Report No. 4
December 1993

INTUITIVE TOOLS FOR INFORMATION RETRIEVAL

– Requirements and Architecture

Ulf Wingstedt
Peter Rosengren
Marie Bern
Peeter Kool

SWEDISH INSTITUTE FOR SYSTEMS DEVELOPMENT

SISU

Intuitive Tools for Information Retrieval – Requirements and Architecture

Ulf Wingstedt
Peter Rosengren
Marie Bern
Peeter Kool

Swedish Institute for Systems Development (SISU)
Electrum 212, S-164 40 Kista, Sweden

ulf@sisu.se
peterros@sisu.se
marie@sisu.se
kool@sisu.se

Abstract

This report focuses on end-user tools for information retrieval and addresses requirements on their functionality and user interface as specified in the Intuitive Project. The purpose of the end-user tools is to make it possible to use multimedia information stored in a set of databases. The tools will support the user in the following important subtasks of information retrieval: Navigation, Selection, Browsing, and Presentation, providing the user with an overview of available information, support for query formulation as well as seeking by inspection and understanding of results.

Publisher: Thomas Falk

Editor-in-Chief: Janis A Bubenko Jr

Editorial Advisory Board: Eva Lindencrona, Björn Nilsson, Benkt Wangler

Editorial Production Manager: Yngve Pavasson

ISSN 1103-1700
ISRN SISU-REP--04--SE

Contents

1. Introduction

This report is the second in a series of SISU reports from the Intuitive Project. The Intuitive Project is an ESPRIT III project in which SISU is a participant. The objective of the Intuitive project is to provide efficient and easy to use solutions for end-users to access heterogeneous information sources.

The reports document the work carried out by SISU during the first year of the Intuitive Project. Two other SISU reports are currently available; Rosengren et al [Rosengren93c] give an overview of the Intuitive approach while Bern et al [Bern93] report on two prototypes being built. The Intuitive Project has also been described in two published research papers where Rosengren et al place a special focus on the use of ER models for information retrieval [Rosengren93a] and experiences from prototyping [Rosengren93b].

The four key components in the architecture of the Intuitive System are:
- Multimodal Interaction Manager
- End-user Tools
- Intelligent Dialogue Manager
- Data Access Layer

The role of the Multimodal Interaction Manager is to allow the user to input his request with a combination of speech and pointing. The End-user Tools provide the user with a visual interface and functionality for the retrieval of information from large heterogeneous databases. The Dialogue Manager is responsible for interpreting user requests according to the user task and initiating appropriate actions in the system. Finally, the Data Access Layer is the glue that binds the different information resources together, providing the end-user tools with data.

This report focuses on the end-user tools and addresses requirements on their functionality and user interface. The aim of the end-user tools is to make it possible to use multimedia information stored in a set of databases. The four main objectives to be met by these tools are:

- Provide an overview and understanding of available information.

- Support the user in expressing his information needs as a query to the system. The tools should provide a short mental distance between users' information needs and expressing a query with the tools.

- Help users understand and interpret the results from querying the databases.

- Provide a seamless interaction allowing the users to switch between different operations in a non-obtrusive way.

The end-user tool's knowledge regarding the underlying databases will be based entirely on conceptual models. The idea of conceptual models is to capture and represent the semantics of data stored in a database in terms familiar to users, not in terms of the underlying data structure. The use of conceptual models will thus provide increased usability as well as shield the tools' implementation from specific database technologies.

A variety of information types will be supported by Intuitive, such as pictures, drawings, video, and business data (structured data). These information types will reside in a set of databases which are integrated through the data access layer of the Intuitive system architecture. As Rosengren et al explain [Rosengren93c], the Tools are built around a 3-schema level architecture - database level, conceptual level and presentation level, see figure 1.1.

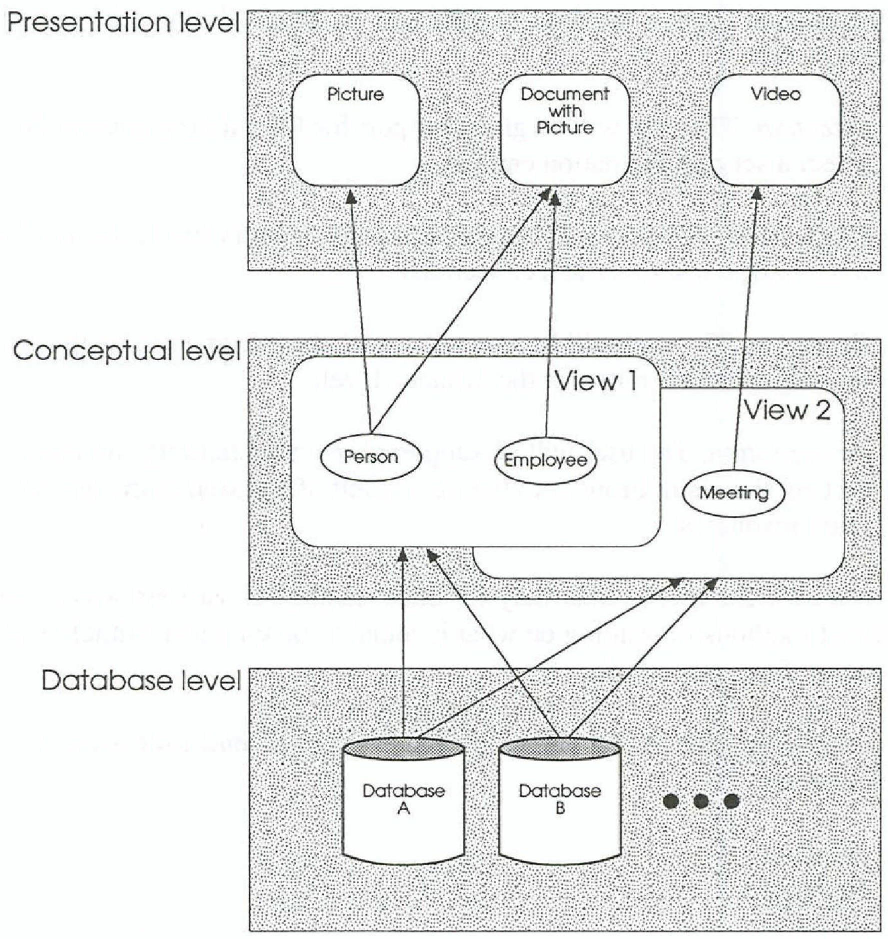


Fig. 1.1 The three schema levels of the architecture

At the *database level* each database is described by a logical database model and has a specific access language. Examples of databases include a relational database with SQL, a document category system with free text search and a picture database with keyword searches. The *conceptual level* constitutes the conceptual modelling of real-world objects within a particular application. Information about an object (or *entity class*) at this level might be distributed over several databases; for instance, information about a Project Entity could be project data stored in a relational database, the project description found in a full text database, and a picture with project participants in a picture database.

