Augmenting CASE Tools with Hypertext: Desired Functionality and Implementation Issues

Janne Kaipala
University of Jyväskylä
Department of Computer Science and Information Systems
P.O. Box 35, FIN-403351 Jyväskylä, Finland

Email: jka@jytko.jyu.fi

Abstract. Information systems have become bigger and more complex as their support has expanded to cover larger business domains, communication and work. At the same time technical design options such as client/server architectures and graphical user interfaces have increased the size and complexity of applications. In addition, pressures to build better systems more quickly have motivated the use of integrated design environments, such as CASE. Several integration approaches such as process modeling, frameworks and hypertext technology have been proposed. Of these we consider the least analyzed, hypertext technology, in this paper. Because of the novelty of hypertext in CASE there are several unresolved issues related to this approach. Present hypertext technology has been mainly applied to non-structured representations such as text, which is radically different from complex structured representations such as the diagrams and matrices used in CASE. CASE tools also imply that a design object has both representational and conceptual aspect, which has not been investigated in relation to hypertext. In this paper we discuss how hypertext can be incorporated into a metaCASE tool which uses all the common representation paradigms: diagrams, matrices and tables. We also report the implementation and architecture of such an environment.

1. Introduction

Information systems have become bigger and more complex while their support has expanded to cover larger business domains, communication and work. At the same time technical design options such as client/server architectures and graphical user interfaces have increased the size and complexity of applications. These changes have been reflected in design practices. First, design methodologies have evolved from text based ones to more complex ones containing multiple methods and representation paradigms. Secondly, CASE (Computer Aided Software/Systems Engineering) tools are being used more extensively by organizations to manage this design information.

A notable trend in CASE is the effort to integrate different types of design information and tools, to better support the management of complex design documentation. In current practice design is captured solely by semi- and non-structured specifications. Semi-structured documents, such as entity-relationship diagrams, follow the rules defined in the method, and they are generally represented using diagrams and tables. While semi-structured specifications capture the critical aspects of the system, non-structured information such as textual descriptions are still used for commenting about and arguing over design.

Integration of this information has been approached using strategies such as metamodeling and hypertext technology. The metamodeling approach aims at defining methods and providing formal integration mechanisms such as decomposition relationships between design objects. Hypertext functionality (Oinas-Kukkonen 1997b), in contrast, provides method-independent linkages between design objects which can be either semi-structured or non-structured. Examples of such linkages are links that support navigation between design diagrams, and links from design objects to issues concerning them.

Several attempts to provide hypertext linkages between design objects have been reported. Their basic features include attaching annotations to certain types of design objects and providing navigation between them. The major defect in the previous research is that the complexities of CASE information, such as the structure of design information and their different representations, have not been considered. Yet, when

designing hypertext functionality into CASE it has to be recognized that design objects have both conceptual and representational aspects. This has been noted as important since designers benefit from several representations of a concept, and different kinds of users need different viewpoints on the same information. Another issue is how to provide links into different types of representations forms used in CASE. Previous research into hypertext functionality has not addressed other representation forms than diagrams, although matrices and tables are also considered as necessary in systems design (e.g. Kelly et al. 1996). In this paper we illustrate how hypertext functionality can be implemented to provide consistent linkage between different types of representations while also respecting the structure of design information.

The first research issue, *to which aspects of design objects should links be attached,* arises from the division between conceptual and representational aspects of design information. The second problem of this paper is to investigate *how to represent links in CASE.* We have integrated a hypertext tool called Linking Ability (LA, Oinas-Kukkonen 1997a) with the MetaEdit+ metaCASE tool (Kelly at al. 1996) in order to systematically address the above research problems. Linking Ability is based on the concept of an intermediary hypertext agent (Kerola and Oinas-Kukkonen 1992) and hypertext functionality as defined by Oinas-Kukkonen (1997b). In this paper we follow Oinas-Kukkonen's classification of link types for CASE into annotation, argumentation, association and traceability links while considering the link attachment issue.

The paper is organized as follows. First, in Section 2 we review related work. Next, in Section 3 we introduce hypertext functionality in CASE. In Section 4 we illustrate the MetaEdit+ environment and its LA hypertext model. In Section 5 we discuss the link attachment, while in Section 6 we address link representation issues. Finally, the architecture and the implementation of the environment are described in Section 7.

2. Related work

Several hypertext approaches to CASE have been reported, though the research issues represented above are largely neglected in these works. Previous research has not systematically considered link representation issues and largely underestimates the complexities of CASE documents. In a nutshell, either the available functionality provides only hypertext features, or the hypertext support functionality is limited to some model areas. In Table 1 we compare our approach with related implementations.

