TDE: Supporting Geographically Distributed Software Design with Shared, Collaborative Workspaces

Antero Taivalsaari
Sami Vaaraniemi

Nokia Research Center
Software Technology Laboratory
P.O. Box 45, 00211 Helsinki, Finland
{taivalsa, vaaranie}@research.nokia.com

Abstract. Telecom Design Environment (TDE) is a software product that combines three challenging technology dimensions: 1) *Information management dimension*: TDE allows direct, interactive, visual management of documents and other design information in a shared, versioned, graphical working space; 2) *CASE tool dimension*: TDE provides graphical design tools for supporting different design notations that are helpful in designing and communicating about software systems; 3) *Collaboration dimension*: TDE serves as a virtual whiteboard system that allows designers to share the same graphical working space location-transparently, and to work on the same graphical designs even at the same time if necessary.

The key focus area in TDE is *team collaboration*. Unlike most other information management systems, TDE never locks anything, but it allows multiple designers to access the same diagrams and designs simultaneously, providing notification services for keeping multiple designers' work up to date. In a way, TDE can be seen as a "writable, collaborative web" that provides many of the same services as the World-Wide Web, except that TDE is always fully interactive, supports shared, direct data manipulation, and enables online collaboration with the other persons using the system. TDE is based on client-server approach and distributed object-oriented database technology, and it utilizes metamodeling technology to achieve better flexibility and tailorability.

1 Introduction

One of the most critical issues in large-scale software development, especially in the telecommunication business, is the rapidly growing size and complexity of the systems. As a result of the rapid growth, there is a need to understand and manage an ever-increasing number of system versions and configurations consisting of thousands of software and hardware components that must be able to interact and cooperate with each other in meaningful ways. Such understanding and management is becoming increasingly difficult without proper support for system *visualization*. Consequently, the availability of easy-to-use, graphical design and reverse engineering tools is becoming essential.

Another characteristic need in large-scale software system development is the support for *designer collaboration.* Due to the growing complexity of software systems, the components of these systems cannot be developed in one location, and designers frequently need to collaborate not only with their immediate peers, but also with colleagues located in other departments, cities or even countries. This requires that the designers be able to effectively share and work on the same design information (documents, diagrams, code, test material, etc.) in a collaborative, location-transparent manner. Such a way of working is supported poorly by current software engineering tools and environments, especially when it comes to sharing non-textual design information.

We have developed a system called *Telecom Design Environment (TDE)* that has been designed and implemented specifically to address the above mentioned issues. TDE is a new kind of a system that combines many challenging, but previously more or less unrelated technology dimensions, including information management dimension, CASE tool dimension, and collaboration dimension. The key objective of TDE is to support geographically distributed, location-transparent collaboration of software designers. This is made possible by providing the designers with visual workspaces that can contain various types of design information (graphical diagrams, text, pictures, links to documents, hyperlinks to other structures within TDE, sound files, and so on), that can be shared freely between the users, and that can be accessed and manipulated jointly by multiple designers. In a way, TDE can be characterized as a "writable, collaborative web" that provides many of the same services as the World-Wide Web, except that TDE is always fully interactive, supports shared, direct manipulation of visual information, and enables online collaboration with the other persons using the system. The TDE system is currently in pilot use in several business units of Nokia, and the experiences with it so far have been very positive.

In this paper we provide a general introduction to TDE. We start by discussing the conceptual background behind TDE, including the reasons and aspirations that lead to its development. Then we present the key ideas and principles of TDE, taking a look at the user interface principles and the features that TDE provides to the users. Some usage scenarios are also given, followed by a brief discussion on TDE implementation issues. Finally, a summary of related work is provided.

2 Motivation and Goals

The TDE system discussed in this paper has evolved as a result of our experiences and observations with many commercial software engineering environments and CASE tools. The following pitfalls characterize many of them:

- *Visual programming pitfall.* Many CASE tools are marketed using arguments such as "fully automated development from specification to implementation", or

"not a single line of code has to be written". In practice, most software design methodologies are not formal enough to support full-fledged code generation, and, if extended to support that end, the visual notations become hopelessly space-consuming. What many tool vendors seem to forget is that from the organizational perspective the primary reason for using design methodologies and visual design notations is not the aversion of programming but the facilitation of communication: common design methods and notations serve as communication vehicles that help designers communicate more clearly and concisely about their work and to understand the role of their work in a larger context.

- *Process enforcement pitfall.* Many tools seem to assume that there exists a certain "ideal" software development process that must be adopted when taking the tool into use. In practice, most projects already have an established set of work processes, methods and practices that cannot be changed just in order to take a certain new tool in use. Thus, tools should be flexible enough to be able to adapt to current work practices.

- *Legacy system pitfall.* Many tool vendors fail to recognize that in most software projects there exists a large body of existing design information that cannot easily be exported into newer systems and that must still be accessible even though new tools are being taken into use. Thus, the tools should provide interfaces that allow information from existing systems to be accessed whenever necessary.

- *Integration pitfall.* Especially in the 1980's there was a tendency to assume that all the phases in software development, including the development aspects, project management aspects, and process management aspects could be covered with a single integrated environment. In reality these integrated environments have mostly failed to meet their promises, and in most situations well-focused, independent, less formal tools have proven to work much better, at least as long as the number of tools remains moderate and if there is not too much redundancy in the roles of the tools.

- *Buzzword pitfall.* Many CASE tool vendors have a tendency to run after the latest buzzwords and trends in the software engineering field, and as a result implement all kinds of awkward new features that provide little added value to the users. In contrast, the really important aspects in supporting large-scale software development, such as the support for accessing information in existing systems and designer collaboration, are often forgotten.

The deficiencies in current software engineering environments and tools have led us to adopt certain key principles in the development of the TDE system. The most important principles are summarized in the following list.

- *Focus on design rather than programming.* In TDE, we try to avoid the above described pitfalls by emphasizing that TDE is primarily a *design environment* rather than a programming environment. In other words, rather than focusing on programming-oriented features such as automatic code generation, integrated compiling and debugging facilities, syntax-oriented editing, and simulation capabilities, the main emphasis in TDE is on facilitating communication among designers and supporting collaborative work.