At the *presentation level*, presentation aspects such as visual appearance and sequence of presentation, are described. Depending on a user and his task, a Project Entity can be presented at different levels of detail and with different types of presentation methods depending on the presentation model for that entity class.

The end-user tools will support the user in the following information retrieval tasks:

- *Selection.* The user will be given support for formulating queries that select a set of information entities.

- *Navigation.* The user will be provided with an overview of the available information resources at a conceptual level.

- *Browsing.* The user will be assisted in seeking information by inspecting information entities at the instance level.

- *Presentation.* The user will be supported in understanding and interpreting the result brought to him as a result of a search in the information resources.

The support for these tasks may be implemented in various ways, using different methods depending on what is found to be the most suitable implementation for a given task.

To support the user in the above tasks four types of end-user tools will be implemented:

- Selector
- Navigator
- Browser
- Presenter

These tools will be general tools, which means that they can be applied in different applications for different sets of databases without reprogramming. Intuitive should therefore be quick and easily customisable. This requirement will be met by the Tools since they will read all relevant domain and application specific information such as conceptual model structures and graphics from the Intuitive Dictionary and configure themselves according to this information. These issues are discussed in more detail in Bern et al [Bern93].

The different tools will provide a seamless manner of interaction where the user can switch between different subtasks of the Information Retrieval process. The tools will have specific graphical interfaces and may be controlled by mouse, voice or keyboard. They will also be able to be controlled by the Dialogue Manager. In this way Intuitive will provide a *mixed-initiative* style of interaction, i.e. the tools may be controlled by either the user or the system.

The Tools will be built using a modular architecture providing several advantages:

- Flexibility, which makes it possible to change parts of the Tools layer even at a late stage in the project. For instance, if a major research breakthrough is made in the area of pictorial query languages we will be able to exploit this by simply changing the Picture Selector of Intuitive, without needing to change the whole Tools layer.

- The complexity of implementation is broken down into several smaller well-defined pieces.

- The various subtasks in the user's information retrieval process are reflected by software modules that support the subtasks. This will allow the user to handle complex information retrieval problems.

- The architecture is extendable, which makes it possible to add new tools and support for new types of data if necessary. It also meets the requirement that Intuitive should be configurable to meet different demands from different users. Some users may only want support for Pictures and Business Data while others may want support for a whole range of information types.

The report is organised as follows:

Chapter 2 discusses navigation issues and identifies the need for a user to look at a conceptual map at different levels of detail. Three types of view of a conceptual model are described. The conclusion from the views discussion is that the Navigator Tools must be highly flexible with regard to definition and selection of views.

Chapter 3 discusses various aspects of visual query systems are discussed, leading to requirements for the Selector Tools.

Chapter 4 discusses browsing in data spaces. The problem of getting an overview of large quantities of data is analysed in detail and the difference between a Fish-eye View and a Bird's-eye View approach to browsing is described. Thumbnail representations for different information types are discussed.

Chapter 5 addresses the need to support the user in understanding a query result. The use of background maps for the presentation of results is discussed and the extent to which this technique is dependent upon a specific application is analysed. A need to have presentation models is identified. Presentation of links to other data items and the kind of link marker the Presenter Tools should use are also analysed in this chapter.

The internal architecture of the tools is presented in Chapter 6.

Chapter 7, finally, outlines our future work.

2. Navigator

The purpose of navigation is to enable the user to express his or her information requirements by providing an overview of the available classes of information, their relationships, and their semantic definitions. Intuitive tools will include support for the navigational task in order to guide and encourage the user to explore data abstractions and meta data to obtain a general understanding of the database contents. The specific tool responsible for this support is called a Navigator.

In literature, as for instance in the SNAP system [Bryce86], the task of Navigation is often called *Schema Browsing*. Within Intuitive, however, we will use the term Navigation when considering abstractions of data, and restrict Browsing to actual database instances.

Within Intuitive, the support for Navigation is based on conceptual models having at least the same level of semantic expressiveness as traditional ER models, although we envisage a need for more expressive models. An extended ER approach is therefore considered, that includes the notion of generalisation/specialisation as well as aggregation (partly). In particular, we will use ER concepts developed by the Tempora project [Loucopoulos91] in order to achieve the required expressiveness.

Navigation within the available information resources at an abstract level requires that the conceptual model describing the information resources can be visualised for the user.

As is recognised (by for instance the developers of the SNAP system), the single most important component in a conceptual model management system is the visual presentation of the model. The basic approach for visualisation of the conceptual model in an Intuitive Application is a variant of ER diagrams with sufficient expressiveness. However, in specific Intuitive applications, other visualisation techniques may be used if regarded as more suitable according to user characteristics, the task to support, or the organisation in which the application is used. For instance, maps and pictures may be used for visualisation. Two alternative methods of visualising a conceptual model can be seen in figures 2.1 and 2.2, where icons are used in figure 2.1 to illustrate different concepts.

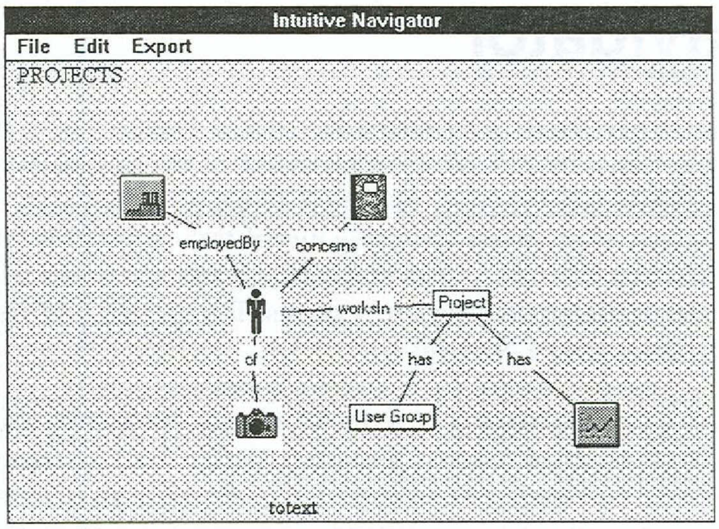


Figure 2.1 Navigator with conceptual model visualised using icons.