Dynamic Design (Bigelow 1988) is a configuration management environment, which uses hypertext links for specifying configurations, attaching annotations and associating design objects and source code. Dynamic Design does not address conceptual and representational aspects of design objects, but allows link attachment to explicit coordinates. The HyperSoft (Cybulski & Reed 1992) environment offers tools for constructing software documents and creating navigable links among them at the representation level. PRO-ART environment (Pohl et al. 1994) allows the addition of design rationale and annotations to semi-structured design objects at the conceptual level through a process model. Consequently, links are automatically activated by a process model when design objects are manipulated instead of explict link traversal, thus making link representation consideration unnecessary. Hyper Analysis Toolkit (HAT, JingXiang and Griggs, 1994) connects description narratives to ER diagrams and vice versa, but neither discusses the conceptual and representation levels, or link representation.

Common to all implementations is that they link semi-structured information with non-structured information by using ad-hoc solutions specially suitable for diagram representations. In contrast, the approach suggested here implements hypertext

functionality into a CASE tool so that it allows multiple representation paradigms and definable methods.

Table 1: Hypertext features in CASE tools

	Dynamic Design	HyperSoft	HAT	PRO-ART	MetaEdit+
Semi-structured objects	Yes	Yes	Yes	Yes	Yes
Non-structured objects (text representations)	Yes	Yes	Yes	Yes	Yes
Concepts vs. Representation problematics addressed	No	No	No	No	Yes
Links to a position	Yes	Yes	No	No	No
Types of semi-structured representation paradigms considered	Diagram	Diagram (ER+)	Diagram (ER)	Diagram (ER, SA)	Diagram, Matrix, Table
Link representation in semi-structured objects addressed	No	No	No	No	Yes
Definable methods	No	Object types	No	No	Yes

3. Hypertext functionality in CASE

Hypertext functionality (HTF) provides a hypertext-like user interface for linking and navigation between design objects (Oinas-Kukkonen 1997b). A fundamental difference from pure hypertext systems is that hypertext links are used as a value adding feature. In addition to link creation and traversing, features such as bookmarks, history list and queries are used to support navigation. Hypertext functionality should enable linking any design objects independent of their structure and representation form.

The need for linking different types of design information is evident. Non-structured information such as annotations and argumentation components are needed in design work and should consequently be allowed to be attached to any kind of design document. Also links between non-structured pieces of information can be used to organize them into a consistent whole (cf. SGML/HTML documents). For example, to-do lists can be divided into sub-lists and then linked to each other. In addition, these non-structured to-do lists can contain links to semi-structured design documents. Moreover, links between semi-stuctured information can be added to support navigation. Consider for example how traceability mappings can be established between design objects to support requirements tracing.

By using hypertext terms, links form mappings between anchors (design objects), while a network of linked anchors is called a hyperspace. Link traversing is initiated by activating a link representation such as an underlined text or an image. A link's source is called a source-anchor and a link's endpoint is called a target-anchor, respectively.

3.1 Conceptual and representational information

While traditional hypertext systems are based on non-structured representations such as text and pictures, semi-structured information in CASE benefits from the differentiation between its representational and conceptual aspects. (e.g. Smolander 1991a, Tolvanen 1995, Wijers et al. 1990). This difference provides *representation independence* which has the advantage that conceptual design objects can exist independently of their alternative representations (Kelly at al. 1996). Therefore, design documents can be represented in text, matrix and graphical forms and moreover, the conceptual design objects used there can be shared. For example a 'Sales system' conceptual graph can have several representations (e.g. diagram, matrix, table, see Figure 1) and different representations of an 'Order' concept are

possible . While the conceptual side of a design object stores information that is representation independent such as the name of a Process in a Data Flow Diagram, the representation side contains properties such as the objects' position and appearance.

The above characteristics of semi-structured design objects is employed during design work which have to respect it when providing linking functionality. Consider a situation where a user is working on a 'Sales system' table and annotates an 'Order' by the statement '*Is this object needed?*'. In this example we must consider whether the annotation should be attached to the order in the table representation, or also in other representations where the order appears. In other words, we have to decide whether to attach the annotation to the conceptual or representational aspect of the design object.

3.2 Link representation

In CASE semi-structured information is often represented in the form of diagrams, matrices and tables, and the need for all these representation forms is undisputable. Diagrams are the most common representation as they are favoured by structured graphical methods, such as ISAC (Lundberg et al. 1981) and SA (Yourdon 1989), or object-oriented methodologies such as OMT (Rumbaugh et al. 1991), OOA/OOD (Coad and Yourdon 1990) and Fusion (Coleman et al. 1994). Matrix representations are useful for supplementing graphical diagrams, but are also used as the sole representation in methods like IBM's Business Systems Planning method (IBM 1975). The third main representation paradigm used by CASE tools is a table representation which is used in methods like Critical Success Factors (Rockart 1989) and Use Cases (Jacobson et al. 1992).

