Tools for Stored Interactive Multimedia

Ole Vedel Villumsen

Thesis submitted for the PhD degree

July 1996

Acknowledgments

Thanks are due to my advisor Jørgen Lindskov Knudsen, Computer Science Department at Aarhus University, for being available from the formulation of my project until several months after it should have been finished, and for many constructive comments, especially in the form of excellent and indispensable methodological advice. Jørgen has been very good at asking the right questions and leaving it to myself to provide the answers.

Thanks to Peter Bøgh Andersen, Bjørn Laursen, Søren Kolstrup and everyone in the Jean de l'Ours, Wodan's Eye, and 'The Transparent Computer' projects for the inspiration that eventually lead me into and through my PhD study. More thanks to Peter Bøgh Andersen for many useful comments. Thanks to Edvin Kau for assisting my literature search in narrative theory related to non-textual media and for useful discussions.

I am indebted to David Madigan for arranging my invitation to spend six months at University of Washington and Fred Hutchinson Cancer Research Center in Seattle, USA, and for valuable co-operation and interaction while I was there. Not least because of David Madigan and his wife Áine, they also turned out to be very pleasant six months. My coauthors and I are grateful to Jeff Bradshaw, Peter Dunbar, and Robert Jacobsen for helpful contributions to chapter 10 on Talaria.

Many thanks to Helen Gray for proof-reading and for helping me translate the sample output from the Petri net 'Kristendom' (Christianity) into English. Thanks to Karen Kjær Møller for proof-reading. Thanks to Tim Caudery and his colleagues at Institute of English, Aarhus University, for excellent help with translating the extended layer model from Danish.

Thanks to Soren 'Petri' Christensen, Computer Science Department at Aarhus

University, for standing by while I learned to use Petri nets and the Design/CPN tool, and for valuable discussions and criticism. Thanks to Torben Bisgaard Haagh for assistance on ML. Thanks to Patrick Sénac for a useful, kind and encouraging comment on the section on HTSPN.

I used to find it a bit ridiculous when authors in their acknowledgements thank their parents, wife, husband, children and pets. I have had to change my mind a bit. Since a PhD study is not compatible with a proper, well-ordered family life, I would like to thank my wife Annette Schachner for bearing with me during the last three and a half years, including my six months' absence when I was in Seattle. I promise to take on more of the cooking and other housework from now on or at least tidy up after myself, and to join Annette now and then for a horse ride in the woods too. Thanks also to my father, Povl Vedel Villumsen, for comments and encouragement.

The Ph.D. study has been conducted within a scholarship ('datalogistipendium' or 'computer science scholarship') from the faculty of natural sciences at Aarhus University for most of the time. The study was finished in a leave-of-absence for education from Magistrenes A-kasse (masters' unemployment fund). The work on Hejmdal was initiated while I was employed in a position financed by the Danish Research Programme for Informatics, grant number 5.26.18.19. The research on Talaria is funded in part by a SBIR (Small Business Innovation Research) grant from National Institute of Health, USA, to Statistical Sciences Inc. and NCI grant CA 38552. The Computer Science Department and the DEVISE project at Aarhus University have placed office, computers and computer software at my disposal. Forskerakademiet (Danish Research Academy) supported my travel to and stay in Seattle. A considerable tax reduction from Danish authorities also supported the stay.

What kind of language is that?

I attempt throughout the thesis to write a British English for an international audience. I deliberately keep French accents, Danish letters æ, ø and å, etc. in the text where appropriate. Though personally I like a personal style (writing 'I' when I mean I), I have tried to avoid it in the thesis.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background: interactive multimedia

The field of computer-based multimedia seems to be emerging from at least two end-points: On the one hand, ordinary computer applications include more and more elements of graphics, sound and animation, just as they have been including more and more graphics over the last decade. Live video is becoming widely available on computers and can be expected to be included in all kinds of computer applications. On the other hand, computer-based multimedia presentations which have their closest relatives in the media worlds are appearing.

As an example of the latter, more and more museums use computers to communicate information to visitors. These computers seem to come as a supplement to the slide show with an audio tape. One reason is probably the possibilities of interaction, which can make the presentations more interesting. Often, a kind of menu is used to let the user go to different parts of the presentation, but new kinds of interaction are also evolving. The new kinks of interaction appearing in this field are often used to simulate the user exploring a world. Depending on the kind of museum, it could be the world of Pablo Picasso or the world of the bronze age.

As an example of ordinary computer applications including new media, many hypermedia systems now include video with sound besides text, graphics and

sometimes animations (hypermedia will be discussed more closely shortly).

Stored computer-based multimedia is rapidly spreading. They are used for communication in many different areas, including museums, education, advertising and entertainment. Stored multimedia, though interactive, is most often used in a one-way communication from a group of authors or developers to an audience of users. (By contrast, live multimedia is more often used in two-way communication, e.g., computer conferencing with video.) One reason for this situation is the relatively high cost of producing stored multimedia presentations; only if there are a number of potential readers is the production of multimedia worthwhile.

Interaction is important in computer-based multimedia. The above examples show that interaction with multimedia is useful. While non-interactive multimedia is not much different from traditional films or slide shows with audio tapes, interactive multimedia is a whole new world. Interactive multimedia constitutes a way for an author to convey new experiences to the user; experiences that have neither been possible with traditional media (film, animation, etc.), nor with traditional interactive computer programs. New kinds of interaction in multimedia are evolving and will probably continue to evolve in the years to come.

Computer-based multimedia is a new world arising between the world of computing and the worlds of different media. Interactive, computer-based multimedia is expected to be used in more and more fields in the future. Research in computer-based multimedia is ongoing in many different directions, both within the uses of computer-based multimedia and within the software and hardware used.

1.2 Problems and contributions

On this background, the demand for advanced tools for working with interactive multimedia is increasing. This thesis explores tools for development of stored, interactive multimedia. The thesis first makes some general observations about requirements for such tools. Since a common complaint among multimedia authors is that available tools always require some scripting or programming, the thesis goes on to investigate the need for scripting

or programming in multimedia development. The rest of the thesis develops tools and techniques for specific purposes and for specific developers within the area of multimedia. The largest part of the thesis is concerned with tools for building a kind of story known as *elastic stories* in multimedia. Other parts present tools for semi-automatic generation and maintenance of links in hypermedia documents, and an object-oriented multimedia tool for programmers.

Many observations were made about requirements for multimedia tools. These will be reported. Many of the requirements correspond to requirements for system development tools in general. In addition, it was found that the multimedia development team must have tools for digitizing, creating and editing material in each medium including time-based media. The tools should be separate in the sense that one can work with one of them at a time, independently of the others, still integrated in the sense that thev can work on the same materials, and in the sense that they have a similar user interface where appropriate.

It has been an interesting question how far one can expect to help multimedia authors many authors by development of new and better tools. Specifically, since dislike programming or cannot program, or both, it has been found worthwhile to investigate questions such as: Is it a necessity that multimedia tools always require some scripting or programming, at least as soon as the developer wants to go just a little bit beyond the core of the tool's intention and metaphor? If so, how much scripting is necessary, or how far can we limit the requirement that the author has to program? It has been found that programming has its place in most including the most interesting multimedia development projects; but it is still worth striving for tools that are more powerful and easier to use than the tools available on the market today.

The main new feature of multimedia is the introduction of time-based media in computers: media that can only be meaningfully recorded and played back over time, such as sound, animation and video. There is a challenge in developing techniques and tools for dealing with time-based media in multimedia development. On this background, the thesis presents Hejmdal, an object-oriented class library for interactive editing and play-back of QuickTime[1] movies, a de-facto standard architecture for time-based documents.

[1]QuickTime is a trademark of Apple Computer Inc.

QuickTime movies can consist of graphics, photographs, live video and animations and sounds. The interactive playback facilities offered by Hejmdal are start and stop playing, random positioning within a movie, and stepping a single frame forward and backward. Movie segments can be interactively cut, copied and pasted. Movies are stored in digital files, thus avoiding the need for additional hardware during playback and editing. It turns out that the object-oriented model in Hejmdal is simple, clear, powerful and flexible. Especially concerning interaction, use of an object-oriented model is advantageous. Hejmdal was originally developed for use in the remainder of the thesis work, where new and more powerful tools were planned on top of Hejmdal.