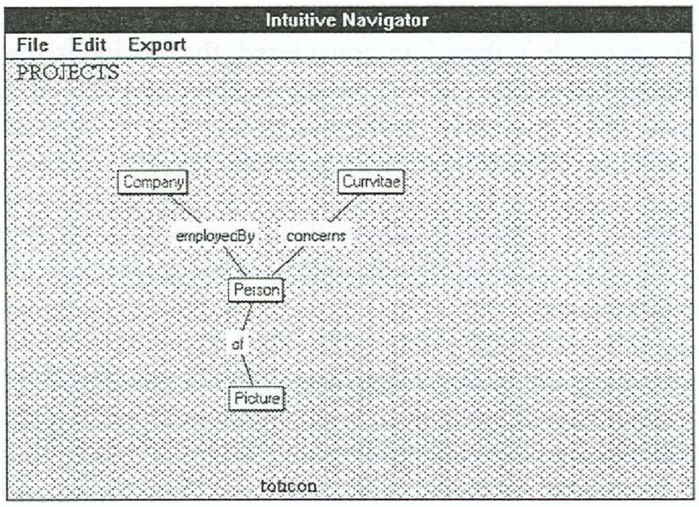


Figure 2.2 Navigator with conceptual model visualised using boxes.

2.1 Views

A visualisation of a conceptual model, an ER diagram (or ER schema), often includes a vast number of schema items such as entity and relationship classes to describe the information resource. In order for the Navigator to support the user in locating the schema items of interest, it must support a division of the complete ER schema into various sub-views.

The grouping of schema items into views should be guided by requirements according to domain and user characteristics such as tasks and skills. It should be possible to create not only application general views common to all users, but also special views for groups of users as well as user-defined views.

The overall conceptual model may be divided into views in three main ways, one horizontal and two vertical view types. Firstly, there may be horizontal views at a specific level of abstraction and detail but showing different parts of the model. The sub-part of the conceptual model included in the view is a *conceptual unit*, i.e. it consists of entity classes that belong together semantically.

Secondly, there may be vertical views that for a specific part of the model show a different number of details. The conceptual model will include a large amount of information that may not be of the same importance to all users at all times. The user should therefore be able to hide and show details.

Thirdly, the model may be divided into several levels of abstraction. Entity classes at a lower level of abstraction may be aggregated into new entity classes at a higher level. Consider, for example, entity and attribute classes such as Door, Wheel, Trunk etc, that may be seen at a specific level of abstraction. At the higher level, these (and others) may be aggregated into the entity class Car. Entities based on aggregation of other entities may be derived (computed) in run-time, i.e. they are not required to be explicitly stored in the information resource.

The important task of designing the application general views will be performed by the Intuitive designer.

In order to support user-defined views, the Navigator will include functionality that allows the user to pick schema items from existing views (e.g. the Complete Model view) and include them in the new view. The user should also be able to change the layout of the view, placement of schema items, choice of visualisation icons etc.

The above requirements suggest a highly flexible tool from the point of view of definition and selection of views. However, all applications will probably not supply all this functionality for all users, but rather use a predefined set of general views.

2.2 Control

In cases where the visualisation of the model, the view, is too large to be viewed in a single piece, the Navigator will provide functionality such as scrolling and panning. In addition, a zooming function will be provided in order to show an enlargement of a part of the visualisation.

While navigating in the conceptual model, the user should always have the opportunity of relating the current visible sub-part of the model to the overall context.

2.3 Queries About Data

Keeping in mind the large information resources that will be supported by an Intuitive system, we believe that users regularly need to query not only for instances from the databases, but also for meta data, i.e. data about the instances. Such a query might be "What information do we have about Patients?" where the answer might be a list of attributes for the entity class Patient: Name, Age, Treatments, Symptoms, Pictures.

The Navigator should also support the user in finding certain entity classes, relationship classes, attributes etc, i.e. finding the view(s) where they appear.

3. Selector

Since the Intuitive System is targeted for general access to corporate information resources we can expect that the number of infrequent database users will be high. The ease of use and naturalness of the query language provided by different tools is of therefore utmost importance.

Several recommendations for the design of database user interfaces have been documented: Elmasri and Larson identify the following requirements for a user-friendly database interface [Elmasri85]:

- The interface should not assume that the user knows the contents of the database or the exact procedure to formulate a query.
- The interface should allow the user to view the database in the way he finds most comfortable when formulating a particular query, and to proceed through query formulation differently.
- Formulation of queries that only differ slightly should be possible without having to perform all major reformulation steps.
- The query formulation mechanisms should be powerful enough to allow formulation of complex queries.

As discussed by Rosengren et al, we propose to base the Intuitive User Interface as far as possible on concepts familiar to the user allowing him to express his information needs in terms of concepts used in his daily working life [Rosengren93c]. This means that the user will perceive querying Intuitive as selecting information entities of interest, rather than selecting tuples in a table as is often the case in traditional systems. From the Hybris project our experience is that this type of metaphor is easy to comprehend, at least in business data applications [Karlgren91], [Lundh89], [Sahlin90]. We also believe that this will be true for other types of data that can be modelled and stored in relational databases.

The selection problems are closely related to the navigation problems described in the previous chapter. A user cannot make selections unless he has a clear view and understanding of the information content in the database. It might be natural to consider integrating these two types of interface and letting the user express his query directly by pointing and clicking in a conceptual map.

The other choice is to have two separate interfaces - one that only visualises the conceptual model and a separate query interface.

The separation of the query interface from the navigation interface has three implications:

- It is possible to express more semantics in the conceptual model. With a combined query/navigation interface the user will suffer from information overload if there are too many symbols expressing relationship, cardinalities, is-a hierarchies etc, while the user is trying to formulate a query.
- The visual query language can be made more powerful and allow the user to express complex types of queries.
- The use of the tools can be more complicated and require more inter action steps for the user, unless the interaction is carefully designed.

The combined query/navigation approach has been used in Hybris [Lundh89], where a simple conceptual model is visualised, showing only basic relationships between entities. The user can query the database by directly selecting entities in the ER diagram and then making restrictions. This has proved to be a solution that is easy to use, although more complex queries are difficult and in some cases even impossible to formulate.