- *Focus on representation and communication rather than formalization.* The success of large software development efforts relies largely on how easily designers can understand each others' work and how well they can communicate with each other in general. When the designers communicate, they do not often use very formal notations. Rather, the majority of real design work takes place at a rather informal level, e.g., in the form of informal meetings, gatherings, coffee break discussions, and so on. Conventional systems are poorly suited for capturing the informal aspects of the design work, and therefore a great deal of essential design information is easily lost. A key goal for TDE is to serve as a collaborative media that allows the designers to effortlessly and clearly express what they really mean, regardless of whether the information is formal, semi-formal or informal.

- *Going away from document-driven design.* Currently software design -- and especially the design of large-scale systems -- is often very document-driven. The primary visible results of specification and design work are documents, which means that a great deal of the design decisions that have been made during the actual design work is often lost. Also, different workproducts such as documents, design diagrams, code and test material are often maintained separately, meaning that the relationships between them are unclear and difficult to trace. TDE aims at replacing the document-driven design approach with a more interactive, collaborative approach in which design information -- whether contained in TDE itself or in legacy systems -- is represented primarily visually, and all the information is accessible to the designers location-transparently.

- *Supporting location-transparent teamwork and informal collaboration.* Most CASE tools today do not provide any support for teamwork, and if they do, they usually rely on rather formal and restrictive approaches to designer collaboration; e.g., use a conventional check-out/check-in model in which diagrams must be explicitly and exclusively checked out from the repository. This approach does not foster true designer collaboration and is often counterproductive, since the designers have a tendency to keep their designs in their own private work areas until the very last minute.

In summary, we feel that one of the foremost goals for a good software design environment is the ability to *lower the communication barrier* between designers, i.e., to make it easier for the designers to communicate about their work, to find and

understand the other designers' work, to contact each other, to work on the same designs regardless of their physical location, and generally to facilitate communication and cooperation via powerful visualization and collaboration techniques. Representation and communication aspects are thus considered more valuable than formal aspects of software engineering. Generality is also very important, and access to existing legacy information must be provided.

3 Central Concepts and Features

TDE (Telecom Design Environment) is a project that is aimed at producing easy-to-use, graphical design and reverse engineering tools for Nokia's software developers. The central goal of the project is to build an advanced, collaborative software design and reverse engineering environment that caters to the needs of the developers of large-scale telecom systems such as switches, network management systems, and network elements. The main result of the project is a new kind of a software product that combines three challenging, but previously somewhat unrelated technology dimensions:

1) *Information management dimension.* TDE allows direct, interactive, visual management of documents and other design information in a shared, versioned, graphical working space.

2) *CASE tool dimension.* TDE provides graphical design tools for supporting different design notations that are helpful in designing and communicating about software systems.

3) *Collaboration dimension.* TDE serves as a virtual whiteboard or "liveboard" system that allows designers to share the same graphical working space location-transparently, and to work on the same graphical designs even simultaneously if necessary.

A key focus area in TDE is *team collaboration.* Unlike most other information management systems, TDE avoids locking and exclusive access as much as possible, and allows multiple designers to access the same visual diagrams even simultaneously, providing notification services for keeping multiple designers' work up to date. In a way, TDE can be seen as a "writable, collaborative web" that provides many of the same services as the World-Wide Web, except that TDE is always fully interactive, supports shared direct manipulation, and enables online collaboration with the other persons in the system.

TDE is based on client-server approach and object-oriented database technology, and it utilizes a technology called *metamodeling* to achieve more flexibility and tailorability. In the rest of this section the most central features of TDE are described. For brevity, some features such as the versioning capabilities have been left out.

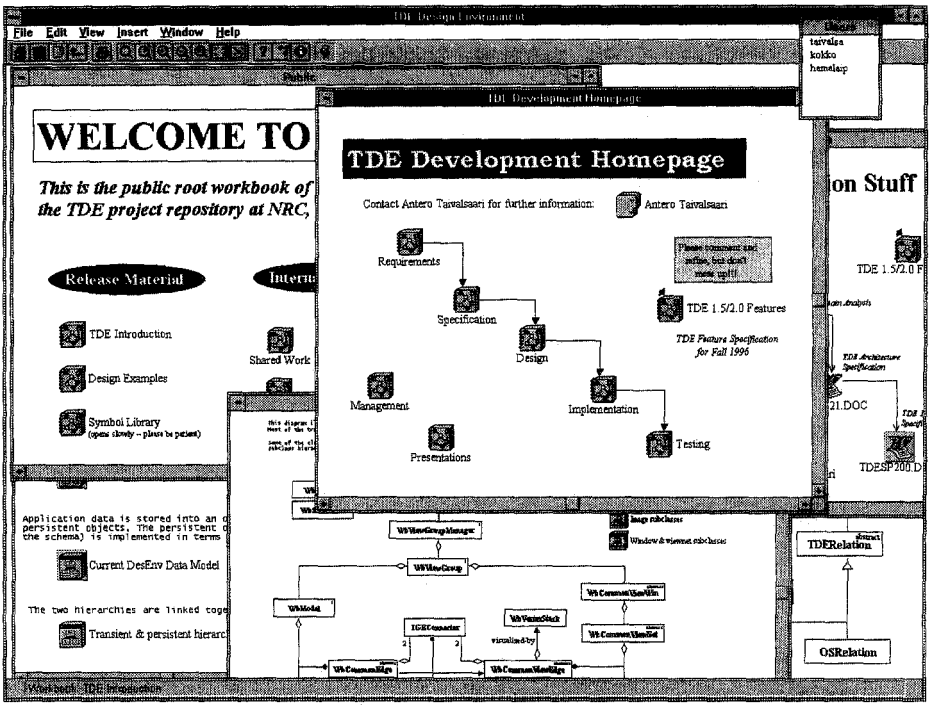

Fig. 1. Telecom Design Environment

3.1 Workbooks

The most central concept in TDE is that of a *workbook*. TDE workbooks are large, flat, shared, graph-structured, versioned work areas or "sheets" that can be used for holding and representing various types of information, including text, pictures, sound, graphical diagrams, references to files in external systems, references to WWW pages, and so on. Workbooks can be linked flexibly with each other, can contain links to arbitrary workproducts, can be edited and annotated freely (even by multiple persons at the same time), and can be shared by the designers in a location-transparent manner. In general, *icons* in workbooks represent workproducts and other design entities (such as documents, design diagrams, code files, screen snapshots, and so on) and *links* (arrows) between them the different kinds of relationships between the workproducts.