Hejmdal is built on the Macintosh[2] extension QuickTime. QuickTime movies cannot include definition of interaction. The thesis suggests that a standard document architecture should be developed which allows interaction to be defined in the multimedia documents, not only in the application programs.

The systems known as *elastic* systems were found to be a particularly interesting class of multimedia systems. Elastic systems form a middle ground between user-controlled and developer-controlled systems, the two paradigms traditionally used in multimedia and other computer systems. The use of elastic systems for telling *elastic stories* is explored. An elastic story is an interactive story in which the reader can try to influence the course of events, without any guarantee that he or she will succeed every time. An elastic story gives both the author and the user some control, but gives none of them the unconstrained power over the course of events. It is believed that elastic systems have a great potential, since the radically new in computer-based multimedia is rather with these than with traditional user-controlled and author-controlled systems. Elastic stories constitute a new use of multimedia. Current multimedia tools do not support the construction of elastic stories very well, which is the rationale behind exploring tools specifically for this purpose.[3] The thesis shows that Petri nets are well suited for formal description of elastic stories. Purposes of describing an elastic story as a

[2]Macintosh is a registered trademark of Apple Computer Inc.

[3]Glorianna Davenport writes: 'What fascinated me over the years is the complementarity which binds the generation of content and the design of tools. In fact we cannot talk about form without discussing content and the tools for accessing that content.' Glorianna Davenport: Bridging Across Content and Tools. Computer Graphics, newsletter of ACM SIGGRAPH, Volume 28, number 1, February 1994, pages 31–32.

Petri net may be:

1. To give a precise, formal specification of the story.
2. To implement the story in a computer system.

Some advantages of using Petri nets for elastic stories are:

1. Petri nets have formal, precise semantics.
2. Petri nets model elastic stories in a straightforward way, thus building elastic stories in Petri nets is relatively easy. While Petri net construction may be considered programming, using Petri nets specifically for elastic stories is considered easier than other forms of implementation.

The vision carrying the work presented here is to be able to fulfil both of the above purposes with one computerized tool. A current obstacle to implementation using Petri nets is the lack of Petri net tools with multimedia capabilities. Work is going on to remove that obstacle, some of it building on Hejmdal.

For the sake of the study, only one style of multimedia interface is under consideration in the work on elastic stories. This multimedia interface has a big, scrollable background picture with moveable objects on it, with additional windows for pictures and video, and it includes sound.

Attention will be given to the question: is the Petri net formalism easy enough to use so that multimedia authors with no background in programming, Petri nets or similar formal specifications can learn to use them for building interactive stories? If this is found not to be the case, other options exist:

1. Petri nets may be used in an informal way in a multimedia project and a Petri net programmer be hired to formalize and refine them into nets that work as the formal descriptions they are intended to be.
2. A syntactic layer may be defined on top of the Petri nets that is easier to use and specifically targeted towards describing elastic stories. In this case, the Petri net constructs given in this thesis can be used to give a precise semantics for the new syntactic layer.

Both options will relieve the author of programming and yet retain the advantages of Petri nets given above.

Finally, the thesis includes a chapter on hypermedia. Hypermedia is often advantageously used in conjunction with multimedia. In practice, it can be difficult to separate the two concepts at all (a theoretical separation of them follows in the next section). One problem with large hypermedia documents is that creation and maintenance of links is difficult and timeconsuming. Motivated by an application to an American federal clinical practice guideline for cancer pain management, the mentioned chapter develops a scheme for automatic linking based on repertory grids.

To evaluate the scheme, a protocol analysis is conducted. Six users of the guideline addressing typical cancer pain management tasks made 25 different links. The repertory grid using a neighbourhood size of 17 captures 20 of these links. With optimization, it captures 23 of the links within a neighbourhood size of 13.

1.3 Some concepts: multimedia, interactive multimedia and hypermedia

Computer-based multimedia is combinations of more than one medium on a computer. Each medium may be text, graphics, photographs, animations, videos or sound. Multimedia can be divided into stored multimedia presentations and multimedia data that are transmitted with no intermediate storing. When the developer and the user or viewer of multimedia are temporally separated, we talk about *stored* multimedia. The opposite could be called live media. The focus of this thesis is on the stored multimedia and the word 'multimedia' is often taken to mean stored multimedia.

Some multimedia systems, such as virtual reality systems, flight simulators and certain computer games, generate images and other material 'on the fly', while the program is running. They are still considered stored multimedia, as long as the author and user are temporally separated.

Often people involved in multimedia development view it as an authoring process. They may see themselves as *multimedia authors* rather than mul-

timedia developers. The two terms are used interchangeably in the thesis. The people who are not in a project as programmers or computer specialists are sometimes referred to as *content persons*. In the thesis, *user* means the receiver in the communication, even though it can be argued that an author is a kind of multimedia user too.

As already mentioned, computer-based multimedia includes one or more new media, i.e. video, sound and animation, besides the traditional computer-based media of text and graphics. An important characteristic of the new media is that they are *time-based*, which the traditional media are not. They are sometimes referred to as *dynamic*, as opposed to traditional *static* (non-time-based) media. Time-based data are also called *temporal*, and their media are known as temporal media. Seen from the view of a multimedia programmer, the distinction between time-based and non-time-based data is a major distinction. Time-based data are those that can only be recorded and played back over time: video, animation and sound. Non-time-based data are stationary in time, like graphics and text. 'Classic' electronic data (numbers, text and graphics) are not time-based. That is, one of the new things in multimedia is the handling of time-based data, and techniques for doing this are being developed. (Strictly speaking, the classic beep is time-based. It has traditionally been conveniently handled as non-time-based data, e.g. like a character in the character set.)

Another distinction is between analogue and digital multimedia data. Most often, a multimedia presentation consists solely of digital data. Multimedia authors often use analogue material, but digitize it for use with the computer. However, analogue data have been used in multi-media presentations, too, e.g. as a video tape or a video disc, controlled from the computer.

A good reason for using analogue data can be the storage space that digital video and audio data occupy, or more precisely, the lower cost of analogue storage media. As better compression schemes for digital video and audio become commercially available, the use of digital data becomes even more widespread, since these are more easily integrated in a computer system.

1.3.1 Multimedia is orthogonal to hypermedia

Hypermedia is the generalization of hypertext to other media than text. This means that hypertext is an example of hypermedia. Ted Nelson defines hypertext in turn as 'a combination of natural language text with the computer's capacity for interactive branching, or dynamic display ... of a nonlinear text ... which cannot be printed conveniently on a conventional page'[4]. The Dexter hypertext reference model[5] is a widely accepted attempt to capture the essence of existing and future hypertext systems. Akscyn, McCracken and Yoder[6] state that most hypermedia systems can be characterized by the following features:

- The material (text or other media, such as images, sound and animation) is 'chunked' into small units or *nodes*.
- Nodes are displayed one per window.
- Nodes are interconnected by *links*. Users navigate in a hypermedia database by traversing links.
- Users can create structures by creating, editing and linking nodes.

(This characterization may not cover hypermedia in general; but it does cover the concept as it is used in this thesis.) Links, or *hyperlinks*, connect nodes that have a semantic connection: if one node triggers an association with another, then a user links them and potentially all users can get from one node to the other. (Each link may be private or shared.) It is said that the user traverses the link from the *source node* to one or more *destination nodes*. A link can have two or more endpoints. Each endpoint of a link may serve as source or destination or both. The location in a hypermedia document

[4]From Theodor H . Nelson: Getting It Out of Our System. In G. Schechter (editor): Information Retrieval: A Critical Review. Thompson Books 1967. Here quoted from Jeff Conklin: Hypertext: An Introduction and Survey, *IEEE Computer*, vol. 20 no. 9, pages 17-41, September 1987.

[5]F Halasz & M Schwartz: The Dexter Hypertext Reference Model. In Communications of ACM, vol. 37 no: 2, pages 30-39. 1994.

[6]Robert M Akscyn, Donald L. McCracken and Elise A. Yoder: KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations. In Communications of the ACM, vol. 31 No. 7, July 1988.

where a link can start or end is known as an *anchor*. Each anchor may serve as endpoint for zero or more links.

Nielsen[7] and Balasubramaniar[8] give brief histories of hypertext and descriptions of the best known applications.