For Intuitive we believe that it is wise to start with the assumption that the navigation interface and query interface are separate.

3.1 Graphical Query Interface

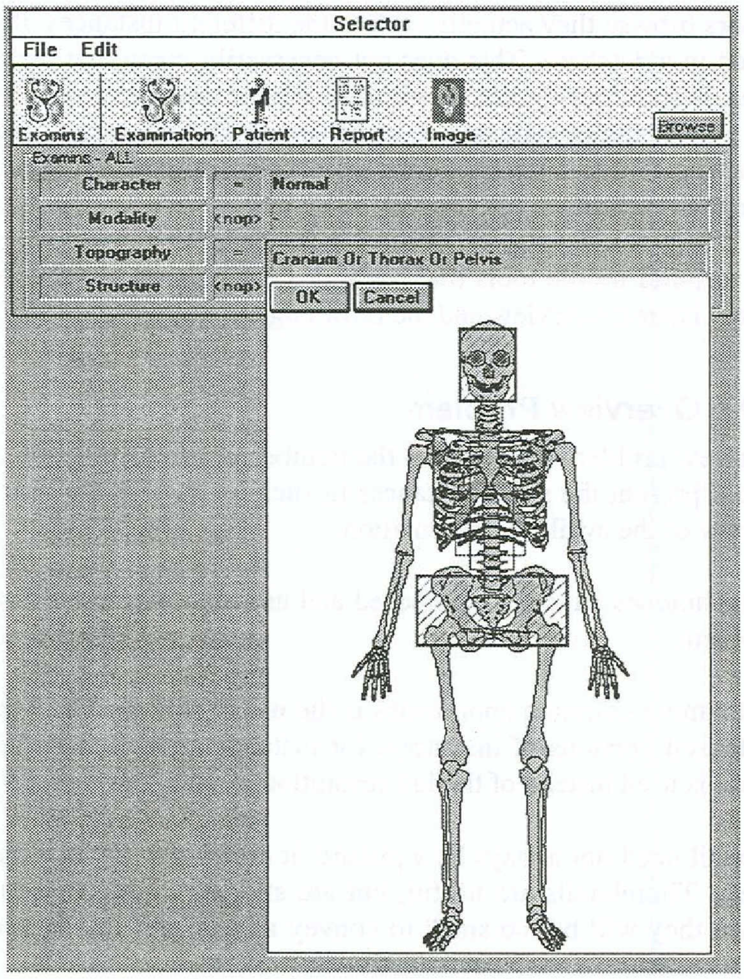
Two types of graphical query interface can be considered - *diagrammatic* or *iconic* [Catarci91]. The diagrammatic approach means that the user expresses his queries directly in a diagram representing the conceptual model. An iconic interface means that the user forms a language expression by choosing different icons representing not only entities and attributes but also operators and other language constructs.

The iconic approach implies more work when customising Intuitive for a specific application. Icons have to be chosen or designed for this particular application. Also, many information entities represent abstract concepts which may not be easy to visualise. We therefore choose the diagrammatic approach initially, but since the Selectors are modular we will have the option of switching to an iconic approach later. The iconic language approach might also work better with the multi-modal style of interaction.

A requirement is that the graphical query interface of the Selector should only allow syntactically correct queries to be formulated.

Although the main purpose of the query interface is to allow formulation of ad-hoc queries, requirements elicited from real world applications have shown the need for adaptable predefined queries [Bern93]. The Selector will therefore support the inclusion of such queries where the user can supply control parameters when querying at run time.

The query interface supports value domains and also hides complex numerical codes from the user. One example is the Medical Demonstrator where codes for representing different parts of a body such as CRA for Cranium [Bern93] are stored in the database. In this case, the system will allow the user to input the full name 'Cranium' instead of the code 'CRA'. An alternative method of input could be to point at various parts in a picture of a skeleton, see figure 3.1 below:



Examins - ALL

Character	=	Normal
Modality	<nop>	-
Topography	=	Cranium Or Thorax Or Pelvis
Structure	<nop>	

Figure 3.1 Pictures can be used to represent value domains.

4. Browser

The role of the Browser is to allow the user to browse at an instance level as opposed to the conceptual level. In this way our tools support the user in *understanding* the database's content both with navigation and browsing. Note, however, that in a multimedia database, browsing is also an important technique in searching for information since users seem to be using this technique intuitively. Selection of a picture from a picture library usually involves selecting a subject and then browsing the picture instances in order to find the one sought.

When users browse they actually look at the different instances of information stored in a database. This does not necessarily mean that users look at real items, it could mean that they look at a representation of the items, for instance picture thumbnails or document icons. The important thing is that there is a one-to-one mapping between what you see when browsing and what is in the resource.

In order to build usable tools that support the browsing task there are two general key issues: overview and the browsing structure (ordering).

4.1 The Overview Problem

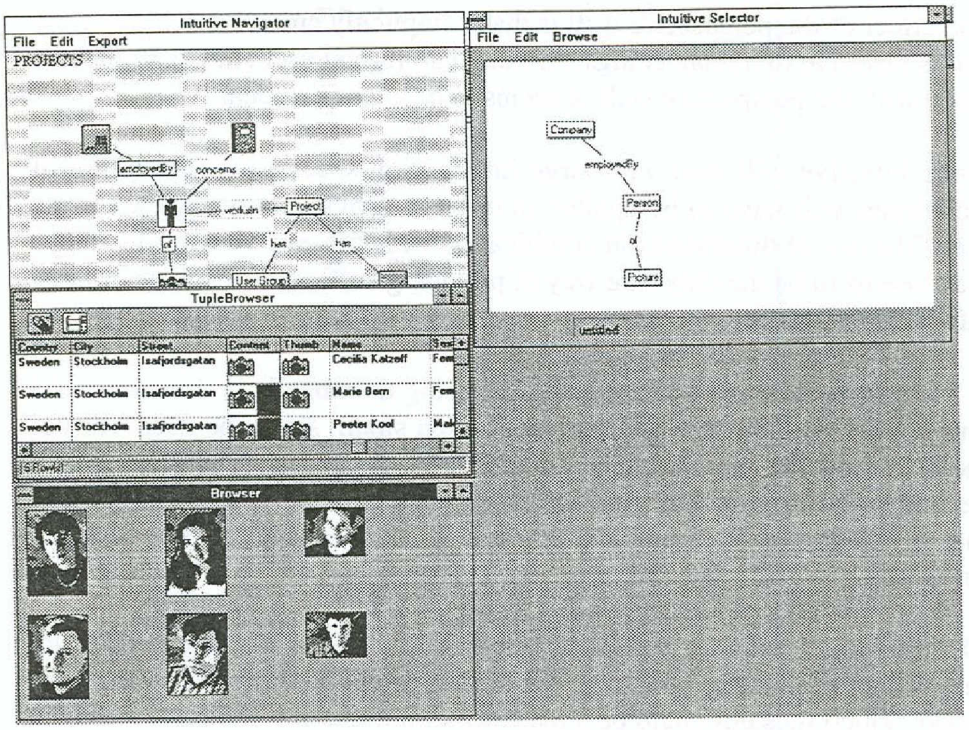
The overview problem increases as the number of instances grow. The system should present the set of instances in such a way that the user is given an overview of the available information.

