National Library of Canada
Acquisitions and Bibliographic Services Branch
395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale du Canada
Direction des acquisitions et des services bibliographiques
395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

RECONCILING FORMAL AND INFORMAL DOCUMENTATION IN BUSINESS MODELLING

by

Amin Yousef Noaman

School of Computer Science
McGill University
Montréal, Canada

February 1995

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE.

ISBN 0-612-05607-4

Table of Contents

List Of Figures

Abstract

Business modelling, the modelling of architectures and processes of organizations, should have a broad scope. It should not exclusively capture the basic information of the processes, but also address the various kinds of documentation related to the processes under consideration. In this combination, organizational models will be more expressive and useful.

The research reported here describes and demonstrates a new approach for reconciling formal and informal documentation in business modelling. It is based on the integration of an underlying formal modelling approach with hypertext concepts that provide mechanisms for capturing, manipulating and viewing informal model documentation.

We have developed the Hypertec tool which complements the Macrotec environment. Macrotec is a business modelling environment that is based on the formalism of extended colored Petri nets. Hypertec is a hypertext-based component supporting authoring, display and navigation of all the process documentation that cannot be captured by Macrotec. Our experience with Macrotec/Hypertec shows that their combined functionality substantially facilitates the understanding of business processes and clearly reduces problems such as miscommunication, misinterpretation, and misunderstandings about entire processes or some of their components.

Résumé

La modélisation des entreprises, la modélisation des architectures et des processus d'entreprises et d'organisations, demande une vision étendue. Elle doit saisir non seulement l'information de base appropriée aux processus mais également les différents aspects concernant la documentation de ces processus. Avec une telle approche, les modèles organisationnels seront plus efficaces et plus expressifs.

La présente recherche décrit et démontre une nouvelle approche pour réconcilier les aspects formels et informels de la documentation dans la modélisation des entreprises. L'idée maîtresse consiste à intégrer l'approche formelle sous-jacente avec les concepts hypertexte qui prévoient des mécanismes pour saisir, manipuler et visualiser la documentation informelle des modèles.

Nous avons développé l'outil Hypertec qui est un complément de l'environnement Macrotec. Macrotec est un environnement de modélisation des entreprises basé sur le formalisme des réseaux de Petri colorés étendus. Hypertec est un composant basé sur les concepts d'hypertexte qui supporte la rédaction, la visualisation et la navigation de toute la documentation des processus qui n'est pas supportée par Macrotec. Notre expérience avec Macrotec/Hypertec montre que la combinaison de leur fonctionnalité facilite grandement la compréhension des processus à modéliser. Également, cette approche réduit considérablement les problèmes de communication, d'interprétation erronée ou de mauvaise compréhension concernant le processus global ou une partie de ce processus.

Acknowledgements

First and foremost I would like to thank my parents and my wife for giving me every support needed during my master program studies. My special thanks go to my supervisor, Rudolf K. Keller, for his sincere encouragements and his diplomatic way of getting me back on the right track when I was digressing, and most of all for his most valuable resource, his time, of which he was never scarce. I am also grateful to my co-supervisor, Prof. Nazim Madhavji, for teaching me the necessary software process skills and supervising me during my master program. I would also like to thank Lorraine Harper and Franca Cianci who gave me administrative support during my studies at the School of Computer Science at McGill. I would also like to thank my best friend, Nasser Elmasri. He is the most understanding and patient person I know. My thanks also go to the Centre de recherche informatique de Montréal (CRIM), for providing an excellent work environment, and to Jean-Michel Goutal, who was always supportive during my stay at CRIM.

Many thanks go to my close friend, Anurag Garg. We motivated each other throughout many exceptionally long coding sessions. I hope that I will have again the opportunity to work with him in the near future. Finally I would like to say that I will never forget the days when all the Tigers were working together, it was such a nice experience to be in this team: Anurag Garg, Ghassan Youssef, Xijin Shen, and Natalia Pagliarulo.

Chapter 1 Introduction

1.1 Motivation

A lot of current research in software engineering is directed toward modelling and model analysis methods. It is recognized that model analysis methods together with their supporting environment are playing an increasingly important role in many areas of system design and refinement. Business modelling is one such area where the overall goal is to identify, design, and evaluate value-adding opportunities for business improvement.

There is a growing consensus that business modelling, the modelling of architectures and processes of organizations, should have a broad scope. It should not exclusively encompass an organization's IT (information technology) component, but rather represent and reason about IT as a component of a wider environment. As stated by Medina-Mora [MMWFF92], we believe that this environment should comprise three domains: materiel, information, and business processes.

Business modelling, especially if it is to exhibit a broad scope, should be complemented with various kinds of documentation related to the processes under consideration, such as forms, database information, pieces of software, and anecdotal

experience. This way, the resulting models can be seen as a multi-level documentation of the underlying processes, comprising both formal and informal elements, and allowing for the execution of their formal parts.

Our goal in this work is (1) to complement business modelling and analysis of organizational architectures and processes with formal and informal documentation, and (2) to provide mechanisms for creating, manipulating, and navigating among these documents.

1.2 Approach

In a joint research project between Group DMR Inc. and the CRIM, a new formalism for organizational modelling, Macronet [1], was developed together with a supporting environment, called Macrotec [2] [KLO+93, KOS94]. The overall goal of Macrotec is to provide an environment that comprehensively supports architectural modelling, high-level requirements specification, and documentation of business processes and information systems.

To address the documentation aspect of business modelling, we identified hypertext as the most appropriate concept. Hypertext is a medium-grained, entity relationship like data model that lets information be structured arbitrarily and keeps a complete version history of both information and structure [Big88]. Moreover, a feasibility study [Tao93] convinced us that applying hypertext to business modelling is an appropriate and effective mechanism. From this study and a preliminary prototype built in a course project [3], we started our work, developing and refining theoretical concepts, implementing the Hypertec tool, and validating our approach with real life examples.

1. "Macronet" is a contraction of the words Macroscope, the project's name, and Petri net.
2. "Macrotec" is a contraction of the words Macroscope, the project's name, and architecture.
3. Course project as part of course CS765 "Software Architectures for User Interfaces" taught at McGill University in winter'94 by R. Keller. The project participants are: A. Garg, N. Pagliarulo, G. Youssef, and Amin Noaman.

1.3 Main Results

The main results of our work are:

- A new approach for the seamless integration of formal and informal documentation in business modelling, both in terms of process and tool support, making modelling more expressive, comprehensive, and comprehensible;

- the design and implementation of Hypertec, a software component that complements Macrotec, our underlying modelling environment, with facilities for integrated model documentation;

- the validation of our approach by modelling with Hypertec a number of examples from various application domains [KGNT94].

1.4 Organization of Thesis

The thesis is organized as follows: Chapter 2 starts with a presentation of basic business modelling concepts. Then, model documentation requirements are discussed. Chapter 3 describes hypertext concepts and object modelling facilities, the major elements of our solution approach, and puts them into the business modelling context. Chapter 4 presents our approach for integrating hypertext into business modelling. It begins with an overview of the Macrotec environment. Then, the integration of hypertext into Macrotec is explained. Thereafter, we discuss in detail the Hypertec tool components for capturing and managing formal and informal documentation. Chapter 5 covers the aspect of architecture and environment integration of Hypertec, from both a design and an implementation point of view. Chapter 6 provides a case study, illustrating the key features and capabilities of Macrotec/Hypertec. Chapter 7 addresses the related systems we found most relevant for our work. In chapter 8, finally, we discuss future plans for Hypertec and draw some conclusions.

Chapter 2 Business Modelling and its Documentation

In this chapter, we shall provide an overview of two major business modelling concepts, namely, petri nets and object modelling. Then, we shall discuss model documentation requirements.

2.1 Business Modelling Concepts/Approaches

A lot of research in software engineering is directed to modelling and model analysis methods. Model analysis methods together with their supporting environment are playing an increasingly important role in many areas of system design and refinement. Business modelling is one such area where the overall goal is to identify, design and evaluate value-adding opportunities for business improvement.

Business modelling is a technique for specifying and analyzing an enterprise's architectural structures and their information technology components. Diverse approaches to business modelling have been taken, including entity-relationship models, data flow diagrams, queuing networks, Petri nets, object models, and knowledge-based approaches [BDD+92]. In our work, we have developed Macronet, a formalism that combines several of these approaches into a uniform modelling framework through the formalism of Petri nets. Furthermore, Macronet exhibits object-oriented features.

In this section, we shall present Macronet and put it into the broader context of Petri nets and of object modelling.

2.1.1 Petri Nets

Petri nets have been used extensively and with good success in modelling of concurrent asynchronous tasks in programming and communications systems. They are at the base of modelling techniques that are effective, accurate, and simple to use.

One important aspect of Petri nets is their ability to accurately model system properties. These properties include support for [PS93]:

- ***Execution time simulation:*** *Petri nets provide a useful and relatively easy means of estimating the amount of time a system will take to proceed from some initial state to some desired final state.*
- ***State Set Cataloging:*** *execution of the network proceeds through a large number of intermediate states.*
- ***Simulation of Indeterminate Situations:*** *By introducing the element of time and indeterminacy that accompanies it, Petri nets are capable of reflecting the statistical nature of many simulation propositions.*
- ***The management of Shared Resources:*** *Petri nets are extremely useful in managing and modelling shared resources.*

A Petri net structure *PN* is defined as tuple *(P, T, I, O, M)* in which *P* is a finite set of places, *T* is a finite set of transitions, *I* is a set of directed input arcs and *O* is a set of directed output arcs connecting places to transitions and transitions to places respectively, and a marking *M*. The marking, or state, of a *PN* is defined by the number of tokens in each place. The marking is represented by a vector whose *i-th* component represents the number of tokens in the net's *i-th* place. A *PN* can formally be described as [MCB84]:

$$PN = (P, T, I, O, M)$$

$$P = \{p_1, p_2, \ldots, p_n\}$$

$$T = \{t_1, t_2, \ldots, t_m\}$$

$$I \subset P \times T$$

$$O \subset T \times P$$

$$M' = \{m_1', m_2', \ldots, m_n'\}$$

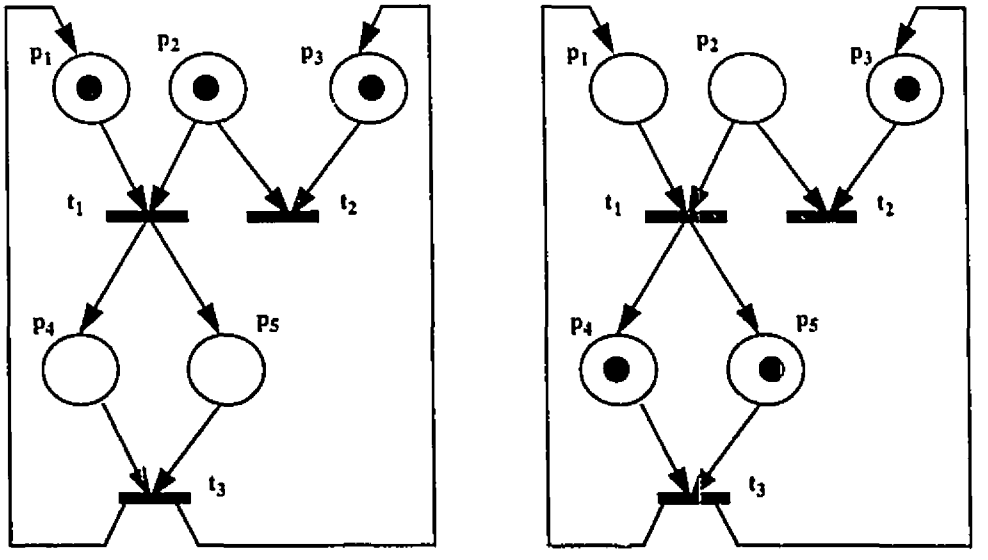


Figure 2.1: Petri Net before (a) and after (b) the Firing of Transition t1.

The execution of a Petri net consists of a sequence of markings, beginning with the initial marking *Mi* and ending in some final marking *Mf*. To go from the current state *M* to some next state *M'*, any non-conflicting transition *t* that is enabled under *M* is chosen and fired. In case of conflicts, different resolution techniques are applied [MCB84].

For example, Figure 2.1 shows a Petri net before (a) and after (b) transition firing. Places are drawn as circles, transitions as bars, and tokens as dots in places. The initial marking, shown in part (a), is $M = (11100)$. The transitions under M are t_1 and t_2. If t_1 is chosen to fire, the next state will be $M' = (00111)$, as shown in part (b). If t_2 is chosen to fire in the initial state, the result would be $M' = (10000)$.

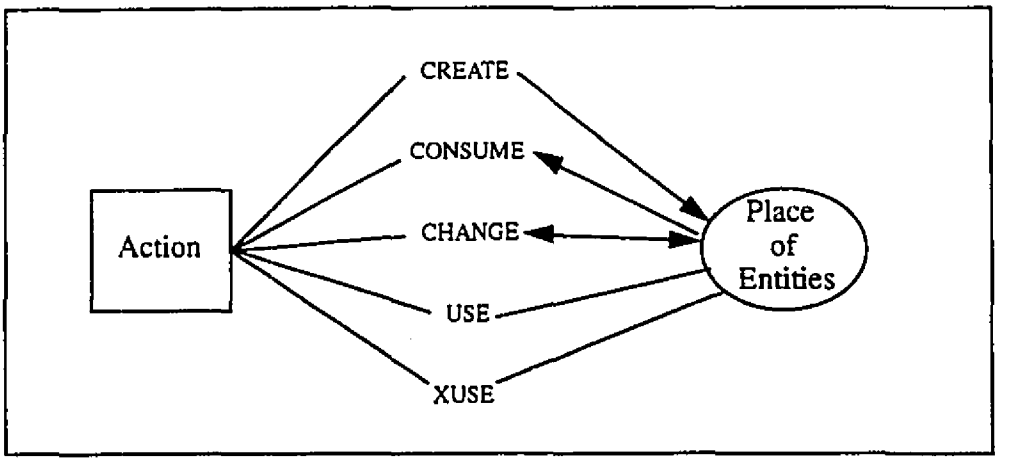


Figure 2.2: Generic Model for Macronet.

2.1.1.1 The Macronet Formalism

Macronet [1] [BDD+92, KOS94] combines several concepts originally developed in separate contexts, such as entity-relationship modelling, specialization and inheritance in the sense of object-oriented methodologies, event analysis, and data flow as well as resource utilization. These concepts have been integrated into a uniform modelling framework with a precise semantics for the dynamic aspects, which has been defined through the formalism of Petri nets.

Figure 2.2 shows the generic model for Macronet. It consists of actions, places of entities, and relations connecting places to actions. The Petri net equivalent to the term "entity" is "token", whereas "actions" are equivalent to Petri net "transitions".

An ***action*** (represented graphically as rectangle) is a unit of work which produces a change in the system's state. Each action is characterized by a name attribute, a firing time which can be immediate, random exponentially distributed, or deterministic, as well as a weight used in situations of conflicts.

1. "Macronet" is a contraction of the words *Macro*scope, our project's name, and Petri *net*.

An ***entity***, which resides inside a place, may represent a resource, for example, a person required for the accomplishment of an action, or a product created by an action.

A ***place*** (represented graphically as oval) is a container which holds entities of a specific type. There is no limit to the number of entities in a place. Each place has a name and a type.

Finally, there are five types of ***relations***, connecting actions to places: Create, Consume, Change, Use, and XUse. They force, respectively, entity creation, deletion, alteration, participation and exclusive participation in the action. Relations may be directly mapped into Petri net arcs, except for the Use relation which has no Petri net equivalent [BDD+92].

Macronets are a combination of Generalized Stochastic Petri nets and deterministic nets. They may be automatically translated into Petri nets without loss of important information. Macronet constitutes a powerful formalism for addressing the specifics of business modelling.

2.1.1.2 Other Petri Net-based Formalisms

There are a number of business modelling approaches that are, like Macronet, based on Petri nets formalisms. Notably in software process modelling, which can be considered as a sub-domain of business modelling, several such formalisms have been suggested [Mad91, CKO92, DG90, BFG93]. More recently, a few Petri net approaches directly addressing business modelling have been suggested, such as the ones reported in [TGH94, BLWW95]. While models founded on formal approaches other than Petri nets are analyzable for syntactic correctness, consistency, completeness, correctness, risks, and opportunities for improvement, PN-based models also lend themselves to more advanced analyses such as reachability of various process states, detection of deadlocks and race conditions in the process, and detection of behavioral ambiguities. These latter characteristics make them particularly relevant for business modelling, where evolving models and dynamic analysis are of great importance. A discussion of these approaches can be found in [KSvB94, REF94].

2.1.2 Object Modelling

Object-oriented techniques have proven to be quite effective in modelling and simulation [PS93]. The analysis stage of object-oriented techniques is represented by the object model. Object modelling is a new way of thinking about problems, using models organized around real-world concepts. Object models are useful in many activities, including understanding problems, communicating with application experts, modelling enterprises, preparing documentation, and designing programs and databases. Such models capture the static structure of a system by showing the objects in the system, relationships between the objects, and the attributes and operations that characterize each class of objects [RBP+91].

The major abstract element in object modelling are the objects, which integrate data and the transformations necessary to manipulate the data. An object is a "thing" that can be distinctly identified. At the appropriate level of abstraction, almost anything can be considered to be an object. Thus, a specific person, organization, machine, or event can be regarded as an object [CAB+94].

An object can have one or more named values associated with it, called attributes. For example, a person object might have three attributes, name, address, and occupation.

An object may have methods that perform the necessary actions. Methods are introduced (1) to access attributes and (2) to support inheritance and the semantic constraints of the problem at hand. Methods are invoked when objects send messages to each other. Objects are grouped into classes when they respond to the same messages in the same ways. Classes are the building blocks of most object-oriented languages.

In the following, we shall give an example of the object classes in the Macronet formalism. Then we shall give an overview of some current object modelling techniques.

2.1.2.1 Object-Oriented Aspect of Macronet

Object-oriented techniques allow the user to derive new classes, which are called subclasses, from the existing classes, which are called superclasses. A subclass inherits the attributes and methods of its superclasses, and may specify additional attributes and methods. This inheritance relation leads to a class hierarchy.

In Macronet, the basic components (action, entity, place, and relation) are object classes. Each class contains attributes, some of which may be predefined attributes, such as class identifier and commonly used attributes. For example, a place represents the base class for a collection of entities and it has "Name" and "Type" attributes. The "Type" attribute specifies the type of entities allowed to reside in the place. A relation, to take another example, represents the base class for Consume, Create, Change, Use, and XUse relations. All the actions and places of a particular model may be specified as class instances in an overall class hierarchy of actions and places, respectively. Figure 2.3 shows the inheritance relation between the superclasses: action, place, and relation and their subclasses. For example, manual and automated actions are subclasses of the superclass action.

2.1.2.2 Current Object Modelling Approaches

There are many different methods supporting object modelling, such as Rumbaugh/ OMT [RBP+91], Booch OOAD [Boo94], Fusion [CAB+94], Coad/Yourdon OOAD [CY91], and Shlaer/Mellor OOA [SM88]. The concepts implemented in the above methods are similar. However, the methods differ in representing the objects (notations).

Below, we shall illustrate the object modelling approach taken in the Rumbaugh/ OMT [RBP+91] and Booch [Boo94] methodologies.

