

**Visualisation of domain knowledge: methods and
techniques**

31.07.2003

IST-2001-32429

ICONS

Intelligent Content Management System

www.icons.rodan.pl

Project Partners

Rodan Systems (PL)

The Polish Academy of Sciences (PL)

Centro di Ingegneria Economica e Sociale (IT)

InfoVide (PL)

SchlumbergerSema (BE)

University Paris 9 Dauphine (FR)

University of Ulster (UK)



Visualisation of domain knowledge: methods and techniques

-

Project name	Intelligent Content Management System
Acronym	ICONS
Workpackage	WP3
Task	T2
Document type	report
Title	Visualisation of domain knowledge: methods and techniques
Subtitle	
Document acronym	D12
Author(s)	Jacek Płodzień, Ewa Stemposz, Kazimierz Subieta
Reviewer(s)	Bartosz Nowicki, Sebastan Lisek
Accepting	Witold Staniszkis
Location	I:\WP3 Advanced graphic user interface\ICONS WP3 T2 D12 0106.doc
Version	1.06
Date	31.07.2003
Status	final version after revision
Distribution	public

31.07.2003

History of changes

Date	Version	Author	Change description
31.07.2003	1.05	Jacek Płodzień, Ewa Stemposz	revisions following from comments
28.12.2002	1.03	Jacek Płodzień	document modification
1.10.2002	1.01	Jacek Płodzień, Ewa Stemposz, Kazimierz Subieta, Zbyszek Wróblewski, Marek Mazan, Robert Karaś	document creation

Executive summary

This report discusses a number of issues related to visualisation. These issues include for instance visual data presentation, visual data access, query result presentation. Furthermore, a general architecture of the ICONS prototype is presented.

Investigations concerning selected tools and methods for GUI interfaces concerns selected publicly available software. The requirements for a GUI are based on visual graphical querying metaphors assumed for the ICONS project, which are the result of literature search, research and experiments. It is assumed that three basic metaphors are to be implemented. One of them concerns storing intermediate and final results of query evaluation (*basket*). The others concern navigation between items of information/data: *extensional navigation* in an explicitly presented graph of objects, and *intentional navigation* in an implicitly presented graph of objects representing the UML-like database schema. These metaphors along with other ideas are supposed to solve such problems as: the support for user awareness, recording visual querying results, graph layouts, graph manipulation etc.

Table of contents

History of changes.....	3
Executive summary	4
Table of contents	5
List of figures	5
1. Introduction	6
1.1 Objectives.....	6
1.2 Scope	6
1.3 Relations to Other Documents.....	6
1.4 Intended Audience.....	6
1.5 Usage Guidelines.....	6
1.6 Notation Conventions.....	6
2. The ICONS Prototype GUI: General Ideas	7
2.1 Information Visualisation Aspects in ICONS.....	7
2.2 Architecture of ICONS	9
3. Structural Knowledge Graph Manager (SKGM).....	11
3.1 Introduction	11
3.2 Visual Representation Paradigm.....	12
3.3 Visual Data Access and Query Formulation.....	13
3.4 Query Result Presentation	15
3.5 Usability Improvement.....	16
4. Electronic Forms Paradigm	17
5. Other Modules.....	19
5.1 Knowledge Schema Manager.....	19
5.2 Knowledge Map Graph Manager	20
5.3 Process Graph Manager.....	20
5.4 Visualisation Aspects	20
6. Summary	21
Bibliography.....	22
Dictionary.....	23

List of figures

Fig. 1. Dependencies between the components of the ICONS architecture	9
Fig. 2. Architecture of the Structural Knowledge Graph Manager.....	11
Fig. 3. A graph of objects	14
Fig. 4. The idea of basket	15
Fig. 5. Electronic form manager data flow	17
Fig. 6. Content Presentation Manager data flow	18
Fig. 7. Elements of the Knowledge Schema Manager.....	19

1. Introduction

1.1 Objectives

The goal of this document is to discuss the question of visualisation for the ICONS project.

1.2 Scope

This report concentrates on visualisation and describes basic requirements for a graphical user interface (GUI) of the application being developed in the project. The interface is supposed to enable one to manage a graph with data through two general kinds of navigation: *extensional navigation* in an explicitly presented graph of objects, and *intentional navigation* in an implicitly presented graph of objects representing the UML-like database schema.

1.3 Relations to Other Documents

This report presents ideas of the proposal. The general assumptions of it can be found in the deliverable [ICONS D16]. The deliverable [ICONS D11] presenting the state-of-art in the field of data visualisation and the deliverable [ICONS D13] outlining technical details of the proposal are also be of relevance to this document.

1.4 Intended Audience

The intended audience is all the participants of the project, in particular those responsible for developing the prototype and its graphical user interface.

1.5 Usage Guidelines

This report should be treated as a general introduction to the problem, where several issues and ideas are discussed, sometimes without giving final answers and/or without going into detail. The final answers are/will be presented in [ICONS D13] and [ICONS D16].

1.6 Notation Conventions

No special notation conventions are used in this report.

2. The ICONS Prototype GUI: General Ideas

2.1 Information Visualisation Aspects in ICONS

The large amount of non-professional, common users, with limited computer qualification, needs the development of easy-to-use interfaces, which make database/repository systems user-friendly and effective for them. This is where the idea of information visualisation systems comes into play. On the other hand, designing a system, which would be perceptually effective and easy-to-use, is very hard and usually user- and task-dependent. The most important problems concerning information visualisation that should be solved during the process of designing ICONS modules are the following:

1. choosing appropriate visual representation paradigms – the methods of presenting data/information on the display. This aspect involves first of all the problem of visualization information in the manner the user would find the most convenient;
2. visual data access – the methods of retrieving the desired data/information from a database by means of visual tools simulating “traditional” textual queries;
3. query results presentation – the methods of visual presentation of query results;
4. usability improvement – the methods of increasing the user-friendliness of interfaces.

One of the most significant aspects is usability, which means a set of the sub-terms like effectiveness, efficiency, user satisfaction, user awareness, etc. The effectiveness refers to the extent to which the intended goals of interaction can be achieved (i.e. accuracy and completeness of data). The efficiency is connected with the time, the money and the effort spent on that activity. A measure of user satisfaction could be for example: how comfortable she/he feels using the system, how easy-to-use the system is. The notion of user awareness concerns the limited ability of humans to memorize information, to plan behaviour during querying, to memorize implicit relationships between various pieces of information, and so forth. Even during a relatively short session the user may forget the goal of his/her search, he/she may forget navigation paths that led him/her to particular information; he/she may lose awareness about related pieces of information, and so on. On the one hand, the aspect of user awareness implies minimizing the amount of data/information on the user screen. Furthermore, the number of screens should be minimal as well, in order to make it easy for the user to memorize the relationships between information items.

Those four above requirements are sometimes contradictory, therefore some reasonable tradeoff is needed. Hence, usability results in additional requirements to graphical query interfaces, to take into account human capabilities and limits.

According to [ICONS D11] the following important suggestions connected with usability that should be taken into consideration when designing an easy-to-use and effective GUI for ICONS:

- The desired activities of the potential end user should be recognized.
- Visuals must be arranged to reveal key answers to task-specific data.
- Appropriate both perceptually effective visual representation of data and interaction paradigms should be chosen.
- Furthermore, The user should be allowed to shift between different available representation paradigms.
- Query languages should have conceptual simplicity, expressiveness, and user-friendliness.
- Context menus should be provided to manipulate the properties. They should be customized for particular users and/or particular applications.
- Interaction operations should be natural and intuitive.
- Undo/redo command stack should be exploited to maintain the history of changes.
- Some supporting operations like: back tracking, history logging, user context preserving, smooth transition between locations should be provided.
- Data conditioning appears to be necessary.
- What is especially important, the perceptual abilities of potential end-users should not be neglected.
- Maintaining the knowledge of users can be useful – based on the history of a user’s interactions with systems.
- Some additional mechanisms, for instance coordinated views, should be exploited.