In a way, workbooks are similar to *folders* in graphical file systems or browsers such as the Macintosh Finder or Microsoft Windows File Manager. However, workbooks offer capabilities that go beyond the capabilities of these conventional systems, such as the freely editable & persistent layout, extensive linking capabilities, property sheet & documentation (annotation) mechanisms, versioning support, zooming capabilities, and collaborative editing support (the ability to edit workbooks simultaneously by multiple designers). Additionally, workbooks provide the ability to

include and transparently access files stored in external configuration management systems. Workbooks are versioned in order to ensure that old versions of designs cannot be removed accidentally and that an audit trail of the evolution of the designs can be maintained.

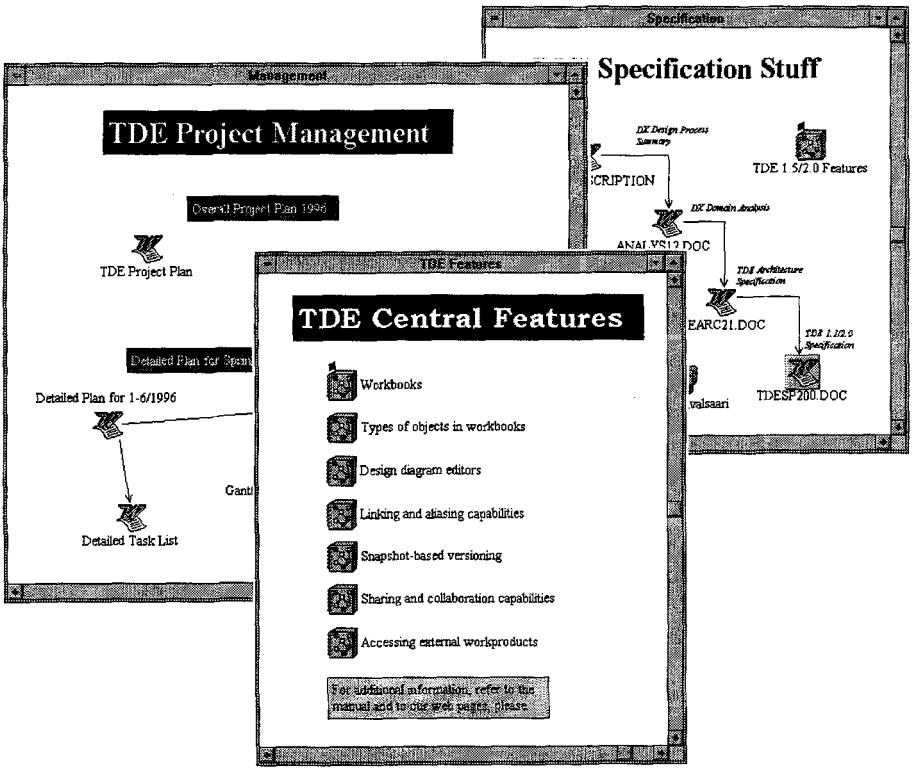

Fig. 2. Workbooks

A central aspect of workbooks in TDE is that they can be *shared* among users. Each user can have any number of workbooks stored in TDE repository along with the other users' workbooks. Connections between workbooks prepared by different designers can be created flexibly to construct different kinds of hierarchical or non-hierarchical networks of interrelated design information. Also, each workbook can contain links to arbitrary external workproducts (e.g., documents, source code files, test material) that can be opened upon request.

In typical usage, workbooks serve as kind of "visual project plans" that allow the designers to build visual descriptions of the information belonging to a certain (sub)project or design phase. By linking together workbooks describing individual design phases, larger and larger designs can be created. Thanks to the availability of aliasing capabilities, the same pieces of designs can easily be described from multiple perspectives. The users can navigate in these designs using context-sensitive menus and other well-known GUI mechanisms.

3.2 Characteristic Features of Workbooks

The following list summarizes the central high-level user interface conventions and visual principles used in TDE workbooks.

- *Graph-based representation.* TDE workbooks are essentially visual containers that can hold different kind of information represented in a graph-based format (see Figure 2). Icons in workbooks represent objects, while links between the icons denote relationships between the objects. The semantics of workbooks have not been fixed in advance, and thus they can be used for representing basically any kind of information. Workbooks can be infinitely large, and zooming capabilities are provided for examining their contents in varying levels of detail.

- *Editable, persistent layout.* The layout of workbooks is fully editable, allowing the users to group related information close to each other and otherwise utilize the spatial dimension to assist the visualization of design information. Also, workbooks are persistent, meaning that when a workbook is closed, its layout will persist so that when it is reopened later, it will look exactly the same as it did upon closing, unless somebody else has modified the workbook meanwhile.

- *Context-sensitive menus.* Context-sensitive menus are used throughout the environment. Each object and relation in workbooks and design diagrams has a specific menu that makes available the operations that can be performed on that particular object.

- *Property sheets.* Every object stored in TDE workbooks has an associated property sheet that provides additional information about the object. Some of the information in property sheets, such as the creation and modification information, is maintained automatically by the system.

3.3 Types of Objects in Workbooks

TDE workbooks can contain a variety of different kinds of objects, each of which has a different set of operations and services available. The list below summarizes the basic object types.

- *Workbook objects.* Workbooks can contain other workbooks that are represented using the workbook icon shown in Figure 3.

- *Diagram objects.* Diagram icons denote design diagrams such as object diagrams, message sequence charts, object interaction graphs, state charts, dataflow diagrams and use case diagrams. Diagram objects are similar to

workbooks in the sense that similar user interface conventions, editing practices and linking capabilities apply to them.

- *User objects*. User object icons serve as "visit cards" by using which the users can leave comments to the other users, and by which they can identify themselves so that they can be contacted easily later.

- *Host system objects*. Host system files can be dragged and dropped into TDE workbooks and design diagrams in the expected fashion, and double-clicking the icon will open the corresponding application (such as Microsoft Word or Excel) for manipulating the file. Note that the files that are imported into workbooks need not be documents only, but they can be applications, sound files, video clips, or basically anything that is storable on a hard disc or other mountable device in a local area network.