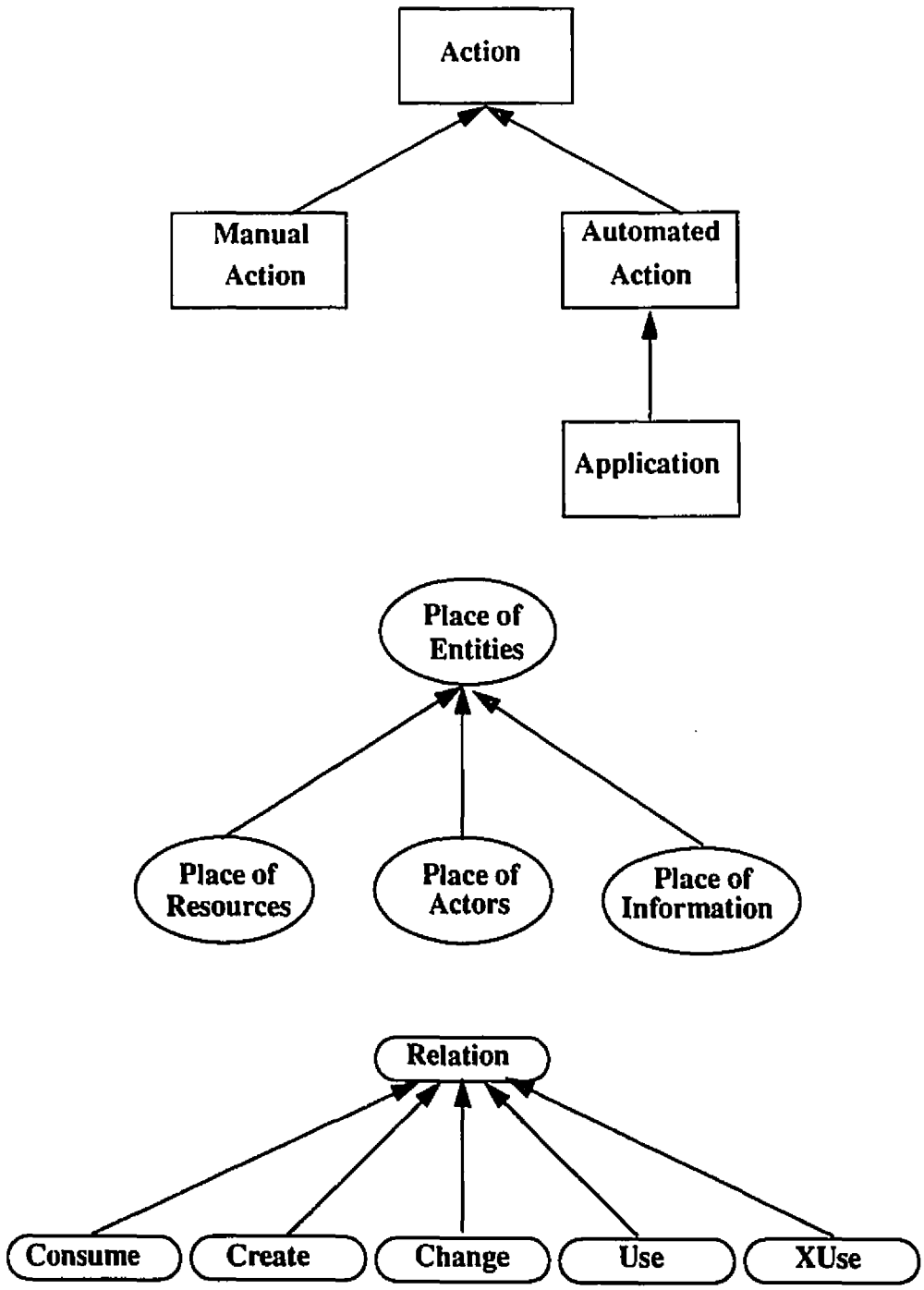


Figure 2.3: Object-Oriented Architecture View of Macronet.

Rumbaugh or OMT:

The Object Modelling Technique (OMT) was developed by James Rumbaugh, Michael Blaha, and William Lorensen. The Rumbaugh methodology is divided into three phases: analysis, which is concerned with modelling the real world; design, which decides on subsystems and overall architecture; and implementation, which encodes the design in a programming language.

In the analysis phase there are three models: the object model, the dynamic model, and the functional model. We shall describe the object model in detail, and for more information about the dynamic and functional model see [RBP+91].

The object model describes the structure of the objects in a system - their identity, their relationships to other objects, their attributes, and their operations. The object model notation is an extended entity-relationship diagram, which describes objects, classes, and relationships. Moreover, it can display methods and invariants. The object model is represented graphically with object diagrams containing object classes. Classes are arranged into hierarchies sharing common structure and behavior and are associated with other classes. Classes define the attribute values carried by each object instance and the operations which each object performs or undergoes [RBP+91].

Figure 2.4 shows an object model of customers ordering products. The Corporate Customer and Personal Customer classes are subclasses of the Customer class, which has a one-to-many association relation with the Order class. The Order class consists of several Order Lines that include an attribute named Quantity. Finally, the Product class has a one-to-many association relation with Order Line and has a many-to-many association relation with Corporate Customer.

Booch:

The "Object-oriented Analysis and Design Methodology" was developed by Grady Booch. The Booch methodology is an iterative approach. Prototypes are developed and refined. At each phase, earlier phases are revisited and revised.

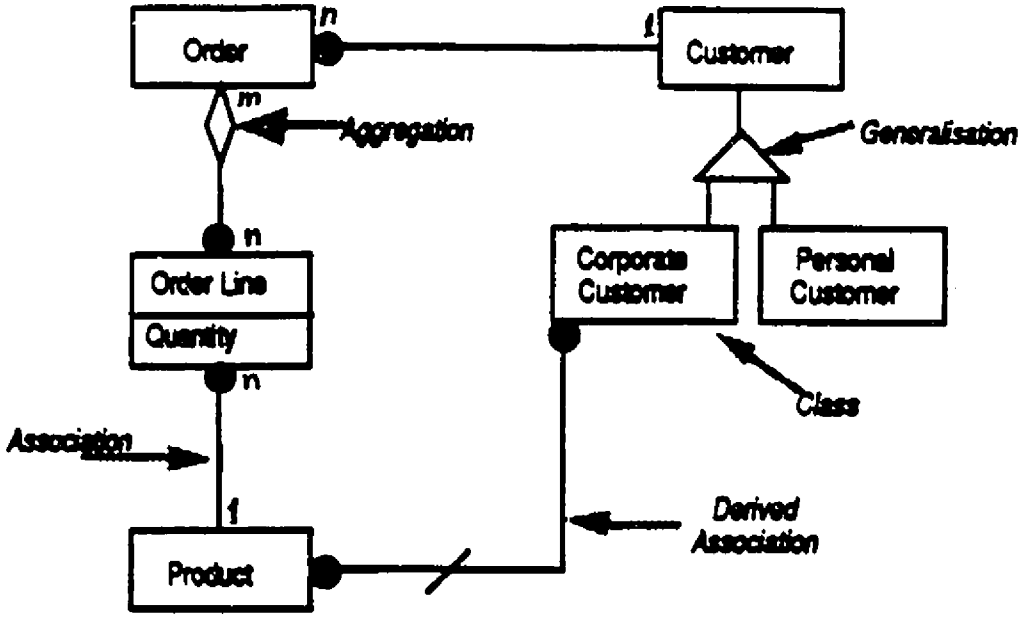


Figure 2.4: Rumbaugh Object Model.

There are six diagram types in the Booch method: Class Diagrams, which show classes and their relationships; Object Diagrams, which show objects and their relationships; Timing Diagrams, which show interactions between objects; State Diagrams, which show the dynamic behavior of objects; and Module Diagrams and Process Diagrams, showing the physical design of the system.

Class diagrams in Booch represent object models. A class diagram is used to show the existence of classes and their relationships in the logical view of a system. A single class diagram represents a view of the class structure of a system [Boo94]. A class diagram is a notational variation of entity-relationship diagrams. The notation allows class categories that group classes into namespaces. Each category is itself a class diagram, and the classes it exports to other categories, or imports from other categories, are so labeled [CAB+94].

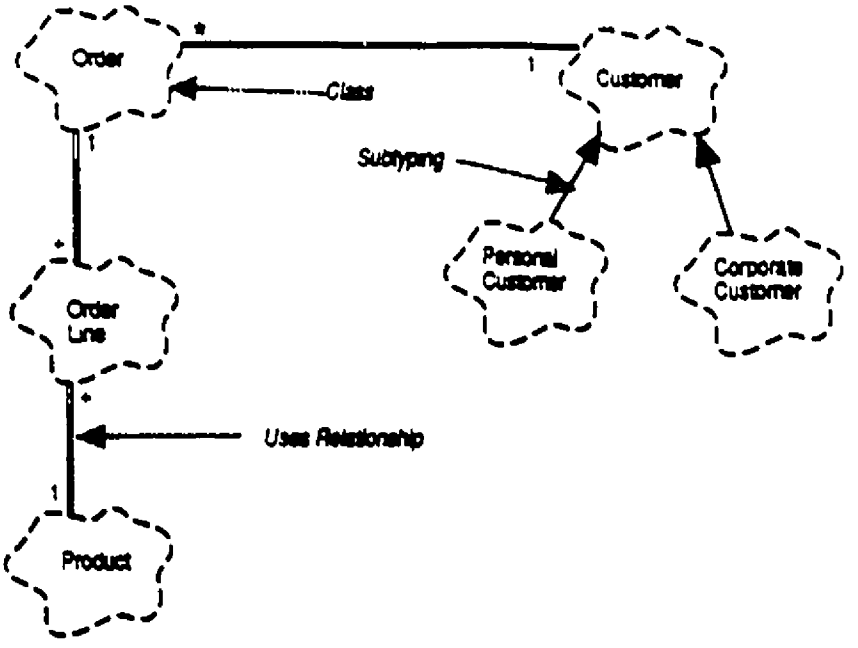


Figure 2.5: Booch Class Diagram.

Figure 2.5 shows the class diagram of customer orders product, which we presented in Figure 2.4. The diagram indicates that the Corporate Customer and Personal Customer classes are both subclasses of the abstract class Customer. Moreover, it shows the association relation between the different classes.

2.1.3 Discussion

We and others [PS93, WW93] believe that both Petri nets (or extensions thereof) and object modelling are key concepts and mechanisms in business modelling. Petri nets are fit for capturing the behavioral aspects of models and are a basis for model simulation. Object modelling, on the other side, is most appropriate for capturing data and relations between them. Combining these concepts into a uniform modelling framework such as Macronet constitutes a key step towards successful business modelling.

2.2 Model Documentation Requirements

Business modelling environment should be complemented with the various kinds of documentation related to the processes under consideration, such as forms, database information, pieces of software, and anecdotal experience. In this way, business models will be more expressive and useful. In addition to the modelling and re-engineering of processes (descriptive and prescriptive usage), they may serve as on-line training vehicles and means for communication in a multi-person environment. Thus, the resulting models can be seen as multi-level documentation of the underlying business modelling, comprising both formal and informal elements, and allowing for the execution of their formal parts [KGNT94].

To this end we would have to be able to access a variety of documents at the same time. Furthermore, we should have mechanisms for manipulating documents and creating navigable links among them. These mechanisms should then be integrated with the underlying business modelling system. A feasibility and usability study [Tao93] convinced us that hypertext is an appropriate and effective mechanism for our needs. In hypertext, information fragments in various forms are grouped and linked together in a way that allows the user to browse and search through them non sequentially. In addition, hypertext does not place any constraints on node formats, which makes it particularly useful for holding the variety of information (formal and informal) found in business modelling. In section 3.1 we shall further discuss the hypertext concepts and its elements.

Other authors advocate hypertext tools to support the software development life cycle, in which a large number of specification and implementation documents is produced (see, for instance, [Big88, CR92]).

Chapter 3 Hypertext and Object Modelling Facilities

In this chapter, we shall discuss some of the concepts which we believe are most relevant in our solution approach. In the first part, we shall elaborate on hypertext concepts, in the second, we shall address object modelling facilities.

3.1 Hypertext Concepts

3.1.1 Introduction to Hypertext

Even though computer technology is still in its infancy, the concept of hypertext has come a long way. In 1945, Vannevar Bush [Bus45] proposed the *Memex* system, a machine based on storing different information such as books, records, and letters on microfilm. The original concept was to give readers the ability to link pieces of text whenever they saw a relationship between the contents of these pieces. Bush stated that "*It affords an immediate step, however, to associative indexing, the basic idea of which is a provision whereby any item may be caused at will to select immediately and automatically another. This is the essential feature of the Memex. The process of tying two items together is the important thing.*" Thus, *Memex*, conceived almost fifty years ago, already exhibited major elements of hypertext. However, *Memex* has never been implemented.

In 1965, Ted Nelson coined the word *Hypertext*. Soon after, hypertext systems entered the real world. These early hypertext systems include:

- Engelbart's Augment/NLS (oN-Line System) (1962-1967)[EE+68]
- Nelson's Xanadu (1965)[Nel80]
- Van Dam's Hypertext Editing System (1967) and FRESS (File Retrieval and Editing System) (1968) [Nie90].

These pioneer systems had a lot of influence on the later development of hypertext.

Hypertext has become popular in recent years, and various organizations such as government, business, and education are using hypertext in a wide variety of situations. With its spread, the whole idea of hypertext has been fragmented and a number of definitions for the different hypertext elements has been introduced. The concepts and definitions that we consider important to our work are presented in depth in subsection 3.1.2 *Elements of Hypertext.*

Before elaborating on the elements of hypertext, we shall define hypertext. We adopt the definition given in [BD91]: "*A hypertext is a an assemblage of texts, images, and sounds-nodes connected by electronic links so as to form a system whose existence is contingent upon the computer. The user/reader moves from node to node either by following established links or by creating new ones.*"

3.1.2 Elements of Hypertext

In its almost fifty years of history, hypertext has been applied to a broad range of applications to enhance the organization, management and presentation of information, as well as to support knowledge acquisition. However, there is still some ambiguity in the definition and usage of hypertext concepts. Below, we shall get a quick overview of the basic elements of hypertext which support our point of view.

Nodes

The fundamental unit of information in a hypertext document is called a node. A node can contain a combination of text, graphics and/or other forms of data. More broadly, a node is some sort of stimulus material that can be activated and presented to a user.

The representation of nodes is an important factor in determining the effects of browsing through a hypertext document. There are two major styles of representing nodes in hypertext: *frame-based* and *scrolling-based*. In a frame-based hypertext system, fragments of information are presented in individual frames. To follow a link is to go from the current frame context to a new frame context. Different frames are usually displayed in different windows. In scrolling-based hypertext system, the non-sequentialism of information structure is embodied by hiding away some information with buttons inside a document. To browse around a document is to bring information of interest into view by means of document scrolling and in-line expansion and contraction of buttons. New information is shown within the context of the current document. A well known representative of the frame-based style is *HyperCard* [Sey91], one of the scrolling-based style is *Hyperties* [Shn87].

Links

Besides nodes, links are the other fundamental building block of hypertext systems. A link is a pointer from one hypertext node to another, and it can be either unidirectional or bi-directional. Each link is a directed connection between related pieces of stored information within the underlying database. Links can have names, types, and other properties.

There are three main classes of links [BD91]:

- ***Association links*** are the most common, and reflect various ways in which one node brings another node to mind. For example, one node might give the reason for the assertions in another node.
- ***Aggregation links*** join a node that represents a whole with its parts. For example, a node representing a book is the aggregate of the nodes representing its

chapters, and would be joined to them by aggregation links of type chapter.

- ***Revision links*** join a node to earlier and later versions of itself, and have no obvious parallel in discourse grammar.

Anchor

Anchors are the representation form of links on the screen. An anchor could be highlighted text, a button, an icon, or some other graphics. Only via selecting its anchor, the underlying link can be activated.

Webs

Web is a set of links that is stored separately from the material (the information stored in the nodes) that is linked together. By opening a web, a hypertext reader imposes a particular set of anchors and links on the underlying hypertext database. This principle has been supported, for instance, in the ***Intermedia*** system [SZ87]: "*Webs present users with contexts in which to collect a set of links that are superimposed on a set of documents. A document can exist within the context of many webs, but only those markers in the current web are shown in the document.*".

Editors

Editors are an important part of hypertext systems in that they enable users to create nodes and link them into the network. Most hypertext systems are based on the principle that users also need to write. For example, ***Concordia*** [Wal88] is a structure-oriented editor, giving writers templates for their nodes with special slots for standard information like keywords and section headings.

Navigation

Navigation is a facility that helps readers to effectively browse through the hypertext network. Several kinds of navigation tools can be found in hypertext systems. Some of these navigation tools are:

- **Browser Tool:** A hypertext browser is a program or subprogram that can display

a diagram of a network of nodes, or node component that has already been created and cannot be changed significantly. For example ***gIBIS*** [CB88] has a graphical browser which provides a visual presentation of the network of nodes and their interconnecting links. It also provides a direct manipulation style interface to the node and link (i.e., display node contents).

- **Tour Tool**: A tour is a whole stream of nodes connected together, as opposed to the pairs of nodes connected together. With a tour tool, reader can just select a button or menu command to specify and follow a tour.
- **History Tool**: A history is a tours record of the nodes that a user has accessed in viewing a hypertext network. A history tool supports the user in visiting nodes that are contained in the nistory record.

3.2 Object Modelling Facilities

In subsection 2.1.2 *Object Modelling*, we have introduced the notions of object, attribute, and method. To capture and maintain the information about objects, their properties and relations, modelling facilities are required. Facilities found in state-of-the art systems support the following activities:

- Viewing the information about the objects in different representations. For example, you may only view the properties of a specific object, or you may view all the objects and their relations in a matrix or table format. These different views support designers in model creation, refinement, and comprehension.
- Generating reports to check and validate the model in terms of consistency and correctness.
- Generating code from the object properties, one of the great benefits of object modelling.

We shall illustrate these ideas with *Paradigm Plus*, a recently developed tool for object modelling. Paradigm Plus is a Meta-CASE tool that provides support for the entire software development life cycle [Pro93]. The tool supports diverse object-oriented methods (Rumbaugh/OMT, Booch, Fusion, Coad/Yourdon, Shlaer/Mellor) with the capability of transferring models from one method into another one.

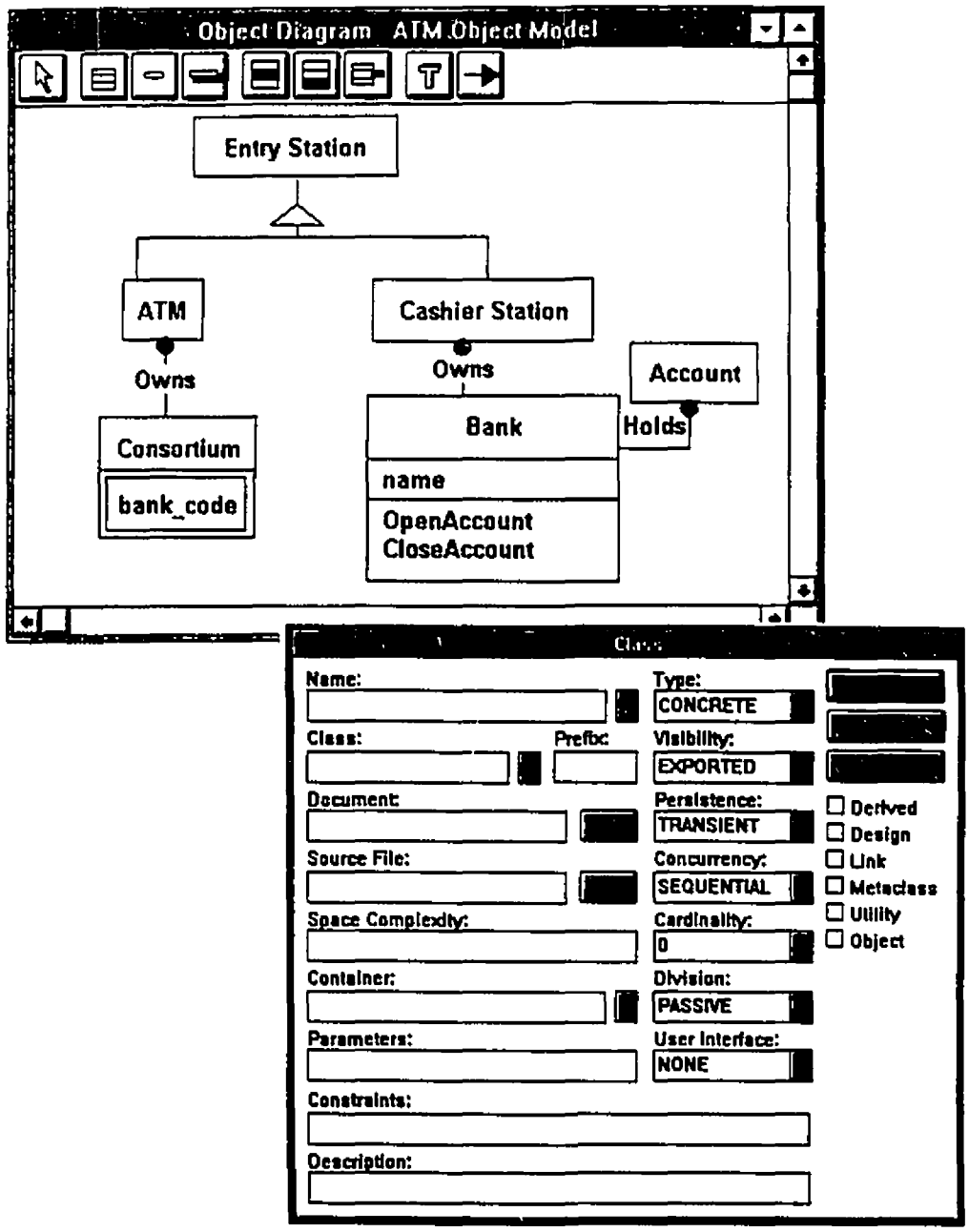


Figure 3.1 : ATM Object Model and the Class Dialog Box.