Some hypermedia documents distinguish between two kinds of users; authors and readers. The author or authors create the document, while readers can read node contents and (most often) create and edit their own links. In some cases they can also add their own annotations. In other hypermedia documents, all users can create and edit nodes on an equal footing. Mixtures of the two approaches exist too.

Hypermedia is interactive by its nature, since user input steers the navigation. In addition, nodes and links may be interactively edited. On the other hand, interactive multimedia may include other kinds of interaction than editing and link traversal.

Multimedia and hypermedia are orthogonal; 'multimedia' refers to the content belonging to different media; 'hypermedia' refers primarily to a chunking and navigation principle. The two are often advantageously used in combination. On the one hand, most multimedia presentations until now can be conveniently described as hypermedia. In particular, *all* the four primary navigational structures used in multimedia according to Tay Vaughan[9] presuppose a hypermedia perspective on the multimedia.

On the other hand, the term 'hypermedia' presupposes the existence of other media than text. Thinking in terms of hypermedia rather than hypertext, the need for mixing the media arises naturally in many applications. On the World Wide Web (WWW), maybe the best known example of hypermedia, images are often embedded in the text. Many web pages offer downloading of sound files, QuickTime movies or other files containing video and sound. These cannot be played back in real time over the net, only because net

[7]J. Nielsen *Hypertext and Hypermedia.* New York: Academic Press, 1990.

[8]V Balasubramanian *Hypermedia issues and Applications, A State-of-the-Art Review,* Independent Research Report as part of Ph.D. Program, Graduate School of Management, Rutgers University, December 1993.

[9]Tay Vaughan. Multimedia: Making It Work. Second edition. Macromedia/Osbome McGraw-Hill 1994. Pages 390-91. The four primary navigational structures are called linear, hierarchical, non-linear (neither linear nor hierarchical) and composite (a mix of the three).

capacity is insufficient.

1.4 Guide to the thesis

Chapter 2 discusses requirements for tools for developing stored interactive multimedia programs. The following requirements are identified: Editors are needed for each medium used in the multimedia. Editors should allow the importing and digitizing of analogue material, as well as on-line creation and editing of digital material. The editing tools should be separate (allowing use of one at a time), yet integrated with each other (sharing the same material) and with the programming environment. Conventional database technology seems insufficient for building a well-structured media database. The need for interpretative execution of the program during development is found to be even greater in multimedia than in other fields. At the same time it is advantageous *also* to have a compiler.

Furthermore, the following requirements are found to be little or no different from the requirements for programmer's tools in other fields: strong typing, integration of code and media data, and facilities for structuring of code and data.

Chapter 3 discusses the role of programmers in development of computer-based multimedia. It argues that scripting or programming is necessary in most multimedia development projects, and that the multimedia developer who wants to exploit the computer's potential and his or her own creativity should learn programming.

Chapter 4 presents *Hejmdal*, an object-oriented platform for working with interactive multimedia. Hejmdal supports interactive playback and editing of multimedia. The chapter discusses the benefits of using an object-oriented platform and the requirements for a platform for working with interactive multimedia, and future work in the field. It argues that the object-oriented model is simple, clear and powerful. It suggests that a platform for working with multimedia should support various kinds of interaction with multimedia.

Chapters 5 through 9 present work on the use of Petri nets for telling a kind of stories known as elastic stories.

Chapter 5 is devoted to narratology, the theory of narration. The purpose is to establish an understanding of what a narrative (story) is, which in turn will be used to develop a model of a narrative of sufficient quality for use in the following chapters. In the context of story construction it is highly relevant to look at concepts and tools developed to *analyse* narratives. It is assumed that the same concepts and tools are useful in *synthesis and construction* of stories, and thus can serve as a good inspiration for computerized tool support. Chapter 5 gives a brief account of a common theory of narration: New Criticism, with emphasis on the Extended Layer Model.

Chapter 6 explains what an elastic story is. It describes the user interface employed and elicits from the theory of narration a set of concepts that is considered sufficient for covering elastic stories in the described interface as they are known today. That set of concepts constitutes the model of an elastic story on which the following chapters are based.

Chapter 7 demonstrates how the elicited concepts are modelled in a straightforward and convenient way using Coloured Petri Nets.

Chapter 8 describes experiments implementing three small elastic stories in Coloured Petri Nets, thereby using all the concepts in the model.

Chapter 9 contrasts this work with related work in interactive story telling and in Petri nets.

Chapter 10 presents Talaria, a multimedia reference tool on cancer pain management for health care providers. The chapter develops a linking scheme based on repertory grids. Harnessing knowledge acquisition techniques established in the field called artificial intelligence, the repertory grid assigns each node a location in 'context space'. A node links to another node if they are both close in context space. Chapter 10 also presents an evaluation of the linking scheme. The final chapter 11 summarizes and discusses the results and elaborates on the connections between them.

1.4.1 How to read footnotes

Some passages of the work contain many footnotes. To avoid interrupting the reading, many readers can benefit from *ignoring* the footnotes. Only the reader who seeks more depth, explanation or evidence, for instance in the

form of references, should read a given footnote.

When several language versions of the same work exist, the footnote typically only gives one of them. Sometimes, other language versions can be found in the bibliography in the back of the thesis.

Chapter 2

Requirements for Tools

2.1 Introduction

As discussed in the thesis introduction, the demand for advanced tools for working with multimedia is increasing. This chapter embarks on the discussion of requirements for such tools. It turns out that in some, but not all, cases the requirements are not very different from requirements for programmer's tools in development of conventional computer systems.

The chapter takes on the perspective that multimedia development (or multimedia authoring or production) is a kind of computer system development. This is only one of a number of valid perspectives on multimedia development. One obvious different perspective would view multimedia development as an authoring activity, parallel to book writing, moving picture production, etc.[1]

Most of the observations contained in this chapter were made during the work on Talaria, which is reported in chapter 10.

Section 2.2 describes an important characteristic of many multimedia programs, namely the use of metaphors. Section 2.3 lists the different roles in a

[1]Erling Maartmann-Moe writes in 'Multimedia' (Universitetsforlaget, Oslo 1991. (In Norwegian)) that multimedia is developed in the intersection between broadcasting, publishing, computing and telecommunication. Each of these four areas can probably provide perspective(s) on multimedia development.

multimedia development team and derives from them some requirements for tools. Section 2.4 presents a number of further observations of requirements. Section 2.5 presents characteristics of media data in multimedia and derived tool requirements. Section 2.6 summarizes the observations.

2.2 Metaphors in multimedia and in the tools

Metaphors are used in multimedia, maybe more extensively than in other computer programs. As in these, metaphors in multimedia seem to help the user by allowing him or her to transfer experience from some familiar domain to the new one, the multimedia. As in other domains, metaphors have their serious limitations though. Firstly, they break down quickly; multimedia systems do include features that are in no way related to their metaphor. Secondly, multimedia systems built entirely on metaphors of familiar phenomena can only convey experiences of those familiar phenomena. Thus to be truly innovative, multimedia will have to go beyond the metaphors.

A common metaphor in multimedia is that if a book or magazine. In hypermedia, a travel metaphor is often used. An example is Talaria, where a travel metaphor structures the navigation tools and provides the user with an intuitive context mechanism. Each node represents a place to visit. The user can travel alone or take guided tours. Section 10.2 pages 176–177 explains the use of the travel metaphor in Talaria.

The tools used for building multimedia often have their own metaphors, which are sometimes visible in the presentations (programs) produced with those tools.[2] Again, such metaphors can allow inexperienced developers to use the tool within the limits of the metaphor.

[2]The best known tool metaphors may be the movie metaphor (QuickTime), the card index metaphor (HyperCard among other programs) and the animation metaphor (Macromedia Director).

2.3 Skill requirements

Multimedia development is often carried out in a cross-disciplinary co-operation, since many different skills are needed. A development team may include:[3]

- an artist or different kinds of artists;
- for non-fiction (and conceivably for fiction), a subject matter expert;
- for educational multimedia, a teacher or other person with didactic knowledge;
- a programmer and/or computer specialist;
- and an end-user representative and/or person with an understanding of users' background, qualifications and expectations.