Several techniques have been proposed and used in order to solve the overview problem.

One of the more common approaches is the use of thumbnails. Thumbnails are condensed versions of instances, for instance a smaller picture (lower resolution) is used instead of the full resolution picture, see figure 4.1.

A thumbnail need not always be a picture, it could also be an icon, a piece of text etc. Thumbnails are useful, but are still not enough when sets are large since they will be too small to convey meaningful information to the user.

Thumbnails can be used in combination with other techniques for improving the overview. For instance, there is the possibility to use a "Bird's-eye view", [Erickson91], showing the position of the current (displayed) instances in the set. This gives the user feedback on the size of the set and the current position in that set. This is useful when it is impossible to display the whole set.



Country	City	Street	Content	Thumb	Name	Sex
Sweden	Stockholm	Isafjordsgatan			Cecilia Katzeff	Fem
Sweden	Stockholm	Isafjordsgatan			Marie Bern	Fem
Sweden	Stockholm	Isafjordsgatan			Peeter Kool	Mal

5 Row(s)

Figure 4.1 The use of thumbnails can provide the user with an overview of large data sets. In the lower left corner a Browser displays thumbnails of pictures.

Fish-eye views [Furnas86] are another proposed technique. The fish-eye view works in the same way as a fish-eye camera lens. The parts that are visible in the middle of the picture are larger than the parts in the peripheral. In this way the overview is maintained without losing too much detail. A problem with pure fish-eye views, however, is that the views are distorted.

A very interesting implementation of a sort of fish-eye view is the Perspective Wall [Mackinley 91] and Cone Trees [Robertson91] that are part of the Information Visualizer [Card91]. These implementations use 3-D animation to create the views.

The Perspective Wall is especially interesting since it represents a fish-eye view of an ordered list. It gives the user feedback on the set size and current position. The metaphor used in the Perspective Wall is a wall represented in 3-D with three surfaces (figure 4.2) on which the items are placed.

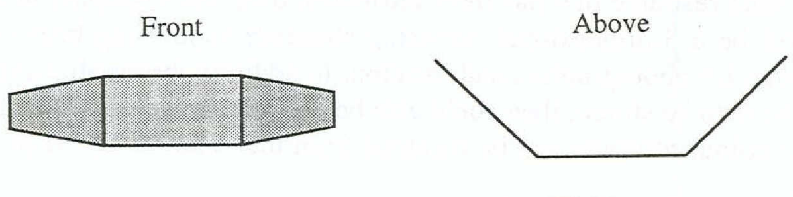


Figure 4.2: The Perspective Wall

The effect of the perspective wall is that it implicitly creates a fish-eye view that is intuitive to human beings. Although no formal user studies have been made with the perspective wall, it seems to be a feasible idea.

Since Intuitive will deal with large information spaces, a combination of techniques will have to be applied in order to provide an overview for the user. The use of thumbnails in combination with fish-eye view or bird's-eye view seems to be the only safe way of providing sufficient overview of large information spaces.

A specific problem in picture browsing using thumbnails is the size of the thumbnails. The thumbnails must not be too small, since the user will use them to find the pictures. The smallest size for picture thumbnails will depend on the application. An X-ray thumbnail might have to be larger than that of a car picture thumbnail in order to convey enough selection information.

The thumbnail used for videos in current applications is a descriptive picture. There might also be a need for time information, i.e. the length of the video. Video mosaics could be a reasonable solution in some cases. A video mosaic is a collection of videos that are played simultaneously without sound. The video showed in a mosaic should be compressed in both size and time in order to give an acceptable overview.

4.2 Browsing Structures

Browsing through items in a set implies that some kind of browsing structure is used. Examples of browsing structures are:

- No structure at all. One or more items are arbitrarily brought to the user upon request.
- Some sort of spatial orientation. For instance, when the user moves to the left, pictures become more and more blue.
- Hierarchy. Consider, for instance, a document database that uses a hierarchical subject model.

In general, browsing structures will have *dimensionality* depending on the number of attributes that are available for each instance. For example, a picture data resource that has three attributes: date, photographer and location, will be a 3-dimensional browsing structure supporting browsing by date order, by photographer, and location. In addition, browsable attributes do not have to be stored, they could also be deduced in the same way that an average colour attribute can be deduced from the value of all pixels in the picture.

Using n-dimensional browsing structures will introduce an orientation problem or, to be more exact, what degree of flexibility the system will allow the user for moving arbitrarily along any dimension. There are three ways in which a dimension can work:

- Movable: The user can "scroll" freely back and forth along this dimension.
- Fixed: The dimension is set to a fixed value, for instance photographer = "Linda Cartney"
- Any: When it is not used in the browsing structure.

Visualising more than three dimensions simultaneously is very complex and may cause usability problems. It is suitable therefore to set the absolute maximum number of movable dimensions to three. In fact, only user testing can resolve the maximum number of movable dimensions that can be of practical use; it might even be less than three.

The problems of building meaningful general browsers that do not use a browsing structure at all are probably outside the scope of Intuitive. Instead, Intuitive will require that all resources have some kind of browsing structures either explicit or deduced.

Usually, structures that can be used for browsing are already contained in the information resource. The typical case today is that the data are ordered in a list or in a hierarchy directly in the information resources. But it is questionable as to whether these structures are optimal for the browsing process. The result of a user task analysis for a specific application will determine which browsing structures are necessary.

Since a user of our system navigates and formulates queries at a conceptual level, there is also a need for the Browser to explain the result in conceptual terms. This will allow the user to have a better understanding of the result and increase the usability of the system. For instance, if a user asks for information regarding projects, persons and organisations, the answer should be brought to the user in terms of those entity classes, and not using the codes of the databases as is often the case in current commercial products.