From the points above, we want to emphasize the issue of querying which is an important part of the development of graphical retrieval interfaces. Typical query languages, such as SQL and OQL, are essentially programming interfaces to be used by programmers rather than by end users. In spite of various hopes and claims existing at early stages of database technologies, it becomes more and more clear that end users usually do not accept keyboard-oriented query languages with complex, recursive syntax, sophisticated (mathematical) semantics and complex pragmatics (rules of use). Practice shows that most end users prefer direct manipulation to that based on command languages. This is mainly due to the fact that command language interfaces are rather difficult for non-professional users, difficult to learn and remember. This expectation is to a big extent amplified by the popularity of Web browsers and GUI-based operating systems such as Microsoft Windows. In other words, end users, especially those non-professional ones, prefer interfaces with friendly, easy to learn and use, and self-explained graphical metaphors based on choices from menu, clicking, dragging and dropping, etc. Direct manipulation interfaces have become popular because they draw analogies to existing human skills (like: pointing, grabbing and moving objects in space) rather than trained behaviours. Menus are also especially beneficial for novice and intermittent users. Since as a rule only a few items to select from are presented at a time, they make a clear structure to decision making. Menus are also appropriate for professional users when the display and selection mechanisms are quick enough. Unfortunately, the expressive power of graphical user interfaces is lower than the power of keyboard-oriented query languages. Command languages interfaces – exploiting artificial languages similar to programming languages – are concise and unambiguous and permit the user to rapidly express complex possibilities through combination of constructs in new and complex ways. The tradeoff between expressive power and end-user friendliness is one of the main problems in the field of visual interfaces.

Another problem is querying itself. In the classical information retrieval scenario the user does the following: tries to understand what information he/she wants to retrieve from a database; defines a query in a query language; waits for an answer; utilizes the answer in his/her job; forgets the query and the answer. However, there are several exceptions from this:

- The user frequently does not realize from the very beginning what he/she wants to find. He/she realizes that only after some interaction with the database.
- The user is frequently unable to precisely define a query expressing his/her needs, because of his/her insufficient knowledge of the query interface (and no interest in this knowledge), and/or because the query interface has no options to express the need precisely.
- The user does not want to wait for query results for a long time.
- The answer can be irrelevant (imprecise), relevant but not increasing his/her knowledge, incomplete, and/or with a lot of information noise. In such cases the user cannot fully utilize the answer; hence he/she may want to use other information retrieval options.
- The query and the answer can be utilized for an indefinite period of time, hence both cannot be forgotten. The user may want to apply some requests many times, with slightly modified forms or applied to different information sources. He/she may accomplish his information need by many queries and other actions, in many days and sessions, and the result of a query/action can be used to formulate another query or as an input to another query.

Due to the above aspects, there is a tendency in visual query languages to create easy-to-use graphical interfaces by combining multiple paradigms, in particular:

- Information access through manual browsing.
- Schema concepts simplification, for example through removing items that the user regards as irrelevant.
- Queries through simple direct manipulation on widgets realizing graphical query metaphors, e.g. choices from menus, moving some attention/control points in a graph, etc.
- Storing information on the user's initial need for querying, storing a query being the result of this need, combining subqueries, using predefined and user-defined stored queries, storing partial and final results of querying together with some additional work that the user has done on the results, e.g. a query result without manually deleted items, grouping the result, augmenting the result with annotations and comments, etc. There should be no limits concerning how long such information has to be stored.
- Typing some additional information on keyboard, e.g. keywords, query conditions (which can be expressed also by means of icons), etc.

Our discussion concerning the information visualisation aspects in ICONS we start with the presentation of the ICONS architecture.

2.2 Architecture of ICONS

The ICONS architecture involves several components. In order to give a general view of the architecture, we present the dependencies between those components in the UML component notation as illustrated in Fig. 1. In this report we present those of the components that directly involve the problems of information visualization and graphical user interfaces (GUIs). First, we briefly present the components as such; afterwards we discuss their most important aspects with regard to visualization.

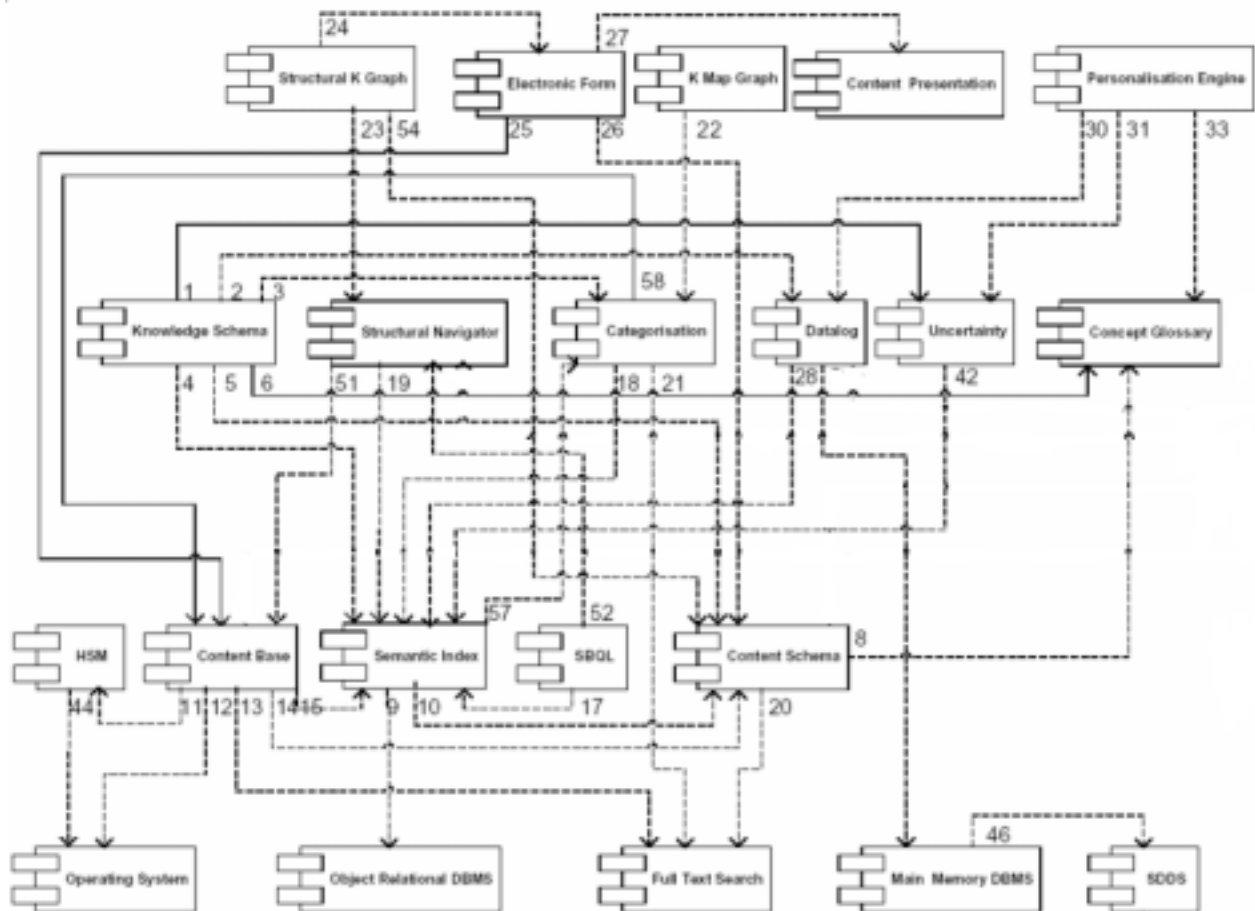


Fig. 1. Dependencies between the components of the ICONS architecture

The components relevant for this report are the following:

- Knowledge Schema Manager is responsible for keeping different types (and subtypes) of knowledge consistent on the global schema level. It stores business data/objects to be displayed and searched via SKGM module.
- Content Presentation Manager is generally used to present in a web browser content that is stored in native formats.
- Electronic Form Manager is used by Structural Knowledge Graph Manager to present the content.
- Content Categorisation Engine is used to search for desired information and produce as a result so-called knowledge maps.
- Knowledge Map Graph Manager is used by end users to display and operate on resulting knowledge maps from the Content Categorisation Engine.
- The Structural Knowledge Navigator (SKN) is a wrapper operating upon the Content Base's API and providing other modules with a set of enhanced and more specific services. It works on the Knowledge

Base level providing access to information objects (modification is not provided). Objects are delivered in the graph structure being a subset of the graph stored in the content repository. The two identified services are transitive closure and generation of a graph representing the neighbourhood of a given object.