It stores all the information about objects and their relations in an integrated active object repository. This information is entered through different dialog boxes.

Figure 3.1 shows an ATM (Automated Teller Machine) object model created by

using the object diagram editor of Paradigm Plus. It also shows the class dialog box where the user can enter the various properties of the class.

There are three important components in the tool that match the facilities that we discussed above. These features are:

- Generic Editors: they support the editing and viewing of all objects stored in the object repository. In its current version, the tool comprises a diagram editor, a matrix editor, and a browser. As an example, Figure 3.2 shows the matrix editor, where the user can edit and view the relationships among objects.
- Report Generators: the user creates reports from the object repository through the use of a script language supported by the tool. Some of these reports are Object summary, Diagram summary, and Consistency Check.
- Code Generators: the user generates automatically source code from the object repository, once sufficient design information has been entered. The tool supports the C++, C, Ada, Smalltalk, and SQL languages.

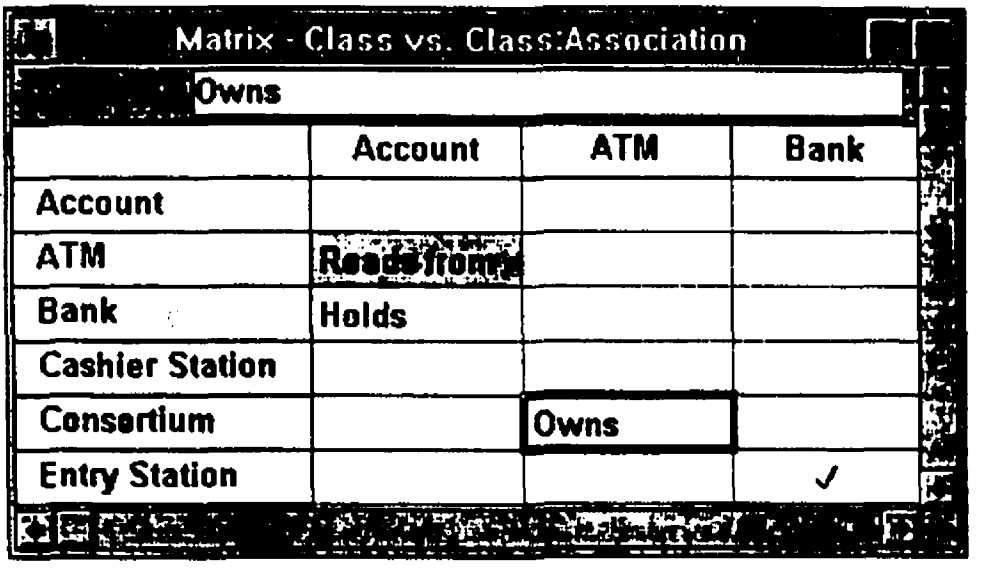


Figure 3.2 : The Matrix Editor.

We conclude that object modelling has great success in supporting software development. This success should be carried over to business modelling.

3.3 Using Hypertext Concepts in Business Modelling

In order to meet the requirements discussed in section 2.2 *Model Documentation Requirements*, and to support formal and informal documentation in business modelling, we decided to introduce the concepts of hypertext into our business modelling approach.

In the following points, we shall illustrate how we relate the features of hypertext to our business modelling concepts.

- **Nodes in hypertext are similar to the different objects in business modelling:**

  Nodes in hypertext hold small pieces of information such as text, a bitmap image, a picture, and so on. In the business modelling context, places, transitions (in Petri net) and objects (in object modelling) have their own information (formal and informal) attached to them, such as attributes, time constraints, dependencies, database information, forms, and graphics.

- **Links in hypertext are similar to arcs and relations in business modelling:**

  Links in hypertext connect pieces of information to each other. In business modelling, arcs relate places to transitions (actions), and relations connect objects in object modelling.

- **Document authoring in hypertext is similar to model editing in business modelling:**

  Document authoring in hypertext enables a user to construct, edit, and display nodes and links. In business modelling, model editing allows the user to draw, edit, and display the different objects of the business model.

- **Navigation in hypertext is similar to model traversal in business modelling:**

  Navigation in hypertext helps users to browse through the hypertext network. In business modelling, model traversal is comparable to the navigation mechanisms of hypertext.

From the above points we can conclude that the entire hypertext structure provides an appropriate natural, and complementary data model for business modelling. Table 1 summaries these points.

Table 1: Relate the features of Hypertext to Business Modelling

Hypertext	Business Modelling
Nodes	Objects
Links	Arcs and relations
Document authoring	Model editing
Navigation	Traversal of model

Chapter 4 From Macrotec to Hypertec

We believe that business modelling should comprise detailed and possibly complex information about model components and their interactions. We also envision that hypertext is the best tool to capture, organize, and present formal and informal documentation in business modelling.

In this chapter, we shall present our approach for integrating hypertext into business modelling, which consists of complementing the Macrotec environment with "Hypertec". We shall begin with an overview of the Macrotec environment, providing both a system and a user perspective. Then, we shall explain the integration of hypertext into Macrotec. Finally, we shall discuss in detail the Hypertec tool, Macrotec's component for capturing and managing formal and informal documentation.

4.1 Macrotec

Macrotec is the result of a joint project between Group DMR Inc. and the CRIM. The project, started in 1991, has been supported in part by research grants from NSERC, Canada, and by Macroscope Informatique Inc. (project managed by Group DMR Inc., Montreal).

The overall goal of Macrotec is to provide an environment that comprehensively supports architectural modelling and high-level requirements specification of business processes and information systems.

4.1.1 Macrotec Environment

The modelling formalism underlying Macrotec is *Macronet* [1], a simple, yet expressive and flexible formalism for business modelling [KSvB94]. Macronet combines concepts such as entity-relationship modelling of information, object-oriented modelling, event and data flow analysis into a uniform modelling framework. Its semantics has been defined through the formalism of hierarchical, colored Petri nets (see section 2.1.1.1).

In order to support and validate the Macronet approach, the Macrotec environment has been developed [KLO+93, KOS94]. Its overall architecture and functionality is depicted in Figure 4.1. The environment is conceived as a toolset, comprising the following tools: *Modelling*, *Alternate Viewing*, *Performance Analysis*, *Graphical Simulation*, *Substitution*, and *Utilities Access*.

- ***Modelling Tool:***

  A tool for the graphical editing and validation of models;

- ***Alternate Viewing Tool:***

  Permits user-defined iconic representations of actions and places as well as event and entity-flow representations;

- ***Performance Analysis Tool:***

  Generates quantitative results such as action throughput, resource utilization, waiting time for actions and average occupancies of places;

1. "Macronet" is a contraction of the words *Macro*scope, our project's name, and Petri *net*.

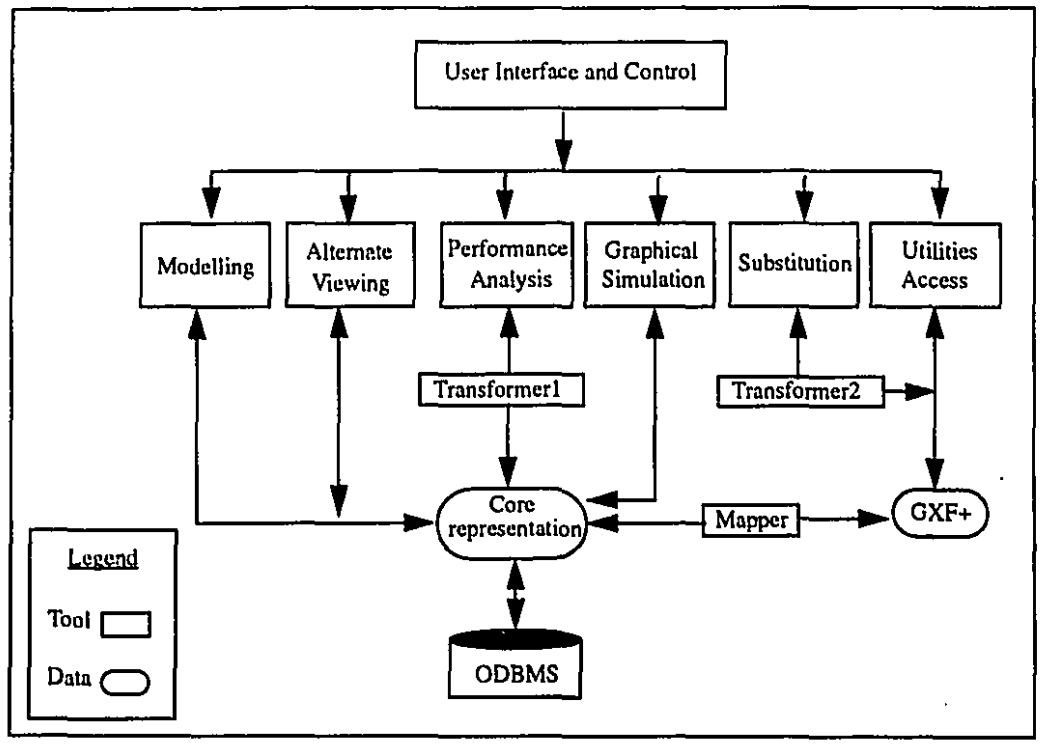


Figure 4.1 : Macrotec Environment.

- ***Graphical Simulation Tool:***

  Facilitates the graphical and interactive simulation of the model, and thus the visualization of the model components' interaction;

- ***Substitution Tool:***

  Supports multi-level modelling. The tool provides mechanisms for the decomposition and abstraction of network parts into sub-nets and super-nets, describing, respectively, lower and higher levels of abstraction.

- ***Utilities Access Tool:***

  Provides facilities for automatic graphical layout and database visualization, and gives access to a data exchange format.

All these tools are integrated at the user interface level, where the user can access the functionality of their components and switch between them.

All information to and from the tools is managed by the *core representation*, which can be mapped into the *GXF+* graphical exchange format and which interacts with the underlying *database management system*. The *database* is the back-end storage facility of the system. It is responsible for the storage and retrieval of the core representation and for handling different model versions. In our current implementation, we use *GemStone* [2], an object-oriented database management system.

Macrotec consists of two categories of tools. The first category includes the tools that manipulate graph layout data. Such tools store their data in the *GXF+* representation. *GXF+* is our customized version of *GXF*, a standardized graph exchange format [MEN92]. Supporting GXF+/GXF allows us to easily exchange data with other, special-purpose, GXF-based systems such as the automatic layout tools being developed at University of Toronto. Non-GXF+-based systems require data transformation programs. For instance, integrating our substitution tool (implemented before adopting the GXF+ standard) required the development of the *Transformer2* program. Mapping of the core into the GXF+ representation and vice versa is carried out by the *Mapper* component.

The second group of tools manipulates the model data that are not related to graph representation. In the case where these tools are part of the Macrotec process, for example, the graphical simulation tool, they interact with the core representation directly. Otherwise, a data transfer program to and from the core is required. For instance, the performance analysis tool, running as a separate process, interacts with the main Macrotec process via *Transformer1*.

4.1.2 A User's View of Macrotec

Users interact with the Macrotec system via a single main window, giving them access to the full functionality of the system and allowing for easy switching between the different tools. Figure 4.2 shows the Macrotec main window containing the model of a delivery system with products being ordered, transported, and delivered to their destination.

2. Gemstone is a registered trademark of Servio Corporation.

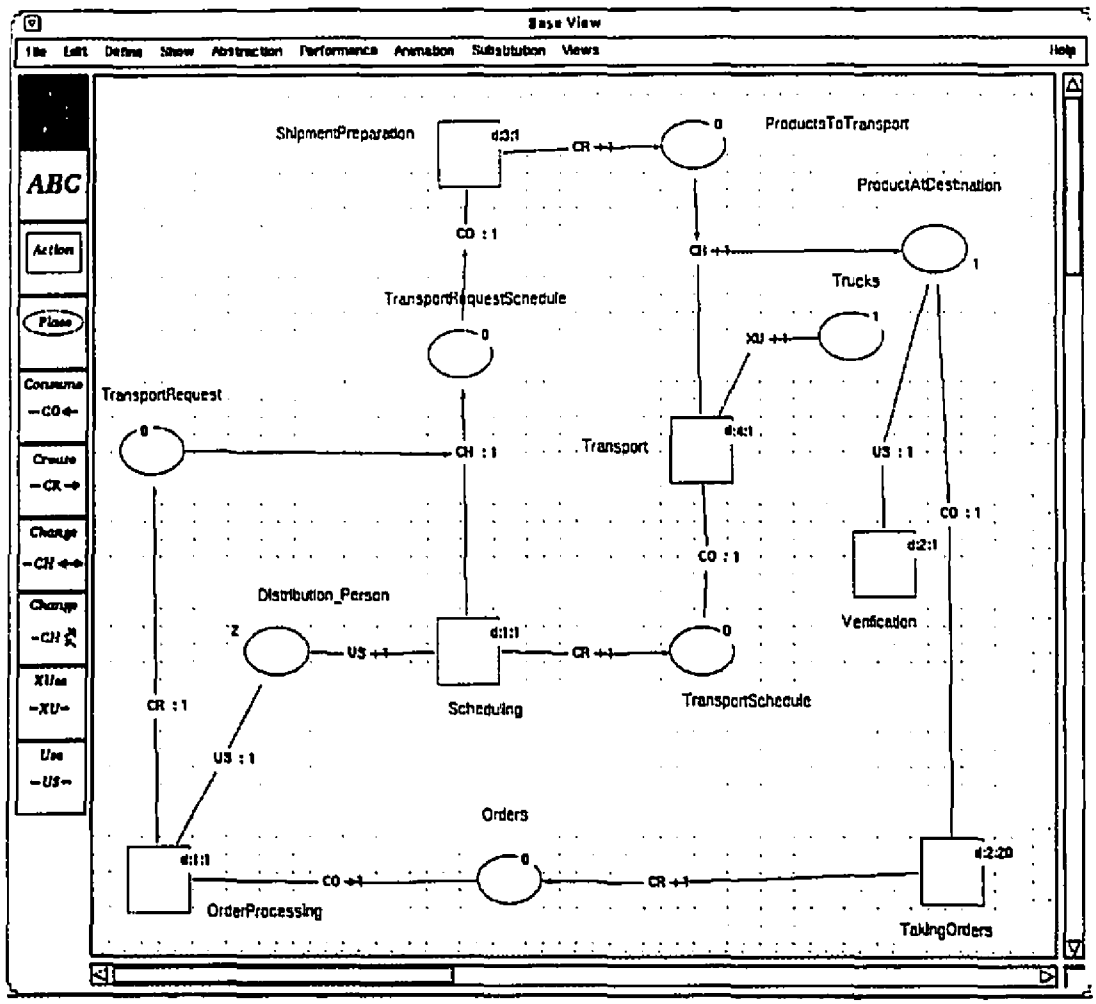


Figure 4.2 : Macrotec Model of a Delivery System.

The main window consists of three parts: the menu bar, the palette, and the drawing area.

- The *menu bar* gives users access to the different functional components of the system, including graphical editing, simulation, performance analysis, substitution and automatic layout;
- *The palette* contains the models' basic building blocks, where places are represented as ovals, actions as rectangles, and relations as directed arcs broken by the relation's acronym. The user may select an icon from the palette for creating

and editing text, actions, places and relations;

- *The drawing area* is the place where the network is displayed and edited.

4.2 Requirements for Integrating Hypertext into Macrotec

Based on our discussion in section 3.3 *Using Hypertext Concepts In Business Modelling*, the integration of hypertext components into Macrotec imposes some requirements and constraints. The components should contain state-of-the-art, general hypertext features: support for browsing, authoring, ease of use, etc. [Nie90]. In the following, we shall emphasize a few issues in the user interface and functionality that we consider important in the integration of hypertext into Macrotec:

- **User Interface:**

  - The hypertext system should be an additional tool of Macrotec, whose use is optional (see for instance [IG90]);

  - Interaction with the hypertext system should be in the main window of Macrotec, and its user interface should be an extension of Macrotec's user interface, for example, additional pull-down menu, etc.;

  - Graphical diagrams present only the main or crucial parts of the overall information according to some regulation and computation. In business modelling, such diagrams can be used as navigational aid which provides the users with an overview of the whole information they deal with. To our convenience, the Macrotec modelling network can be interpreted as graphical diagram of hypertext. This way, users will not "be lost in hyperspace" (see for instance [EH89]);

- The hypertext system should support users with an editor to create, edit, and display the annotations components;

- Links in the hypertext system should be divided into two groups: original Macrotec relations interpreted as links ("system links"), and links created by the user (see for instance [Big88]). It should be possible to turn these links on and off in graphical views;

- The hypertext system should support a history mechanism, since business modelling systems are interactive systems and usually contain a large information base. Thus, the user should be supported by good navigation facilities such as history tool.

- **Functionality:**

  - Macrotec is a single user system, and so the hypertext system needs not take into account multi-user issues;

  - Annotations in the hypertext system should be available for all Macrotec units, i.e., nodes, groups of nodes, pages, sub-pages;

  - No restrictions on annotations should be imposed. The user should be free to add text, graphics, images, source code, etc. into nodes, i.e., annotations;

  - Linking should be non-intrusive. This way, no special program is needed to convert hypertext documents into linear documents;

  - The attributes of Macrotec nodes, i.e., places and actions, should be displayed, together with their annotations (see for instance [CG91]).

4.3 Hypertec

In order to validate our approach discussed in section 3.3 *Using Hypertext Concepts in Business Modelling*, and to integrate hypertext into Macrotec in the way we detailed in the previous section, we have developed Hypertec. Hypertec, which stands for "*Hyper* Macro*tec*", is a hypertext-based component supporting authoring, displaying, modifying, manipulating, and navigating through the business modelling information base.

In the following, we shall discuss Hypertec from a user's perspective. First, we shall present the basic building blocks of Hypertec. Then, we shall describe the different ways of navigation in Hypertec. The final section will address Hypertec's annotation facilities.

4.3.1 Basic Building Blocks of Hypertec

Nodes and links are the basic building blocks of hypertext. Hypertec is also designed around these two basic constructs.

- **Nodes**

In Macrotec, models are represented as networks of components which are either of type action or place. Hypertec interprets these components as nodes in the hypertext sense and allows the user to complement the node information provided by Macrotec. The user may add, for instance, comments or other informal documentation, both in textual or graphical form. There are two types of node information, *annotation information* and *attribute information*:

- *Annotation information*, or in short, "annotation", can be any kind of information except attribute information. It can contain, for example, user comments concerning a certain result from the performance analysis of the model, some product's distribution schedule, forms, database information, etc. Annotations

can be textual or graphical and are editable. On the screen, the existence of annotation information for a particular component is indicated by a small black rectangle icon, displayed inside the action or place shape. This feature helps users recognize that there is an annotation with that particular place or action. For example, in Figure 4.3, the black icon in the "Distribution" action indicates that this action has an annotation.

- *Attribute information* provides standard information about a node. The user specifies this information upon model editing. It is then made available in the Hypertec node for read-only use. This information includes node *type* (it's value could be action or place), node *name*, node *entity-class*, and data which is dependent on the type of the node. In the case of a place, for instance, a data item is the number of entities. In the case of an action, a data item is action type.

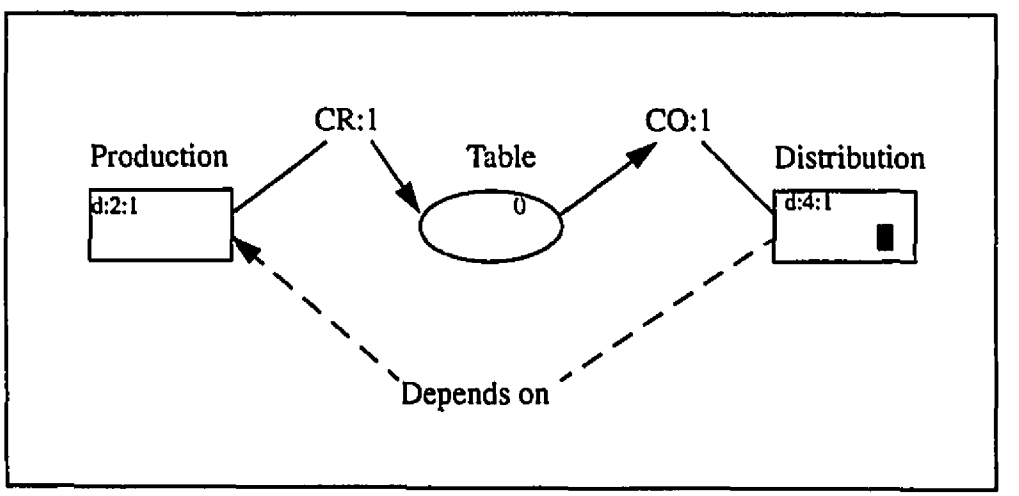


Figure 4.3 : Annotation Icon and user-defined Link in Hypertec.