From a link representation point of view these three representation paradigms are not that new as they have been used e.g. in WWW pages defined in a HTML language. However, these hyperdocuments differ from systems design documents in that they are static, representation oriented and links form an essential part in constituting the documentation. Therefore, link representations in them can and need to be carefully designed. In contrast, CASE tools operate on "concepts" and their different representations, while links are secondary to the design objects which can change all the time. For these reasons we should not bother designers with link representation issues but instead should provide users with an automatic and consistent link representation scheme in three representation forms used in CASE.

4. Hypertext functionality in MetaEdit+

We use the MetaEdit+ CASE tool and a hypertext subsystem (LA) as a vehicle to demonstrate how we approach the above research questions. In the following we will review underlying representation paradigms (diagram, matrix, table) and corresponding tools. We also describe the underlying data model of MetaEdit+ and show how LA is used to link different types of information managed.

4.1 MetaEdit+ environment

MetaEdit+ is a multi-method, multi-user, multi-tool environment for computer aided software engineering. It establishes a versatile environment for creation, maintenance, manipulation, retrieval and representation of design objects (information) structured and created according to a method (Kelly et al. 1996).

The kernel of the MetaEdit+ functionality and architecture is determined by the underlying conceptual data model called GOPRR. MetaEdit+ uses the GOPRR conceptual data model as a universal and generic meta-metamodel i.e. as a sole language to specify methods (Kelly et al. 1996). The fixed conceptual meta-metamodel (see Table 2) forms the basis on which varied representations of data, including not only the usual graphic diagrams, but also text and matrices can be built.

The GOPRR data model makes a distinction between the representational and the conceptual aspects of a method, thus providing a suitable platform for studying hypertext support for different representations of CASE.

Table 2: GOPRR concepts

Graphs	Containers for Objects, Relationships and Roles.
Properties	Appear as textual labels in diagrams, contain single data entries such as a name, text field or number.
Objects	Appear as shapes in diagrams, contain properties and model concepts such as Entity in an Entity Relationship Diagram.
Relationships	Appear as lines between shapes in diagrams, contain properties, and model concepts such as Data Flow.
Roles	Appear as the end points of Relationships (e.g. an arrowhead), contain properties, and model the part an Object plays in a Relationship.

4.2 Model editing tools in MetaEdit+

Model editing tools form MetaEdit+'s key functionality from the users' point of view as these tools are used to create, modify and delete models and their instances. Moreover, hypertext links between conceptual and representational design objects are created and represented in these tools. All model editing tools are similar in sense that their main purpose is manipulating and creating models, but differ in terms of their focus and representational paradigm supported (Kelly at al. 1996).

4.2.1 Diagram Editor

The Diagram Editor offers the necessary functionality to create, modify and represent models graphically. An example of using the tool in creating a DFD diagram is shown in Figure 1.

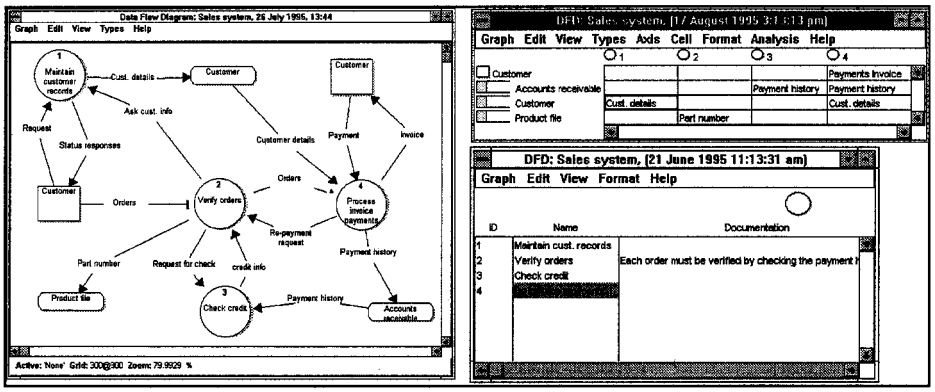

Figure 1: MetaEdit+ modeling tools

4.2.2 Matrix Editor

Matrix editor allows the user to adopt a different representational perspective on his graphical models. The user interface of the matrix editor is visually similar to that of a spreadsheet, with the axes containing objects, and the cells containing information about the relationships between those objects (Kelly at al. 1996).

In the example in Figure 1, the user has opened a Matrix Editor on a Data Flow Diagram of a Sales System (represented in the same figure). He has decided to show all the Processes (1–4) on the horizontal axis, and the Externals and Stores on the vertical axis, with symbols displayed for both. For example, the framed element 'Cust. details' on the left shows that there is a flow of customer details between the Customer Store and Process 1 (Kelly at al. 1996).