2.3.1 Different(-ly tailored) environments for different participants and tasks

As such diverse skills are involved in a multimedia development project, hardly any one development platform will serve all participants unless it is tailored to each participant's needs. Furthermore, often during development the developer focuses on a narrower part, e.g. only one medium, and does not want to deal with functionality not related to the part in focus. The developer may for instance spend a day or more doing video capture, in which case a good video capture tool is essential, and really nothing else. He or she may want to experiment with video size and resolution (number of pixels in each dimension), frame rate, different compression schemes and different tools for doing the job.

Therefore, rather than trying to integrate all relevant functionality into one multimedia development program, it is advantageous to provide the developers with a variety of relatively independent programs for the different tasks involved in the development: independent in the sense that the developer

[3]See for example sections 2 and 3 in Rob Philips: Producing Interactive Multimedia Computer-Based Learning Projects. Computer Graphics, newsletter of ACM SIGGRAPH, Volume 28, number 1, February 1994, pages 20-24.

can use, for instance, the video digitizing program independently without bothering about the existence of the video editing program, let alone all the other tools needed in a project.

Integration between the tools is important: first, it is crucial that the tools can operate on the same media formats (or at least that conversions exist); second, a similar interface for the tools can ease the use of the different tools at different times during development. Note that integration in this sense does not conflict with the independence of tools as described in the previous paragraph.

2.4 Requirements for tools

Experience with different multimedia development tools[4] reveals that the following properties of such tools are desirable:

1. Immediate interpretation

   It has been found very useful to have the opportunity at any time during development to 'press a button' and see the program run. In some phases this is used very often, so a need to compile and link the program before execution would be a hindrance to development. Is this any different from other program development? Yes; the need is greater in multimedia development than in other program development. In multimedia development, often the interface look is developed in a more experimental fashion. This may include experimental development of the content. In traditional program development, content is part of the data, not the program itself, and is therefore not provided by the developers. A screen layout can be evaluated to some extent without running the program. The distinction between program and content is seldom used in stored multimedia; here the development team provides both the media data and the code. In case of an animated interface, you have to see it run to evaluate it.

2. Speed and efficiency

   Development efficiency dictates the need for interpretation as described

[4]Mainly Macromedia Director, HyperCard and SuperCard.

above. At the same time, some multimedia systems are CPU intensive, e.g. in digital video playback or in having multiple elements continuously responding to mouse movements. Such systems will take advantage of a good compiler allowing them to run more smoothly. Ideally, both interpretation and compilation should be available. In cases where the execution speed is essential to the experience, the developer will of course have to compile the program to get the right impression of how it will run.

3. Strong typing

   Cases are made for and against strong typing in experimental development. Multimedia development, experimental as it often is, is no different. In the work on Talaria, it has been found that with typeless, undeclared variables it is very easy to make mistakes that a type checking facility could very easily have found. Unfortunately, most scripting languages are typeless.

4. Flexibility

   While a tool metaphor is helpful for some time, developers very often find themselves wanting to do things that do not fit within the metaphor. One example is the developer building a metaphor in the multimedia system that is different from the tool's metaphor. Tools are thus needed that allow programmers to go beyond the tool's primary intention. Common ways to do this are (1) adding scripts (2) accessing code written in a different programming language, e.g., C. While a script language epically offers good support for the same metaphor as the tool as a whole, it may at the same time be general enough to allow the determined programmer to obtain what he or she wants. For a script language to give the full flexibility it would have to be a full programming language. At the same time, all the media used in the multimedia program must be accessible from the script language. The next chapter discusses programming in multimedia development and the need for it.

5. Integration of code and media data

   In the work on Talaria, it was found very convenient to have elements of the presentation and the code guiding their behaviour (e.g., their reaction to mouse clicks) together. For instance, HyperCard allows code

(scripts) to be attached to PICTs, cards, buttons, texts and QuickTime movies. An example of a poor design is Macromedia Director. The script language Lingo is object based, that is, objects integrate code and data. The important limitation is that the media data with which Director works (primarily cast members) have to be stored separately from any objects and therefore are not integrated with the code. While at least in the object-oriented world there is nothing new in integrating code and data, having the data be all kinds of media is new. The next section returns to the treatment of media data.

6. Structuring facilities

   The basic need for structure in code and data is no different in multimedia programs and other programs. Some multimedia programs contain hundreds or thousands of images, sounds or video clips. However, tools for developing multimedia presentations often lack structuring facilities. While relational databases solve many data structuring problems in traditional programming, they are not suited for multimedia data. A field in a relation (table) cannot contain a picture or a movie segment. Techniques for searching multimedia databases are only beginning to be developed. Also multimedia data often needs specialized storage formats optimized for fast playback, which traditional relational database management systems do not offer. Object-oriented databases look more promising than relational databases.

2.5 Media data in multimedia

The media data make up a substantial majority of the data in multimedia programs. The amount of media data in a stored multimedia program can be large, not only measured in megabytes, but also perceived as large by the user.[5] Usually, only a few data are not media (e.g., counters and screen coordinates).

As mentioned in the introduction, programming tools for multimedia must be able to handle time-based data. Apple's QuickTime is a good tool of today

[5] "A picture is worth a thousand words" (Chinese saying).

for handling time-based data from a programming language. QuickTime is discussed more closely in chapter 4.

Typically in stored multimedia presentations, all the media data are prepared in their final form beforehand. While often the order of the presentation and sometimes also the positions or motions of certain elements are decided interactively at runtime, the basic content seldom is. In traditional programming terms, the data consist largely of constants, variables being used much less.

Therefore, the multimedia development team will need tools to create and edit these 'constants'; e.g. text editors, draw and paint programs, a scanner and/or a digital camera, video digitizers and video editing programs, sound recording and editing programs and animation programs. Furthermore these tools will have to be integrated with the programming tools (if any), so that the media can be edited after they have been integrated with the code as described above.[6]

The above is not to say that the media in a multimedia program cannot be variable. The use of more variables will probably contribute to innovations in multimedia in the future. As mentioned in the introduction, virtual reality and flight simulators are examples in which the media content is largely generated at runtime.[7]

2.6 Summary of tool requirements

In this chapter the following requirements for tools for multimedia developers have been identified: interpretation and compilation; strong typing; integration of code and media; facilities for structuring code as well as media, and a media database; flexibility, that is, ways to go beyond the tools' primary intention and metaphor, e.g. using scripting; and tools for digitizing, creating and editing material in each medium including time-based media. The tools should be separate in the sense that one can work with one of them while ignoring the others, still integrated in the sense that they can work on

[6]One way to accomplish this in practice has been to have objects in the programming language contain not the actual media data, but only a filename or other pointer to the media. In this way, the media can still be edited independently.

[7]Also in non-stored multimedial e.g. computer-based video conferencing, the content is variable.

the same materials, and in the sense that they have a similar user interface where appropriate. Many, but not all of these requirements correspond to requirements for programming tools for other fields.

While the above requirements for multimedia tools are very general, the remainder of the thesis deals primarily with tools for specific purposes or developers: Chapter 4 presents an object-oriented platform for multimedia programmers. Chapters 5–9 discuss tools for building elastic stories in multimedia. Finally in chapter 10, tools for hypermedia linking are discussed.

Chapter 3

The Need for Programming

This chapter discusses the role of programming in the development of stored interactive multimedia, and the role of the programmer in the multimedia development team.

Real-world multimedia developers often seek to avoid scripting or programming, or at least limit the amount of it in the development process.[1] For good reasons; working with a tool with a graphical WYSIWYG and/or metaphorical interface is often nicer and more productive. Furthermore, many people with creative ideas or other valuable contributions to multimedia development are not capable of computer programming. At the same time, some of them find it is not satisfactory for them to have someone else—a computer programmer—realize their ideas.[2] That also makes experimentation cumbersome. Is it a necessity that multimedia tools always require some scripting or programming, at least as soon as the developer wants to go just a little bit beyond the core of the tools intention and metaphor? If so, how much scripting is necessary, or how far can we limit the requirement that the

[1] Obviously in this context 'programming' is defied as a process involving explicit algorithms and/or data structures and perceived as difficult by the average multimedia author.

[2] This has even been compared to the imaginary situation that a painter had someone else put the brush on the canvas for him or her. That would take away much of the painter's power over his or her work. At the same time, many artists do have people realize their works for them. For instance, a playwright only *writes* his or her theatre pieces; others instruct and play them. It is said that painter Rembrandt Harmenszoon van Rijn (1606- 1669) and novelist Honoré de Balzac (1799-1850) also had people work on their works for them.

author has to do scripting or programming?