- *Objects in external configuration management systems*. It is also possible to have icons that refer to structures (e.g., source code files, documents, test scripts, picture files, and so on) stored in external configuration management systems. Upon double-clicking such an icon, the corresponding file is fetched transparently from the external configuration management system.

- *WWW objects*. WWW icons denote documents and other entities stored in the World-Wide Web. A web browser will be launched automatically upon double-clicking such an icon.

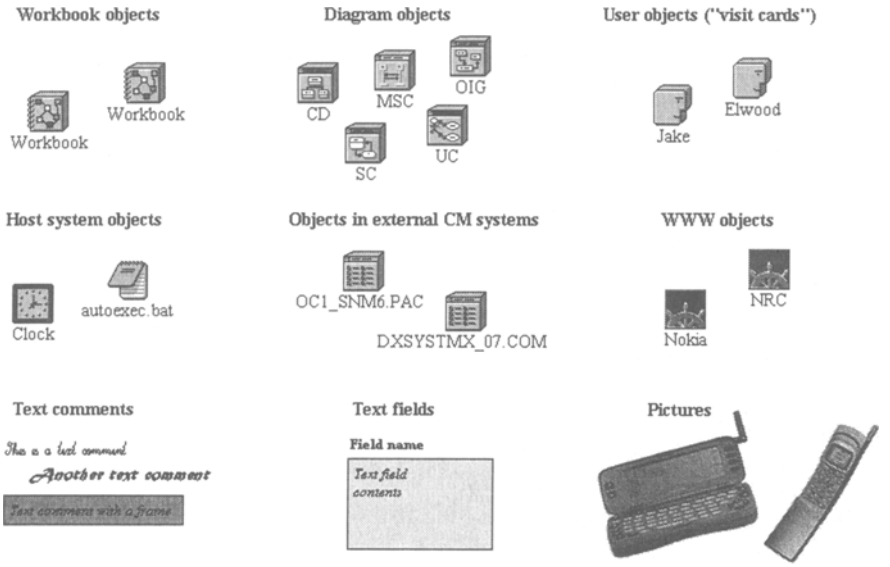

Fig. 3. Standard TDE object types

- *Text comments and text fields.* Text comments can be added to workbooks and design diagrams as necessary. Font types, sizes, colors and frame types in text comments can be changed freely. Text fields are similar to text comments except that they have a fixed size and can thus be used, e.g., for creating different kinds of *forms* (e.g., use case sheets, operation sheets) or to augment diagram with form-structured data.

- *Pictures.* Arbitrary pictures can be inserted into workbooks and design diagrams. Pictures are stored in compressed format to conserve space.

All the object types are manipulated using context-sensitive menus and have the same object linking capabilities available.

3.4 Object Linking Capabilities

TDE provides flexible mechanisms for creating different kinds of relationships between objects. Three basic types of relationships are supported.

1) *Relations within diagrams.* TDE provides a possibility to establish visible relations between any two objects contained in TDE workbooks. Relations are displayed as links (arrows) between the objects. Relations can be given names, and different kinds of line types and end node types are available. The semantics of relations in workbooks have not been fixed in advance, and thus the users can use relations in various ways to illustrate the connections between objects.

2) *Relations between diagrams (hyperlinks).* In addition to relations *within* diagrams, it is possible to establish relationships *between* different workbooks and design diagrams. These kinds of relationships are called *hyperlinks.* Hyperlinks are bidirectional connections that can be established between any two objects in any TDE workbook or design diagram by using the drag-and-drop mechanism. Like relations within diagrams, hyperlinks can be named, and the names of the hyperlinks are used for categorizing them in the context-sensitive menus of the objects.

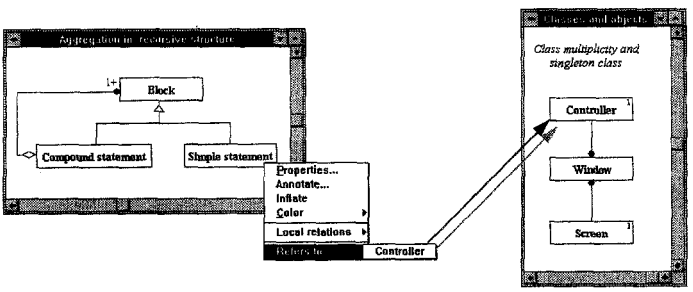

Fig. 4. Hyperlinks

3) *Aliases.* TDE provides the possibility to create *aliases* of workbooks and design diagrams. Aliasing means that the same workbook or design diagram is contained in several other diagrams. Aliasing is very important and frequently used in TDE, since aliasing facilitates *multi-perspective design*, i.e., the description of the same pieces of design from multiple viewpoints. For instance, when describing the new features of a certain system such as a telephone switch, the workbook describing a certain new feature can be referred to simultaneously from several other higher-level workbooks that summarize the new features 1) from *structural perspective* ("in which parts of the system the features belong to"), 2) from *timing perspective* ("in which order are the features to be implemented"), and 3) from *responsibility perspective* ("which designers are responsible for designing and implementing these features"), and so on (Figure 5). Thus, the designs need not be restricted to just one viewpoint, but the same information can be accessed effortlessly from multiple perspectives. This has turned out to be very powerful in practice.