- Structural Knowledge Graph Manager is used to retrieve the content through navigational search.
- Process Graph Manager represents visually both the definitions of processes and their instances.

We start our report with Structural Knowledge Graph Manager (SKGM), because this component involves most of the basic aspects that we intend to consider while developing a GUI for ICONS.

3. Structural Knowledge Graph Manager (SKGM)

3.1 Introduction

SKGM is built on top of the ICONS Content Base. A part of the ICONS architecture which is the context for the SKGM module is presented in Fig. 1.

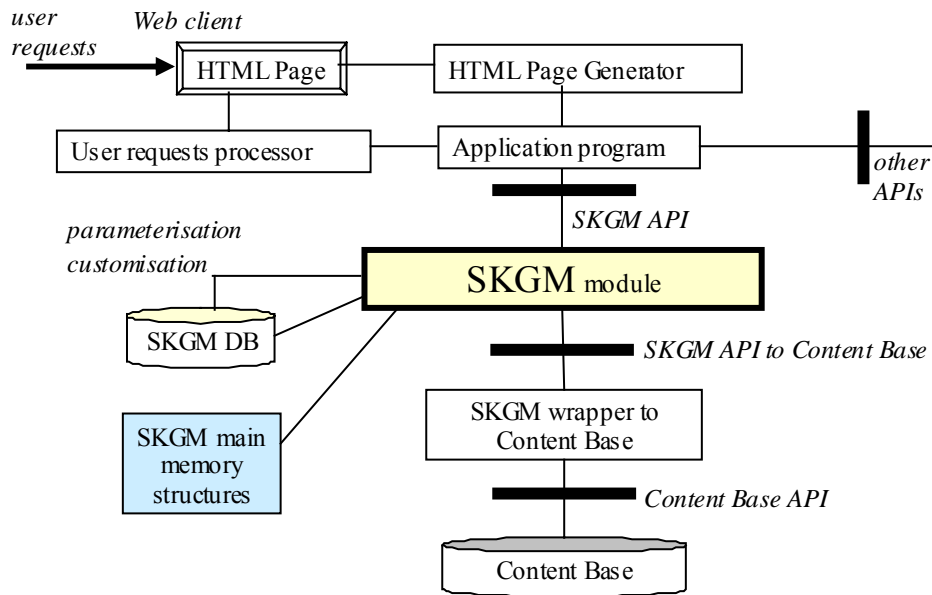


Fig. 1. Architecture of the Structural Knowledge Graph Manager

The modules here are the following:

- Content Base – It stores information to be displayed and searched via the SKGM module. It is assumed that SKGM will use only retrieval operations, that is, in the current version updating of the base will not be available through SKGM. This should concern both persistent and temporary data.
- Content Base API – It is a set of Java interfaces that can be used to search, display etc. the resources of the Content Base. As stated previously, SKGM should not use operations that may update the Content Base.
- SKGM wrapper to Content Base (represented as the Structural Knowledge Navigator (SKN) in Fig.) – This module contains Java classes and interfaces that customize the Content Base API to the needs of SKGM. For example, some typical operation of navigation in a graph may consist of several Content Base API operations, thus it might be useful to encapsulate them as a single operation. This wrapper may also be necessary as a layer in a multi-tier architecture, isolating user interfaces from particular solutions in the Rodan Portal Repository (hence supporting maintenance of the software). The isolation has twofold meaning: the Content Base may be changed without changing SKGM (by changing the wrapper instead), and the Content Base can be substituted by another base/repository by changing the wrapper.
- SKGM API to Content Base – It contains operations on Content Base from the point of view of SKGM wrapper.
- SKGM API – It is a set of Java interfaces that could be used by an application program developer to parameterize SKGM and to call its external functions. The graphical format in which the result of SKGM functions is to be sent to an application program is still a subject of future decisions. SKGM API should determine a visual metaphor to be managed, should determine a database context that has to be visualized, should determine various utilities that allow to reduce the amount of information on the user screen (e.g. user views), etc.

- SKGM module – It contains Java classes and interfaces that use SKGM API to Content base and present the retrieved data stored in the Content Base. It prompts the user to make various operations on the graph and other visual structures, e.g. dragging distinguished points, dragging and dropping, etc.

3.2 Visual Representation Paradigm

Because from the SKGM's point of view the Content Base is an object-oriented database, the natural choice of the representation paradigm is the graph metaphor. Currently graphs are a very popular visual paradigm, because they are especially easy and natural to express the structural representation of data with inherent relations among the data elements. Graph visualization has a wide area of application, including file hierarchy on computer systems, data structures in compilers, evolutionary trees, genetic maps in biology, molecular maps, biochemical paths in chemistry, class browsers in object-oriented systems, document management systems, real-time systems, Petri nets, semantic networks, virtual reality, WEB site maps, etc.

One of the main problems with graphs used for the visualisation of data structures is that they can be large, even too large to be shown on a single screen (this concerns for instance the size of a query result which may consist of hundreds, thousands or more items, thus it will be difficult or impossible to browse them using only manual operations; the issue of data access is discussed in Section 3.3). Moreover, graphs can be difficult to present without crossing edges. This can result in increased complexity and can decrease the user perception. Although there are algorithms to optimize graph layout, implementation of them may not improve essentially the quality of the user's work.

A simple, but effective method of reducing the amount of information on the user screen is to present only a small portion of the graph on the screen at a time. This requires additional mechanisms to support user perception:

- Presenting only the near context of the current navigation attention point, that is, those objects and links among objects that are closer to the central object than the customized presentation depth.
- Changing the neighborhood of the presented content by applying the zooming technique, both geometric zooming (blowing up the graph content) and semantic zooming (displaying more details when a particular part of a graph is approached). To support user awareness all those operations should be performed smoothly – the radical change of the graph drawing can make the user feel lost, which could not be acceptable for interactive usage.
- Removing unnecessary objects/classes and links/associations from the view according to his/her choice.
- Hiding names of objects, of attributes and of links.
- Changing the size of icons.
- Altering manually graph layouts. The user should be able to create a graph of his/her own layout by moving nodes and edges on the screen. With reducing the amount of information on the screen when necessary, the possibility to alter graph layout is even more important than implementation of automatic layout algorithms. Layouts should be personalized. Altering graph layout involves the question of predictability, that is, having for a given graph similar layouts with different runs of the algorithm applied to alter layout.

Graph layout should be legible for the end user, even that non-professional. There should be a possibility to choose between different layout according to one's needs. Those above mentioned features should be personalized.

Additional techniques reducing the amount of information on the screen are: focus + context, clustering, and incremental exploration techniques [ICONS D11].

Because we suggest using the direct manipulation interaction style, it is important to emphasize here that the main difficulty in designing such interfaces is to find a suitable graphical representation or visual metaphor for classes, objects, relationships and so on. We suggest to exploit icons with a textual description because they give the user a faster comprehension than textual or pictorial icons. It is especially important both for non-professional (those not familiar with database concepts) and intermittent users. Icons are also usually much more concise than textual notions, which may in turn be also useful for advanced users.