An answer to these questions is found with Paul G. Brown[3]. Inspired by the semiology of the American philosopher Charles Saunders Pierce, Paul Brown makes the distinction between *iconic* and *symbolic* interfaces.[4] The modern graphical interfaces are highly (but not exclusively) iconic, consisting of icons. Icons are simplified representations of real things, with which they still have some similarity. Programming or scripting languages, in contrast, are symbolic interfaces, characterized by the relation between the symbol and its meaning being established by convention. It is in no way obvious if you do not know it. Symbolic programming languages (still contrary to iconic languages) press the user to become intimate with the inner workings of the computer and thereby get a better understanding of its potential. This understanding again can support creativity.

It is assumed that the reason we use symbolic languages at all is that they allow us to do things we cannot do in purely iconic languages. (This thesis, for example, is written in symbolic language. It could hardly have been written in purely iconic language.) If this is to be believed, we can conclude that a script or programming language will always give us power that one cannot have from an iconic interface.[5]

This does not give the final answer to the question of how far we can go without scripting, nor does the answer given include many nuances. Striving for powerful metaphorical tools is still worthwhile (like SuperCard and Director). One way to obtain more power might be to provide a number of different metaphors for the developer to choose from.

The above does however suggest that the multimedia developer who wants to exploit the medium's potential should learn programming. This is the

[3]Paul Brown: The Ethics and Aesthetics of the Image Interface. Presented at ASIS Mid Year Meeting i993. Computer Graphics, newsletter of ACM SIGGRAPH, Volume 28, number 1, February 1994, pages 28-30.

[4]Paul Brown includes a third kind of interface, the *indexical* interface, which is the rich kind of interface used in virtual reality.

[5]Elmer Sandvad writes: "It is a well-known phrase that 'a picture can tell more than a thousand words', but there exist also situations where a few words can tell more than a thousand pictures." Elmer Sandvad attributes the saying to Kristen Nygaard (personal communication). The quotation is from Elmer Sandvad: Object-Oriented Development — Integrating Analysis, Design and Implementation. PB–302. Computer Science Department, Aarhus University 1990.

only way to gain full control over the work. As the ultimate consequence, programming tools for non-professional programmers should be developed.

The alternative for the developer who does not want to learn programming will be to let a programmer do some work on the realization of his or her work.[6]

In practice it is very easy for multimedia developers to come in a situation where they want to go beyond the core capabilities of their tools. When going outside the core intention of the tool, scripting is often the way to realize one's ideas. Sometimes other methods are available that include using the tool in an unnatural way that it was not meant for One may ask, if more powerful non-programmer's tools are developed, will it relieve the situation? This is not likely. Rather, as in other areas, users' expectations and developers' ambitions will grow. The need for fully flexible tools will always dictate the need to go beyond non-programmer's tools.

For example, in the Talaria project, a travel metaphor was planned. Even though this is probably the most often used metaphor in hypermedia, none of the tools considered for the project offered direct support for the parts of this metaphor, such as a map; it would have to be programmed 'by hand'. Had a tool with automatic map generation been found, the wish for a fisheye view of the map (see section 10.2) might have rendered that tool useless. This is not necessarily a criticism of the available tools. The generalization of the observation is that each project has its own style and requirements, so it is likely to go beyond what is offered by any specialized tool.

3.1 Conclusion

This chapter has argued that programming has its place in most and in the most interesting multimedia development projects. The programming can be carried out by creative multimedia developers having learned to program, or by programmers in the traditional sense of the word. The choice would depend on the degree of direct a control the creative multimedia author wants over his or her work, and on his or her inclination towards learning to program, among other factors.

[6] See footnote 2 on page 45.

On the background of this conclusion, the next chapter discusses *programmers'* tools for multimedia. That chapter presents Hejmdal, a platform for the creation, editing and playback of time-based data. It also discusses the use of Hejmdal as a basis for new tools for programmers and for non-programmers. Furthermore it discusses the introduction of some new kinds of interaction that could be developed using Hejmdal.

Chapter 4

Hejmdal: Object-oriented Handling of Interactive Media

This chapter presents *Hejmdal* an object-oriented model for manipulation of interactive, computer-based multimedia developed in the PhD study. Hejmdal supports interactive playback and editing of multimedia. The multimedia documents, called *movies*, can consist of graphics, photographs, live video and animations, and sounds. The interactive playback facilities are start and stop playing, random positioning within a movie, and stepping a single frame forward and backward. Movie segments can be interactively cut, copied and pasted. Movies are stored in digital files, thus avoiding the need for additional hardware during playback and editing.

Hejmdal is built on top of the Macintosh extension *QuickTime* from Apple Computer. The movie document architecture is used by QuickTime, and is a de-facto standard for documents that include time-based data. However, movies cannot include interaction. The chapter suggests that a standard document architecture should be developed that allows interaction to be defined in the multimedia documents, not only in the application programs.

The motivation for building Hejmdal is the need for advanced tools for interactive multimedia mentioned in the thesis introduction. As mentioned there, programming tools for multimedia must be able to handle time-based data. QuickTime is a good present-day tool for handling time-based data from a programming language. However, though objects can be clearly identified in

the movie metaphor[1] there is no object orientation in the application programmer's interface of QuickTime, and insufficient integration of code and data. Hejmdal provides object-orientation. This chapter discusses the benefits of using an object-oriented platform. With respect to integration of code and data, there would be advantages of being able to store with the QuickTime movie code guiding its behaviour, e.g. other playback orders than the default linear playback, or specification of interaction: response to mouse clicks, etc.

The thesis introduction mentioned that ordinary computer programs include more and more elements of multimedia, while multimedia presentations that have their closest relatives in the media worlds are also appearing. It is intended that it should be possible to use Hejmdal at both of these extremes: for adding elements of multimedia to existing computer programs and for assembling entire multimedia presentations. In addition, new multimedia tools for both programmers and non-programmers can be built on Hejmdal.

This chapter is largely based on a previous workshop paper[2].

The chapter first includes a section on the background and motivation for Hejmdal. This section briefly describes QuickTime, upon which Hejmdal is built, and MacEnv, of which Hejmdal is an extension. The section discusses the need for tools for working with interactive multimedia.

The following section presents Hejmdal, the object-oriented platform for working with interactive multimedia. It explains in detail how the movies can be played, edited and saved in files using Hejmdal and goes through the various classes in Hejmdal.

Then a section discusses the merits of Hejmdal. This discussion unveils some benefits of using an object-oriented model: the object-oriented model is simpler and clearer than QuickTime itself, and still more powerful with respect to interaction.

The last section of the chapter describes further requirements for Hejmdal and some future work in the field. Hejmdal may include better support for

[1] Most evidently, a *movie* object with part objects *poster, preview* and *track*.

[2] Ole Villumsen: 'Hejmdal — an object-oriented platform for working with interactive multimedia. In Third Eurographics Workshop on Object-Oriented Graphics Preprints, Centre universitaire d'informatique, Université de Genève, October 1992, pages 467-481.[110]

interaction in multimedia. Another idea is to have it support the import of data from external devices and the compression of image and sound data.

4.1 Background

This section describes QuickTime, upon which Hejmdal is built, and MacEnv, of which Hejmdal is an extension. Readers who know QuickTime can skip subsection 4.1.1. Readers who know MacEnv can skip subsection 4.1.2.

4.1.1 QuickTime

This subsection briefly presents QuickTime. QuickTime is a software product from Apple Computer Inc. for the Macintosh and Windows computers (also licenced to Silicon Graphics Inc. for the Indy).

The central part of QuickTime is the *movie*, an architecture for multimedia documents, and a toolbox for manipulation of movies, called *the movie toolbox*. Movies contain time-based data, such as video, sound and animation. (A movie can contain static graphics too; see subsection 4.2.6 Tracks and media, pages 62–63.) A movie is conceptually linear in time; it cannot contain branches, loops or the like. Around the movie toolbox QuickTime includes support for adding *components*. Components can provide functionality of different kinds, for instance image compression, or grabbing or digitizing data from external sources.

The data in a movie can be multimedia or other types of time-based data. Movies can be exchanged between application programs, even over the Macintosh clipboard without problems. The QuickTime movie is virtually the only architecture used for cross-platform distribution of digital video with and without sound. QuickTime movie players are available for Macintosh, Windows, Amiga and various Unix platforms. QuickTime movies are played back without the use of additional hardware.