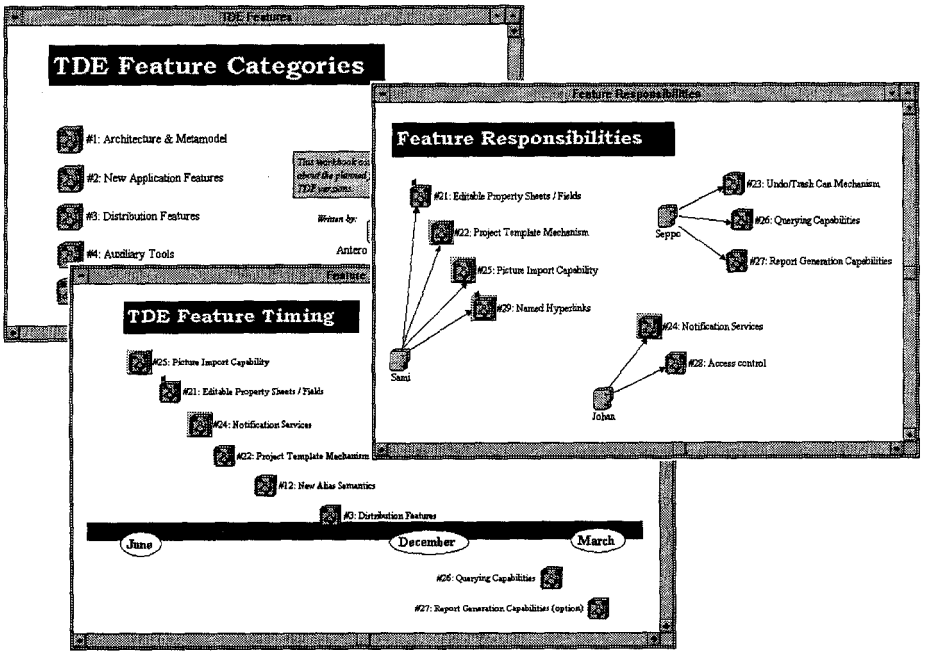

Fig. 5. Using aliases for multi-perspective design

3.5 Diagram Editors

TDE provides the possibility to create, edit and share various kinds of *design diagrams*. These diagram editors are similar to those in many commercial CASE tools such as Paradigm Plus, Rational Rose, or System Architect. Currently TDE provides support for those diagram types that are most frequently used at Nokia,

including class diagrams, object interaction graphs, message sequence charts, state charts, dataflow diagrams and use case diagrams, but thanks to the use of metamodeling new diagram types can be added flexibly. The user interface conventions in design diagrams are similar to those used in workbooks. This implies, e.g., that context-sensitive menus, version control, object linking capabilities and other more detailed workbook editing features are available also for design diagrams. This also means that design diagrams -- like workbooks -- are treated as "places" that can be accessed ("visited") and edited by multiple users even simultaneously (see next section). Figure 6 provides examples of TDE design diagrams.

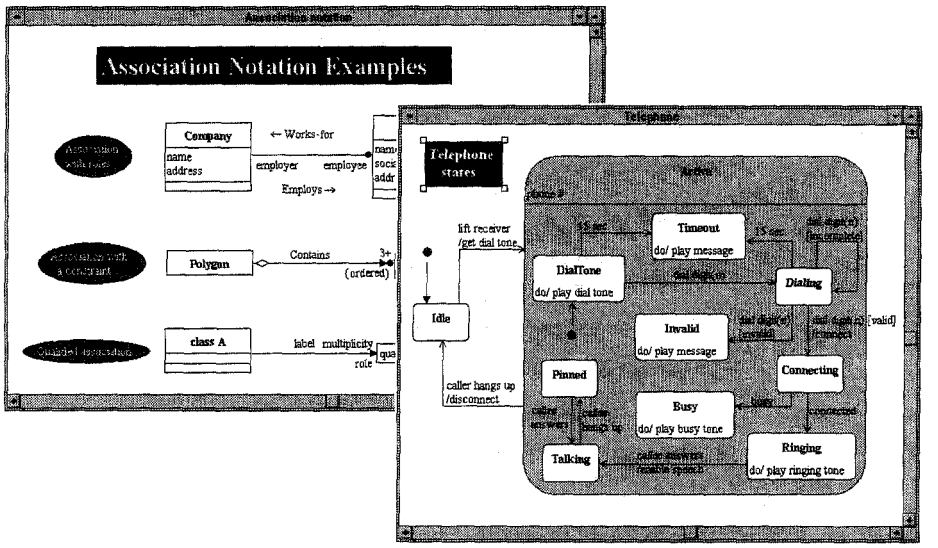

Fig. 6. TDE design diagram examples

3.6 Collaboration and Sharing Capabilities

The most unique aspect of TDE is the emphasis it places on information sharing and designer collaboration. Most software design environments today do not provide any support for teamwork, and if they do, they usually rely on rather formal and restrictive approaches to designer collaboration, such as the traditional check-out/check-in version control model which prevents designers from accessing the same diagrams at the same time. TDE does not suffer from these limitations. Rather, a guiding principle in designing TDE has been that by default no information should be locked. The environment should foster teamwork and make it possible for the designers to work on the same designs and otherwise collaborate with each other as much as necessary.

Places, things and people metaphor. In designing TDE, an idea known as the "*places, things and people*" *metaphor* has been followed. Workbooks and other diagrams are seen as "places" that TDE users can "visit" either at different times or

simultaneously. Each workbook can contain different kinds of "things" that can be manipulated by the users. TDE users are "people" who have an identity within TDE repository and who can leave visit cards identifying themselves in those places that they visit. This metaphor seems like a natural starting point for environments supporting collaborative work. A similar metaphor has been adopted in some other research projects, including the *Jupiter* project at Xerox PARC.

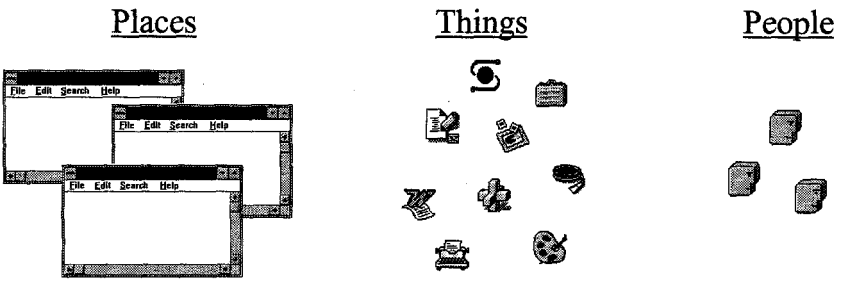

Fig. 7. Places, things and people metaphor

Note that in TDE the places, things and people metaphor is not limited to workbooks only but applies also to all the design diagrams such as class diagrams, message sequence charts or use case diagrams. Like workbooks, these diagrams are thus treated as places that multiple users can visit and manipulate simultaneously if necessary.

Online collaboration. TDE supports *online collaboration*, i.e., makes it possible for the designers not only to share the same diagrams and other designs but also to work on them *simultaneously*. By this we mean that several designers can have the same workbooks and design diagrams open, make modifications to them interactively, and the other users will see the corresponding changes reflected on their screens. Notification mechanisms are provided for keeping the users' displays up to date. When editing diagrams at the same time, all TDE users are treated equally, i.e., none of the users is dominating or moderating the editing process.