Several attempts to build browsers that reflect the conceptual structure have been reported. One of the early attempts was LID "Living in a database" [Fogg84]. In LID, browsing was the *main* technique for database querying. We do not think that this is a realistic strategy in real databases mainly for reasons of performance. In industrial applications, the number of instances of an entity class will be too large to view on the screen. Browsers in our system therefore work mainly on limited data sets, but it is still possible for the user to work according to LID principles.

Each of the entity classes in a result set is rendered as visually separated objects in the Browser user interface with the relationships connecting them together. The exact rendering depends on the type of data or media, pictures are rendered as a set of thumbnails, business data as tuples etc. The interaction style is an extension of *Synchronised Browsing* in the Pasta-3 system [Kuntz89]. Synchronised browsing occurs when more than one entity is browsed and the entities are associated with relationships. The main idea is that the user selects one entity from an entity class and can see all the related entities in other entity classes.

5. Presenter

The way in which retrieved results are presented has a great impact on two specific stages in the user's information retrieval process - (1) *evaluation* of results, and (2) *refinement* of needs.

It has been shown that the evaluation phase of an information retrieval process is decisive for whether a query is successful or not [Katzeff89]. When interpreting results of the query, users not only evaluate whether the results fit their information needs, but also whether they have formulated a query that correctly corresponds to their information needs. The system's response should display the result in a way that supports these two types of evaluation in the decision process. Moreover, the form of presentation should be congruent with the user's expectations within the task being undertaken.

In our system the main task for the *Presenter* is to support the understanding of the information retrieved. As explained by Rosengren et al [Rosengren93a] [Rosengren93b], the Dictionary contains a presentation model. The Presenter decides on how to present an entity based on the presentation model and user preferences.

In addition, the Presenter is responsible for the visualisation of hyperlinks, see [Rosengren93c] for an explanation of this feature. We are combining two different methods of link visualisation. The first link type is called *overlay links* and is visualised as a colour indicated area in the entity presented, e.g. a picture. The second link visualisation method is to use *link buttons* with descriptive names.

It is an important user requirement that the Intuitive tools support the export of retrieved data into external tools such as word processors and spreadsheets. Besides having default presentation capabilities, the Presenter will therefore allow the export of Intuitive data formats to other programs to enhance the usability of the information retrieved.

5.1 Understanding of Query Results

The user should be able to read a text and see a picture easily, not merely see symbols that represent the data instance. This means that functionality to present the data in full resolution is needed. Presentation models that display the relationships between data are also needed. Further, to enhance the understanding of the selected information, presentation models that display relevant data connected to the selected concept are needed.

Kerner and Thiel [Kerner91] suggest the *box presentation* method for supporting the user in the evaluation of results. The box presentation method supports the understanding of a retrieved data instance by presenting the information requested in a context of other related data not explicitly asked for in the original query. The positions of data instances within the box express the relative importance of the information. In western civilisation information is read from top left to bottom right. The most important information is presented in the upper left corner of the presentation area. The concept to be focused on should therefore be displayed in this corner.

Another solution is to present the result of a query in the context of the query itself. The information "15 000" has for example little meaning in isolation. If the user knows that this was the result of the query "What is the salary for Mr. X?" the information is meaningful. To help the user form a mental view of the context in which the information is presented, the query retrieving the information should always be visible in parallel with the presentation.

An alternative method of visualising the context of the result is in a map. The map should represent something static and known to the user, a context in which the user could perceive the result. The map should be visible to the user during the complete work session with the system.

The best context in which to present a result of a selection is however not the same for every result in the system. It must be possible to change context for the presentation. It is also central that the context corresponds to the specific task the user is dealing with. How maps can be used for presentation purposes is an issue for further work.

5.2 Link Markers

An important issue to consider is what link marker the system should use. A link marker is a visible or audible indicator in the node to mark the presence of a link. Some examples are:

- Data that serve as source or destination for links are highlighted or encircled. No information about the type or destination of the link is shown.
- Data that serve as source or destination for links are highlighted or encircled, but only when the cursor moves over the area. No information about the type or destination of the link is shown.
- The source and destination for links are marked by icons. No information about the type or destination of the link is shown.
- The source and destination for links are marked by icons. The icon contains information about the type and destination of the link.

A Presenter should be able to handle all of these types of link visualisation. Moreover, since the links will be named, a pop-up menu can be used for link access.

The Dialogue Manager can use information in the Dictionary regarding users and tasks to enable the showing of, among many different links, the links relevant for a specific user in a certain situation.

The most difficult question concerning hyperlinks however, is how the user gets an overview of the nodes connected by the links [Katzeff92]. The purpose of the hyperlinks is to encourage the user to explore the data. To encourage the user successfully, the user needs to be secure in his interaction with the system. Quick, easy back-tracking and help in maintaining the global sense of location during link traversal are important issues to consider. As a consequence, when an instance is displayed by a Presenter, the Navigator will highlight the entity class to which the instance belongs.

6. Tools Internal Architecture

The development of a generic Tool architecture has been driven by the following design goals derived from requirements on usability, exploitation, implementation and maintenance:

- *Automatic configuration at run-time.* Tools' appearances will be configured automatically from Dictionary information at run-time. This means that it is possible, without programming, to change tools' appearance, feel and behaviour by changing the contents of the Dictionary.

- *Allowing external program control.* It should be possible for other software components of the Intuitive system to control the tools' behaviour at run-time.

- *Modularity at Tool level.* As shown by Bern et al, the set of tools actually used may vary from application to application [Bern93]. For instance, if video is to be incorporated in an application, a Video Presenter should be able to do this without changing other tools.

- *Look and feel independence.* It should be possible for a certain tool to have variable layout, within limits, in the user interface in different applications. This implies that program code for user interface graphics should be separated from the tool's functionality code.

- *Reuse of tool sub-modules in new tools.* Parts of a tool's functionality may be reused in new tools. These modules should be constructed in a general way to allow reuse.

The first two goals require tools to respond dynamically at run-time to requests from other programs or changes in the Dictionary. Tools should therefore provide mechanisms for retrieval of Dictionary information and a messaging protocol for external control.