3.3 Visual Data Access and Query Formulation

One of the basic visual querying metaphors assumed by the ICONS prototype should be the interaction style based on navigation (the so-called direct manipulation style) in a graph representing stored data structures. This is due to the fact that the idea to present data description and data structures as graphs with named nodes and edges is natural for object-orientation. On the level of database schema, the graph should resemble – due to its simplicity – the well-known entity-relationship, for instance as the UML-based class diagram. Such a diagram would precisely present the user view on database content and thus can be directly used for querying. On the level of objects there would be an object graph, with named nodes being objects and named links (pointers) representing connections between objects.

The problems connected with data access should be considered in ICONS with regard to the following navigation paradigms:

- *extensional navigation* – the user directly navigates through a graph of objects moving his/her focus according to links connecting objects. After reaching an object of his/her interest he/she can display it, copy it, record its reference, etc.
- *intentional navigation* – the user navigates through a graph representing a database schema, where icons are used to denote classes and relationships between classes (a class is defined here as a set of objects of the same type). The navigation utility is similar to the previous case, but each node of the graph represents a set of objects rather than a single object. The user must therefore be additionally equipped with a utility to select a subset from the set that he/she has reached on the graph – such a utility could be a context menu from which the user can select only those of the objects that he/she needs.

The user should be allowed to determine and execute any predicates that select the elements retrieved in the given navigation step. Both types of navigation should be accomplished through the same consistent visualisation mechanisms.

The user should be able to freely switch from one type of navigation to another. Extensional navigation resembles for example browsing the Web, where objects correspond to HTML pages, and pointer links correspond to hyperlinks on HTML pages. This is not the case for intentional navigation, which so far has no corresponding facilities in Web browsers, mainly due to the fact that Web pages lack structuralization. This type of navigation seems to be much more important for end users and can be thus very important for further Web organization based on XML resources having well developed schema (meta-model, ontology).

We should note that there are several problems concerning this kind of visual data access:

- The size of the end user's screen: usually it is impossible to present a large and complex graph, hence it must be presented partly, with zooming facilities, context + focus, panning, clustering etc perhaps with 3D views and with hiding details of objects depending on the mode or stage of searching.
- User awareness: the user can very quickly lose orientation while navigating in a complex graph, thus special graphical facilities are necessary to keep him/her aware of the current sub-goals or sub-results of his/her search.
- Ergonomics: On the one hand, the effort necessary to retrieve the required information should be minimal from the end user viewpoint. On the other hand, the retrieved data should be displayed in a clear form. This problem is especially important if a graph used as visualisation of object-oriented data structures is large and it is difficult (impossible) to show the whole of it on the computer screen without compromising the quality of the presentation.
- The problems concern extensional navigation. Extensional navigation may imply (in typical cases) thousands (even millions) of nodes and edges. Intentional navigation involves smaller graphs, but in typical cases even such graphs may consist of dozens of nodes and edges, thus it would be difficult for the user to present it on a single screen.
- Combining manual and predicate-based automatic navigation.
- Elliptic queries: for some kinds of navigation it would be useful for the user to omit some details of navigation.

Because the data structure used to present data is the graph, for querying ICONS should apply the query strategy by schema navigation, i.e., based on concentrating on a concept (or group of concepts) and moving from it to

reach another concept(s) of interest, on which some conditions may be specified. This kind of strategy adopts as such typical query operators like the selection of visual concepts, the traversal on adjacent concepts, and the creation of a bridge between disconnected concepts.

Two mechanisms necessary to make a navigational system a powerful end-user tool are:

- recording query results – navigation in a graph requires a special utility to store temporary and final retrieval results;
- manual browsing – the user can manually browse through retrieval results (obtained by him/her or other users), with possibility to remove some items from the results, to insert new items, to merge results, etc.

These tools should be freely combined with extensional and intentional navigation. The graph navigational manager should provide support for the above mechanisms.

To illustrate navigation in a net of objects, we present in Fig. 3 a graph, where objects (named *A*, *B*, *C*, and *D*) are connected to each other through directed edges (*x*, *y*, *z*, *t*, *w*, and *v*). We do not require the names of objects and the names of edges to be unique. Objects can store information (attributes and their values), which can be displayed for the user. The user can select starting objects navigation through the following actions:

- Manual choice through clicking and marking proper objects on the screen (e.g. on the basis of their content, which can be optionally displayed).
- Introducing by typing the name of objects and a condition on their contents.
- Taking proper objects from his/her store (which has been filled in during a previous search).

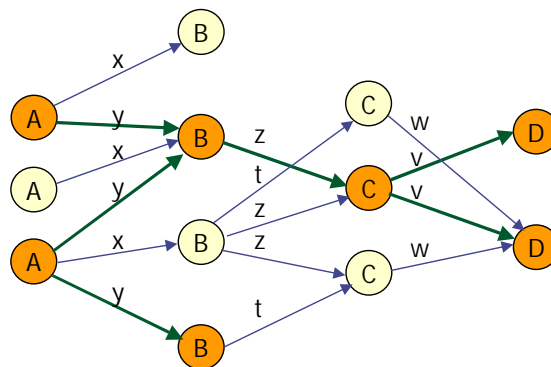


Fig. 3. A graph of objects

After selecting initial objects, the user can navigate in the graph through its named edges (selected from the menu). Suppose he/she initially selects the first and the third *A* objects (denoted as gray circles), and then he/she uses the *y* edges (denoted as thick lines). This means that he/she is moving to the second and the fourth *B* objects. If then he/she uses the *z* edge, then he/she is moving to the second *C* object. Then if he/she uses the *v* edges, he/she is moving to both of the *D* objects. The objects selected during this search we will call *marked*; the other objects are *unmarked*. During the interaction the user is allowed to do any actions by means of context menus, such as selecting/deselecting objects, displaying object, moving references to objects to his/her private store etc.

With regard to more advanced issues of data access through querying, the user should be able to:

- Make use both of a visual querying style and a command query language style. While visual querying is a modern paradigm that could be tested during the ICONS research project, a command query language is a tool for advanced users, especially in cases when it is easier to use a command language than a visual language to express more sophisticated queries.
- Use results of other queries to define new queries. Because the query and the answer could be utilized for an indefinite period of time, the user may want to apply some requests many times, with modified forms or applied to different information sources. He/she may accomplish his information need by many queries and other actions, in many days and sessions, and the result of a query/action can be used to formulate another query or as an input to another query.

3.4 Query Result Presentation

In order to enable the user to store query results and manipulate them, we propose to introduce the concept of basket (Fig. 4), which directly corresponds to the virtual shops metaphor. A basket is a graphical element with icons representing selected data items (objects, classes etc. depending on the kind of navigation). Baskets should be organized hierarchically, similarly to operating system catalogs. They should be persistent structures, i.e. they should store in the database (SKGM DB). In this way a single search could be subdivided into many user sessions.

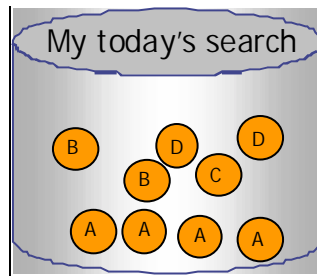


Fig. 4. The idea of basket

Icon	Id	Object Name	Retrieval date	Object content	Comment
☺	23156	Person	02.01.03	John Smith	I checked him yesterday.
☺	23456	Person	02.04.05	Mike Brown	Smart client!
📁	766585	Document	02.08.19	Order 234527	Currently processed, ready in 2 days
🏢	3453453	Company	02.07.19	Brainstorm Ltd.	Our best supplier.