QuickTime is used to include elements of video, animation and sound in more and more application programs. Popular Macintosh programs like HyperCard, SuperCard, Macromedia Director and the drawing program Canvas

support QuickTime in various ways. QuickTime is distributed with the Macintosh system software and with the application programs supporting it, so very many Macintosh users have access to it.

A natural consequence of this development is a demand for tools for making movies and integrating them in application programs. QuickTime can itself be regarded as such a tool. Hejmdal is an object-oriented tool that can be used for editing movies and integrating them in application programs. Since the movie is a de facto standard architecture, a large amount of multimedia material is already available for manipulation through Hejmdal, and more is expected to come.

4.1.2 MacEnv

MacEnv is an existing object-oriented model built on the Macintosh toolbox. It is a family of class libraries for writing application programs for the Macintosh. MacEnv supports the building of an advanced graphical user interface according to Apple's user interface guidelines. Among other things it includes support for interactive graphics.

One of the strengths of MacEnv is the ease with which the event handling is conducted. MacEnv takes care of all the details of the Macintosh event dispatching and handling. The MacEnv libraries essentially convert all Macintosh event occurrences into invocations of special virtual procedures within the appropriate user interface object (e.g. the one below the mouse pointer). These virtual procedures are often called event patterns or simply events. The application programmer only has to specialize[3] these virtual procedures (events) to specify the actions to be taken in response to user interaction.

The class hierarchy in the figure[4] shows the most important MacEnv classes and their subclass and superclass relations.

[3] 'further bind' in the BETA terminology.

[4] The figure is redrawn from Mjølner BETA System. Macintosh Libraries. Mjølner Informatics Report[83]. MIA 90-10(0.6). March 1992.

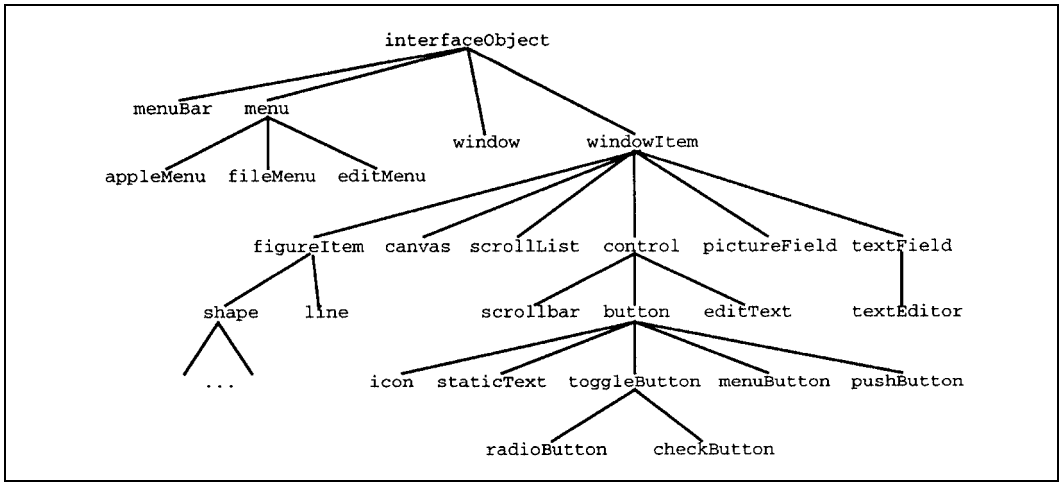


Figure 4.1: The most important MacEnv classes.

4.2 Hejmdal

This section presents Hejmdal[5] as seen from the programmer using it.

Hejmdal is a class library that supports the playback, editing and creation of multimedia consisting of videos, still pictures, sounds and animations. Hejmdal, as well as MacEnv, is developed using the object-oriented language BETA[6]. The central class in Hejmdal is *movie*.

In the following, Hejmdal is first presented by means of an example of how to play back a QuickTime movie. The major classes of Hejmdal are then

[5]There is a tradition for assigning names from Nordic mythology to the tools and libraries used with the BETA system. Hejmdal (also spelled Heimdal) is the whitest of Gods, on guard on Bifrost, the rainbow. At Ragnarok—the twilight of the Gods—he shall blow his horn, the Gjallarhorn, to call the Gods. Since Hejmdal combines the colours of the rainbow with the sound of the horn, Lars Peter Abildskov and Paul Erik Dahl gave his name to the class library that combines the colour video and graphics with sound. References: H. Ellekilde (after A. Olrik): Heimdal. Entry in Salmonsens konversationsleksikon. Anden udgave, Bind XI: Hasselmus-Hven (2nd edition, volume XI). A/S J.H. Schultz Forlagsbog-handel 1921. Pages 149 – 150. (In Danish.)[40] Lars Peter Abildskov & Paul Erik Dahl: Quick-Time. Unpublished project report, Department of Computer Science, Aarhus University, December 15,199l. (In Danish.)[1]

[6]The reference on BETA is Ole Lehrmann Madsen, Birger Møller-Pedersen & Kristen Nygaard: Object-Oriented Programming in the BETA Programming Language. Addison-Wesley Publishing Company 1993.[77]

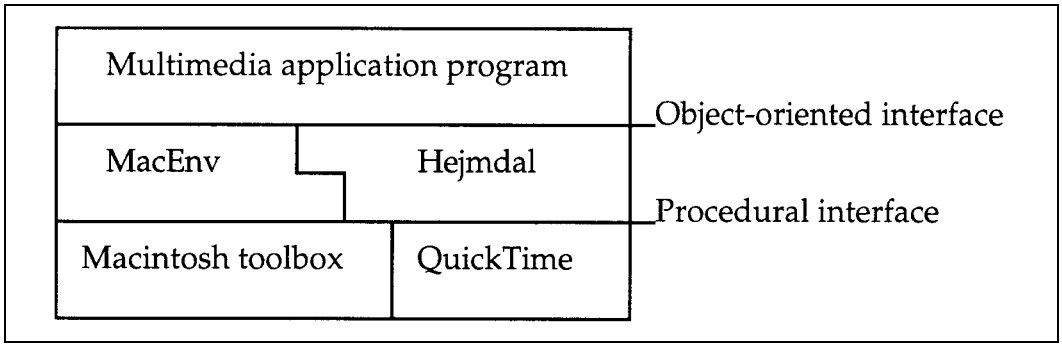


Figure 4.2: Hejmdal is an extension of MacEnv built on top of QuickTime.

presented in turn:

- `Movie`
- `MovieFile`
- `MovieField`
- `MovieEditor`
- `Track`, with subclasses `soundTrack` and `videoTrack`
- `Media`

Finally some further issues are presented:

- Poster and preview objects
- Movies on the clipboard
- Movies and resources

4.2.1 Playing a movie

Every application programmer using Hejmdal will probably want to play back a movie. The movie will most often be retrieved from a file. To play a movie from a file, instances of three Hejmdal classes are needed:

- `movie`: models a QuickTime movie.
- `movieField`: a field in a window in which a movie can be played[7].
- `movieFile`: a file containing one or more movies.

The steps involved in retrieving and playing a movie are illustrated by the following code example. In the example, one object is declared, an instance of a specialization of the class `movieField`[8]. When the method `open` of the object is called (as in the last line of the example), the field is first initialized by the open method that applies to all movieFields, which then calls the code that is special for this movieField's `open` method (`QTInit` and the long if statement).[9] This code retrieves the movie and makes it ready for playing.

```
MacEnv
   (# MyWindow:  @window
        (# type::< windowTypes.plain; (* a plain window ... *)
           hasClose::< trueObject; (* ... with a close box *)
             (* declare theField an instance of a specialization of)
              * movieField (@ denotes an instance):  *)
           theField:  @movieField
             (# theMovie:  @movie;
                (* extend the definition of the open method thus:  *)
             open::<
                  (# theFile:     @movieFile;
                     w,h,dummy1:  @integer;
                     dummy2:      @text;
                     boo,dummy3:  @boolean;
                  do QTInit;
                     (if movieFileType -> theFile.getFile then
                        (* if the user selected a file.  the -> operator is
                         * used for parameter passing and for assignment.  *)
```

[7]More precisely stated, the video part of the movie is played in the field in the window, and the audio part through the computer's speaker.