4 Usage

TDE is a general-purpose environment in the sense that the contents or the semantics of the information that can be imported into TDE workbooks has not been predetermined or fixed in any way. Nor does the environment predefine the types of relations that can be established within and between workbooks and the things contained in them. In general, we have aimed at keeping TDE as an open-ended design environment that is not bound to any specific design process, and that enables the weaving of related information together in a hypertext/Web-like fashion, with the difference that TDE is always fully interactive, supports shared, direct manipulation of visual information, and enables interactive collaboration with the other persons using the system.

As a result of its generality, TDE can be used in different ways. One way to use TDE is to use it simply as a graphical interface or a visual workbench to existing file systems or configuration management systems. However, such environments are already commonly available, and thus such usage is not very exciting. It is also possible to use TDE as a visual document management system that provides rapid access to existing documents and that allows these documents to be annotated and decorated in various ways. This is an area for which it is quite difficult to find commercial tools that would be flexible and adaptive enough to suit to the needs of industrial users.

However, the real potential of TDE is elsewhere. As mentioned earlier, a central goal for TDE is to make possible a transition away from document-driven design, and to eventually replace the traditional documentation and other design material altogether with an active, shared, visual media that can contain hyperlinked design information in various formats, including text, graphical design diagrams, pictures, audio, video and so on. The real benefits of TDE do not appear until steps towards this direction are taken. As obvious, such a transition is not just a tool issue, but it also involves a number of methodological and process aspects that must be addressed, too. Fortunately, since TDE supports information and document management also in a more conventional fashion, and allows different combinations of old and new representation formats, it is possible to perform this transition gradually, i.e., without any sudden changes in the existing work practices. This is very important when dealing with large-scale projects.

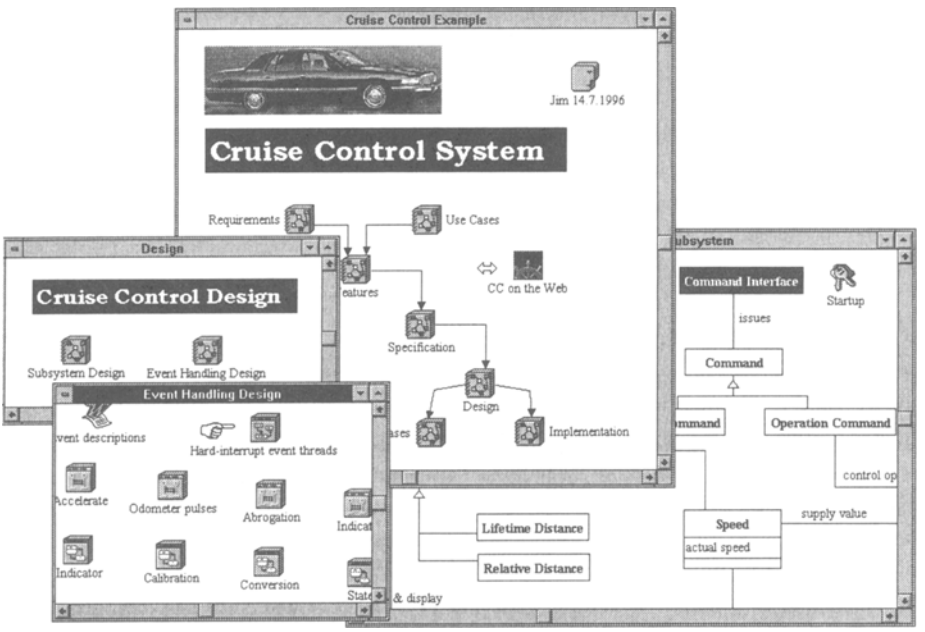

Fig. 8. Workbook usage scenario

Figure 8 provides a simple example of TDE usage. In this example, workbooks are used to form a hierarchical description of a cruise control subsystem software design process. A simple waterfall model has been followed, which consists of different phases including requirements, features, specification, design, implementation and testing. The workproducts of each phase are gathered in their own workbooks, with relationships within the workbooks representing the structural relationships of the workproducts belonging to a certain design phase, and hyperlinks between workbooks representing other types of relationships between workproducts. At the lowest level, there are various graphical design diagrams that describe the cruise control subsystem in detail. Some links to related external documents are also included.

5 Comments on Implementation Technology

TDE is based on a client-server architecture where multiple clients share a common repository server holding shared data. In the current version each TDE application can be connected to only one TDE server at a time, but the implementation of a multi-server version is underway. In order to provide more flexibility and tailorability, TDE is built on top of an object-oriented database (ObjectStore by Object Design, Inc) and it utilizes *metamodeling* technology [ToK93, Ode95, KLR96] to a great extent. In other words, rather than using a hard-coded schema for information representation, TDE uses a straightforward ORP (Object-Relation-Property) metamodel for information representation. Thanks to the use of metamodeling, new diagram types and other features can be added to TDE relatively easily. We have observed that metamodeling is particularly useful for the implementation of collaborative systems because the fine-grained nature of objects facilitates the manipulation of objects independently of each other. Further, the metamodel interface serves as a natural location for the notification services that must be present in a collaborative system such as TDE. In spite of the apparent synergy between metamodeling and collaboration, a lot of effort in TDE implementation has been spent on mutual access mechanisms and conflict prevention.

In addition to the use of client-server technology and metamodeling, there are some other areas that have been central in implementing TDE. For instance, special emphasis has been placed on implementing a generic external workproduct interface that allows various types of external objects (host system files, files in configuration management systems, files in legacy systems, WWW documents) to be accessed uniformly. Also, the implementation of the notification server for keeping multiple designers' work up to date and for enabling online collaboration has required a substantial amount of design and implementation work. The recent TDE versions also provide advanced Intranet capabilities for displaying the contents of TDE repositories via a regular WWW browser. The description of these facilities is out of the scope of this paper, but further information is available from the authors. Figure 9 provides a summary of the general design principles underlying TDE repository.