Tab. 1. Example basket content

The basic characteristics of basket should be the following:

- Basket stores the results of extensional navigation.
- Baskets have unique names.
- The user can assign to a basket some longer description or comment.
- Icons representing particular kinds of objects can be different; hence they should be customized by the user.
- The content of each object in the basket should be displayed if necessary.
- Each object in the basket should be supported by the following information, for instance, presented as a list of properties:
 - An icon representing this object.
 - Object unique identifier.
 - Object name.
 - Representation of the object content (for instance, for objects named *Person* this could be a concatenation of the value of attribute *Name* and attribute *Family Name*), which is determined by the application developer.
 - Date/time of placing the reference to an object to the basket.
 - Any string comment (annotation) introduced by the user.

An example content of a basket is illustrated in Tab. 1.

An alternative solution for the idea of baskets can be folders like the Window's Explorer fashion with the option to display the details of content on demand.

3.5 Usability Improvement

The usability in the SKGM module could be obtained through a number of methods and techniques, like the following:

- Visual representation of data should be perceptually effective. The user should be allowed to shift between different available representation mechanisms based on different paradigms.
- Interaction operations should be natural and intuitive.
- The current status of the user session should be shown on the screen (constantly or on demand).
- A query language should have conceptual simplicity, expressiveness, and user-friendliness. If designed language aims at non-programmer users, the graphical interface (menus and/or forms and/or icons) should be chosen.
- The user should be able to record any result of his/her search in a convenient manner. To this end we propose the idea of user basket. The user should be able to annotate the results of his/her queries.
- The user should be able to use the stored results of previously evaluated queries in order to start new navigation.
- The context menus should be provided to manipulate the properties. The context menus may be customized for particular users and/or particular applications.
- Parameterization and customization information should be exploited. This information concerns for example the sets of available icons, visual preferences of particular users, user views – i.e. the methods of reducing amount of information to be displayed (for instance, the names of objects and relationships, the form and size of graphs, colours, etc.).
- Information about the users of a content management system. This information should allow the programmer to identify the user after login and to associate it with personalization information.
- In the case of extensional navigation when too many nodes are connected to a single node, the user should have the possibility to switch the mode to intentional navigation. Alternatively, there is also an idea to combine extensional and intentional navigation into a single interface.
- The idea of multiple views (especially with the possibility of mutual coordination) should also be considered. Multiple views are especially useful for large portions of data, because they enable for example to see simultaneously all the data in one view (this view certainly does not show details but increases user awareness) and some data in another view (hence those data can be shown in more details than in the former view).
- The user should be able to undo any operation he/she performed, including navigation, i.e. to go back to some previous navigation step so undo/redo command stack appears useful to be necessary.
- Maintaining the knowledge of users can be useful. The classification of users can be done on a basis of the history of their interactions with the systems, as a result the system will be able to help the user by suggesting the more appropriate both data representation and interaction paradigm. All systems should be developed taking into consideration such possibilities.
- Furthermore, the abilities of the end user should not be neglected, then assistance for query construction, troubleshooting and context-sensitive help should be provided. For instance, a system could help the user by suggesting the more appropriate both data representation and interaction paradigm.

4. Electronic Forms Paradigm

Forms have proved to be a user-friendly paradigm of computer-human interaction thus they should be a part of the ICONS project. They should be key elements for ICONS as a mechanism to create, update etc content objects in the form of XML documents. The most significant problems concerning forms, including Web-oriented solutions, are the size and complexity of the processed data, as well as the semi-structureness of XML data. As usual, one must take into account the aspects of ergonomics and user friendliness.

The overall data flow in the ICONS application with regard to electronic forms is presented in Fig. 5. The form editor takes the information about objects to be edited from their corresponding classes' XML schemas; they provide information about attribute types, cardinalities and some general constraints.

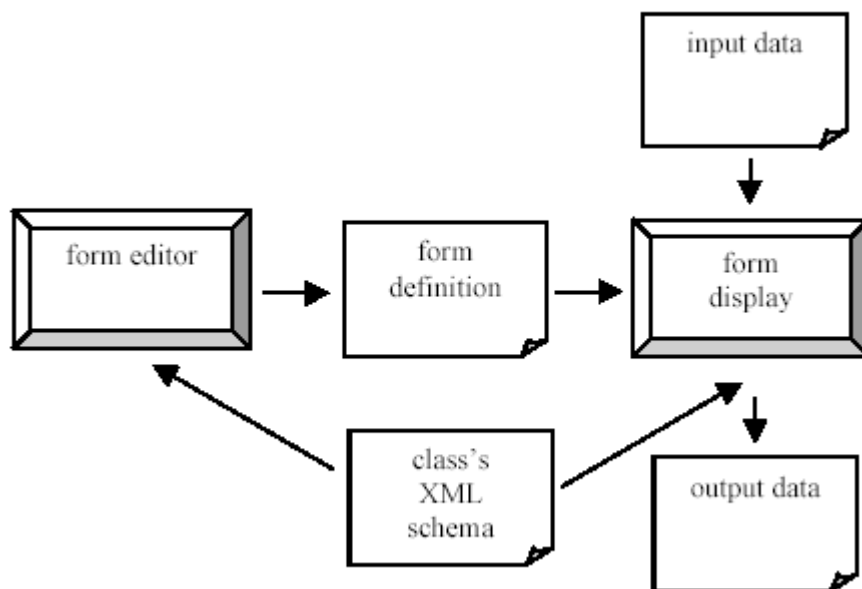


Fig. 5. Electronic form manager data flow

A form should be able to show an arbitrary number of classes, including complex classes of tree structure. For such classes a form should present attributes selected by the user; each attribute should be available in one of the modes: edition, display etc.

An attribute field should be of such types as, for example: textbox, combobox, listbox, multichoice listbox. The user should be able to input data both by typing and through selection from dictionaries assigned to the form; furthermore, there should be a possibility to set initial data for a data input field. The validity of the data in data input fields should be continuously checked by the module as well as the servicing of erroneous situations should be supported.

For data fields with binary content (e.g., video, audio) the user should be able to open a specialized (sub) window with special functions to operate on this content.

A form should be able to show any necessary graphical element, such as additional text, borders, logos etc. For visible elements of forms there should be a possibility to choose a style – this concerns for instance font size, font style (bold, italic, underline), colour, border, etc.

The form editor, after interactions with a form designer (human operator), should generate the form definition in some XML-based language. The form display should render the form on the user's screen along with the form definition. Input data must conform to the appropriate XML schemas. Output data should be guaranteed to conform to the same XML schemas.

The ICONS application should offer a special graphical WYSIWYG editor for creating and modifying electronic forms. The editor could be similar to popular electronic form editors available in such programs as Web browsers.

An important characteristic of electronic form managers is that the time of form display/redisplay should be acceptable for the end user who is seldom willing to wait long for a form to (re) display. This may mean some kind of trade-off. Because we assume that the size of a form (in terms of number of classes and attributes) should not be limited, the user should be able to split complex forms when necessary.

Generally, in order to improve the usability of electronic forms the following features should be taken into account:

- the user should be able to use any shortcuts he/she needs;
- during long operations the user should be constantly informed about the progress;
- such useful functions as undo/redo, history of operations etc should be available;
- assistance, troubleshooting and context-sensitive help should be provided during electronic form construction.

Forms use the Content Presentation Manager to present data stored in native formats by means of web browsers.

The issue of the great number of native formats is important for Web-based applications, thus the problem of content presentation should be solved in the ICONS project. The native formats to be considered should include for instance: db (DBase), drw (Micrografx), dxf (AutoCad), lwp (Lotus), mpp (Microsoft Project), msg (Microsoft Outlook), msw (Microsoft Word), pcx (Paintbrush), ppt (PowerPoint), qp (Quattro Pro), sow (StarOffice Writer), tif (Tagged Image File Format), visio (Visio 4), wp (WordPerfect), xls (Microsoft Excel), doc (MS Word), pdf (Acrobat). Furthermore, the two most popular web browsers: Internet Explorer and Mozilla, should be considered as a web browsing tool.