4.2.3 Table Editor

Table representation complements the matrix and diagram representations and it has two useful features. First, in the Table Editor design objects are represented as rows, and properties of the objects form columns. This provides a natural and economic way to view design information concerning multiple objects in a compact form. Thus, whereas matrix and diagrams represent the overall structure of the IS models and show connections between model components, Table Editor allows detailed browsing through individual model elements. A second useful feature is to support methods that are not graphical nor matrix based. Figure 1 shows an example of a model instance as seen through Table Editor. The underlying conceptual graph is the same that is shown as diagram and matrix, but it is represented differently. (Kelly at al. 1996).

4.3 Nodes and anchors

Generally, a hypertext network is constructed by linking either whole nodes or parts of them. For example, in HTML documents nodes can be structured using tags which define the linkable positions in documents. Similarly, in CASE tools one can manage design documents which are structured according to the methods used. In MetaEdit+ they are defined using the GOPRR meta-model. This implies that each design object of the used method is based on some GOPRR construct (cf. Table 2). Therefore, we have used the GOPRR data model as a basis to define the (meta-) structure of the hyperspace.

We consider representation graphs (diagrams, matrices and tables) as nodes in hyperspace. Furthermore, we consider Graphs, Objects and Relationships as anchors, setting the granularity of the hypertext nodes to a reasonable level but yet not limiting the usability. We do not consider Properties and Roles as anchors, but instead encourage designers to use objects that are associated with them. For example, if a user considers the issue "*Should this class be named as a manager or a controller?*", he will in all likelihood attach the question to the class object instead of the name-property (*name* is a property of the class). Similarly, if a user is considering a role he can use the relationship of which the role is a part.

In summary, in Table 3 we describe how GOPRR elements are represented by different tools. In the table "Representation" means that there is a representation for an element (and a concept behind it) and "Concept" means that only a concept is used for the representation. Stroked-through elements indicate non-linkable design objects.

Table 3: Anchors and nodes in GOPRR

Nodes/anchors ⇒ ⇓ Anchors	*Matrix*	*Table*	*Diagram*
Graph	Representation	Representation	Representation
Object	Representation	Representation	Representation
Relationship	Concept	Not represented	Representation
~~*Role*~~	Concept	Not represented	Representation
~~*Property*~~	Concept, partially	Concept	Concept, partially

4.4 Linking Ability -hypertext model

LA (Linking Ability) is a stand-alone hypertext system with annotation, argumentation and model linking capability (Oinas-Kukkonen, 1997a). It supports linking of semi-structured information (GOPRR data model concepts) and non-structured information (annotations and design rationale nodes). The basic features of the LA implementation include creation, deleting, navigation and querying of links. The design rationale system of the environment is described in Oinas-Kukkonen (1996).

The design objects (i.e. semi-structured and non-stuctured information) are linked using four types of links, namely annotation-, debate-, association- and traceability links. Annotation and debate links lead to non-structured design objects, while association and traceability links lead to semi-structured design objects managed by MetaEdit+ tools (see Table 4). All the links can start from any design object, such as a piece of text in an annotation or a Process in a DFD diagram.

Table 4: LA links and node types

	Source node	*Target node*
Annotation link	Any design object	Annotation
Debate Link	Any design object	Debate node
Association link	Any design object	GOPRR design objects
Traceability link	Any design object	GOPRR design objects

In addition to links, LA offers bookmarks and landmarks for information location. They enable users to "mark" locations that are visited often or are otherwise meaningful (Oinas-Kukkonen 1997a). Landmarks mark central places in the design, whereas a designer can embed a bookmark to a diagram while he interrupts his work, to support fast refocusing to his unfinished design.

Landmark and bookmark lists provide quick jumps to these marks. For this reason they are considered as links from anywhere to the source-anchor. Consequently, we include them in our discussion of link attachment.

5. Linking considerations

As noted, linking of semi-structured information presents a problem due to its conceptual and representational distinction. In our environment semi-structured information can form both source and target anchors of links. For example, all links have semi-structured design objects as their *sources* and also *targets* of association and traceability links are semi-structured objects. In the following we will demonstrate that links are useful both at the conceptual and representational levels of semi-structured information managed in CASE. Thereby, for each link type the question "*Is a meaningful (source/target) anchor for this link type a concept, a representation or both?*" will be answered.

Before we judge whether links are needed on both the conceptual and the representational level we have to be aware of link representation and traversing conventions. Since a conceptual object forms the foundation of all its representations, links whose source is a concept are displayed in the context of their representations. For example, if a link is attached to the Order concept in Figure 1, the link will appear in matrix, table and diagram representations of the Sales System. The link traversing is affected, too, since when a user follows a link whose target is a concept, the system has to request which representation he prefers.

The user's decision whether to link to concepts or representations depends on the desired link semantics and the desired impact on user interface. The semantic aspect is considered when we want to explicitly indicate that a link anchor is a *certain* representation or a *plain* concept, while the user interface criteria is used when users

want to limit the scope of a link for some reason. For example, when designers are considering particular representations' during the design activity they benefit from links explicitly attached to those representations. This makes navigation easy as the link whose anchor is a representation leads directly to the representation without asking the user to choose between all possible representations.