[8]In BETA specialisation and instantiation can be done in one declaration. This is useful when it is known in advance that only one instance of the specialisation will be needed, which is often the case with interface objects.

[9]In BETA the method in the superclass (`movieField`) is in control of when to 'call' the code of the specialisation in the subclass (the anonymous subclass that `theField` belongs to). The INNER imperative in BETA is used for the 'call'. In this respect BETA is unlike C++ and Object Pascal, in which the method in the subclass (optionally) calls the method in the superclass.

```
                    (* get the first movie from the file:  *)
                    theFile.openRead
                    theFile.getFirstMovie -> (theMovie, dummy1,  dummy2,
                                                                 dummy3);
                    theFile.close:
                    theFile.name -> myWindow.title;
                    theMovie.getDisplaySize -> (w,h);
                    (* adjust the movie field's size so that the
                     * movie + controller fit:  *)
                    (w, h+16) -> size;
                    (* let the window be a bit bigger:  *)
                    (w+40, h+56) -> myWindow.size;
                    (* position the movieFielf in the window:  *)
                    (20,20) -> position;
                    (* enter a reference to the movie into the movie
                     * field:  *)
                    theMovie[ ] -> contents;
                  if);
                #) ; (* end extension of open *)
                close::< (# do QTShutDown; #);
              #) ; (* end theField *)
            open::< (* extend the open method of the window:  *)
               (# do (100,150) -> position; theField.open #);
      #); (* end myWindow *)
do myWindow.open;
#)
```

After the call `theField.open` in the fourth line from the bottom has been completed, the movie can be played. The user can start playing the movie by double-clicking in the movie field. Normally this will be the right way to do it. Apple's user interface guidelines state that the movie should not start playing by itself. This could confuse the user, and she or he might lose the first few seconds of the movie. A screen snapshot from a run of the above program is shown in figure 4.4, page 61.

A movie field's default handling of a double-click is to start playing the movie. Similarly, a single click stops playing. This default behaviour conforms with Apple's user interface guidelines; but it can still be overridden in specializations of `movieField` if desired.

On this point, Hejmdal is reusing the general scheme for interface objects of MacEnv; many of these have a default event handling, but it can always be overridden in specializations.

Should the programmer wish to start playing the movie, this can be done by calling the `movieField` method `start`.

The `getFirstMovie` operation used in the example is the most commonly used operation for retrieving movies, because a file often contains only one movie. Two other operations, `getMovieById` and `getMovieByName`, retrieve a specified movie.

4.2.2 Movies in files

Movies are stored in files.[10] As mentioned in the example, the class `movieFile` models such files. The programmer using Hejmdal needs not be concerned whether the files are stored on a hard disk, diskette, CD-ROM or elsewhere, since QuickTime and the Macintosh operating system take care of this.

The following movieFile methods are defined for retrieving, saving and deleting movies in files. Note that `addMovie` assigns a unique ID to the saved movie and returns it. (This is called a resource ID or resID in the Macintosh terminology.) In the method declarations, the `enter` and `exit` parts of each method describe the in and out parameters respectively. The types of the parameters are declared separately in the declaration part in the beginning of each method.

```
getFirstMovie:
   (* retrieves a movie from the movie file.  Returns the movie,
     * its resource id, its resource name, and a boolean indicating
     * whether references from the movie to the media were changed
     * during the retrieval.
   *)
   (#     newMovie:            @movie;
          theId:               @integer;
          newMovieName:        @text;
          dataRefWasChanged:   @boolean;
   do ...
   exit (newMovie, theId, newMovieName, dataRefWasChanged)
   #);

getMovieById:  (* retrieves a movie from the movie file.  *)
   (#     theId:               @integer;
```

[10]Note fot Machintosh judges: Movies are stored as resources of type 'MooV' in the resource fork of a file

```
        newMovie:            @movie;
        newMovieName:        @text;
        dataRefWasChanged:   @boolean;
   enter theId
   do ...
   exit (newMovie, newMovieName, dataRefWasChanged)
   #);

getMovieByName:  (* retrieves a movie from the movie file.  *)
   (#   theId:               @integer;
        theName:             @movie;
        newMovie:            @text;
        dataRefWasChanged:   @boolean;
   enter theName
   do ...
   exit (newMovie, theId, dataRefWasChanged)
   #);

updateMovie:
   (* Overwrites an existing movie in this(movieFile).
   (* Used for saving changes.
   *)
   (#   theMovie:            @movie;
        theId:               @integer;
   enter (theMovie, theId)
   do ...
   #);

addMovie:  (* adds a new movie to the movie file *)
   (#   theMovie:            @movie;
        theName:             @text;
        theId:               @integer;
   enter (theMovie, theName)
   do ...
   exit theId (* the unique id given to the movie.  *)
   #);

removeMovie:  (* deletes a movie resource from this(movieFile).  *)
   (#   theId:  @integer;
   enter theId
   do ... ;
   #)
```

There is a `movieFile` method `getFile` that presents the user with a dialogue box in which the user can select a file to be opened. The dialogue box is

shown in figure 4.2.2. A similar dialogue for specifying the name of a file to be created and its location in the directory hierarchy can be created using the method `putFile`. The latter dialogue does not contain a file preview.

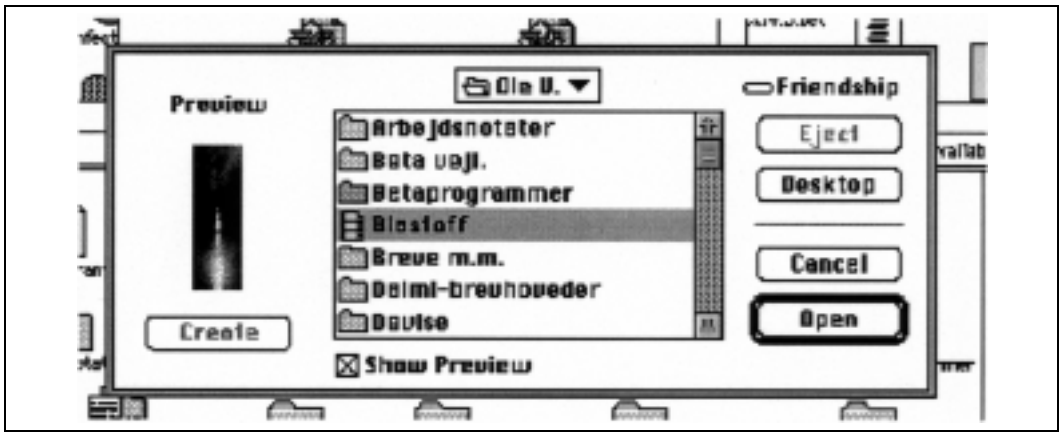


Figure 4.3: File dialogue with preview.

`MovieFile` also includes the methods `openRead,openWrite,openReadWrite` and `close` for opening and closing the file. The method `create` creates a new file, and `delete` deletes the file from the file system.

4.2.3 Movie fields

As shown in the example, a movie can be played back in a `movieField`. The class `movieField` is a subclass (specialization) of the class `windowItem`, which is defined in MacEnv. As the name suggests, every item in a window is a `windowItem`. `WindowItem` has many other subclasses, including `textField`, `icon` and different kinds of buttons, as shown in Figure 4.1 on page 52. `MovieField`, as well as `windowItem`, is declared locally to the `window` class, so an instance of `movieField` object can only be declared where it is meaningful: as belonging in a window.

The following are the most important attributes and methods defined for `movieField:`

- `open:` initializes the movieField.

- `close:` closes the movieField.
- `content:` sets and gets the movie in the movieField.
- `start:` starts playing the movie in the movieField.
- `stop:` stops playing.
- `done:` returns true if the movie is finished playing.
- `rate:` sets and gets the playback rate.
- `goToBeginning:` positions the movie at the start.
- `goToEnd:` positions the movie at the end.
- `time:` the position within the movie. Entering a value into `time` goes to the specified position.
- `playPreview:` plays the movie's preview (described below).
- `previewMode:` sets, clears and gets preview mode. In preview mode, `start` will start playing the preview, not the movie itself.
- `showPoster:` displays the movie's poster (described below).
- `size:` sets and gets the size of the movieField.
- `position:` sets and gets the position of the movieField within the window.
- `show:` makes the movieField visible.
- `hide:` makes the movieField invisible.
- `dragOutline:` lets the user drag the movieField to another position.
- `dragResize:` lets the user resize the movieField by dragging a corner.
- `eventHandler:` defines how events (mouse clicks, etc.) are handled. Can be specialized if the default event handling (as described above) is not appropriate.