The other three represent developers' requirements for changeability and reuse. Modularity at the tool level implies that different tools used in an application should be independent of each other, while look and feel independence implies that the user interface should be independent of functionality within a tool. In addition, a tool should as far as possible be divided into functional modules that may be reused in the construction of new tools.

To summarise the design goals, we see requirements on flexibility in two separate phases, namely at run-time and design time (development time). The architecture presented below has been designed to fulfil these requirements.

The generic architecture for Tools is described in the following section. The generic architecture will be used for each tool.

6.1 Functional Modules

All tools consist of one or more Functional Modules (core functionality) that are separate functional units that handle a certain aspect of functionality needed in the tool.

The functional modules are divided into three main parts as shown in figure 6.1 below.

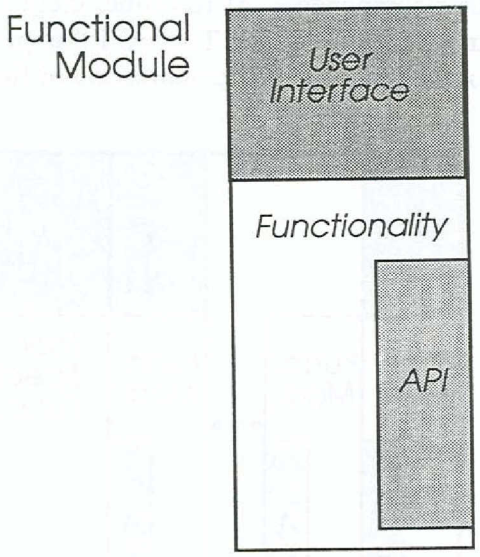


Fig. 6.1 Functional Module architecture

Many of the functional modules used in the Intuitive tools will handle some user interaction aspect such as providing a visual query language. A functional module may thus have a full-featured graphical user interface.

In order to support developers in changing the look and feel of the user interface, it will be separated as much as possible from the code implementing the actual functionality. A call-back structure from interface events to functions will be used.

Finally, every functional module also has an API (Application Programmer's Interface) allowing developers to integrate the module with other software components (including other functional modules) into a Tool.

6.2 Generic Tool Architecture

A developer's view of the generic Tool architecture is described in Figure 6.2. A tool is assembled from a set of functional modules with an optional user interface portion. The glue holding the tool together is the code in the Tool Specific Functionality part, which integrates the various functional modules and also implements additional non-general functionality including additional user interface.

The functional modules should be treated as "black boxes", i.e. the internals of the modules should be of no concern to the developer of a new tool. Rather, the developer will use the API's from the functional modules for access to their functionality.

The use of functional modules in a tool will be hidden from the other components of an Intuitive system, external to the tool. When developing a new tool, the developer has to provide a specific Tool API that includes all functions required by other components. At run-time, these other components only need to have knowledge about the Tool API that provides a coherent access mechanism for the tool's specific combination of functionality.

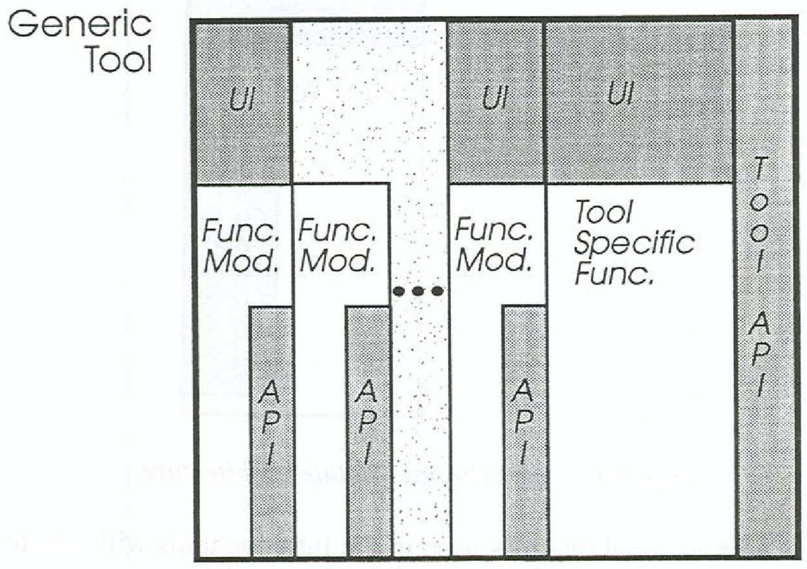


Fig. 6.2 The Generic Tool Architecture; the developers view.

A run-time view of the general architecture is shown in figure 6.3, where the data flow within a Tool and between a Tool and other modules of the Intuitive system is described.

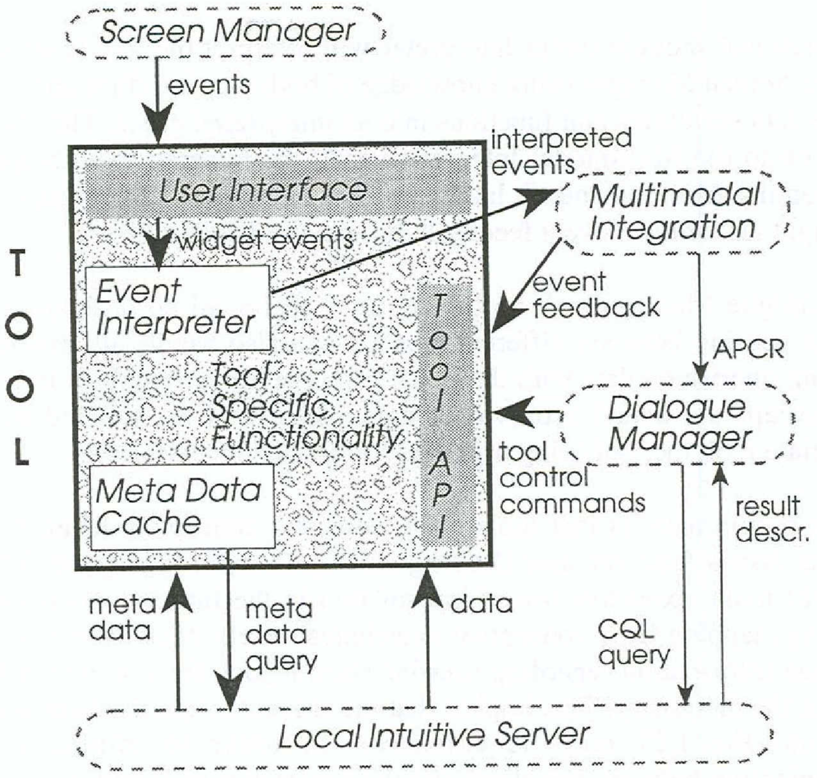


Fig. 6.3 Data flow within a Tool and between a Tool and other modules of the Intuitive system.