In the ICONS project the component responsible for content presentation is the Content Presentation Manager. The data flow in this module is presented in Fig. 2. Data in a native format are passed to the Outside In package that generates the corresponding HTML and provides the Manager with the file path to the rendition.

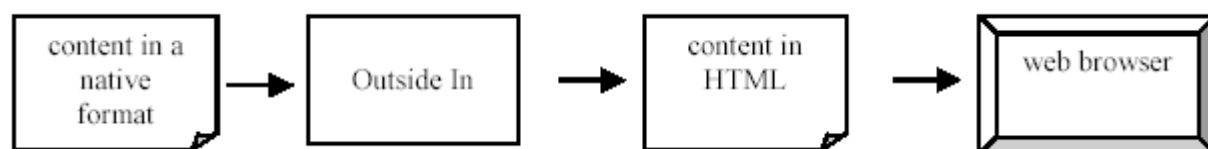


Fig. 2. Content Presentation Manager data flow

5. Other Modules

In this chapter we present other ICONS modules with regard to the problem of information visualisation.

5.1 Knowledge Schema Manager

The ICONS global knowledge schema should be of multi-paradigm nature, that is, it should be formed of formal knowledge representation pertaining to structural, declarative and procedural knowledge concerning a particular application domain. Knowledge schemas are of major importance because nothing, except what is defined on the schema level, can be instantiated on the knowledge base level. The schemas in question are to the same extent different and complementary. Therefore, within ICONS platform they should co-operate closely, forming a global knowledge schema, in order to provide complete functionality of platform for knowledge intensive applications' development. Maintaining knowledge schema integrity within a particular knowledge type is not a trivial task [ICONS D10]. Maintaining the global knowledge schema is a challenge addressed by the Knowledge Schema Manager.

A major problem to be solved by the Knowledge Schema Manager is to guarantee the integrity of the global schema. Usually modules supporting a particular type of knowledge maintain integrity constraints of this type of knowledge (see the intra-integrity constraints within [ICONS D10]). For example, an association can be defined only if classes to be associated are defined. Integrity is imposed both on the API functions level as well as user/administrator interfaces. Modules are however not aware of external knowledge representation; they just refer to external (from their point of view) schema elements (the classification of the elements of the Knowledge Schema Manager is illustrated in Fig. 3) and assume that they exist and are fully operable by another module. This is due to the modules' modularity and encapsulation that results mainly from the software engineering premises. Coupling modules too tightly always increases maintenance costs. Moreover, it is not sure that ICONS list of supported knowledge (sub) types is ultimately closed. Within tightly coupled architecture adding new (sub) type of knowledge would mean severe reconstruction of the existing modules. In the proposed solution, thanks to Knowledge Schema Manager that is aware about all types of knowledge and statuses of their elements, only the module servicing the new (sub) type of knowledge is affected while other modules may need only slight interface modifications.

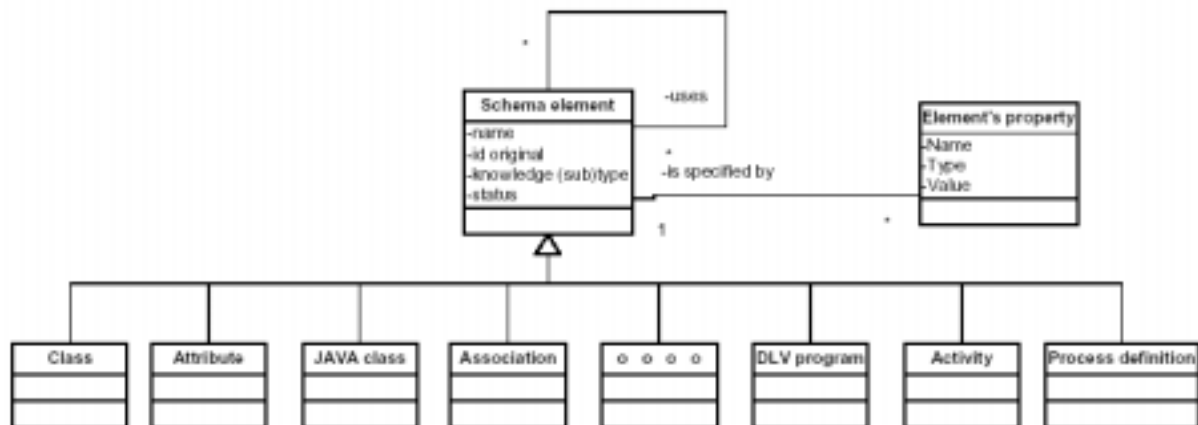


Fig. 3. Elements of the Knowledge Schema Manager

The graphical integrated global schema development environment (with e.g. cross schemas' drag and drop functionality) as well as usage of ICONS inference capabilities to support automatic integrity check are two major vehicles of the Knowledge Schema Manager. Other functions could support the following possibilities:

- One is able to register any knowledge schema element.
- One is able to assign status to registered knowledge schema elements (for example: planned, under development, ready, obsolete etc).
- Relation with other schema elements is inferred automatically.
- An element's definition is parsed; appropriate references are identified and represented.

- One is able to store additional properties of schema elements.

5.2 Knowledge Map Graph Manager

Knowledge maps provide means to categorise concept categories stored in the content repository through the possibility to assign concepts to categories. The imposed tree-structured hierarchical categories provide a powerful navigation and search tool for browsing the content repository along consciously engineered organizational context. The user should be able to: drill-down into consecutive levels of hierarchy to find more and more relevant information among fewer and fewer occurrences, drill-up (opposite to drill-down) and drill-across (the creation of a bridge between disconnected concepts).

Application specific knowledge maps should be defined and materialized (through the Content Categorization Engine). The main idea of the Knowledge Map Graph Manager is to provide access to services offered by the Categorisation Engine (knowledge maps in particular) on the user interface level in an effective and user friendly way. The Knowledge Map Graph Manager enhances their services with respect to navigation and search.

In order to improve the process of browsing, some visual mechanisms, such as colours, different shapes and patterns, icons, should be used to distinguish content in knowledge maps. A special structure – basket – is available for storage of selected objects and knowledge maps; for details see Section 3.4. As for other components of the ICONS application, the time needed for (re) displaying knowledge maps should be acceptable for the user.

For knowledge maps, that is, large trees, which probably will be the case, we suggest to follow the well-known “information visualisation” mantra “overview, zoom, and details on demand”. Hence, the exploiting of the following techniques for effective dealing with large graphs should be considered (for details see [ICONS D11]):

- zooming (both geometric and semantic);
- pan;
- focus and context techniques;
- clustering;
- incremental views;
- coordinated views, for example, a separate window for overview (by means of a tree map) and a separate window for more detailed information.

5.3 Process Graph Manager

One of the most important factors of a workflow management system is providing a simple-to-use but powerful visual language to model business processes. Such a language should not be too technical, so that it could be understood by both customers and vendors, linking the world that already exists (i.e. the customer world) with an artificial, computer-represented world to be developed (i.e. the vendor world). An advantage of a graphical language is that it can make communication between vendors and users easy and help notice and avoid many development problems at early project stages. In addition, it enables end-users to model business processes of their organizations. In order to keep process representation simple and powerful, including history of process execution, information on the current state, and how the process can be continued should be represented using the same or very similar language. In such case, when the users understand the language, they almost immediately are able to read and use information on process execution.

For every work item from a user work list, the user interface should visualize the current state of a process instance, that is, a process instance to which this work item belongs.

5.4 Visualisation Aspects

The most appropriate visual representation paradigm for the modules covered in this chapter are graphs. Hence all the discussion for SKGM (Chapter 3) is relevant for those modules; the only difference is the possibility to modify graphs. Therefore such a possibility has to be delivered. Moreover, because after modification the layout of graphs will change, great attention should be paid to graph predictability.’