5.1 Debate and annotation links

Debate links connect mostly to the conceptual level design objects. For example, there can be questions concerning properties of a design object, or the implementation responsibility of an object. These questions can rarely be considered as attached to a particular representation. However, a question concerning a particular representation graph (diagram, matrix, table) will be attached to its representation. Consider for example a debate link: "*Who should review this diagram?*" (see Table 5).

In a similar vein, uses for annotation at the representation and conceptual levels are easy to find, as we often make notices, comments and suggestions about design objects. For example, we can provide information that is useful in the implementation of a concept (see Table 5). Although annotations are mostly attached to concepts, a general example of an annotation link at a representation is a comment on the quality of a diagram.

Table 5: Examples of use of debate and annotation links

Link type *Source*	*Debate link*	*Annotation link*
Concept	"Who should implement the *Customer* browser?"	"We need an efficient sorting algorithm here. A review on algorithms can be found in http://alg.orit.hms
Representation	"Who should review this (Sales System Architecture) diagram?"	"This diagram needs restructuring"

5.2 Association and traceability links

Designers benefit from associative connections while engaged in design activity, which demands visiting a set of design objects. Consequently, a designer will reference these objects more often than other objects. Association links provide fast navigation between these objects, since navigation can be initiated by activating a link symbol instead of looking for design objects in a browser.

Association links are useful in both representational and conceptual levels. As discussed before, a link leading to the representation level of a design object provides direct access to the target representation while a link to the conceptual level provides a user with a possibility to choose between alternative representations that a concept has. Associations between tightly related concepts are essential in providing easy navigation between all different representations, but they are useful also when used from a concept onto itself. A link from a concept onto itself constitutes a linkage between all representations of a user's problem related concept, enabling easy navigation between its representations (enabled by the link representation strategy presented earlier).

Association links are useful also at the representation level. A user can create a network of representations that are needed to carry out a design task, or he can create an "index" annotation where links exist to the problem related representations. Association links from concepts to representations and vice versa are needed. A link from a concept to a certain representation is useful when a designer prefers a

representation that is central to his *own* current problem (and also his design problems in general). Therefore, designers must be allowed to use explicit links to representations that he prefers. The analysis of association links is summarized in Table 6.

Table 6: Examples of uses for association link

Source \ *Target*	*Concept*	*Representation*
Concept	"To Order" (a link between related concepts)	"To Order" (in my own purchase diagram)
Representation	"To order" (the user does not prefer any representation)	"To Customer in this diagram (that is currently not visible)"

A traceability link differs from an association link in that there is certain semantics behind it, although they are used for navigation. A common usage for a traceability link is "forward" tracing between requirements and a design to ensure that a certain requirement has been met. For example, consider the requirement statement (concept) "*Credit will be given only to regular customers*" that is linked to a *customer* class in an object diagram where the method '*Check for credibility*' is introduced. There can also be backward traceability links from design objects to requirements, which allows tracing all requirements related to a certain design concept. This implies creating traceability links from a design concept to a requirement concept (see Table 7).

Whether to link 'customer' concepts or representations depends on the preferred granularity. On the one hand, a requirement should be logically linked to the conceptual "instance" of a design object to indicate that a requirement is related to that *concept* and all its representations. On the other hand, we can benefit from a more fine-grained traceability relationship to a representation to indicate that the requirement is met by a certain representation of an object.

Table 7: Examples of uses for a traceability link

Source \ *Target*	*Concept*	*Representation*
Concept	Trace where a requirement is implemented	Trace the exact representation that implements a requirement
Representation	Trace to any version of '*Order*'	Go up/down the evolution trace of the dialog of '*Order*'

Traceability links between representations can establish evolution traces of design objects through different design phases. For example, different representations of a customer object can be linked into a chain that can be traversed through links.

5.3 Landmarks and bookmarks

Landmarks can support marking of either concepts or representations. For example, an architecture diagram (representation) can easily be located when a landmark is attached to it (see Table 8), while a landmark attached to a concept provides the user a possibility to choose between alternative representations.

Table 8: Examples of the use of landmark and bookmark

Link type *Source/Target*	*Landmark*	*Bookmark*
Concept	A guide to the system architecture (diagram, matrix or table)	"Sales system"
Representation	A guide to the system architecture diagram	"In the morning, start here"

Bookmarks are used similarly to landmarks but are temporary in their nature. Table 8 illustrates examples of use on both conceptual and representational level. We can attach a bookmark to a Sales System's conceptual or representational graph if they are frequently visited. A user might also embed a bookmark to a graph (concept) while he interrupts his work, to help fast refocusing to his unfinished design.