• Links

In Hypertec, any connection between two component nodes of the Macrotec model is considered a link. They are typed and directed, connecting a source node to a destination node. We distinguish two sets of links, based on their derivation pattern:

- ***System Links:***

  Macrotec supports the five Macronet relations Consume, Change, Create, Use and XUse, which in turn connect an action to a place (except for Change which may connect an action to two places). In Hypertec, these five relations are interpreted as *system links*, since they can be derived automatically from the system, i.e., the underlying Macronet model. Thus, there are five types of system links, which are labelled as CO, CH, CR, US and XU, respectively. The directions of these link types are given by the directions of the corresponding relations (see Figure 2.2). On the screen, system links are represented by solid lines.

- ***User-defined Links:***

  In Hypertec, users can create their own links by specifying a source and a destination node, together with a link label. Each *user-defined link* is typed and unidirectional. A user-defined link may connect an action/place to an action/place. The direction of such a link must be specified by the user. The user may also specify (textual) link labels. Links carrying the same label and directions are considered having the same type. There is no limitation on the number of link types, and users are free to create as many links as they want. The only restriction to link creation is that for two given components, only one link or two counter-directional links are allowed. This policy gives the users maximum freedom of authoring, while preventing them from disorientation by introducing too many links. On the screen, user-defined links are represented by dashed lines helping the user distinguish at a glance system links from user-defined links. For example, Figure 4.3 shows the user-defined link "Depends on" which links "Production" and "Distribution" actions.

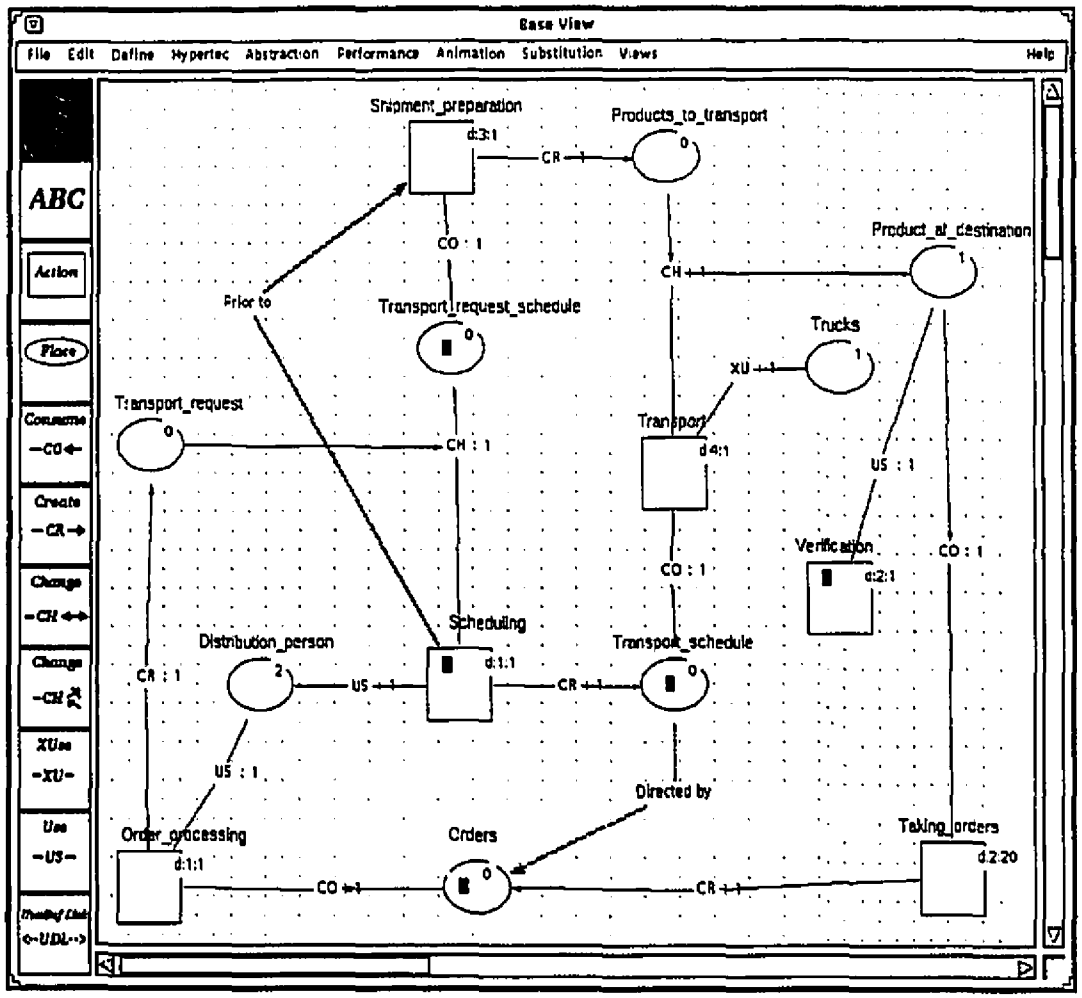


Figure 4.4 : Model of the Delivery System of Figure 4.2, enhanced with Macrotec Version that integrates Hypertec.

4.3.2 Navigation

Navigation support comprises facilities for browsing and managing the potentially large network of nodes and links. In Hypertec, it comes in three facets: graphical diagrams, history list and navigation anchors.

• Graphical diagrams

From a Hypertec perspective, the business modelling networks created with Macrotec can be considered as structural overview diagrams. These diagrams may be used as navigational aids, providing the users with an overview of the whole information they are dealing with. The Macrotec main window allows both construction and browsing of these diagrams.

Figure 4.4 shows the Macrotec main window after integrating Hypertec, displaying the graphical diagram of a delivery system (a non-Hypertec-based version is shown in Figure 4.2). It shows some of the Hypertec features integrated into Macrotec: the Hypertec pull-down menu, the user-defined link icon at the bottom of palette, and the small annotation icons (black rectangles; see below). The Hypertec menu gives users access to the different functions of the Hypertec tool; the user-defined link icon allows them to construct user-defined links.

• History list

Hypertec provides a history mechanism allowing users to return to nodes they have visited before. Hypertec represent the history list via a dialog box that contains a list of the most recently visited nodes (in the current implementation a maximum of ten nodes). A node is considered visited when its annotation window has been opened (see below). The list is ordered chronologically from bottom to top, with the most recent node at the bottom, with duplicate entries removed. The history list also indicates the type of nodes: if it is action, the indicator will be */A/*, and if it is place, the indicator will be */P/*. Selecting an item from the history list allows users to jump back to the desired node and annotation window. The history dialog box can be activated by selecting the "History ..." command of the "Hypertec" menu.

Figure 4.5 depicts the history dialog box and a sample list resulting from the delivery system example (Figure 4.4). Double-clicking an item in the list, or selecting the item and pressing the "OK" button, will get the corresponding node selected, bring up its annotation window, and will close the dialog box.

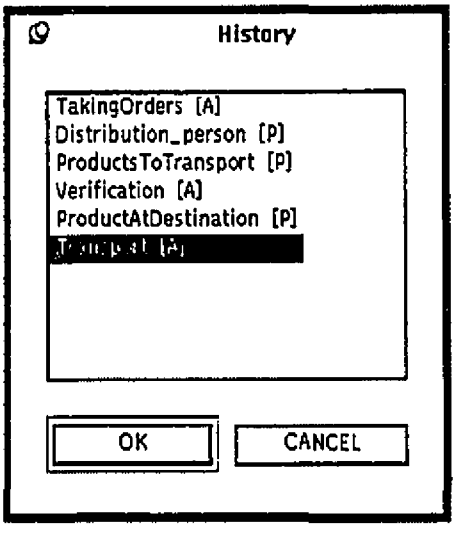


Figure 4.5 : The History Dialog Box of Hypertec.

- **Navigation anchors**

Hypertec provides a navigation anchor mechanism to facilitate browsing through the business modelling information base. For a particular node, all of its related system links and user-defined links are provided as a list. Each item in the list represents an anchor and contains the following (textual) information: link direction, link label, and name of designated node. From this list, users can easily find out about the links that connect to a particular node and navigate to the designated nodes. We will further elaborate on navigation anchors when discussing the navigation subwindow (see below).

4.3.3 Information Window

The Information window is at the heart of the Hypertec tool. We will refer to it as *InfoSheet*. It is a pop-up window for displaying and editing detailed business modelling information for the place or action at hand. The InfoSheet window can be activated in three different ways: by double-clicking on a certain node (place or action) in the drawing area, by selecting a node in the drawing area and invoking the "Open InfoSheet" command of the "Hypertec" menu, or by activating one of the visited nodes from the history dialog box.

An InfoSheet window consists of three subwindows and a button area (see Figure 4.6). The three subwindows are labelled: *Annotation*, *Attributes*, and *Navigation*. The annotation subwindow allows users to create, edit, and display annotation information; the attributes subwindow displays the attributes of the node at hand; and the navigation subwindow displays the navigation anchor information of the node.

The button area at the bottom of the InfoSheet window contains two buttons labelled <-- *Previous*, and *Reusable*. The first one is a push-button allowing users to bring up the InfoSheet data of the previously visited node. This button becomes disabled when there is no previous node. The button labelled *Reusable* is an on/off toggle button which controls the reuse of the displayed window for new InfoSheet data. If the user sets the button to *off* and activates another node, a new InfoSheet window will pop-up. If, however, the user had set the button to *on*, the data of the activated node would have been displayed in the current InfoSheet window, replacing the data of the previously active node. This mechanism gives the user the opportunity to browse the InfoSheet data.

The title of the window is identical to the label of the *node* it belongs to. If the label changes, the title will change accordingly. Furthermore, modified attributes are updated in the "Attributes" subwindow.

In the following, we shall elaborate on the three subwindows of the InfoSheet.

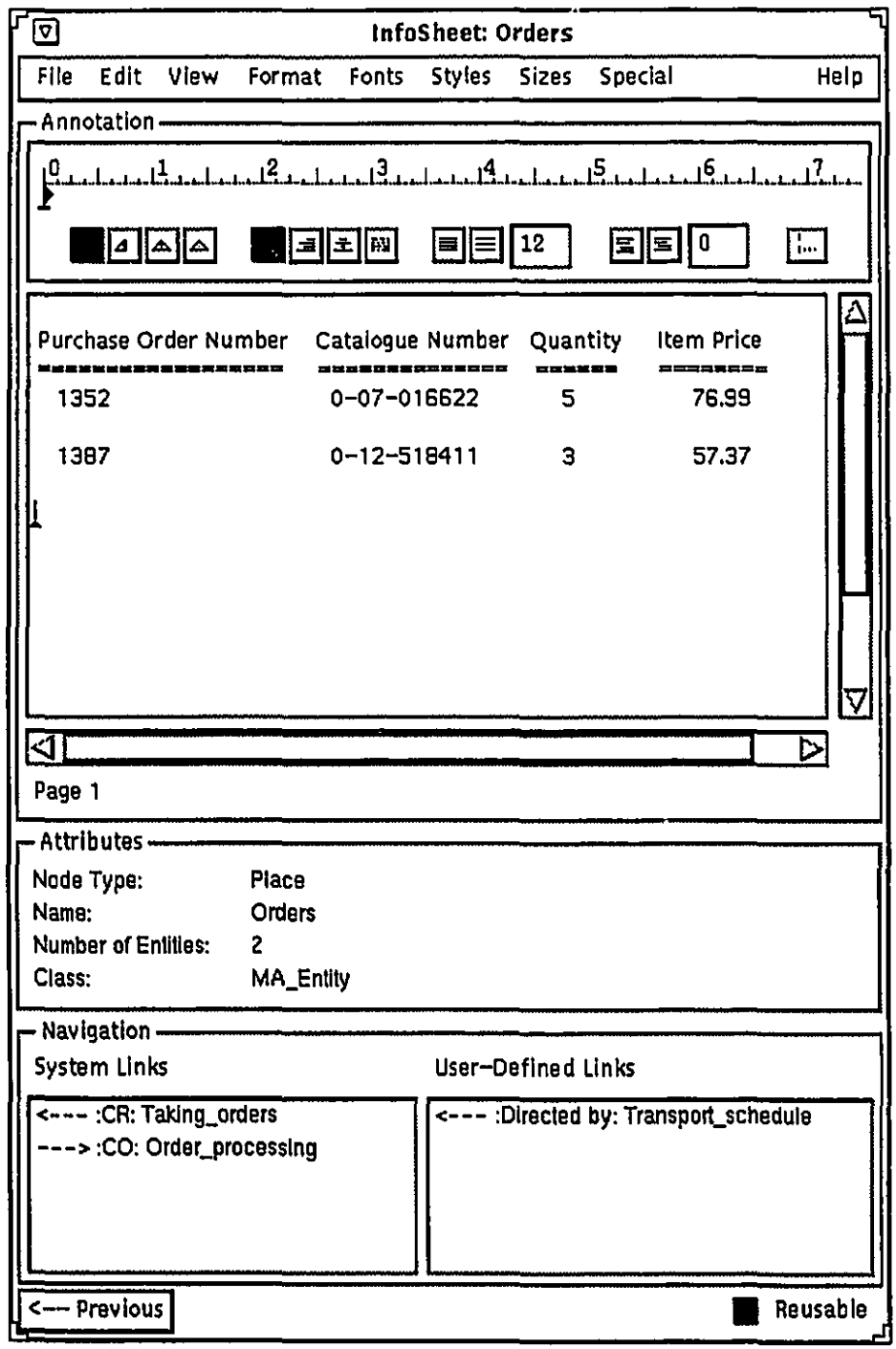


Figure 4.6 : The InfoSheet Window.

• Annotation subwindow

The annotation subwindow allows users to create, edit and display the annotation information of a currently active node (Place or Action). This information could be user comments, documents related to the model, forms, etc. The user manipulates this information via a text editor. This text editor is not only a word processor, it can also import graphics in various formats such as Macintosh pictures, Sun rasterfiles, X11 icons, encapsulated postscript, and portable bitmap.

The menu bar above the annotation subwindow allows the user to perform various commands. For example, besides saving the annotation information to the database along with its corresponding node, the "File" menu allows the user to save the annotation information to a file via the "Save to File ..." command and load from a file via the "Load from File ..." command.

The annotation subwindow is displayed at the top of Figure 4.6. It presents the annotation information of Place "Orders": the purchase order number, catalogue number, quantity, and price per item.

• Attributes subwindow

The attributes subwindow is a read-only area, used for displaying the attributes information of the currently active node. The attributes information is derived automatically from the underlying business model. If the users want to change this information, they do so by editing the model in the network editor. During edition, the attributes information is updated automatically.

The attributes subwindow contains the following items:

- *Node Type*. Its value can be either action or place;
- *Name*. Its value is identical to the node label displayed in the drawing area;

- Data dependent on type of the node:
  - If node type is place, the data will be *Number of Entities*. Its value represents the number of token inside the place;
  - If node type is action, the data will be Action *Type*. Its value represents data relevant to dynamic analysis and simulation tools;
- *Class*. Its value represents the entity-class.

The attributes subwindow is displayed in the middle of Figure 4.6. It shows the attributes information of the place Orders: its type, its name, its class, and the number of entities.

• Navigation subwindow

The navigation subwindow displays the list of navigation anchors we discussed in subsection 4.3.2 *Navigation*. This subwindow consists of two scrollable lists labelled: *System Links* and *User-Defined Links*. The system links list, displayed on the left hand side of the navigation subwindow, contains all the system links connected to the active node. The user-defined links list, displayed on the right hand side of the navigation subwindow, contains all the user-defined links connected to the active node. Clicking on an item in either list activates the designated node and brings up its InfoSheet data.

Each item listed in the two scrollable lists has the following format:

<Link Direction>: <Linked Label>: <Name of Linked Node>

<Link Direction>: it represents the link direction via one of the following symbols: "----", "--->", "<--", "<-->". Figure 4.7 shows the complete list of representations of systems links.

<Link Label>: its value depends on the type of the link. For system links, it is one of: "CO", "CR", "CH", "XU", "US". For user-defined links, it is identical to the link label displayed in the drawing area.

<Name of Linked Node>: its value is identical to the node label of the designated node in the drawing area.

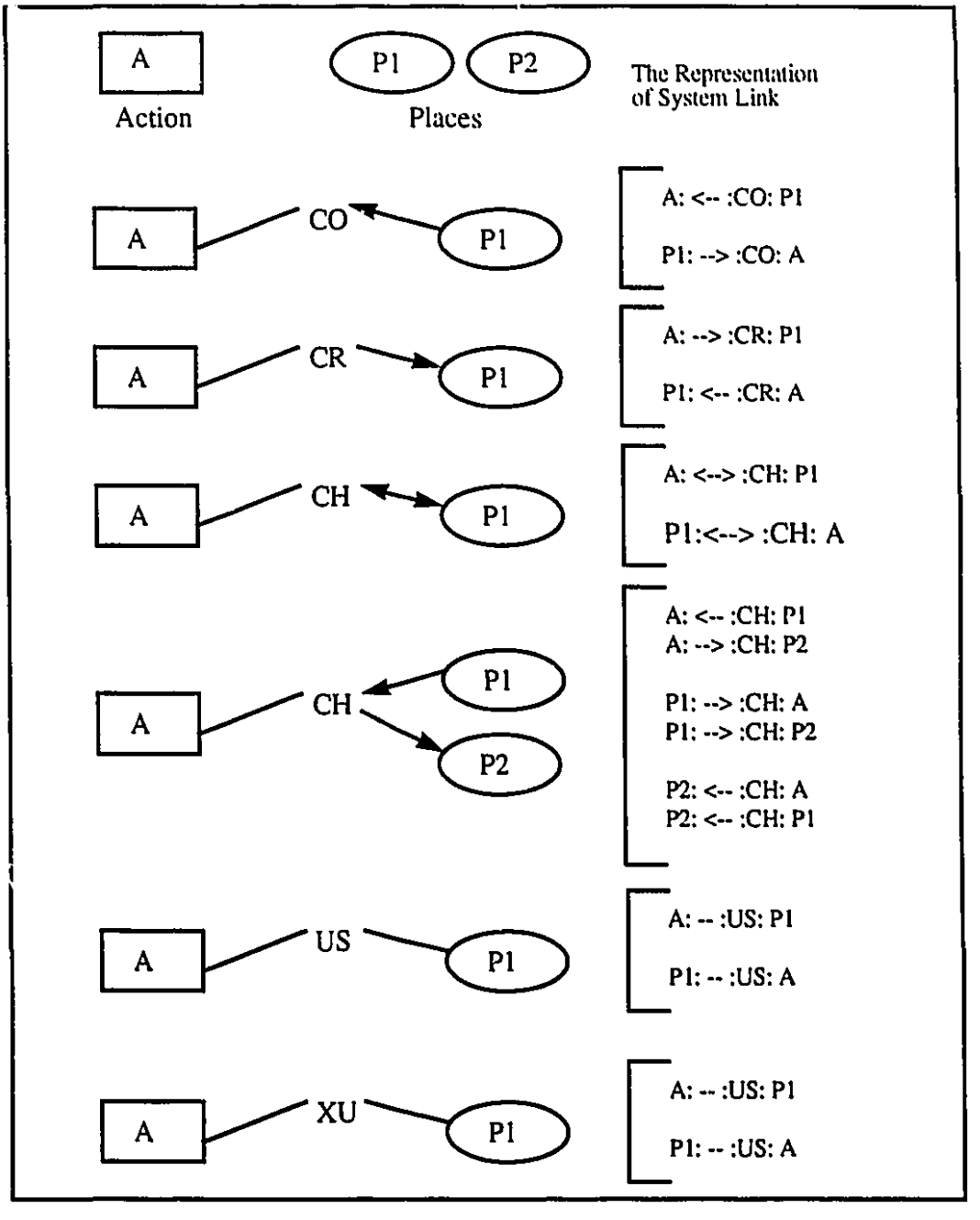


Figure 4.7 : All Cases of System Link Representations.