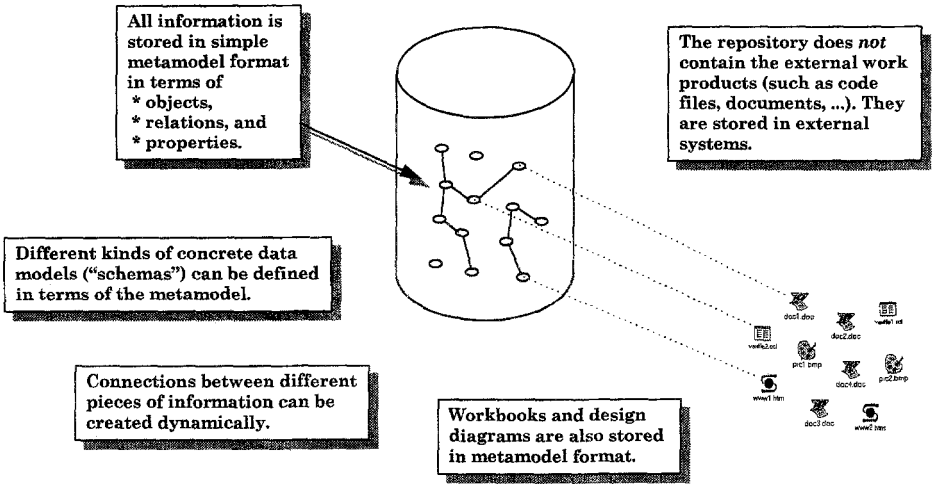

Fig. 9. Summary of TDE repository principles

6 Related Work

The TDE system described in this paper is related to various kinds of software systems and tools, ranging from software engineering environments (SEE's), CASE tools and document management systems to CSCW systems, groupware and virtual whiteboard technology. At the technical level, TDE covers a broad spectrum of research issues such as metamodeling, hypertext, the advanced use of object technology including object-oriented databases and distributed objects, and advanced graphical user interface (GUI) technology. Due to the broad spectrum and variety of the related areas, it is impossible to cover all the related work here. Furthermore, we are aware that none of the technical solutions used in TDE is new in itself, but the way we have combined these solutions seems truly novel. In other words, although many of the individual ideas behind TDE have been presented before, in TDE the whole is much more than just the sum of the parts. Also, it should be emphasized that TDE is not just an experiment or a research prototype but a concrete system that has been designed for production-level usage.

In the area of software engineering environments there are several areas that bear resemblance to TDE. For instance, the functionality provided by advanced programming environments such as Mjølner [HeM88], HP SoftBench [Cag90] and Sniff [Bis92, Sni95] overlaps with TDE in some respects. However, the focus in these systems is much more on the programming side, e.g., on providing an integrated source code editing, compilation, linking and debugging services, whereas TDE places much more emphasis on information management, visualization and collaborative design.

The relationship between TDE and CASE tools is close. If we compare TDE to the various CASE tools available on the market today, such as System Architect, Rational Rose, Paradigm Plus, Graphical Designer, and Software Through Pictures, we notice that the basic capabilities of these systems are rather similar. All of them provide an integrated set of graphical diagram editors that are augmented with additional services and functions. Traditional CASE tools put a lot of emphasis on features such as automatic code generation, consistency checking, and simulation. In contrast, TDE's main focus is on features which require less formality but which we have found to be much more important in large-scale software development, including multi-user support, legacy information access and designer collaboration. Although multi-user support is provided in some CASE tools, e.g., Paradigm Plus, System Architect and MetaEdit+ [KLR96], most of these systems use traditional locking and version control policies that have a tendency to restrict the creativity of the designers. Further, none of the widely used CASE tools we are aware of provides support for online collaboration, i.e., the ability of multiple designers to work on the same diagrams at the same time.

TDE is closely related to the work done in the areas of groupware and Computer-Supported Collaborative Work (CSCW) [Gru94]. Support for collaboration in software engineering environments has been investigated, e.g., in [GMH95] and [BiK95]. Experiences with virtual whiteboard and notebook technology have been reported in [EPG96]. Technological solutions for building collaborative software engineering environments have been studied, e.g., in [Lon95], and communication solutions for collaborative systems in [Cri96]. The idea of using the World-Wide Web as a model for supporting interactive concurrent engineering has been presented, e.g., in [HMC96], and similar solutions are pursued also in the *Habanero* project at NCSA/University of Illinois (*http://www.ncsa.uiuc.edu/SDG/Software/Habanero*). The functionality of TDE also partially overlaps with collaborative document management systems such as Lotus Notes (*http://www.lotus.com/*).

One of the central technological solutions behind TDE repository, *metamodeling*, has been investigated a lot in the literature in the recent years. The basic technical requirements and solutions have been discussed, e.g., in [Bri90, ToK93, Ode95]. Also, some CASE tools that utilize metamodeling technology have been built, including Paradigm Plus 3.0 and MetaEdit+ [SML91, KLR96]. In the same vein, there is a lot of material on the use object-oriented database technology that bears resemblance to the solutions underlying TDE repository (e.g., [Mac96]). When implementing TDE, we decided to build TDE on top of a commercial object-oriented database management system and did a lot of experiments with several of them, including ObjectStore, Objectivity, Poet and Versant.

The user interface of TDE bears a lot of resemblance to the ideas that have been presented in the field of hypertext research [Nel65, Con87, SmW88]. The most central common idea is to manage information in terms of interconnected nodes that can contain text, graphics, video as well as links to source code or other forms of data. In supporting the transition away from document-driven design, TDE also

shares some ideas with *interactive documentation* [Bro86, MöK95]. TDE's idea of using virtually infinite, flat, shared workspaces for software design is not fully new either, but has been utilized earlier in some systems, including the *Kansas* user interface of the *Self 4.0* programming language [SMU95].

Internally within Nokia our work on TDE is closely related to the work that is being carried out in the areas of process improvement and software design methods and tools. Results of this work have been reported in [AaJ94, Jaa95, AKZ96, Kän96, Tai97].

7 Conclusion

In this paper we have presented the background, central concepts, high-level functionality and usage of TDE -- Telecom Design Environment. TDE is a collaborative software design environment that allows software designers to communicate about their designs with the other designers across a network of workstations. The most unique feature of this environment is the support that it provides for *designer collaboration*, making it possible for the designers to work on the same graphical designs location-transparently and even simultaneously if necessary. In a way, TDE can be seen as a novel combination of three kinds of systems: a document management system, CASE tool and a virtual whiteboard system. The result could be described as a "writable, collaborative web", i.e., an interactive, collaborative, graphical environment that provides effortless access to existing design information and provides powerful visualization and linking capabilities for weaving related pieces of design together.