5.4 Default link attachments

One fundamental contribution of our solution to hypertext functionality is that for all LA link types we can simultaneously link concepts and representations. This affects the link *creation* procedure, since we have to ask a user for detailed information whether he wants to link to a concept or its representation. For example, a user may activate an 'Order' in an ER-diagram to indicate a link anchor which however does not indicate whether he "means" the concept or its particular representation. This problem can be partly overcome by automatically creating a link either to the concept or its representation. This suggests that we have to consider which one alternative should be a default in each link type.

Debate and annotation links seem to be mostly attached to conceptual anchors, though attachments to a representation are sometimes handy, especially when considering graph representations. Yet, we suggest that a default attachment type for a debate and an annotation link is a concept. In case of association and traceability links it seems that the default attachment type can be either a concept or a representation. There is no evidence that landmarks and bookmarks are more often used in either concepts or representations. Intuitively, they are used to mark a *certain* place (i.e. representation) that a designer intends, which suggest a default attachment to be a representation.

Overall, the most successful way to determine a default attachment is to empirically study how users apply these links in different contexts. It is possible that a link attachment is a personal and method-related issue and therefore a user should be allowed to tailor the default source and target types. In addition, a user should be able to change the source and target type of a link on demand.

6. Interface

6.1 Link representation considerations

A link marker indicates visually that there is a link connected to an anchor. If there are several links attached to a single anchor there are two representation alternatives. We can either use one link marker in conjunction with a separate list that provides the links or we can show each link's marker separately besides its anchor. Separate markers are advantageous since a user perceives them immediately and also a descriptive text for a link can be shown. However, this approach is deficient in the case of compact structured representations such as a matrix and a table, where possibilities for tailoring the representation are more limited. Therefore, we use one link marker to indicate the existence of links. In addition, we suggest that if there is

only one link a symbol indicating the link type should be shown. In the current implementation we use a figure to indicate the number of links, although in future we plan to use symbols.

6.2 Link placement

Link markers need to be positioned in different representation formats of MetaEdit+. The main principle is to show a link marker near the link's anchor (a graph, an object or a relationship). Therefore, we first have to consider anchors' representation in different types of representations (see Table 9). Another issue is to analyze what alternatives there are for positioning their links.

Table 9: User interface elements that contain GOPRR constructs

	Matrix editor	**Table editor**	**Diagram editor**
Graph	Top left corner	Top left corner	Top left corner
Object	Row/Col Label	Row Label	Spatial symbol
Relationship	Cell	-	Spatial Symbol

Although graphs are represented in different formats such as matrices, diagrams and tables, we strive for consistency in representing links. Therefore, we suggest that a link marker is shown in a separate area in each tool. Accordingly, we display the link marker on the top right corner of matrix, table and diagram windows.

In a diagram editor link markers can be represented easily. Because of the relatively low density of the representation, we can choose to represent the link marker on the top of anchors (a design GOPRR object or relationship). This solution is adequate although it must be noted that the link marker may collide with other representations such as relationship lines. An example link is illustrated in Figure 2(1) where two links ('2') have been attached to the 'Link' object.

Table and matrix representations differ from diagram representations in that their layout is more dense and structured. Design objects are laid on rows and cells, enabling the link placement only on the right or left side of design objects. In tables the anchors (GOPRR objects) consist of a whole row each, thus we display the link marker on the left side of an anchor. An example is illustrated in Figure 2 where the 'Link' object in the table has two links. Similarly, in matrices link markers are represented on the left side of their GOPRR object-anchors residing in rows and columns. An example of this is illustrated in Figure 2, where e.g. the 'Node' object has one link.

In contrast to these "easy" cases, displaying relationships' link markers in a matrix is problematic. In a matrix representation cells are used to display relationships between objects within corresponding rows and columns. The problem arises when the number of relationships is so high that relationships do not fit into the cell. In a tool implementation this can be alleviated by providing an additional 'cell browser' that displays any number relationships. We chose to display the link markers on the left side of the relationships on a cell or in a browser.

6.3 An overview on the user interface

The user interface implemented is illustrated in Figure 2, which depicts a typical design scenario. In the case a designer has developed a hypertext system, and uses diagram (1), matrix (2) and table representations (3) to describe its node-link model. During the design he documents design problems (4) and has developed a 'Guide for visitors' (5) for accessing central aspects of the design. In addition, he has left two landmarks in his design, one in the guide and another in the node-link model graph.

Now, consider a situation where another designer is interested in the design. He examines a landmark list (6) where by clicking the 'Guide for visitors' link he opens

the guide containing further links. In the guide he clicks the 'See the node-link model' link and a dialog requests him to choose whether to open a diagram, a matrix or a table representation (dialog not visible in the figure). He chooses the diagram representation and observes that two links are attached to the 'Link' object. He opens the link list (7) by clicking the link marker ('2') from where he navigates to a question: 'How to represent links?' (8), and adds his own contribution to the ongoing discussion.

