshop on Information Visualization and Mobile Computing (IMC'96)*, Rostock, Germany, February 26–27 1996.