Acknowledgments

The authors would like to thank all TDE project team members for their talented work on TDE. Thanks go also to our pilot users at Nokia Telecommunications, especially Jari A. Lehto and Pekka Immonen.

References

AaJ94 Aalto, J-M., Jaaksi, A., Object-oriented development of interactive systems with OMT++. In Ege, R., Singh, M., Meyer, B. (eds): TOOLS'14 -- Technology of object-oriented languages and systems, Prentice-Hall, 1994, pp.205-218.

AKZ96 Awad, M., Kuusela, J., Ziegler, J., Object-oriented technology for real-time systems: a practical approach using OMT and Fusion. Prentice-Hall, 1996.

Bis92 Bischofberger, W.R., Sniff -- a pragmatic approach to a C++ programming environment. In Proceedings of the USENIX C++ Conference, Portland, Oregon, August 1992.

BiK95 Bischofberger, W.R., Kofler, T., Computer-supported cooperative software engineering with Beyond-Sniff. In Proceedings of SEE'95 (Noordwijkerhout, the Netherlands, April 5-7), IEEE Computer Society Press, 1995, pp.135-143.

Bri90 Brinkkemper, S., Formalization of information systems modeling. Ph.D. thesis, University of Nijmegen, Thesis Publishers, Amsterdam, 1990.

Bro86 Brown, P.J., Interactive documentation. Software -- Practice & Experience vol 16, nr 3 (Mar) 1986, pp.291-299.

Cag90 Cagan, M.R., The HP Softbench environment: an architecture for a new generation of software tools. Hewlett-Packard Journal vol 41, nr 3, 1990.

Con87 Conclin, J., Hypertext: an introduction and survey. IEEE Computer vol 20, nr 9 (Sep) 1987, pp.17-41.

Cri96 Cristian, F., Synchronous and asynchronous group communication. Communications of the ACM vol 39, nr 4 (Apr) 1996, pp.88-97.

EPG96 Edelson, D.C., Pea, R.D., Gomez, L.M., The collaboratory notebook. Communications of the ACM vol 39, nr 4 (Apr) 1996, pp.32-33.

GMH95 Grundy, J.C., Mugridge, W.B., Hosking, J.G., Amor, R.W, Support for collaborative, integrated software development. In Proceedings of SEE'95 (Noordwijkerhout, the Netherlands, April 5-7), IEEE Computer Society Press, 1995, pp.84-94.

Gru94 Grudin, J., CSCW: history and focus. IEEE Computer vol 27, nr 5 (May) 1994.

HMC96 Hanneghan, M., Merabti, M., Conquhoun, G., The World-Wide Web as a platform for supporting interactive concurrent engineering. In Constantopoulos, P., Mylopoulos, J., Vassiliou, Y. (eds): Proceedings of CAiSE'96 (Heraklion, Crete, Greece, May 20-24), Advanced Information Systems Engineering, LNCS 1080, Springer-Verlag, 1996, pp.301-318.

HeM88 Hedin, G., Magnusson, B., The Mjølner environment: direct interaction with abstractions. In Proceedings of ECOOP'88, Lecture Notes in Computer Science vol 322, Springer-Verlag, 1988.

Jaa95 Jaaksi, A., Object-oriented specification of user interfaces. Software Practice & Experience vol 25, nr 11 (Nov) 1995, pp.1203-1221.

Kän96 Känsälä Kari, Software process improvement experiences within Nokia. In Proceedings of SPI'97 (Brighton, England, December 2-5), 1996.

KLR96 Kelly, S., Lyytinen, K., Rossi, M., MetaEdit+ -- a fully configurable multi-user and multi-tool CASE and CAME environment. In Constantopoulos, P., Mylopoulos, J., Vassiliou, Y. (eds): Proceedings of CAiSE'96 (Heraklion, Crete, Greece, May 20-24), Advanced Information Systems Engineering, LNCS 1080, Springer-Verlag, 1996, pp.1-21.

Lon95 Lonchamp, J., CPCE: A kernel for building flexible collaborative process-centered environments. In Proceedings of SEE'95 (Noordwijkerhout, the Netherlands, April 5-7), IEEE Computer Society Press, 1995, pp.95-105.

Mac96 Machura, M., Managing information is a cooperative object database system. Software -- Practice and Experience vol 26, nr 5 (May) 1996, pp.545-579.

MöK95 Mössenböck, H., Koskimies, K., Active text for structuring and understanding source code. Johannes Kepler University of Linz, Institut für Informatik, Technical Report 3, August 1995.

Nel65 Nelson, T., A file structure for the complex, the changing, and the indeterminate. ACM 20th National Conference, 1965.

Ode95 Odell, J.J., Introduction to method engineering. Object Magazine vol 5, nr 5 (Sep) 1995, pp.69-72,91.

SMU95 Smith, R.B., Maloney, J., Ungar, D., The Self 4.0 user interface: manifesting a system-wide vision of concreteness, uniformity and flexibility. In OOPSLA'95 Conference Proceedings (Austin, Texas, October 15-19), ACM SIGPLAN Notices vol 30, nr 10 (Oct) 1995, pp.47-60.

SML91 Smolander, K., Marttiin, P., Lyytinen, K., Tahvanainen, V-P., MetaEdit -- a flexible graphical environment for methodology engineering. In Andersen, R., Bubenko, J., Solvberg, A. (eds): Advanced Information Systems Engineering, LNCS 498, Springer-Verlag, 1991, pp.168-193.

SmW88 Smith, J., Weiss, S., An overview of hypertext. Communications of the ACM vol 31, nr 7 (Jul) 1988.

Sni95 SNiFF+ Release 2.0 User's Guide and Reference. TakeFive Software GmbH, Salzburg, Austria, 1995.

Tai97 Taivalsaari, A., Multidimensional browsing. To appear in the Proceedings of Software Engineering Environments (SEE'97) Conference (Cottbus, Germany, April 7-9), 1997.

ToK93 Tolvanen, J-P., Lyytinen, K., Flexible method adaptation in CASE environments -- the metamodeling approach. Scandinavian Journal of Information System vol 5, 1993, pp.51-77.