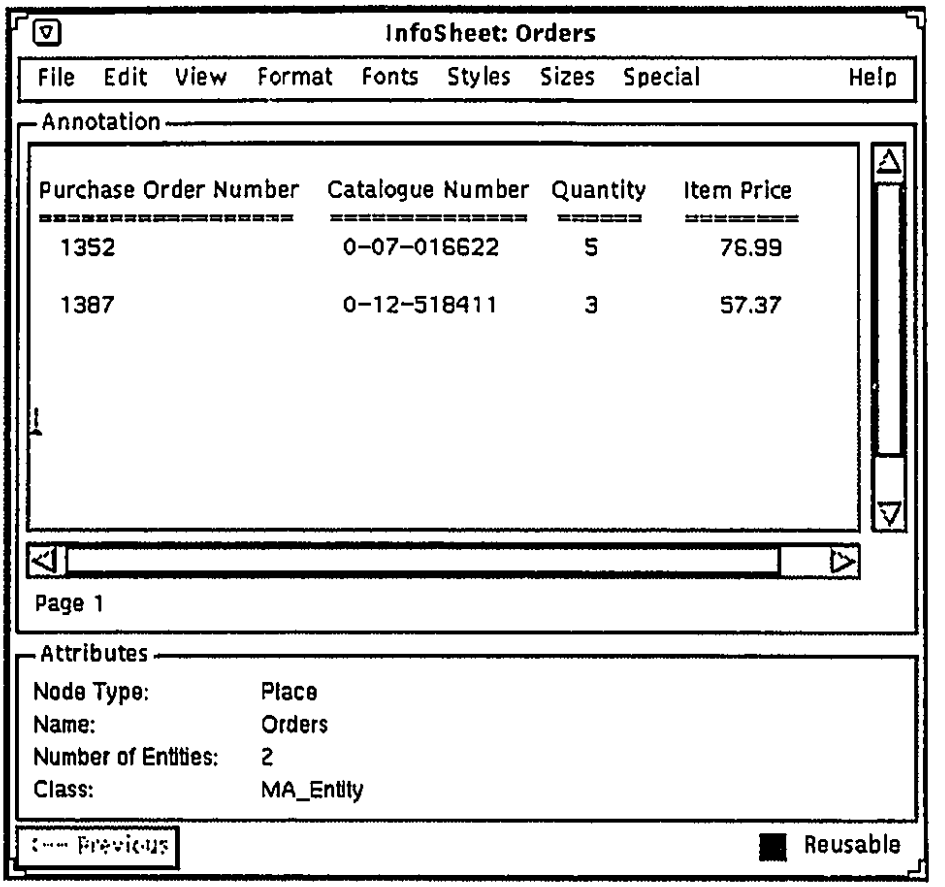


Figure 4.8 : Controlling the View of the three Subwindows.

The navigation subwindow is displayed at the bottom of Figure 4.6. It shows that the currently selected place Orders connects to the two actions: Taking_orders and Order_processing by system links, and also connects to the place: Transport_schedule by a user-defined link. For example, the first item in the system link list denotes an incoming link ("<---") to the place Orders, having as type label CR, and as source the Taking_orders node.

These three subwindows of InfoSheet can be separately toggled on/off via the "Show ..." command of the "Hypertec" menu with options to apply on current or all InfoSheets, or on an individual basis via the "View" menu in an InfoSheet window; the InfoSheet window is appropriately re-sized. In Figure 4.8, the InfoSheet displays only the annotation and attributes subwindows.

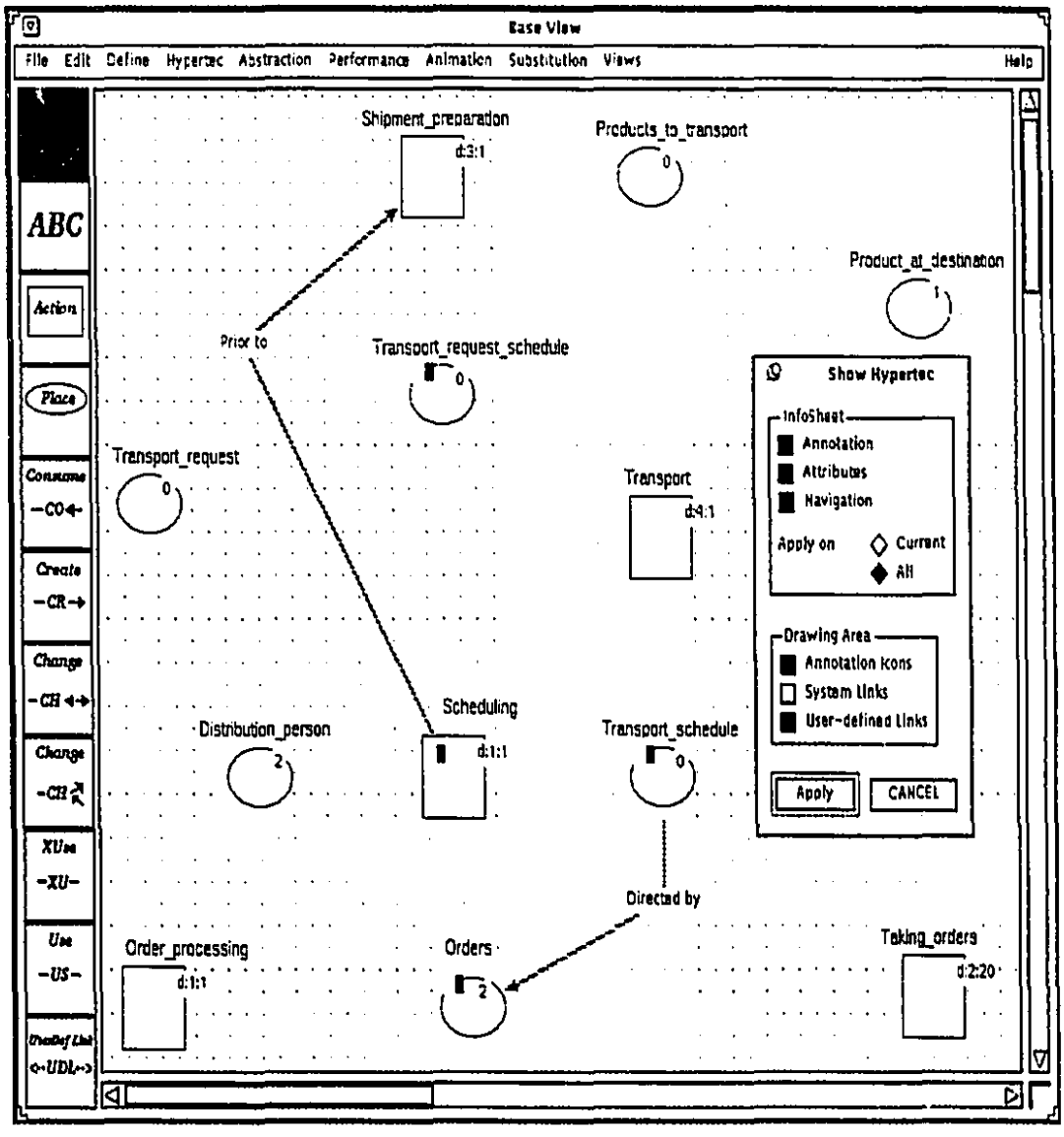


Figure 4.9 : The "Show Hypertec" Dialog Box.

The "Show ..." command of the "Hypertec" menu activates a dialog box labelled "Show Hypertec". This dialog box provides the user with the facility to control the viewing of the Hypertec components. It consists of two groups of options: InfoSheet, and drawing area. The InfoSheet options allow users to view part of the InfoSheet window, i.e., the annotation, attributes or navigation subwindow, or a combination thereof,

and to apply that to the current node or to all nodes. The drawing area options allow the user to turn on/off the Hypertec annotation icons, system links, and user-defined links on the graphical diagram. Figure 4.9 presents the "Show Hypertec" dialog box. In this example, the system links are turned off, and all InfoSheet subwindows are enabled and applied to all InfoSheets.

Chapter 5 Design and Implementation of Hypertec

In this chapter, we shall discuss the design and implementation of Hypertec, with a focus on user interface and data management aspects. Hypertec is implemented in C++, on top of the ET++ framework [WGM89] and the Gemstone object-oriented database management system. The architecture of Macrotec, complemented with the Hypertec tool, is depicted in Figure 5.1.

We shall first address the user interface, discussing the ET++ framework, software reuse in Hypertec, and the integration of the Hypertec user interface into Macrotec. Then, we shall focus on data management, introducing the GemStone system and detailing the extension of the database scheme and the core representation for accommodating Hypertec in Macrotec.

5.1 User Interface

We shall first describe the ET++ framework which we used to build our user interface. Then, we shall briefly discuss the reuse of *Write*, an application developed with ET++ which we reused for building the InfoSheet window, the major component of Hypertec. Finally, we shall explain the integration of the Hypertec user interface into Macrotec.

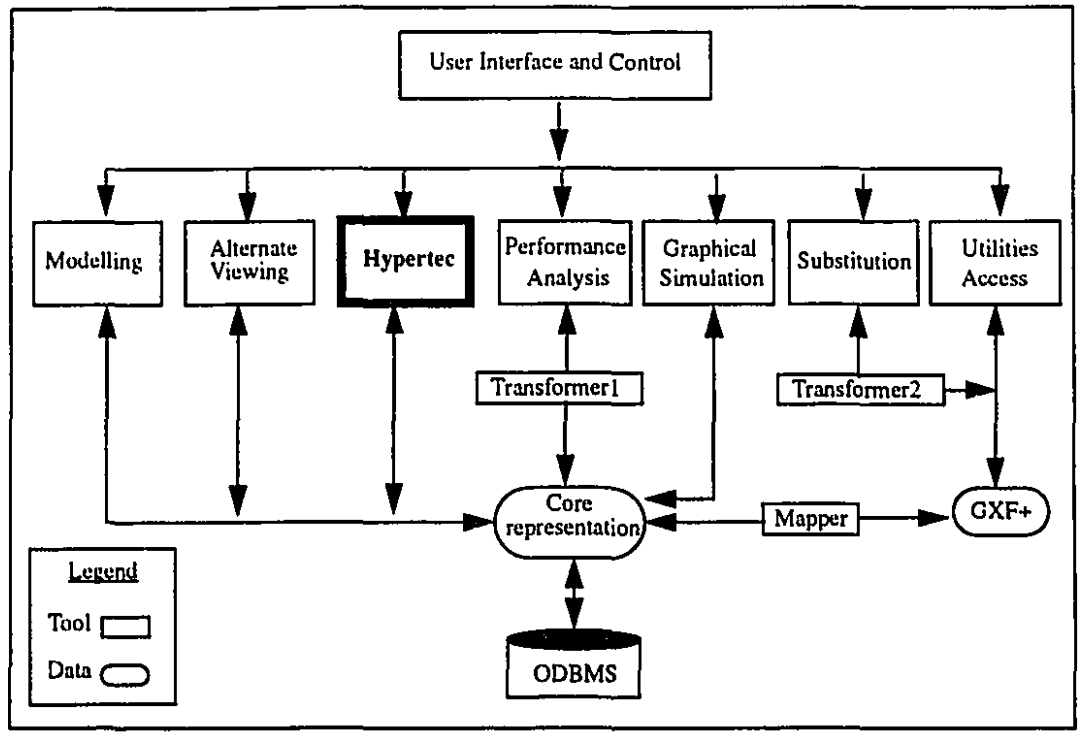


Figure 5.1 : Hypertec Tool within the overall Macrotec Architecture.

5.1.1 ET++ Application Framework

The ET++ framework was written at the University of Zurich by Weinand, Gamma, and Marty [WGM89]. It is based on an abstract window system interface which currently supports SunView, X11, and NeWS and can run on a variety of Unix workstations.

ET stands for "Editor Toolkit", and the original aim of ET++ was to make it easy to build highly interactive tools such as drawing programs, CASE diagram editors, source code browsers, etc. Many of the ideas in Apple's MacApp system [Sch86] were borrowed and used, which gives ET++ applications a strong "Macintosh feel".

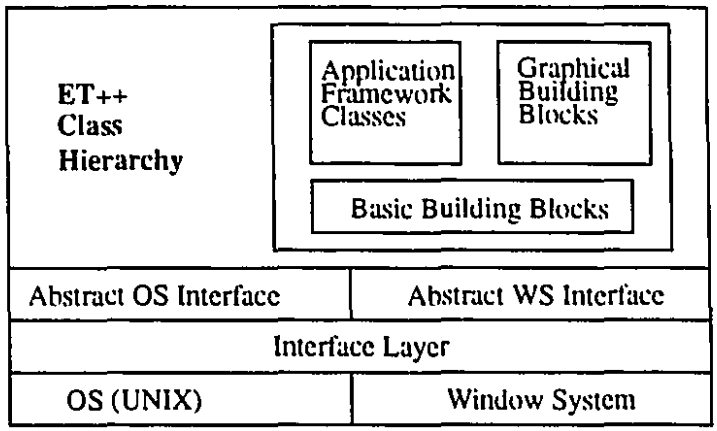


Figure 5.2 : ET++ Architecture.

ET++ is a class library for C++, comprising more than 230 classes, which aims at providing most of the facilities found in the standard Smalltalk class library. It is structured as a single-rooted inheritance hierarchy with many virtual functions available to provide flexibility and opportunities for extension.

The backbone of the ET++ architecture is a class hierarchy and a small device dependent layer mainly mapping an abstract window and operating system interface to an underlying real system. Figure 5.2 shows the ET++ architecture [WGM89]. It consists of four major parts: *Basic Building Blocks*, *Graphical Building Blocks*, *Application Framework Classes*, and *System Interface Layer*.

• *Basic Building Blocks*

The basic building blocks contain the most important abstract classes of the ET++ class hierarchy. The superclass of all classes in ET++ is class *Object*, which defines the protocol for actions common to all classes. Most of the member functions in `Object` are defined as virtual functions, meant to be overridden in subclasses. Every object appear-

ing on the screen is a subclass of *VObject* (visual object). VObject is the graphical foundation class in ET++. It defines an abstract protocol for managing the size and position of visual objects, for drawing them, for handling input, etc. Basic building blocks also define basic data structures such as arrays, lists, sets, etc.

- ***Application Framework Classes***

Application classes are high-level abstract classes that factor out the common control structure of applications running in a graphical environment. The classes *Application*, *Document*, and *Command* are responsible for the overall application behavior. They define an abstract, generic ET++ application. An *Application* object is in overall control, and may manage any number of document objects. The *Document* class holds the data structure or model of the application, and is responsible for loading and storing these data in files. All modification to a Document are implemented as *Commands*, and the Document class permits the user to undo the last Command performed.

- ***Graphic Building Blocks***

The Graphic building blocks comprise the graphical and interactive components found in most user interface toolboxes, such as menus, dialog boxes, buttons, scrollbars, editable text fields, etc. Moreover, they define the framework to easily build new components from existing ones.

- ***System Interface Layer***

The system interface layer provides its own hierarchy of abstract classes for operating system services, window management, input handling, and drawing on various devices. Subclasses exist for implementing the system interface layer's functionality for various window systems such as, News, Sun Window, and X11 window, and run under the Unix operating system. This makes it possible to port ET++ to other environments with a minimum effort. System interface classes are the only classes not derived from the Object class.

Figure 5.3 shows the class hierarchy of the Basic Building Blocks, Graphical Building Blocks, Application Framework, and System Interface Layer.

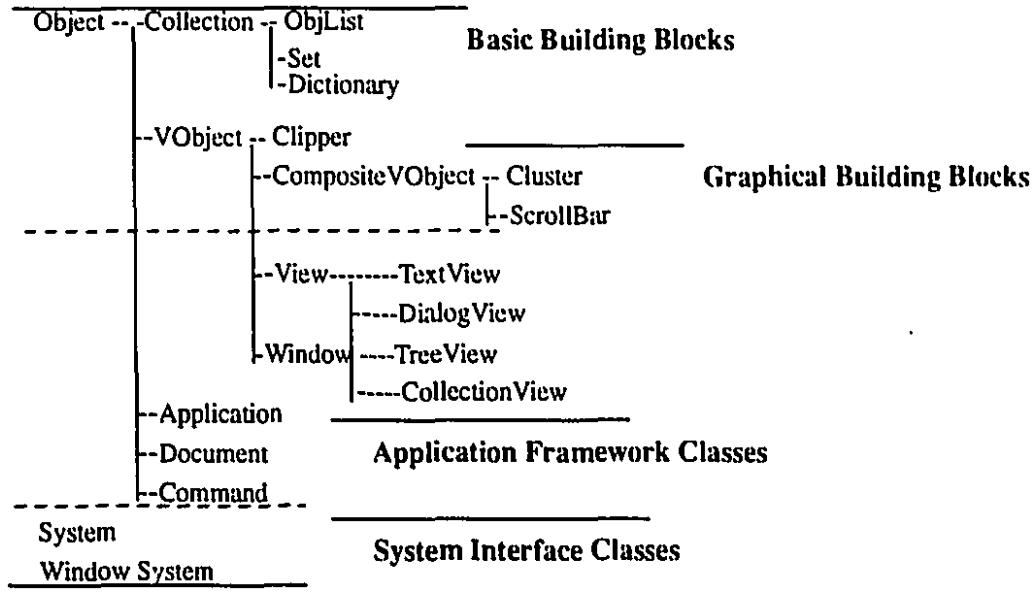


Figure 5.3 : ET++ Class Hierarchy.

5.1.2 Software Reuse in Hypertec

ET++ comes with diverse example applications to help beginners learn it. When studying the *Write* application, a state-of-the-art graphical text editor, we realized that it supported many of the requirements for Hypertec, especially for the InfoSheet window. We then decided to make use of as much of the original Write application's code as possible. We modified many of Write's classes by modifying existing methods, adding new ones and removing others.

For example, one of the classes we modified is the *WriteDoc* class which we renamed *InfoSheet* in our implementation. This class will be responsible for manipulating the three InfoSheet subwindows (annotation, attribute, and navigation) instead of just manipulating the text editor window in the Write application. We added to this class about ten methods and several attributes. One of these methods is called *SetupContents*, which is responsible for setting up the components of the three InfoSheet subwindows for a given node:

```
void InfoSheet::SetupContents(char *title, Shape *node, char *annotation, char *attributes)
{
    SetName(form("InfoSheet: %s", title));
    nodeShape = node;

    //Setup the Annotation content
    if (! annotationTView) annotationTView = new VObjectTextView(this, Point(gPrintMan-
      ager->GetViewRect().Width(), 30), annotationText);
    if (annotation) {
      IStream from(strlen(annotation) + 1, annotation);
      DoRead(from, NULL);
    }
    else {
      annotationText->Empty();
      pageNo = 1;
      InitCounters();
    }
    updateCounters = FALSE;

    //Setup the Attributes content
    if (attributesText) attributesTView->SetString((byte*)attributes);
    else attributesText = new CheapText((byte*)attributes, -1, gApplFont);

    //Setup the Navigation lists content
    MakeSysListView(sysListView);
    MakeUdlListView(udlListView);
}
```

5.1.3 Integrating Hypertec User Interface into Macrotec

In order to integrate the Hypertec user interface into Macrotec, we modified many of Macrotec's classes by adding new attributes and methods, modifying existing ones, and removing others. For example, the BoxShape and OvalShape classes (used to implement actions and places, respectively) were enhanced with new attributes and several methods to manipulate the annotation data, to control the display of the annotation icons, and to access the different member functions of our InfoSheet class. The following is part of the BoxShape class showing the attributes and methods we added:

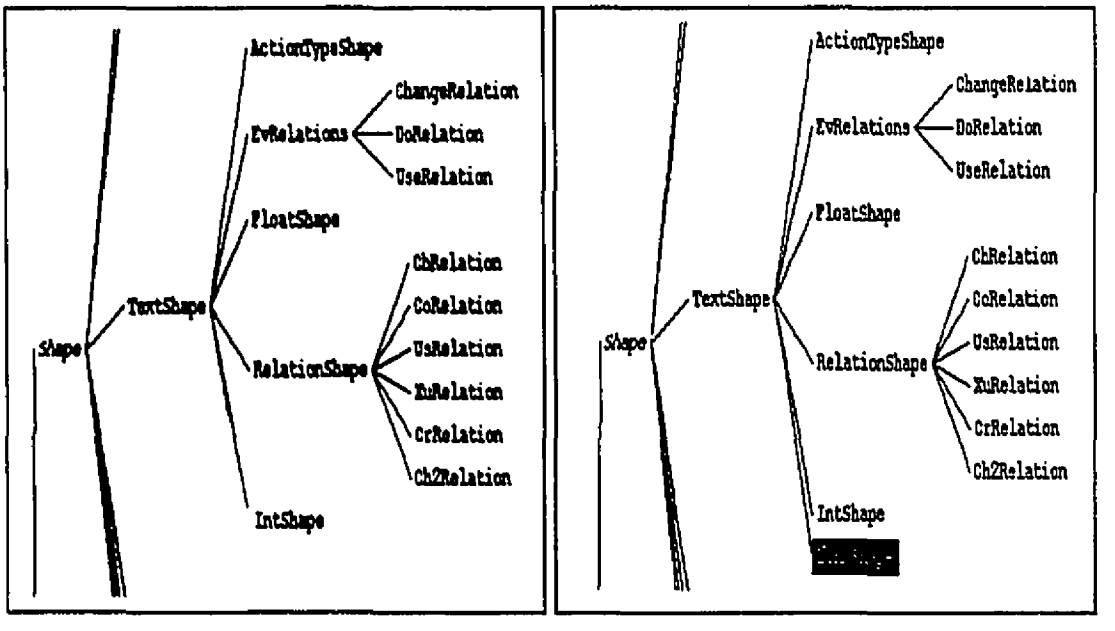
```
class BoxShape : public Shape {
    friend class InfoSheet;
    .....
    //---- InfoSheet ----
    InfoSheet *infoSheet;
    char *annotationData;
    IconShape *icon;
    .....
public:
    .....
    //---- InfoSheet ----
    void UpdateIconShape();
    IconShape *SetIconShape();
    IconShape *GetIconShape();
    void SetAnnotationData(char*);
    .....
}
```

In addition to modifying the existing Macrotec classes, we created various classes and integrated them into the Macrotec class hierarchy. For example, in order to draw the user-defined links in the drawing area, we created a new class called *LinkShape*. This class is derived from the TextShape class, which is responsible for drawing text onto the drawing area (setting display properties, making the drawn text dependent on other objects, etc.). The LinkShape class includes methods to maintain the connection between the nodes and the direction of arrows:

```
class LinkShape: public TextShape {
    ArrowHeads linkArrows;
    Shape *node1, *node2;
    Connection *con1, *con2;

public:
    .....
    Connection *GetCon1() { return con1; }
    Connection *GetCon2() { return con2; }
    void ChangeArrows(ArrowHeads ah);
    bool CanConnectWith(Shape *s);
    .....
}
```

Figure 5.4 depicts part of the Macrotec class hierarchy before and after creating the LinkShape class.