6. Summary

The report discusses the basic issues related to the graphical user interface – visualisation in the ICONS prototype. General ideas of our proposal have been presented. The main part of the discussion concerns the SKGM module – the conclusions and ideas from that discussion are relevant also for the other modules.

The following important topics which should be taken into consideration have been discussed:

- Visual representation paradigms.
- Graph navigation paradigms.
- Query formulation issues.
- Usability improvement: Undo/Redo commands to maintain the history of changes, back tracking, history logging, user context preserving, data conditioning etc.

For the modules whose interfaces are based on graphs, we propose to introduce the idea of baskets to store temporary and final results of navigation in complex linked structures. This metaphor of baskets enables us to deal with the issue of user awareness; in particular, it takes into account the limited human ability to memorize information and implicit relationships between various pieces of information, and makes it possible to manage (large amounts of) retrieved data effectively. Other issues related to graphical user interface are for instance user awareness and ergonomics which concern the effort necessary to retrieve the required information and to display it a clear form on the screen so that the user does not lose orientation. The problems in visualisation require some trade-off between the power of an interface and its user-friendliness.

Bibliography

- [BCMS96] A.N. Badre, T. Catarci, A. Massari, G. Santucci: Comparative Ease of Use of a Diagrammatic vs. an Iconic Query Language. *IDS 1996*: 4
- [CaCo96] T. Catarci, M.F. Costabile: Visual Query Systems. *Journal of Visual Languages and Computing* 7(3), pp. 243-245, 1996
- [CCFL96] T. Catarci, S. Chang, M.F. Costabile, S. Levialdi, G. Santucci: A Graph-Based Framework for Multiparadigmatic Visual Access to Databases. *TKDE* 8(3), pp. 455-475, 1996
- [CCLB97] T. Catarci, M.F. Costabile, S. Levialdi, C. Batini: Visual Query Systems for Databases: A Survey. *Journal of Visual Languages and Computing* 8(2), pp. 215-260, 1997
- [CCS94] T. Catarci, S. K. Chang, and G. Santucci. Query Representations and Management in a Multiparadigmatic Visual Query Environment. *Journal of Intelligent Information Systems*, 3, pp. 299-330, 1994
- [Cruz94] I. F. Cruz. User-defined Visual Query Languages. In *IEEE Symposium on Visual Languages (VL '94)*, pp. 224-231, 1994
- [CSC97] T. Catarci, G. Santucci, J. Cardiff: Graphical Interaction with Heterogeneous Databases. *VLDB Journal* 6(2), pp. 97-120, 1997
- [ESPRIT96] Foundations of Advanced 3D Information Visualization, 1996. European (ESPRIT) Working Group <http://www-cui.darmstadt.gmd.de/visit/Activities/Fadiva/>.
- [HIL95] E.M. Haber, Y.E. Ioannidis, M. Livny: OPOSSUM: Desk-Top Schema Management through Customizable Visualization. *Proc. of VLDB*, pp. 527-538, 1995
- [ICONS D10] The IST-2001-32429 ICONS Consortium, A Multi-Paradigm Integrated Knowledge Schema
- [ICONS D11] The IST-2001-32429 ICONS Consortium, Information Architecture: Evaluation of Tools and Methods
- [ICONS D13] The IST-2001-32429 ICONS Consortium, The ICONS Graphic Interface – Design Specification
- [ICONS D16] The IST-2001-32429 ICONS Consortium, Specification of the ICONS Architecture
- [MGP98a] N. Murray, C.A. Goble, N.W. Paton: Kaleidoscope: A 3D Environment for Querying ODMG Compliant Databases. *Proc. of VDB*, pp. 85-101, 1998
- [MGP98b] N. Murray, C.A. Goble, N.W. Paton: A Framework for Describing Visual Interfaces to Databases. *Journal of Visual Languages and Computing* 9(4), pp. 429-456, 1998
- [MHC96] B. A. Myers, J. D. Hollan, and Isabel F. Cruz (eds.): Strategic Directions in Human Computer Interaction. *ACM Computing Surveys*, 28(4), 1996
- [MPGB00] N. Murray, N.W. Paton, C.A. Goble, J. Bryce: Kaleidoquery-A Flow-based Visual Language and its Evaluation. *Journal of Visual Languages and Computing* 11(2), pp. 151-189, 2000
- [ODMG00] Object Data Management Group: The Object Database Standard ODMG, Release 3.0, R.G.G.Cattel, D.K.Barry, Ed., Morgan Kaufmann, 2000
- [PiCa95] P. Pirolli and S. Card: Information Foraging in Information Access Environments. In *ACM SIGCHI '95*, pp. 51-58, 1995
- [SSC97] S.F. Silva, U. Schiel, T. Catarci: Visual Query Operators for Temporal Databases. *TIME* 1997, pp. 46-53, 1997
- [UML01] Object Management Group: Unified Modeling Language (UML) Specification. Version 1.4, September 2001 [<http://www.omg.org/>]
- [WfMC] <http://www.wfmc.org/standards/docs/tc003v11.pdf>

Dictionary

Notion	Meaning
annotation	A text that the user has attached to a retrieved object in a user basket.
attribute name	A string name used to bind an attribute value of an object.
attribute type	Type assigned to an attribute. Type is used for two purposes: for checking operations on this attribute and to determine representation of an attribute value.
attribute value	A value stored within an attribute.
basket	A visible data structure to store intermediate and final results of retrieval in the database. Baskets can be nested.
basket name	Name assigned to a basket.
button	A widget that after clicking by the user triggers some action.
central point	An object, after selecting by the user, presented in the centre of the user view.
class diagram	A diagram showing the static conceptual structure of a business domain as a graph representing classes, attributes, methods, inheritance relationships, associations, aggregations, and other notions. The idea of class diagrams is based on syntactically modified and extended Entity-Relationship diagrams. A class diagram is an abstract view addressing human conceptual modeling of computer applications. However, it is not precise enough to be considered as a specification of data structures. Some subsets of class diagram capabilities can be used as in this role, e.g. as a database schema language.
conditions for class and association	Retrieval conditions determined by the user for a class/association. The condition involves attributes of an object.
current class	A concept of intentional navigation: the class recently chosen by the user.
customization	Parameterization of the entire GUI, according to the developer's wishes, e.g. fonts, colors, kinds of icons to be displayed etc. The customization may require some (virtual or materialized) database views on the database, which will be used by the developer for conceptualization of an application.
database schema	An expression (possibly in a graphical form) determining the content of a database and organization of data structures.
depth	The maximal distance (the number of edges) of an object from a chosen graph node.
depth of presentation	Determines the distance (the number of edges) from starting object to the most far objects in the given presentation.
dialog on conditions	A concept of intentional navigation denoting a box window where the user can determine conditions that must be satisfied by selected objects.
distance in a graph/network	The minimal number of edges between two graph nodes.
Electronic Form Manager (EFM)	Graphical, knowledge-schema-driven, electronic form development environment.
extensional navigation	Navigation in which a graph displayed on the user screen is isomorphic (conceptually identical) to a subset of the object graph stored in the object repository. The user moves from a displayed object to another object according to displayed links connecting the objects.
far context	Objects that are farther than the depth of presentation.
form display	Functional part of the Electronic Form Manager responsible for displaying and operating