The two white boxes, i.e. Event Interpreter and Meta Data Cache, internal to the tool, are examples of functional modules. In fact, these two modules will appear in all four tools Navigator, Selector, Browser, and Presenter.

The first Intuitive component collecting user events from mouse and keyboard is the Screen Manager. It interprets gesture into events such as *click(x,y)* where x and y are co-ordinates where the mouse was clicked. Also more complex gestures can be resolved, such as *wipe out* (cross over).

The events are passed on from the Screen Manager to the tool. There are three types of event:

- Private events, that are private to the tool and not passed on to the Multimodal Interaction Manager. The tool can give feedback immediately.

- Notifiable events. Feedback can be given directly by the tool on these events. The Multimodal Integration Module is only notified about its occurrence.

- Public events. The tool decides on a default action, but passes the event on to the Multimodal Integration Module.

The functional module Event Interpreter will interpret the low level event from the Screen Manager using knowledge of both user interface and Tools' internal data structures and functions into an *interpreted event*. The event is passed on to the Multimodal Integration (MI) for integration with spoken input coming from the speech hardware. The Multimodal Integration Module might also initiate event feedback through the Tool's API.

The Dialogue Manager will be the central hub for all co-ordination and message passing between different tools. It will also handle all interaction regarding queries on data with the Local Intuitive Server Services Interface (LIS) except the actual retrieval of data which will be handled by the appropriate tool, the tool being triggered by the Dialogue Manager.

A basic requirement is that tools must work at a conceptual level, hiding technical issues from the user [Rosengren93a]. In order to achieve this, all tools will make extensive use of information in the Intuitive Dictionaries regarding mapping from conceptual to technical levels. Moreover, tool configuration information regarding user interface layout, for instance, will be stored in the dictionary. This implies that the amount of queries between the tools and the LIS may be extensive. In order to minimise this communication between the client and the server part of Intuitive, we plan to use a Meta Data Cache in the tools.

7. Future work

An important aspect that needs further research is design methodology. Given an application and a set of users' tasks, application developers will be faced with questions such as, "How do I find a suitable visual language and what visual cues do I need to visualise the underlying conceptual model?" To provide support for fast and efficient construction of highly usable ER-based Information Retrieval Systems we need to develop a design methodology that is easy to follow for application developers [Bern93].

Due to the flexible and modular architecture of our system we will be able to experiment with different approaches for individual tools, that allow us to test different schema visualisation techniques within the Navigator easily without affecting the rest of the system. We will also experiment with different visual languages in the Selector without changing the rest of the system. All these opportunities will be exploited in a series of cognitive and usability tests.

Our further work will also include development and implementation of a presentation model.

References

[Bern93]	M. Bern, P. Kool, P. Rosengren, U. Wingstedt, "Application Design with the Intuitive Tools - Two Case Studies", SISU Report No. 5, December 1993.
[Bryce86]	D. Bryce, R. Hull, "SNAP, a Graphics-Based Schema Manager", Proceedings of the 2nd IEEE International Conference on Data Engineering, pp 151-164, 1986.
[Card91]	S. Card, G. Robertson, J. Mackinlay, "The Information Visualizer, an Information Workspace", Proceedings of CHI´91 Human Factors in Computing Systems, 181-188, New York: ACM, 1991.
[Catarci91]	T. Catarci, A. Massari, G. Santucci, "Iconic and Diagrammatic Interfaces: An Integrated Approach", IEEE Workshop on Visual Languages, 1991, pp 199-204.
[Elmasri85]	R. Elmasri, J. Larson, "A Graphical Query Facility for ER Databases", Proceedings of the 4th International Conference of Entity Relationship Approach, 1985.
[Erickson91]	T. Erickson, G. Salomon, "Designing a Desktop Information System: Observations and Issues", Proceedings CHI '91 Human Factors in Computing Systems, 49-54. New York: ACM, 1991.
[Fogg84]	D. Fogg, "Lessons from a Living in a Database Graphical Query Interface", Proceedings ACM Sigmod, 1984.
[Furnas86]	G. W. Furnas. "Generalized fish-eye views", Proceedings of CHI´86 Human Factors in Computing Systems, 16-23, New York: ACM, 1986.
[Karlgren91]	K. Karlgren, M. Wideroth, "En utvärdering av Hybris", SISU Technical Report 12.

[Katzeff89] C. Katzeff, "Cognitive Aspects of Human-Computer Interaction: Mental Models in Database Query Writing", Doktorsavhandling, Psykologiska institutionen, Stockholms Universitet, 1989.

[Katzeff92] C. Katzeff. "Överblicksproblemet i hypermedia", SISU Technical Report 18, in Swedish, 1992.

[Kerner91] A. Kerner, U. Thiel, "Graphical Support for Users' Inferences within Retrieval Dialogues", IEEE Workshop on Visual Languages, 1991.

[Kuntz89] M. Kuntz, R. Melchert, "Ergonomic Schema Design and Browsing with more Semantics in the Pasta-3 Interface for E-R DBMSs", Proceedings of the 8th International Conference of Entity Relationship Approach, 1989.

[Larson86] J. Larson. "A Visual Approach to Browsing in a Database Environment", IEEE Computer, June 1986.

[Loucopoulos91] P. Loucopoulos, B. Wangler, P. McBrien, F. Schumacker, B. Theodoulidis, V. Kopanas, "Integrating Database Technology, Rule Based Systems and Temporal Reasoning for Effective Information Systems: The Tempora Paradigm", Information Systems Journal, Vol. 1, No. 1, April 1991.

[Lundh89] J. Lundh, P. Rosengren, , "Hybris - A first step towards efficient information resource management", SISU report No. 5, 1989.

[Mackinlay 91] J. Mackinlay, G. Robertson, S. Card. "The Perspective Wall: Detail and Context Smoothly Integrated", Proceedings of CHI'91 Human Factors in Computing Systems, 173-179, New York: ACM, 1991.

[Robertson91] G. Robertson, J. Mackinlay, S. Card, "Cone Trees: Animated 3-D Visualizations of Hierarchical Information", Proceedings of CHI'91 Human Factors in Computing Systems, 189-194, New York: ACM, 1991.