(a) Before (b) After

Figure 5.4 : Macrotec Class Hierarchy before (a) and after (b) creating the LinkShape Class in the User Interface.

5.2 Data Management

The backend of the Macrotec system is GemStone, an object-oriented database management system. GemStone provides interfaces for C, C++, and Smalltalk. We have used GemStone together with its C++ interface which has led to an efficient database interface.

In this section, we shall briefly introduce GemStone. Then, we shall describe the extension of the database scheme and the core representation of Macrotec.

5.2.1 GemStone Database Management System

The GemStone database system [BOS91, MSOP86] is a commercial product developed by Servio Logic Corporation in the USA. The basic approach taken was to examine the Smalltalk language and system, and to identify and support a number of requirements of a Smalltalk-based database systems. The result is the language OPAL, which manipulates persistent data objects controlled by a disk-based storage manager.

GemStone supports simple inheritance for organizing the classes and their attributes. It also supports concurrency control, recovery, dynamic schema modification, transaction management, limited authorization, and queries.

There are two main components in the GemStone architecture *Stone* and *Gem*. They correspond roughly to the object memory and the virtual machine of the standard Smalltalk implementation.

Stone provides persistent storage management, concurrency control, authorization, transaction, recovery, and support for associative access. Stone uses unique surrogates, called object-oriented pointers to refer to objects.

Gem has capabilities of compiling OPAL methods into bytecodes and executing that code, and takes care of user authentications, and session control. Furthermore, it provides the redefined set of OPAL classes and methods which are available in the GemStone system.

GemStone is one of the more visible object-oriented database systems available today in terms of both the database features provided and the contributions to research into object-oriented database architectures.

5.2.2 Extending the Database Scheme

The database scheme underlying Macrotec was built with GemStone's C++ interface. It consists of 26 classes which are responsible for executing SAVE/LOAD operations and ensuring the editor's semantic constraints when actions and places are connected via relations.

In order to extend the database scheme of Macrotec to support the requirements of Hypertec, we created new classes and integrated them to the existing scheme. We also modified several classes in the scheme by adding new attributes and methods to support the saving and loading of the annotation data and links between nodes. For instance, we derived a new class called MA_Link from the existing class called MA_GraphicNode, which is an abstract class responsible for maintaining the name and position of the graphic node.

The following is the MA_Link class, showing its attributes and methods. It is responsible for keeping pointers of its linked nodes, its position, and its direction:

```
class MA_Link: public MA_GraphicNode //INDICT::HMACAToolsClasses
{
    protected:
      MA_GraphicNode_GPTR  GraphicNode1, GraphicNode2;
      MA_CoordinatesXY_GPTR  namePosition;
      int LinkDirection;

    public:
      MA_Link(char*,  MA_GraphicNode_GPTR,  MA_GraphicNode_GPTR,  BoolType
            isAutomatic = FALSE, GS_Class* c = 0);

      MA_GraphicNode_GPTR getGraphicNode1() {return GraphicNode1;}
      MA_GraphicNode_GPTR getGraphicNode2() {return GraphicNode2;}
      void setNamePosition(int,int);
      void setLinkDirection(int);
      MA_CoordinatesXY_GPTR getNamePosition(){return namePosition;}
      int getLinkDirection() {return LinkDirection;}

GS_MEMBERS(MA_Link);
};
```

Figure 5.5 depicts part of the Macrotec class hierarchy before and after creating the MA_Link class in the database scheme.

5.2.3 Extending the Core Representation

In this subsection, we shall first briefly describe the design of the Macrotec core representation. Then, we shall explain the extensions to the core representation which were necessary to support the functionality of Hypertec.

• The Core Representation

Internally, the *core representation* (referred to from this point on as the *Macrotec core*) is the heart of the Macrotec system [KLO+93] (see Figure 5.1). All information to and from the user interface and database is managed in the Macrotec core representation.

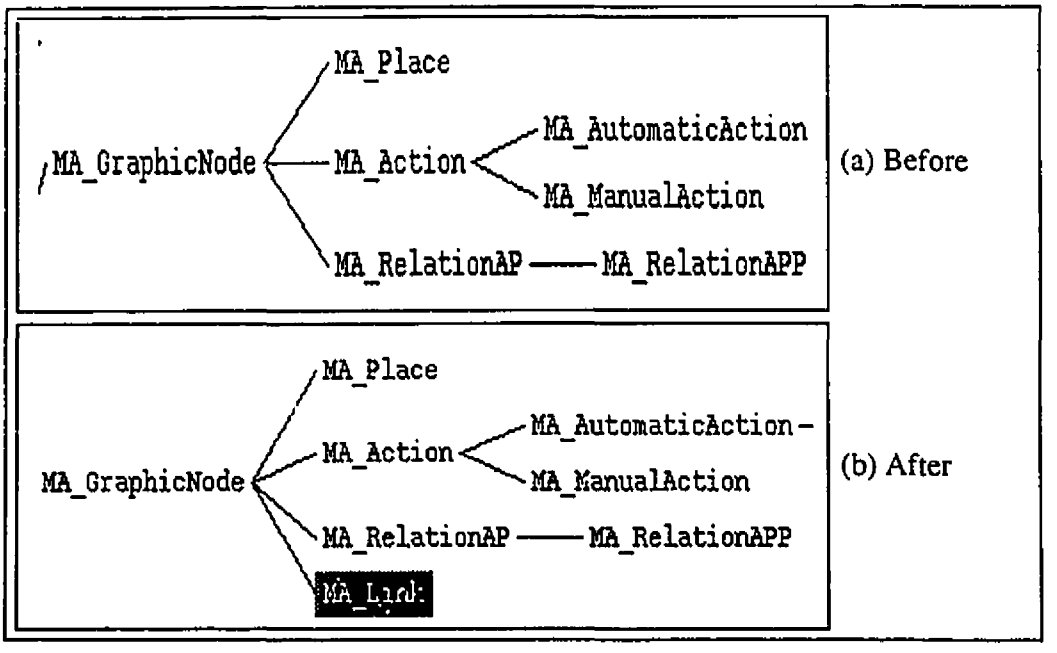


Figure 5.5 : Macrotec Class Hierarchy before and after creating the MA_Link Class in the Database Scheme.

There is another representation called *data representation* (referred to from this point on as the *editor core*) maintained within the user interface. A debatable issue which was struggled with in the past is whether or not to ensure that both representations (editor and Macrotec core's) are identical at all times. The current implementation is such that they are not identical until the user explicitly requests them to be, via a "Save DB" or "Save as DB" menu selection.

There are both pros and cons in this design decision. One big disadvantage of maintaining consistency between the two representations is the implementation complexity involved in having to maintain each and every change made to graphical objects.

There are four commands provided to the user with respect to database storage and retrieval: Save DB, Save as DB, Load Page from DB and DB hierarchy.

The Gemstone database hierarchy of pages is such that we must maintain at most three pointers to access a particular page. They are,

- ***DbArch*** Pointer to the Gemstone architecture which is analogous to the root directory. It contains SuperPages.

- ***DbSuper*** Pointer to the SuperPage within the architecture pointed to by *DbArch*. All SuperPages are displayable. Also, SuperPages may contain SubPages.

- ***DbSub*** Pointer to the SubPage within the SuperPage pointed to by *DbSuper* within the architecture pointed to by *DbArch*.

The user need not deal with SubPages, hence it is possible for DbSub to be a NIL pointer. DbArch and DbSuper will always be valid non-NIL pointers after having loaded a page.

Based on this data management, we shall briefly describe the four commands:

- ***Save as DB***: has two possibilities - whether a SuperPage has been chosen as a target for the save or whether the target is a new page. If DbSuper is NIL, we are dealing with the case of a new page.

- ***Save DB:*** handles the case where the target is NOT a new page, i.e., DbSuper is NOT NIL.

- ***Load Page from DB:*** will require the user to first select a SuperPage or SubPage from the *Page Dialog* in order to properly set the pointers DbArch, DbSuper and DbSub.

- ***DB hierarchy***: allows the user to see the hierarchy of the DB architecture in a newly created window.

- **the Extension**

The SAVE and LOAD operations are the two most important access methods of the core representation. We had to add several extensions to them to support the storage and retrieval of the Hypertec component.

For example, when the user selects the option ***Load Page from DB***, two methods GetGsPlaces and GetGsActions will be invoked.

The functionality of the GetGsPlaces method is to load all the places from the database into an ET++ sequential list of shapes, and to draw each one of these places as an ovalshape with its name and its number of entities into the Macrotec main window.

The functionality of the GetGsActions method is to load all the actions from the database into an ET++ sequential list of shapes. Then, each one of these actions is drawn as a boxshape with its name. Moreover, the method draws the relations between the actions and places.

To support Hypertec, we extended these two methods to do the following operations:

- load the annotation data and user-defined links from the database;
- attach a small black rectangle icon to the place or action shapes in the drawing whose annotation data is not NIL;
- draw the user-defined links between the nodes. These links should be an action/place to an action/place.

Chapter 6 Experience with Hypertec

We have run Macrotec with Hypertec on several examples. In this chapter, we shall begin by presenting one typical example which is realistic, illustrative, and small enough to show the key features and capabilities of Macrotec/Hypertec. Then, we shall describe the scope of Hypertec and report our experience.

6.1 Case Study: Book Purchase Process

In this section, we shall show how Macrotec, complemented with the Hypertec tool, can be used to represent and simulate a real-life business process. The process we studied comprises all the activities related to purchasing a book for CRIM's library. We shall refer to it as *Book Purchase Process*. Briefly, the process starts with a researcher needing a book and preparing a book request. The book request is checked and approved by the librarian, who forwards it to the administration clerk. The clerk processes the request and emits a purchase order for the vendor. The vendor ships the book to the clerk, from where it goes back to the librarian, who in turn informs the researcher of the arrival of the book.

This description of the process is sketchy and is only intended to give an overview of the book purchase process. We conducted a set of interview sessions with all the

CRIM employees that are involved in one way or another in the process. These interview sessions were intended to elicit the entire process. The following part of this section (6.1.1 and 6.1.2) presents the detailed process and the corresponding model that was built using Macrotec/Hypertec.

6.1.1 Elicitation

In order to model the book purchase process, we had to gather and synthesize information about all the roles and activities within the process. This included finding out about the details of the activities being carried out, the actors, and the artifacts generated or used. Eliciting the process information was performed formally through interviews and informally through discussions with book purchase process participants. We carried out the following sequence of steps:

- Preparation of interview questionnaire.
- Interviewing key participants of the book purchase process.
- Informal discussions with participants.
- Synthesis of the collected information.

• Preparation of questionnaire:

In preparing the interview questions, we decided to use the same set of questions for all participants. This meant that the questions had to be general enough to be applicable to all the participants and yet specific enough to draw the detailed responses we needed. From each participant in the book purchase process, we wanted to get the following information:

- Overall structure of the process.
- Roles played by the participant.
- Activities within each of the participants roles.

- Relationships among the roles/activities.
- Pre-conditions and post-conditions for each activity.
- Inputs and outputs for each activity.
- Resources used in performing activities.
- Authorization or validation where applicable.

This was the information that we considered necessary for modeling the process. With this outline as a basis, we prepared a questionnaire consisting of 41 questions. Redundancy was deliberately introduced in order to double-check the accuracy of the information gathered.

• Interviewing the participants:

Having prepared the questionnaire, we made a list of all the key participants in the book purchase process and scheduled interview sessions with them. The following are the participants whom we short-listed for interviews:

- Member of research group.
- Librarian.
- Administration clerk.
- Vendor.

No formal titles have been assigned to any of the participants. The interviews were conducted in a question and answer format, the answers being recorded by us on the interview sheets. Each interview lasted one hour on average. We also examined the documents and forms used by the participants.

• Discussions with participants:

We had several opportunities to discuss the book purchase process informally with the participants. In such discussions, we would seek clarifications on any conflicting information, individual opinions on various aspects of the process, and any other information

that the participants would wish to share with us. Since these were informal sessions, there was no established format.

• Synthesis of information:

After most of the information had been obtained, specifically after the interviews were completed, we started the task of compiling the information we had gathered so as to obtain a formal description of the book purchase process. In the process of synthesizing the information, some gaps or conflicts in the information would emerge. We resolved these by use of several methods including informal discussions as explained above.

The following paragraphs provide a more detailed description of the book purchase process based on the elicited information.

- **Member of research group**

  The book purchase process starts with a researcher needing a book. When the researcher needs a book that does not exist in the library, he or she has to fill out a book request form and send it to the librarian.

- **Librarian**

  The librarian carries out the following tasks:

  - searching for a vendor that can supply the requested book;
  - verifying the information listed in the book request form, for example, vendor address, budget, etc.;
  - approving and sending the approved book request to the administration clerk;
  - informing the researcher about the delivery date as announced by the administration clerk;
  - receiving the new book from the administration clerk;
  - recording the book data into the library database;
  - informing the researcher that the book has arrived.

- **Administration clerk**

  The administration clerk carries out the following tasks:

  - filling out a purchase order form and sending it with payment to the vendor;
  - informing the librarian about the expected delivery date as indicated by the vendor;
  - receiving the new book from the vendor and verifying it with the purchase order;
  - sending the book to the librarian.

- **Vendor**

  The vendor is responsible for:

  - supplying catalogs to the librarian;
  - announcing the delivery date and shipping the book to the administration clerk.

6.1.2 Modelling

In modeling the book purchase process with our system, we used the elicited information to identify all pertinent *Actions*, *Places* and the various documents associated with them, for example, book request form, purchase order form, etc. We also had to determine the standard Macrotec relations connecting actions to places. In addition, we introduced different user-defined links to represent the relationships that are not captured by the standard Macrotec relations.

Figure 6.1 depicts the top-level view of the process, before refining it into submodels, and ignoring exception handling. The user is carrying out a graphical simulation of the process (small control window to the right, "Forward Manual Animation"). Simulation reveals a conflict situation, i.e., the librarian is supposed to *Process New Book* and *Check Book Request* at the same time (the conflicting actions are displayed with a hatched frame). Note that the icons in Macrotec are user-defined and can be associated with different classes of nodes, e.g., human actors, actions dealing with books, with forms, etc.

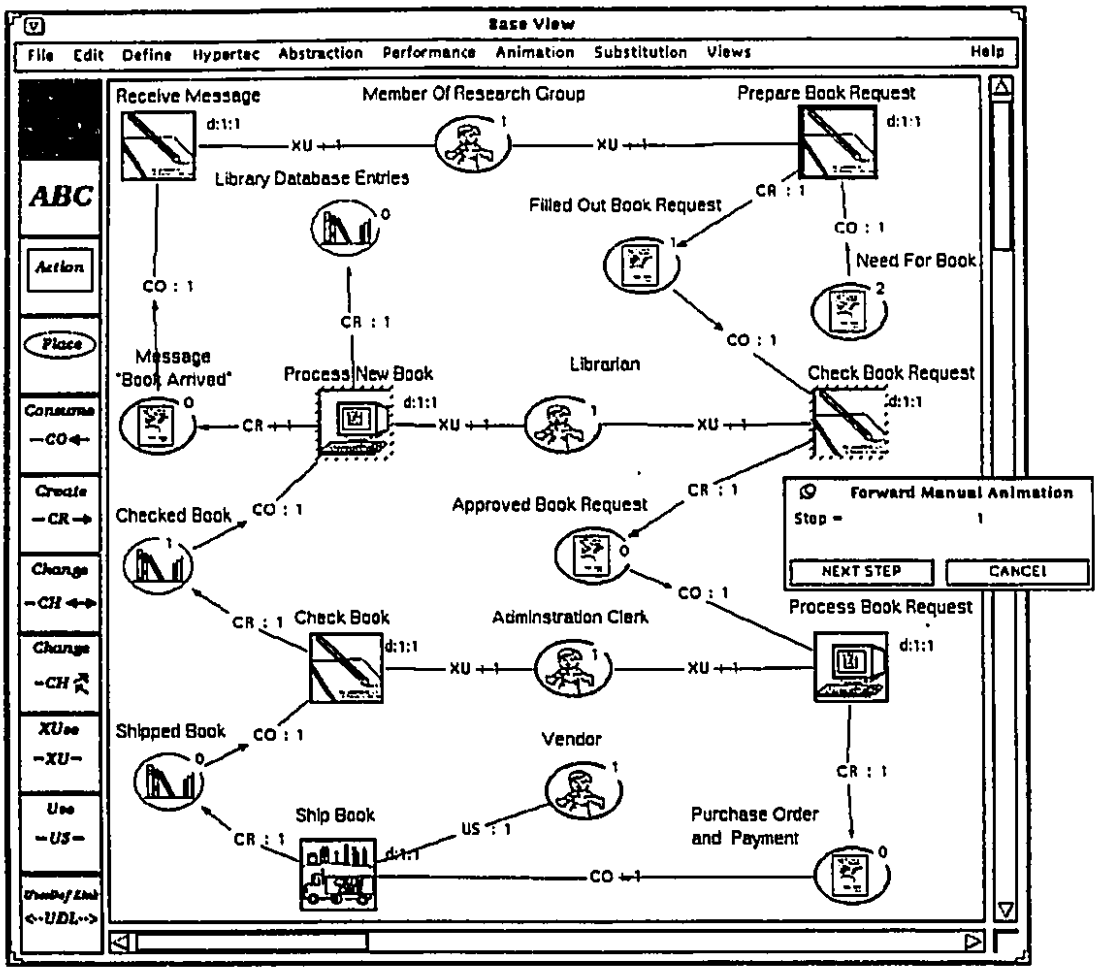


Figure 6.1 : Macrotec Model of Book Purchase Process at CRIM: Top Model Page, Icon View, during Graphical Simulation.

Figures 6.2 and 6.3 screen-shots illustrate the Hypertec component. In Figure 6.2, the user has selected the *Filled Out Book Request* node and triggered the display of that node's info sheet (top left corner window). The info sheet's "Annotation" section contains the scanned request form, and by scrolling, the user can view all the items that should be filled out. The "Navigation" section of the info sheet lists two system links, according to the underlying model. Furthermore, the user has on display the user-defined links that go out from the librarian node (dotted lines, e.g., the link *Announces Delivery Date* to node *Member Of Research Group*).