		an electronic form during interaction with end user.
form editor		Functional part of the Electronic Form Manager supporting development and maintenance of electronic forms.
graph		A structure consisting of nodes and edges, shown on a user screen. The structure is a mapping of corresponding data structures.
graphical metaphor		A graphical icon or more complex graphical structure displayed on the screen, together with operations that the user can perform on the icon/structure. A graphical metaphor, together with its operations, is usually designed in a way that makes resemblance of user operation with some obvious user activity in the real world (e.g. moving an object to another place, walking through a maze of paths, flying in a 3D space, etc.).
graphical metaphor	query	A graphical metaphor that is to be used for querying data.
Graphical User Interface (GUI)	User	A user interface consisting of visual metaphors that organize user interaction with some data and service resources, usually with a database.
icon		A picture being a visual representation of some computer resource, e.g. a data, a program, a file, etc.
ICONS Content Base		The central data store and corresponding API functions for data/knowledge management within the ICONS prototype.
information object		Any database object containing useful information.
information retrieval		The process that the user performs in order to fetch useful information from the database.
intentional navigation		Navigation in which a graph displayed on the user screen is some abstraction over an object graph stored in the object repository. Usually this abstraction is a database object schema, where objects are represented by icons of classes, and links among objects are represented by icons of associations. Navigation in intenso requires mixing a navigation metaphor with selecting object through some other means, e.g. predicates, menus or manual operation.
knowledge schema	base	Synonym for database schema.
knowledge map graph manager (KMGM)		A module responsible for navigation and search within defined knowledge maps on the user interface level.
label		Name of an object, of an association, of a link, etc., displayed on the graph presenting data structures. Synonym for name.
link		A data structure that accomplishes semantic relationships between two objects, usually in a form of a named pointer leading from an object to another one.
main attributes		Attributes of an objects that appear as first on the screen after object visualisation. Main attributes allow the user to recognize at first glance the content of an object.
main basket		The basket assigned to a particular user that contains all his/her subsidiary baskets. The user has only one main basket.
manual browsing		Browsing through a database or through query results by manual displaying particular data objects on the screen.
menu		A vidget that makes it possible for the user to make a choice among several displayed possibilities.
metadata		Data that describe data. Metadata are stored in a metadata repository. Sometimes (especially in AI and Web terminology) referred to as ontology.
metadata repository		A persistent data structure storing metadata in a structural, searchable form. In another view, a metadata repository stores a database schema together with a lot of extra information (for administration, optimization, maintenance, statistics, security, etc.). Usually items in the metadata repository are linked to objects they describe. In RDBMS metadata are collected within catalogs. Other terminology: data dictionary, data

	dictionary/directory, DDD, ontology base, ontology metabase, metabase.
metamodel	A general conceptual frame describing principles, rules and structures for storing metadata. Metamodels can be of two kinds that must not be confused: (1) a reference metamodel, which describes abstract relationships and dependencies between concepts introduced in a particular notation or language; (2) a database metamodel, which describes the structure of a metabase repository.
metaphor	see: graphical metaphor
navigation	A data querying/searching metaphor, in which the user moves (in his/her mind) according to directed arcs in a data structure graph. In query languages (e.g. XPath or SBQL) navigation is accomplished by path expressions, e.g. Doe.wife.company.manager.wife. In visual interfaces navigation is a graphical metaphor accomplished by conscious moving by the user some visually distinguished point along edges and nodes of a graph displayed on the screen.
near context	All objects and links between objects that are closer to the central object than the presentation depth.
nested basket	A basket that is nested within another basket.
network	Synonym for graph.
object diagram	A graph consisting of objects (nodes) and semantic links between objects (edges).
object graph	1. A data structure stored in an object database with explicit named reference links between named objects. 2. A graph displayed on the user screen being a visual shape of a part of the above graph.
object identifier (OID)	An internal identification of an object. Usually it is assumed that object identifiers are unique, non-printable, non-changeable, and assigned automatically by the system.
object name	An external identification of an object, e.g. Person, Student, Company, Document. An object name is invented by the object model/schema designer to identify the object within a database schema or within an application program. Usually it is assumed that an object name is not unique or objects with the same name are enclosed within a named collection, e.g. Persons for Person objects.
object reference	An object identifier (OID) returned or used by some other feature, e.g. delete operation. Usually terms "object reference", "OID", and "pointer" depend on the context of documentation: "OID" is used when one talks on an object itself; "reference" is used when one talks on an operation on the object; and "pointer" is used when one talks on a data structure that stores OID.
ontology	A formal description of meta properties that is relevant for a particular business domain. In some current literature the term ontology substitutes (or is a synonym for) the term "metadata". In database terms, ontology denotes a mix of classical concepts related to metabase or catalogs (a schema repository) with some features that describe the status of information, e.g. authorship, ownership, security, privacy, representation, validity status, last updating time, etc. Some professionals from the AI camp assume that ontology has to be specified in the predicate logic.
personalisation	Mechanisms that customize the data/content shown to the user according to his/her preferences.
pictographic shape	A shape of the class diagram in which classes are represented by visual icons.
predictability, preserving the mental map	Predictability concerns the problem of graph layout. It means that two different runs of a graph layout algorithm, involving the same or similar graphs, produce similar layouts with no radically different visual representations.
presentation identifier	An attribute value or a set of attribute values used in the presented graph to identify/recognize an object. Presentation identifier need not be unique.
query	A statement of a query language.

query condition	A condition formulated in a query language.
query language	A high level language used to retrieve data from databases. Usually query languages must satisfy the following properties: declarativity, macroscopic processing, naturalness for the users, physical data independence, late binding, formal semantics, possibility of automatic optimization.
rendering	The process of representing content stored in a native format in some more uniform format (e.g. html to be presented in web browsers).
screen	An entire area of a computer display.
session basket	A basket with retrieval result that exists temporarily up to the end of the user session.
set of selected objects	A concept of intentional navigation: the set of objects that have been selected by the user through the following actions: navigation is a structural knowledge schema; determining selection conditions; choice of concrete objects.
starting point	An object which begins navigation.
structural knowledge	Knowledge incorporated in ontology-based structure of objects and relation among them.
Structural Knowledge Graph Manager (SKGM)	A package of utilities (collected e.g. as Java classes and interfaces) which enables the programmer, (system administrator, user) to retrieve proper data in a data/knowledge base, process it, and present on the screen as graphic metaphors that can be manipulated by the user. User operations on graphic metaphors (dragging, clicking, activating graphic elements, etc.) are mapped by SKGM to proper operations on database or on internal SKGM data structures (e.g. baskets). SKGM allows for introducing various parameters to the graphic environment, e.g. names that restrict the user view, graphic icons that will be displayed for the user, modes of user work, etc.
usability	Usability is a general term concerning the usefulness and user-friendliness. Usability is usually defined through sub-terms such as effectiveness, efficiency, user satisfaction, user awareness etc. Effectiveness refers to the extent to which the intended goals of interaction can be achieved (i.e. accuracy and completeness of data). Efficiency is connected with the time, the money and the effort spent on that activity. A measure of user satisfaction is for example: how comfortable she/he feels using the system, how easy-to-use the system is. The notion of user awareness is connected with the situation where a user can lose orientation while dealing with complex data structures (e.g. when navigating through large, complex graphs), thus some special facilities (graphical) are necessary to keep the user aware of the current sub-goals and/or sub-results of his/her search.
user awareness	The psychological state of the user in which he/she is fully controlling the goals, the actions and the results of his/her activity with the system.
user basket	A basket assigned to a particular user.
user context	Personalized user data stored in the SKGM database plus information on the current user session.
user interface	All elements that the user deals with during querying or other operation on a system. Graphical user interfaces consist of the following parts: visualisation of the data/object store, visualisation of database schema, visual queries or other operations on the store or on the schema; visualisation of the results of querying or other operations.
widget	Visual gadget: a graphical icon or a manipulation metaphor that the user deals with on the screen; for instance, a button, a menu, a scroll bar, etc.
view	A functional part of the user screen.
viewer	A module responsible for rendering and presentation of content.
visited object	An object participating in the user session which was the subject of some user action (e.g. navigation, display).
visual query language (VQL)	A visual interface allowing the user to ask queries to a database.

visualisation	Graphical presentation of data or knowledge stored in a data/knowledge base. Visualization may employ various graphical techniques (graphs, bars, pies, etc.) and various human perception metaphors.
visualisation system	A system that makes it possible to present data from the database in a visual form and/or to visually process/retrieve data presented in the visual form.
window	A rectangular functional part of the user screen that can be shown, hidden or screened in the user view.