In this scenario the idea of linking both the conceptual and representational aspects of objects can be seen as follows. The link to the node-link model has been attached to the conceptual graph, and thus a dialog asks the user to choose between representations instead of opening a particular representation (e.g. the matrix). Similarly the debate links (see the link list) have been attached to the conceptual level of the 'Link', thus these links (their marker) are visible beside the 'Link' in matrices and tables also. In contrast, one link is attached to a particular 'Node' object representation in a matrix and thus it does not appear in the diagram and table representations.

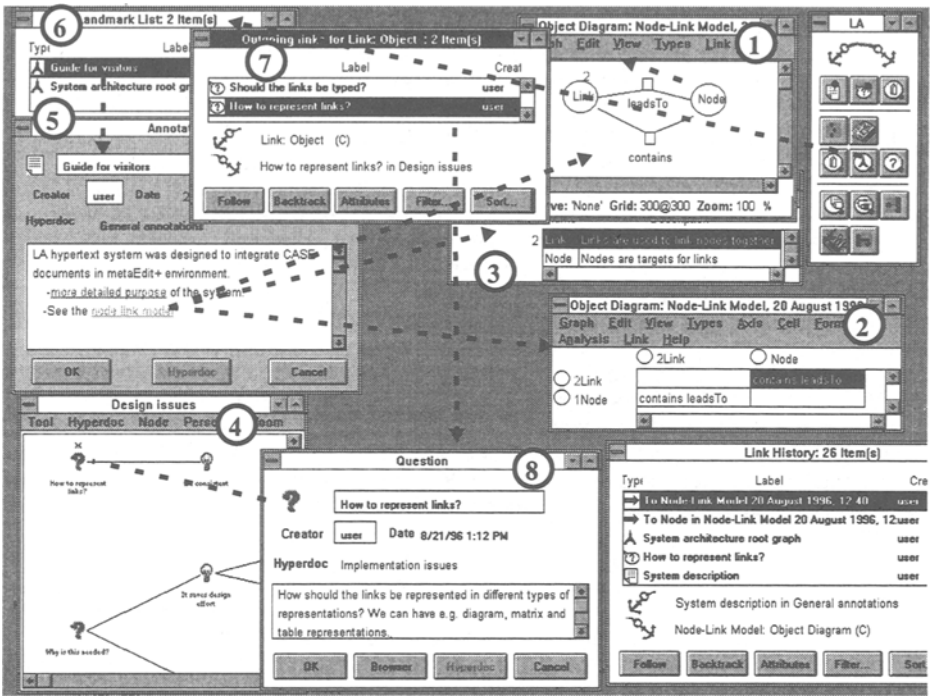

Figure 2: MetaEdit+ user interface

7. Architecture

Both MetaEdit+ and LA have been implemented using VisualWorks™ Smalltalk and object persistency added using the ArtBASE™ repository. The design has been carried out using an object oriented approach and prototyping.

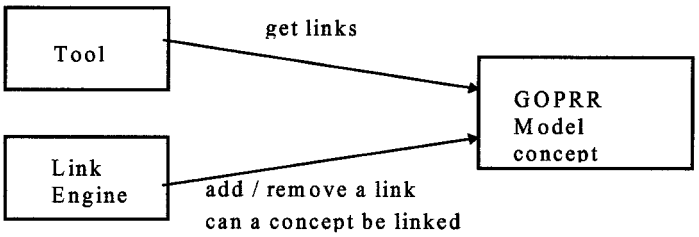

Figure 3: *Message relationships between the system's components*

The linking is carried out by LA objects (Link Engine in Figure 3), while the link representation is handled by each MetaEdit+ tool whose concept instances have links. The generic architecture is illustrated in Figure 3, where the boxes denote objects and the arrows denote messages between them. Links to and from the GOPRR concepts are created by the Link Engine. It communicates with the GOPRR data model concepts by confirming linking requests and by adding and removing links from them.

8. Conclusions

In this paper we have shown how the use of a hypertext functionality can be extended to deal fully with normal CASE tools. A metaCASE tool called MetaEdit+ was enhanced to allow the representation of links provided by the Linking Ability hypertext system.

Traditional hypertext systems operate only on one level: there are no concepts behind the representations the user sees. In CASE, design diagrams use several semi-structured representation paradigms, and the design objects have both representational and conceptual aspects. In this paper we suggested how to represent links in representation paradigms such as a diagram, matrix and table. To achieve a uniform link representation in all tools, a single link marker was applied to indicate the existence of a link. In matrix representations link representation is problematic since due to space problems, some anchors may not be visible.

Our study also shows that it is necessary to attach links to both concepts and representations. To relieve the user from selecting the type attachment, we suggested default strategies for debate and annotation links. This, however, is a rough solution and more empirical evidence is needed.