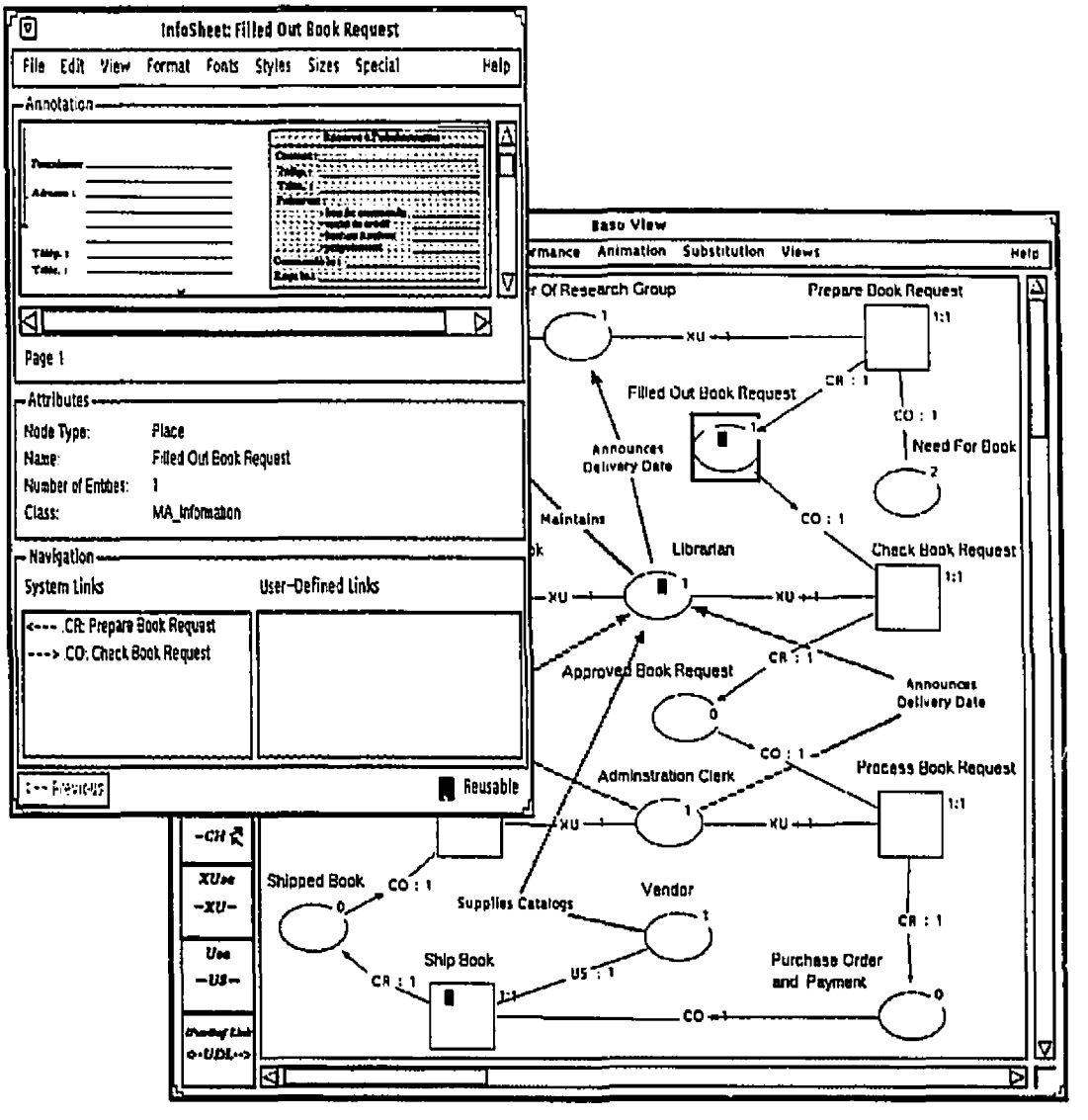


Figure 6.2 : Macrotec Model of Book Purchase Process at CRIM: Top Model Page, InfoSheet of Node "Filled Out Book Request", user-defined Links of Node "Librarian".

In Figure 6.3, the user has suppressed the display of system links, in order to concentrate on the user-defined links.

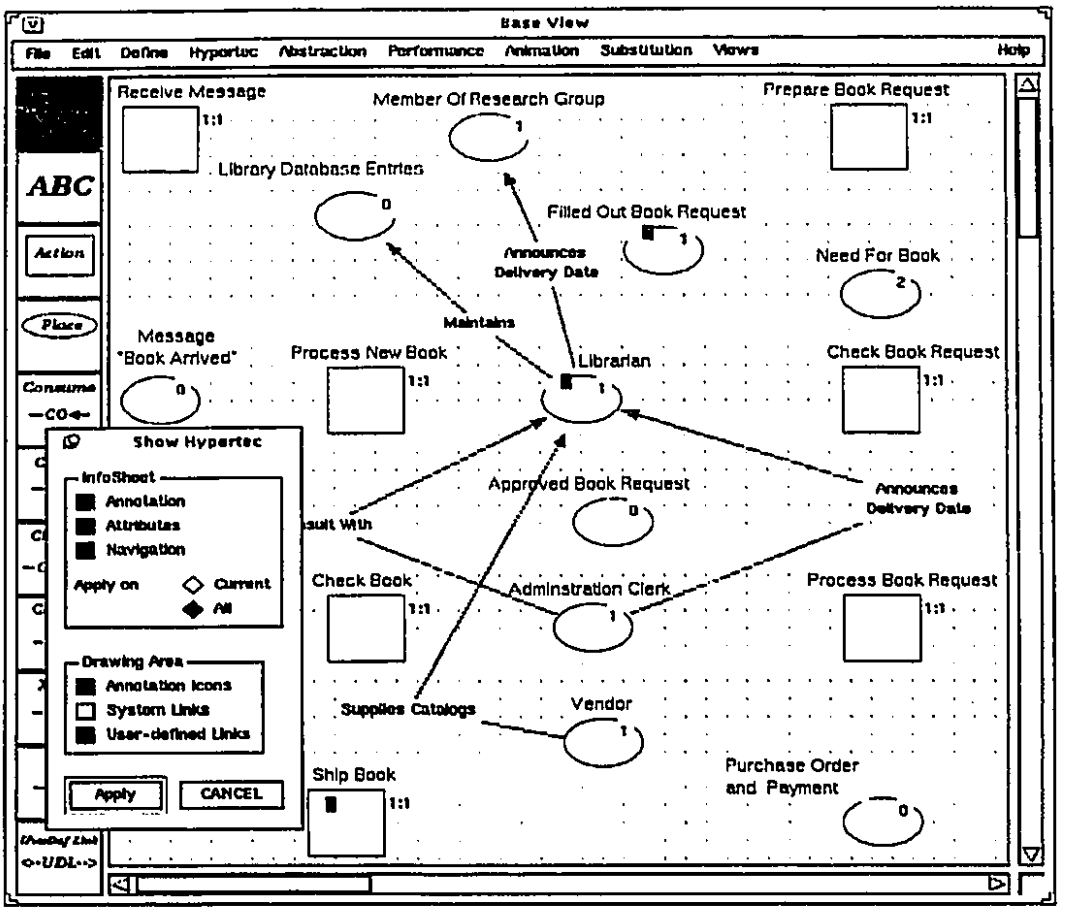


Figure 6.3 : Macrotec Model of Book Purchase Process at CRIM, Selective Display of Information: only user-defined Links are Displayed; Control via Show Hypertec Dialog Box.

In modelling the book purchase process, we valued understandability over completeness and did not reach a level of decomposition beyond two successive refinements. The resulting model is simple, may be even considered simplistic, but has the great advantage of illustrating clearly the main features of the Macrotec/Hypertec tool. In addition, it provides a good understanding of the book purchase process. We applied the tool to other processes, for instance, a bank loan process (see section 6.2, [DFG+94]).

6.1.3 Difficulties Encountered

The ultimate goal in process modelling is to devise a complete, understandable and unambiguous model that describes accurately the activities being modeled. In most situations, a trade-off has to be made between these goals.

Another difficulty is to decide to which level of detail and granularity the model should reflect the actual process, that is, which elements of the process are worth modelling and which are not. Elements should be left out if their modelling does not provide any value-added to the model with respect to understanding, completeness or lack of ambiguity.

Such decisions must be made for the different levels of abstraction that the model deals with. This issue is of particular importance when modelling processes that involve human activities with loose time constraints and frequent exception handling modes.

In modelling the book purchase process, we had to decide which details to capture and which ones to leave out. These choices were made in an ad-hoc fashion using the ideas mentioned above: details that do not provide a substantial value-added to understanding, completeness, and elimination of ambiguity were not modeled.

For example, one of the librarian task is to search for a vendor that can supply the requested book. This task has no formal procedure defined and the person carrying out this task follows only some unstated guidelines and tries to meet some loose constraints. Therefore, trying to refine this task further leads to difficulties without adding much to the its understanding.

6.2 Scope of Hypertec

Macrotec is an environment for organizational modelling and analysis, which is complemented with Hypertec, a hypertext-based component supporting authoring, display and navigation of the documentation that cannot be captured in the formal part of Macrotec.

This combination facilitates the understanding of the process and eliminates the overhead problems generated by miscommunications, misinterpretations or simply, misunderstandings about the entire process or some of its components. They also offer opportunities to gain an insight into the nature of the work and procedures carried out with various kinds of documentation related to these procedures under consideration. This knowledge can be a valuable asset for an organization (reuse, packaging of whatever elements proved valid candidates for such uses). Moreover, it helps process management and project monitoring.

Macrotec, enhanced with Hypertec, makes process models not only more expressive and useful, but may also serve as an on-line training vehicle and means for communication in a multi-person environment. In addition, the resulting models can be seen as a multi-level documentation of the underlying architectures and processes, comprising both formal and informal elements, and allowing for the execution of their formal parts.

An example is the model of the book purchase process described in the previous section (see Figure 6.2). This model might be used by a new employee of the CRIM organization to find out how to proceed in requesting a new book, which people to contact, what form to use, etc. Moreover, he or she might simulate the model to get an intuitive feel of the dynamic behavior of the process.

By using Macrotec/Hypertec, the software engineers, programmers, or maintenance people can easily keep the big picture of the whole documentation in the model, select the nodes they want to visit, follow the links, and navigate through the different parts of the documentation.

We have modelled several medium-sized examples, including a bank loan process and an out-sourcing scenario. Currently, we are modelling documentation processes such as the ones described in [MDL+93, KGT93] and some other processes, for example, software problem reporting and the organization of research meetings.

In order to present an example of a different domain, we are providing a short description of the bank loan process [DFG+94]. The bank loan process is divided into four different steps (see Figure 6.4).

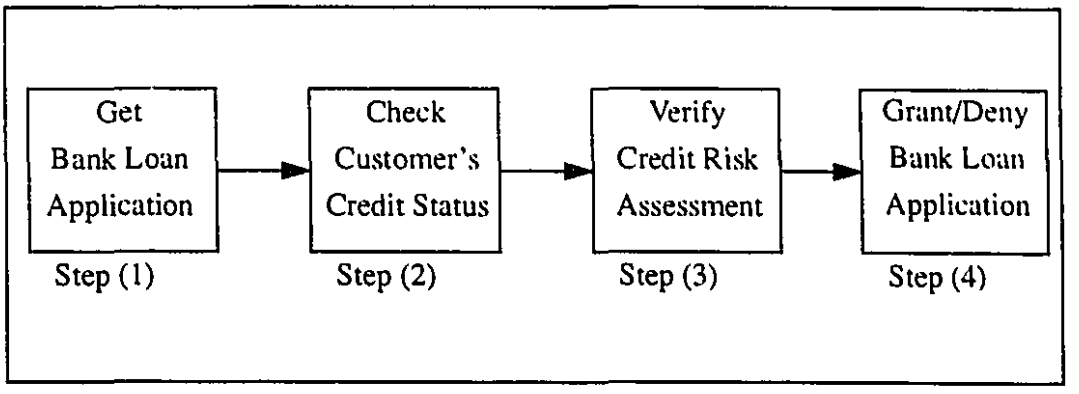


Figure 6.4 : A Four Step Procedure for Processing Bank Loans.

Step (1) begins when a customer applies for a bank loan at the bank branch. The clerk and the customer discuss the type of loan needed. The clerk is responsible for obtaining all the pertinent information and for processing the loan application. At the end of the day, the clerk forwards all applications to the bank's loan agents through internal mail.

Step (2) begins when one of the loan agents receives the customer's loan application, as completed by the bank clerk. The customer's credit status is checked with an external credit rating company. The customer loan application and the customer's file at the bank are then updated with these results. At the end of the day, the loan agent forwards all updated loan applications to the bank's credit risk department through internal mail.

Step (3) begins with the bank's credit officer who is responsible for ensuring that the credit risk for the loan application has been properly assessed. The credit officer then updates the customer's file, and he/she enters the payment instructions. At the end of the day, the payment instructions and applications for both accepted and non-accepted loan requests are returned to the loan agent through internal mail.

Step (4), the loan agent informs the customer as to the outcome of the loan application. If the application is accepted, the customer is asked by the loan agent to come into

the bank branch to sign a promissory note as security for the loan. After having signed the application, the loan agent enters the loan payment instructions into the customer's file. Finally, the loan account is opened for the customer.

Applying Hypertec to these diverse examples, we gained many insights:

- Macrotec/Hypertec is a research prototype and is not as stable as a commercial product;
- Given the size of the out-sourcing example, Hypertec seems to scale up to big examples.
- People to whom we gave demos provided many suggestions for future work (see section 8.1).
- Modelling these processes was an opportunity to experience the difficulties inherent in process modeling activities.

Chapter 7 Related Work

There are several commercial and research packages that have hypertext capabilities such as gIBIS [CB88], Ishys [GS91], DIF [GS90], Kiosk [GCF91], Neptune [DS86], Conversation Builder [CBK+93], HyperCASE [CR92], HyperWeb [FHS+92], and PROXHY [KL91].

In this chapter, we shall discuss briefly some of the existing hypertext systems we found most relevant for our work. Also, we shall describe in detail the Trellis system. Since Trellis is similar to Hypertec, being based on Petri nets, it deserved some special attention.

7.1 Existing Hypertext Systems

In the specification phase of the Hypertec tool, we conducted a literature study of hypertext. This study helped us getting familiar with the diverse hypertext features, finding out about the limitations of hypertext systems, and identifying the features we wanted to implement in our tool. In the following, we shall look through some of the more important hypertext systems.

• KMS

KMS (Knowledge Management System) [AMY88] is a distributed hypermedia system for workstations (Sun and Apollo). It is based on the ZOG system and was developed at Carnegie-Mellon University. KMS is designed to support organization-wide collaboration for a broad range of applications, such as project management, software engineering and computer-aided instruction.

The KMS database consists of screen-sized WYSIWYG work spaces called frames which contain text, graphics and image items. Each item can be linked to other frames or used to invoke programs. The database can be distributed across an indefinite number of file servers and be as large as available disk space permits. The KMS user interface provides a form of direct manipulation designed to exploit a three-button mouse. A combined browser/editor is used to traverse the database and manipulate its contents.

• NoteCards

NoteCards [Hal88] is a general hypermedia environment for workstation systems. Its intended users are authors, researchers, designers, and other intellectual laborers engaged in analyzing information, designing artifacts, and generally processing ideas. The system provides the user with a network of electronic notecards interconnected by typed links; the user represents and organizes collections of related information. The system provides facilities for displaying, modifying, manipulating, and navigating through this network. NoteCards was developed at Xerox PARC, and built around two basic constructs: notecards and links. These two basic constructs are augmented by two types of cards, browsers and fileboxes, which help the user manage large networks of cards and links.

• gIBIS

gIBIS (graphical Issue Based Information System) [CB88] is a hypertext system designed for capturing the rationale of software designs. It is based on the IBIS method [KA70], which, in turn, builds upon the principle of the design process as a conversation among the stakeholders who bring their respective expertise and viewpoints to the resolution of design issues. The people involved in the design process argue about these issues by suggesting positions (i.e. ways to resolve the issue) and arguments which either support or oppose to these positions. All of this is represented in a hypertext structure with three types of nodes (issues, positions, and arguments) and nine kinds of links (responds-to, supports, is-suggested-by, etc.).

gIBIS was developed at MCC (Microelectronics and Computer Corporation) in Austin. It is implemented in C on Sun workstations. gIBIS makes use of a relational database server to facilitate building and browsing of IBIS networks. It supports the collaborative construction of these networks by any number of team members, spread across a local area network.

• Neptune

Neptune [DS86] is a hypertext system for computer-assisted software engineering developed at Tektronix Laboratories. It supports version control for various reports, documents, and code objects. It is designed as a layered architecture. The bottom layer is a transaction-based server named the Hypertext Abstract Machine (HAM). The HAM provides storage and access mechanisms for nodes and links. It also provides distributed access over a computer network and synchronization for multi-user access with recovery and rollback of interrupted or incomplete transactions. On top of the HAM, there are one or more application layers and at the top of them is the user interface layer. The application layer consists of programs that manipulate or transform the hypertext data. The user interface layer provides a window interface for browsing and editing hypertext data and controlling application layer programs.

The HAM presents a generic hypertext model because it defines operations for creating, modifying, and accessing hypertext components. It provides quick access to a complete history of all versions of the information stored in the database.

• Discussion

Each of the above systems has its particular strengths and is more "powerful" than Hypertec according to some criteria. For instance, KMS is very fast in creating text and graphics, and is minimizing system response time (e.g., the time to display a new frame takes about half a second). NoteCards excels in analyzing information, including reading, categorization, interpretation, and technical writing. The gIBIS system is designed (1) to understand the internal structure of design decision and the higher level dependencies which grow up among decisions and (2) to provide effective methods for indexing and retrieval of the large amount of information captured. Neptune has advantages in distributing access over a computer network and synchronization for multi-user access with recovery and rollback of interrupted or incomplete transactions.

On the other hand, Hypertec offers features that are tailored to business modelling which cannot be found in any combination of these tools:

- the business modelling networks can be interpreted as hypertext overview maps which provide the users with an overview of the whole information they deal with;
- the two groups of links: original Macrotec relations interpreted as "system links", and "user-defined links". These two types of links can be turned on and off in the graphical diagram, which gives the user the ability to concentrate on system links, user-defined links, or both of them;
- the InfoSheet windows which display and allow for editing of detailed business modelling information (formal and informal) for the place or action at hand;
- the history mechanism that allows users to inspect their previously visited nodes.

7.2 Trellis System

The Trellis system [SF89, FS89, FS90] was developed by Stotts and Furuta at the University of Maryland and implemented on Sun-3 workstations under the SunView window package. The hypertext model of Trellis is based on Petri nets.

The Trellis screen (see Figure 7.1) is divided into two main parts. On the right side is a Petri net editor and simulator, derived from Molloy's SPAN tool. The left side of the screen is a hypertext browser, subdivided into four windows, each containing a text panel and a button panel. The author uses the net editor to build a Petri net structure and then uses tag string to specify the mappings of places to browsable text elements and transition to buttons.

Each Petri net place is associated with an information element. Browsing this information is accomplished by simply executing the Petri net from its initial marking. During browsing, when a token resides in a place, the text associated with that place is displayed in one of the four text windows [1]. The place name appears in the upper left-hand corner of that window. Any enabled transitions in that post set of the place have their button names displayed in the button menu to the left of the text. Selection of a button in the menu causes its transition to fire, changing the net state and thereby causing a change in the information elements that are displayed in the text windows.

Execution of the hypertext model is controlled either by mouse selection of display buttons in the browser windows or by direct firing of enabled transitions from the net editor.

The unifying formalism in Trellis is net theory, specifically Petri nets. This net theory provides a mathematically rich and notationally flexible basis for hypertext. First, a Petri net is a process model, and browsing a hypertext is a process. Secondly, a Petri net is inherently a concurrency model, so it provides a simple notation and semantics for expressing simultaneous display of multiple information elements. Thirdly, a large body of Petri net analysis techniques has been developed that can be directly applied to problems in the hypertext domain. Together these points allow expression of, and reasoning about, the experience a reader will have while navigating a document.