Ideas presented in this paper can be generalized into most CASE tools, since the underlying GOPRR meta-metamodel is expressive enough to cover most methods supported by CASE tools. Moreover, three representation paradigms and the division between the conceptual and representational aspects are central issues in considering any hypertext support for a CASE tool. The results can also be utilized in any task area, where diagrams, matrices and tables are used simultaneously. For example, drawing tools and spreadsheets would benefit from a similar approach.

In future default link attachment to concepts versus representations should be investigated empirically. There is also a need to examine multi-user functionality and concurrency control with the hypertext functionality — an issue which is rarely touched on in hypertext research.

Acknowledgements

I want to express my thanks to the members of the MetaPHOR project for their essential comments.

9. References

Bigelow, J. (1988), Hypertext and CASE. In IEEE Software, March 1988. pp. 23-27.

Coad, P., E. Yourdon (1990), Object-Oriented Analysis, Englewood Cliffs, New Jersey (1990).

Coleman, D., P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes and P. Jeremaes (1994). "Object-Oriented Development The Fusion Method," Prentice Hall, Englewood Cliffs (1994).

Cybulski, J. L., Reed, C. (1992), A Hypertext Based Software Engineering Environment. In IEEE Software, March 1992.

Gane, C., T. Sarson (1979), "*Structured Systems Analysis: Tools and Techniques,*" Prentice Hall, Englewood Cliffs, NJ (1979).

IBM (1975), IBM Corporation, "Business Systems Planning — Information Systems Planning Guide," Publication #GE20-0527-4, IBM (1975).

Jacobson, I., M. Christeson, P. Jonsson and G. Övergaard (1992), "Object-Oriented Software Engineering — A Use Case Driven Approach," Addison-Wesley, Reading, USA (1992).

JingXiang, Griggs (1994), A Tool for Hypertext-based Systems Analysis and Dynamic Evaluation. Proceedings of the 27th Annual Hawaii International Conference on System Sciences 1994.

Kelly, S., Lyytinen, K and Rossi, M. (1996), MetaEdit+: A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In: Proceedings of the 8th international conference on advanced information systems engineering, CAiSE'96. Heraklion, Crete, Greece, May 1996.

Kerola, P. and Oinas-Kukkonen, H. (1992), Hypertext system as an intermediary agent in CASE environments. IFIP WG 8.2 Confrence on The Impact of Computer Supported Technologies on Information Systems Development, Minneapolis June 15-17, 1992.

Lundeberg, M., G. Goldkuhl and A. Nilsson (1981), "Information Systems Development: a systematic approach," Prentice-Hall (1981).

Oinas-Kukkonen, H. (1997a), Towards Greater Flexibility in Software Design Systems through Hypermedia Functionality, Information & Software Technology (forthcoming), 1997.

Oinas-Kukkonen, H. (1997b) Embedding Hypermedia into Information Systems, Proceedings of the Thirtieth Hawaii International Conference on Systems Sciences (HICSS 97), January 1997.

Oinas-Kukkonen, H. (1996) Debate Browser - an Argumentation Tool for MetaEdit+ environment. Proceedings of the Seventh European Workshop on next generation of CASE tools (NGCT'96), Creta, Creece, May 1996

Pohl, K., R. Dömges and M. Jarke (1994), PRO-ART: PROcess based Approach to Requirements Traceability, in Poster Outlines: 6th Conference on Advanced Information Systems Engineering, Utrecht, Netherlands, June 1994 (1994).

Rockart, J (1979), "Chief Executives Define Their Own Data Needs," Harward Business Review 57(2) (1979).

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy and W. Lorensen (1991), "Object-Oriented Modeling and Design," Prentice–Hall, Englewood Cliffs, NJ, USA (1991).

Smolander, K., Lyytinen, J., Tahvanainen V-P., Marttiin P. (1991a), MetaEdit — A Flexible Graphical Enivnronment for Methodology Modelling. In advanced information systems engineering, Proceedings of the third international conference CAiSE'91. Trondheim, Norway, May 1991. Springer-Verlag, Berlin 1991.

Smolander, K. (1991b), OPRR: A models for modeling systems development methods. Licenciate Thesis WP-20, Univ. of Jyväskylä, Finland. 1991.

Tolvanen (1995), Flexible method adaptation in CASE — the metamodeling approach. Licenciate Thesis, Computer science and information systems report, TR-5. Univ. of Jyväskylä, Finland. 1995.

Welke, R, J. (1992), The CASE repository: More than an Another Database Application. Cotterman, W and Senn, J.A. (eds.). Challenges and strategies for research in systems development. Wiley 1992.

Wijers, G. M, Hofstede, A. H. M., van Oosterom N. E (1990), Representation of Information Modelling Knowledge. SOCRATES. Report 90/09, November 1990. Software Engineering Research Centre 1990, Utrecht.

Yourdon, E (1989), Modern Structured Analysis, Prentice-Hall, Englewood Cliffs, NJ, USA (1989).