1. Providing only four windows has been recognized by Stotts and Furuta as a restriction [SF89].

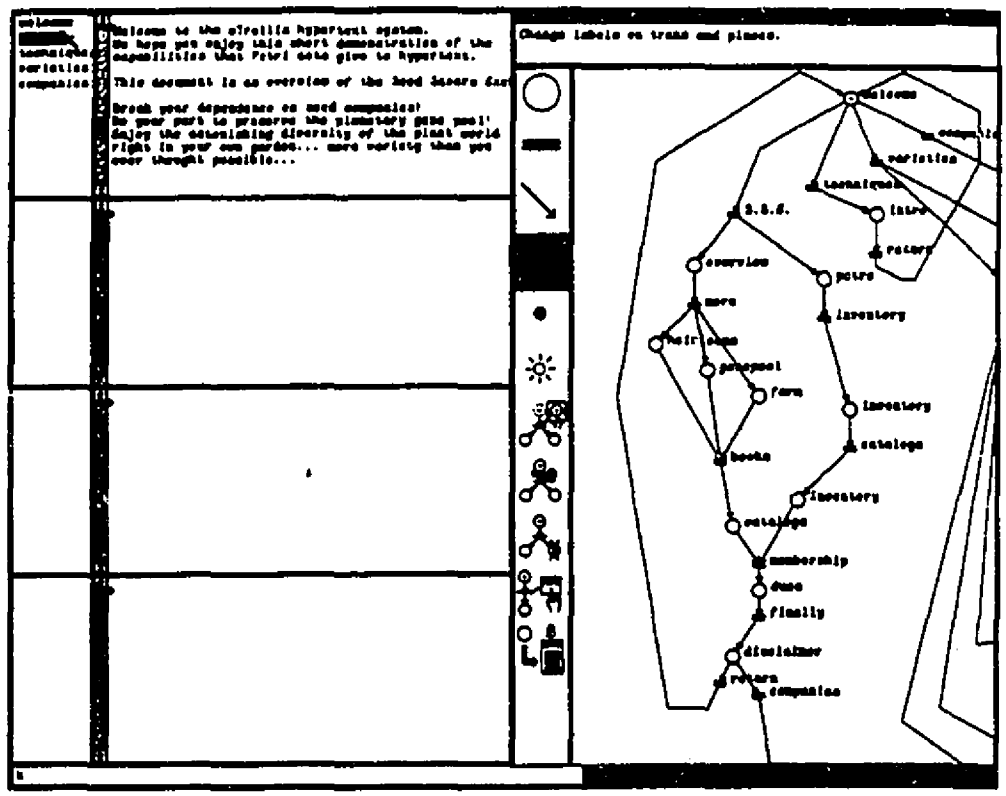


Figure 7.1 : A Sample Screen from the Trellis Hypertext System.

Note that there are hypertext systems that are based on mathematical foundations other than Petri nets. A sample system is hypergraph [Tom89]; Zheng and Pong propose statecharts as modelling formalism [ZP92].

Commonalities and differences between Trellis and Hypertec:

- ***Commonalities:***

Both Hypertec and Trellis are workstation-based systems, and they are both based on the Petri net formalism. They provide facilities for displaying, organizing, manipulating

and navigating through networks. Their way of building the network is quite similar: the use of a Petri net editor in Trellis, and the use of the "Macronet editor" in Hypertec/ Macrotec.

- ***Differences:***

The most important differences we can see between Hypertec and Trellis are:

- The browsing mechanism in Trellis is based on Petri nets: displaying the node component depends on enabling the transition, then users will select from the button panel to change the information that are displayed in the text windows. In Hypertec, however, the user has several ways to display nodes, for example, the user can double-click on a node in modelling network, select one of the nodes from the navigation subwindow of the InfoSheet, or select a node from the history dialog box;

- Trellis doesn't provide a history mechanism that allows users to inspect their previously visited nodes. However, this mechanism is supported in Hypertec;

- Trellis is limited to four windows to display a node's annotation. In Hypertec, users are provided with an unlimited number of InfoSheet windows, with the flexibility in using them (see discussion of reuse button feature in the subsection 4.3.3 *Information Window*);

- Trellis windows are not editable when they display the node's annotation. The Alphα authoring language (Author Language for Petri-net-based Hypertext, α version [FS89]) is used to associate text and graphics elements to the node annotation. In Hypertec, InfoSheet windows are editable and can import graphics in various formats such as Macintosh pictures, Sun rasterfiles, X11 icons, encapsulated postscript, and portable bitmap;

- Hypertec has two kinds of links, user-defined links and system links; furthermore, links are typed;

- In Hypertec, both places and actions, which correspond to the Petri net transitions found in Trellis, may contain information.

Chapter 8 Discussion

In this chapter, we shall first present our plans for future extensions of Hypertec. Then, we shall conclude this work.

8.1 Future Work

Not surprisingly, Hypertec may be extended in many directions. Below, we shall discuss some of the extensions envisioned. They were inspired by our study of similar systems (see previous chapter) and the experience gained from using the current Hypertec version. The extensions include the following:

- Similar to the approach taken by Stotts and Furuta in the Trellis system [SF89], we intend to support "annotated animation" of strict paths through the nodes. In parallel to graphical simulation, already supported in the current version of Macrotec, the system would automatically display the InfoSheets of the active model components (enabled actions together with input places). This way, the user will be able to easily access the active components' information and take action, whenever the simulation stops in step-wise execution or prompts for user input in conflicting situations (step-wise or continuous execution);

- We plan to enrich the InfoSheet facilities by supporting external links. Such support would allow users to link InfoSheets with external documents, for example, nodes of the World Wide Web [LNB+94], and to browse through the targets of these links, for example, pages of the World Wide Web. In the book purchase process discussed in section 6.1, for instance, one of the vendors' responsibilities is to supply book catalogs to the librarian. These catalogs could be made available on-line, provided that such a facility existed and the catalogs were available in electronic form. This can be done by adding another command to the "Special" menu of the annotation subwindow, allowing the user to create an anchor at any position in the document. This anchor should contain the URL (Uniform Resource Locator), the address of a specific node on the Web. By double clicking on the anchor, the underlying link will launch the XMosaic browser. For launching XMosaic, we can borrow some ideas from the work done in *Chimera* [ATW94] where the RPC (Remote Procedure Calls) mechanism is used to establish communication links in a heterogeneous environment;

- Also, we would like to build a component for generating reports about certain aspects of the models at hand, for example, tables of attribute values occurring in the models, list of all the model nodes, textual simulation output, etc. These reports will be based on the information spread over the model and the InfoSheets.

- Moreover, we would like to generate code from the model and take advantage of the object-oriented features of our generic model. This work has not been done yet for lack of resources. To achieve this work, we plan to carry over ideas from the Paradigm Plus tool which we have described in section 3.2 *Object Modelling Facilities* [Pro93].

- Finally, we plan to supplement Hypertec with search facilities. This will include providing a data dictionary, as well as a query mechanism in the base windows and throughout the InfoSheets.

8.2 Conclusion

The goal of this work was to complement business modelling and analysis of organizational architectures and processes with formal and informal documentation, and provide mechanisms for manipulating and viewing the model documents and creating navigable links between them. Hypertext fits the bill. Hypertext provides a clear conceptual model for the organization of complex structure, and a technology that supports viewing and navigating through structures [SS91]. Therefore, we have attempted in our work to adopt hypertext concepts found in various existing systems and tailor them to the domain of business modelling and analysis.

In order to support and validate our approach, we have developed the Hypertec tool which complements the Macrotec environment version 2.0. The Hypertec component was added in June 1994, making up for the Macrotec version 2.1. Hypertec is a hypertext-based component supporting authoring, display and navigation of the process documentation that cannot be captured in Macrotec. It contains state-of-the-art, general hypertext features and is tightly integrated with the rest of the Macrotec environment, both at the user interface and the database level. We have modelled several medium-sized examples, such as a bank loan process, the book purchase process, described in section 6.1, the paper discussion process presented in [KGT93], and others. Our experience with Macrotec/Hypertec indicates that our approach substantially facilitates business modelling, leading to comprehensive models and a more realistic modelling process.

Bibliography

[AMY88] Robert Akscyn, Donald McCracken, and Elise Yoder. KMS: A distributed hypermedia system for managing knowledge in organizations. *Communications of the ACM*, 31(7):820–835, July 1988.

[ATW94] Kenneth M. Anderson, Richard N. Taylor, and E. James Whitehead Jr. Chimera: Hypertext for heterogeneous software environments. In *Proceedings of the European Conference on Hypermedia Technology (ECHT'94)*, Edinburgh, United Kingdom, September 1994. ACM.

[BD91] Emily Berk and Joseph Devlin, editors. *Hypertext/Hypermedia Handbook*. McGraw Hill, New York, NY 10020, 1991.

[BDD+92] G. v. Bochmann, A. Debaque, R. Dssouli, A. Jaoua, R. Keller, N. Rico, and F. Saba. A method for architectural modelling and dynamic analysis of information systems and business processes. Technical Report CRIM-92/12/10, Centre de recherche informatique de Montréal (CRIM), Montreal, Canada, December 1992.

[Big88] James Bigelow. Hypertext and CASE. *IEEE Software*, 5(2):23–27, March 1988.

[BFG93] S. C. Bandinelli, A. Fuggetta, and C. Ghezzi. Software process model evolution in the SPADE environment. *IEEE Transactions on Software Engineering*, 19(12):1128–1144, December 1993.

[BLWW95] Roger W. H. Bons, Ronald M. Lee, Rene W. Wagenaar, and Clive D. Wrigley. Modelling inter-organizational trade procedures using documentary Petri nets. In *Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences*, Hawaii, January 1995.

[Boo94] Grady Booch. *Object Oriented Analysis and Design with Applications*. Benjamin/Cummings Publishing Company Inc., Redwood City, CA, 1994. Second edition.

[BOS91] Paul Butterworth, Allen Otis, and Jacob Stein. The GemStone object database management system. *Communications of the ACM*, 34(10):64–77, October 1991.

[Bus45] Vannevar Bush. As we may think. *The Atlantic Monthly*, pages 101–108, July 1945.

[CAB+94] Derek Coleman, Patrick Arnold, Stephanie Bodoff, Chris Dollin, Helena Gilchrist, Fiona Hayes, and Paul Jeremaes. *Object-Oriented Development: The Fusion Method*. Prentice-Hall, Inc., 1994.

[CB88] Jeff Conklin and Michael L. Begeman. gIBIS: A hypertext tool for exploratory policy discussion. *ACM Transactions on Office Information Systems*, 6(4):303–331, October 1988.

[CBK+92] Alan M. Carroll, Douglas P. Bogia, Simon M. Kacmar, William J. Tolone and Celsina Bignoli. Supporting collaborative software development with conversation builder. In *In Proc. of ACM SIGSOFT '92: fifth Symposum on Doftware Development Environments, Washington D. C.*, pages 11–20. ACM, December 1992.

[CG91] Karen S. Catlin and Nancy Garrett. Hypermedia templates: an author's tool. In *In Proceedings of Hypertext'91*, pages 147–160, San Antonio, Texas, December 1991.

[CKO92] Bill Curtis, Marc I. Kellner, and Jim Over. Process modeling. *Communications of the ACM*, 35(9):75–90, September 1992.

[CR92] Jacob L. Cybulski and Karl Reed. A hypertext based software engineering environment. *IEEE Software*, 9(2):62–68, March 1992.

[CY91] Peter Coad and Edward Yourdon. *Object-oriented Analysis*. Yourdon Press, Englewood Cliffs, New Jersey, 2nd edition, 1991.

[DFG+94] Anne-Claire Debaque, Paul Freedman, Jean-Michel Goutal, Rudolf K. Keller, Michel Levy, and Fayez Saba. The ECORP approach to Petri net tool evaluation. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, pages 814–820, Halifax, N.S., Canada, September 1994. IEEE.

[DG90] Wolfgang Deiters and Volker Gruhn. Managing software processes in the environment MELMAC. In Proc. of 4th Symp. on Softw. Dev. Environments, pp. 193-205, Irvine, CA, Dec. 1990.

[DS86] N. Delisle and M. Schwartz. Neptune: A hypertext system for CAD applications. In *Proceedings of the ACM SIGMOD'86*, pages 132–142, Washington, DC, May 1986.

[EE+68] D.C. Engelbart and W.K. English, "A Research Center for Augmenting Human Intellect," *AFIPS Conf. Proc.*, Vol. 33, Part 1, The Thompson Book Company, Washington, D.C., 1968.

[EH89] D. M. Edwards and L. Hardman. *Lost in HyperSpace: cognitive mapping and navigation in a hypertext environment. In Hypertext: theory into practice*, pages 105–125. Oxford: Intellect Limited, 1989.

[FHS+92] James C. Ferrans, David W. Hurst, Michael A. Sennett, Burton M. Covnot, Wenguang Ji, Peter Kajka, and Wei Ouyang. HyperWeb: A framework for hypermedia-based environments. In *Proceedings of ACM SIGSOFT '92: Fifth Symposium on Software Development Environ-*

ments, pages 1–10, Tyson's Corner, VA, December 1992. Appeared also as ACM Software Engineering Notes, vol. 17, no. 5, dec 92.

[FS89] Richard Furuta and P. David Stotts. Programmable browsing semantics in Trellis. In *Proceedings of the First ACM Conference on Hypertext*, pages 27–42, Pittsburgh, PA, November 1989.

[FS90] Richard Furuta and P David Stotts. *Separating hypertext content from structure in Trellis*, chapter 22, pages 206–213. Ablex Publishing Cor., Norwood, NJ 07648, 1990.

[GCF91] Martin Griss, Michael Creech, Dennis Freeze. Using hypertext in selection reusable software components. In *Hypertext 91*, pages 25–38. ACM, December 1991.

[GS90] Pankaj K. Garg and Walt Scacchi. A Hypertext System to Manage Software Life-Cycle Documents. *IEEE Software*, 7(3):90–98, May 1990.

[GS91] Pankaj K. Garg and Walt Scacchi. Ishys: Designing an Intelligent Software Hypertext System. *IEEE Expert*, 4(3):399–419, Fall 1991.

[Hal88] Frank G. Halasz. Reflections on NoteCards: Seven issues for the next generation of hypermedia systems. *Communications of the ACM*, 31(7):836–852, July 1988.

[IG90] W. Irler and G.Barbieri. Non-intrusive hypertext anchors and individual colour markings. In *echt90*, Versailles, France, November 1990.

[KA70] W. Kunz and R. Adams. Issues as elements of information systems. working paper no. 131, institute of urban and regional development. Univ. Of California, Berkeley., 1970.

[KGT93] Rudolf K. Keller, Anurag Garg, and Tao Tao. HyperRef - on-line support for research literature assessment and documentation. In *Proceedings of the Eleventh Annual International Conference on Systems Documentation*, pages 163–175, Waterloo, Ontario, Canada, October

1993. ACM.

[KGNT94] Rudolf K. Keller, Anurag Garg, Amin Noaman, and Tao Tao. Multi-level documentation of organizational architectures and processes. In *Proceedings of the Twelfth Annual International Conference on Systems Documentation*, pages 139–144, Banff, Alberta, Canada, October 1994. ACM.

[KL91] Charles J. Kacmar and John J. Leggett. PROXHY : A Process-Oriented Extensible Hypertext Architecture. *ACM Transactions on Information Systems*, 9(4):399–419, October 1991.

[KLO+93] Rudolf K. Keller, Richard Lajoie, Marianne Ozkan, Fayez Saba, Xijin Shen, and Tao Tao. Design of the Macrotec toolset version 2.0. Technical report, Centre de recherche informatique de Montréal (CRIM), December 1993.

[KOS94] Rudolf K. Keller, Marianne Ozkan, and Xijin Shen. Towards comprehensive support for the dynamic analysis of Petri net based models. In Robert Valette, editor, *Application and Theory of Petri Nets 1994 (Proc. of 15th Intl. Conf. on ATPN)*, pages 298–317, Zaragoza, Spain, June 1994. Springer-Verlag. LNCS 815.

[KSvB94] Rudolf K. Keller, Xijin Shen, and Gregor v. Bochmann. Macronet - a simple, yet expressive and flexible formalism for business modelling. In *Proceedings of the Workshop on Computer-Supported Cooperative Work, Petri Nets and Related Formalisms*, pages 51–55, Zaragoza, Spain, June 1994. Held in conjunction with the 15th Intl. Conf. on Application and Theory of Petri Nets.

[LNB+94] Air Luotonen, Henrik Frystyk Nielsen, Tim Berners-Lee, Robert Cailliau and Arthur Secret. The world-wide web. *Communication Of The ACM*, 37(8):76–82, August 1994.

[Mad91] Nazim H. Madhavji. The process cycle. *IEE/BCS Software Engineering Journal*, 6(5):234–242, September 1991.

[MCB84] Marco Ajmone Marsan, Gianni Conte, and Gianfranco Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, 2(2):93–122, May 1984.

[MDL+93] Elwin N. McKellar Jr., Ginger Dwyer, Thomas LaJeunesse, Jeffrey Limatta, and Diana Risdon. An interactive online process for developing and producing policy and procedure documentation. In *Proceedings of the Eleventh Annual International Conference on Systems Documentation*, pages 199–207, Waterloo, Ontario, Canada, October 1993. Association for Computing Machinery.

[MEN92] Alberto O. Mendelzon, Frank Ch. Eigler, and Emmanuel G. Noik. GXF: A graph exchange format. Technical report, University of Toronto, Toronto, July 1992.

[MMWFF92] Raul Medina-Mora, Terry Winograd, Rodrigo Flores, and Fernando Flores. The action workflow approach to workflow management technology. In *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'92)*, pages 281–288, Toronto, Canada, October 1992. ACM.

[MSOP86] David Maier, Jacob Stein, Allen Otis, and Alan Purdy. Development of an Object-Oriented DBMS. In *Proceedings of the Conference on Object-Oriented Programming: Systems, Languages and Applications*, pages 472-482, Portland, OR, 1986.

[Nel80] T.H. Nelson, "Replacing the Printed Word: A Complete Literary System," *IFIP Proc.*, October 1980, PP. 1013-1023.

[Nie90] Jakob Nielsen. *Hypertext and Hypermedia.* Academic Press, Inc., San Diego, California, 1990.

[Pro93] ProtoSoft Inc. *Paradigm Plus*. ProtoSoft Inc. 17629 El Camino Real# 202 Houston, Tx 77058, USA. TEL:(713) 480-3233, FAX: (713) 480-6606, release 2.0 edition, 1993.

[PS93] Lawrence Peters and Ron Schultz. The application of Petri-nets in object-oriented enterprise simulations. In *Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences*, Hawaii, January 1993.

[RBP+91] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. *Object-oriented Modeling and Design*. Prentice-Hall, Inc., 1991.

[REF94] Proceedings of the Fourth Reengineering Forum, 2 volumes, Victoria, B.C., Canada, September 1994. Reverse Engineering Forum Inc.

[Sch86] Kurt J. Schmucker. *Object-Oriented Programming for the Macintosh*. Hayden Book Company, Hasbrouck Heights, NJ, 1986.

[Sey91] Philip Seyer. *Understanding Hypertext Concepts and Applications*. Windcrest Books, Blue Ridge Summit, PA 17294-0850, 1991.

[Shn87] B. Shneiderman User interface design for the hyperties electronic encyclopedia. pages 189–194. ACM Hypertext'87, November 1987.

[SF89] P. David Stotts and Richard Furuta. Petri-net-based hypertext: Document structure with browsing semantics. *ACM Transactions on Information Systems*, 7(1):3–29, January 1989.

[SM88] Sally Shlaer and Stephen J. Mellor. *Object-Oriented System Analysis: Modeling the World in data*. Prentice Hall Inc., Englewood Cliffs, NJ., 1988.

[SS91] John B. Smith and F. Donelson Smith. Abc: A hypermedia system for artifact-based collaboration. In *In proceedings of Hypertext'91, San Antonio, Texas.*, December 1991.

[SZ87] Karen E. Smith and Stanley B. Zdonik. Intermedia: A case study of the differences between relational and object-oriented database systems. In *Proceedings of the 1987 Conference on Object-Oriented Programming Systems, Languages and Applications*, pages 452–464, 1987.

[Tao93] Tao Tao. Applying hypertext concepts to business modelling. Master's thesis, McGill University, Montreal, PQ, Canada, July 1993.

[TGH94] Aphrodite Tsalgatidou, Dimitris Gouscos, and Constantin Halatsis. Dynamic process modelling through multi-level RBNs. In *Proceedings of the Fourth International Working Conference on Dynamic Modelling of Information Systems*, pages 327–341, Noordwijkerhout, The Netherlands, September 1994.

[Tom89] F. W. Tompa. A data model for flexible hypertext database system. *ACM Transactions on Information System*, 7(1):85–10, January 1989.

[Wal88] J.H. Walker. Supporting document development with concordia. *IEEE Computer*, 21(1):48–59, January 1988.

[WGM89] André Weinand, Erich Gamma, and Rudolf Marty. Design and implementation of ET++, a seamless object-oriented application framework. *Structured Programming*, 10(2):63–87, April-June 1989.

[WW93] Yair Wand and Carson C. Woo. Object-oriented analysis - is it really that simple? In *Proceedings of the Workshop on Information Technologies and Systems (WITS'93)*, Orlando, Florida, December 1993. IEEE.

[ZP92] Yi Zheng and Man-Chi Pong. Using statecharts to model hypertext. In M. Nanard P. Paolini D. Lucarella, J. Nanard, editor, *ACM ECHT '92 Conference*, pages 242–250. SIGLINK, SIGIR, SIGOIS, ACM, November 1992.