Figure 4.4: A window containing a movie window. In cases where it is field with a movie controller.

Of the mentioned attributes and methods, the first two (`open` and `close`) and the last seven (`size` through `eventHandler`) are iherited from `windowItem`, though most of them are specialized in `movieField`.

The programmer can specify whether the movie field is to have a *movie controller*. A movie field with a movie controller is shown in Figure 4.4. (In the figure, the movie field does not fill out the window. In cases where it is the only item in the window, it often will.) The controls on the controller allow the user to turn the sound up and down, start and stop playing, position randomly in the movie using the slider, and step a single frame backward and forward.

4.2.4 Movie editors

The `movieEditor` classis a specialization of `movieField`. Besides playing amovie, a `movieEditor` allows the user to do simple movie editing, namely cutting, copying and pasting of movie segments. It also has an undo capability. To be more specific, when the user selects (clicks in) a movie editor, it takes over the standard Macintosh Edit menu, so the undo, cut, copy, paste and clear operations in the Edit menu directly affect the current selection of the movie in the movie editor. The selection is made by holding down the shift key while dragging the slider of the movie controller over the segment of the movie to be selected.

The cut or copied movie segments are put on the Macintosh clipboard (also called the scrap), so they can be transferred to and from other applications, including applications that do not use Hejmdal (as long as they use QuickTime).

A movie editor can be *locked*. If it is locked, no changes can be made to the

movie in the editor, but it is still possible to copy a part of it and paste it in somewhere else.

4.2.5 Time-based call-backs

A number of local classes[11] in the movie class model different kinds of time-based call-backs. These classes are subclasses of a common call-back class. A call-back can be set up to be called when the movie reaches a specified time during playback, when the movie time is changed ('jumps'), or movie rate is changed. The way to set up a call-back is to specialize and instantiate the appropriate class and initialize the instance. At the appropriate time, according to the specialization done, Hejmdal calls the call-back.[12]

4.2.6 Tracks and media

A movie consists of a number of tracks. Each track is either a sound track or a video track. In Hejmdal these are modelled by the classes `soundTrack` and `videoTrack`, which have a common abstract superclass `track`. `Track`, `soundTrack` and `videoTrack` are local to `movie`. A track has not necessarily the same duration as the movie. Each track has its own (possibly zero) offset from the beginning of the movie and its own duration, as shown in the above figure.

If there is more than one video track, they are layered according to their `priority` attribute.

The tracks themselves do not contain the actual sound and video data. Rather, they contain pointers to media, which are files or resources containing the data. The list of pointers in a track is known as an *edit list*. The relation between movie, track and media is depicted in the figure on page 63.

The Hejmdal class `media` models the media. Hejmdal supports the creation

[11] To be more precise these are local *patterns*, where a pattern can be a class or a procedure.

[12] Here Hejmdal take advantage of the fact that the call-back can itself serve as a procedure. In certain other languages, a method in the call-back class would have to be used.

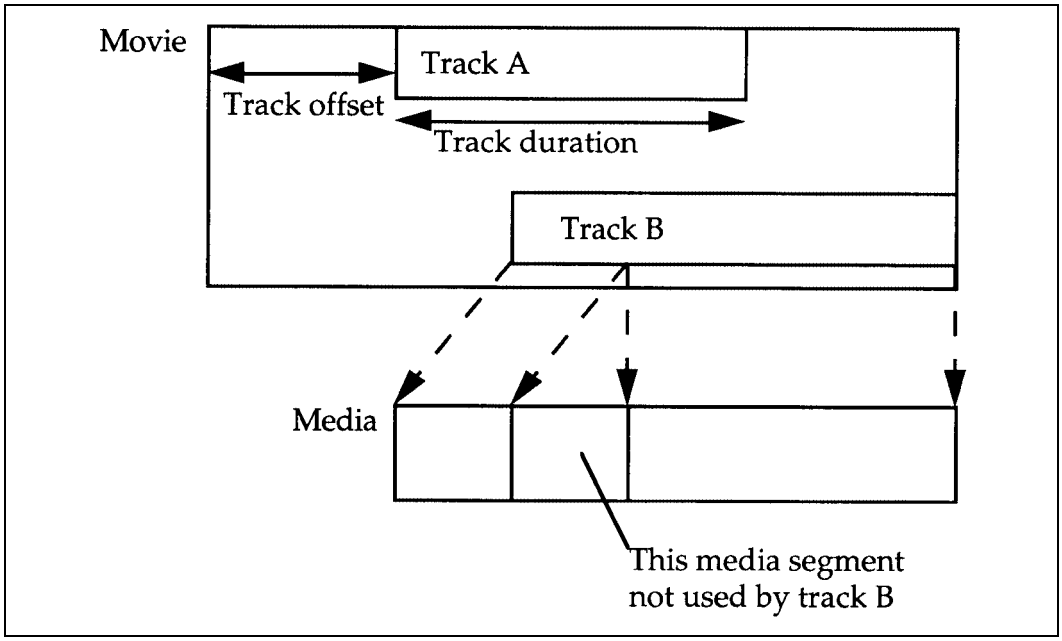


Figure 4.5: Movie, track and media.

of new tracks and media, and the deletion of existing ones. Still pictures can be included in a movie, since an entire video track may point to a single still picture.

There are some advantages of separating the actual data from the movie. In this way several movies can share the same audio and video data without duplicating them. Moreover, the movie itself stays relatively small. This makes it possible to transfer it over the Macintosh clipboard as described in subsection 4.2.8.

4.2.7 Preview and poster

A movie has a *preview* and a *poster*.

A preview is a short sequence from the movie that should give a hint to the content of the movie. In Hejmdal, the preview is a separate object inside the movie with its own methods, but accessible from outside the movie. The `preview` object has the attribute `time`, which sets and gets the preview's

offset from the beginning of the movie and its duration.

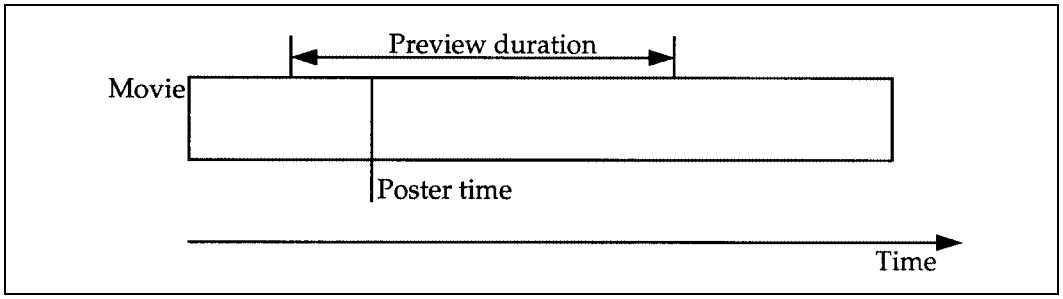


Figure 4.6: Movie with preview and poster.

A poster is a single frame from the movie describing the movie. The `poster` object of a movie has the attributes `time` and `box` and the method `getPict.Time` sets and gets the poster time which selects the frame from the movie to be used for the poster; `box` sets and gets the poster's bounding rectangle; and get `getPict` returns a picture of the poster (in PICT format).

It can be specified that a track is used in the movie's preview or poster, but not in the movie itself, or vice versa. In this way, a preview and poster can be made that are not exact copies of a sequence and a frame from the movie.

4.2.8 Movies on the clipboard

MacEnv contains a model of the Macintosh clipboard, a `clipboard` object with operations for putting and getting texts and pictures and for determining whether the clipboard currently contains a text object or a picture, respectively. Hejmdal enhances the clipboard by adding the attribute `movieContents` that can be used for putting a movie on the clipboard and getting a movie from it. (`aMovie [ ] -> clipboard.movieContents` transfers both the movie and a picture in PICT format to the clipboard, the picture containing the current frame of the movie.) Hejmdal also provides the method `hasMovie` for determining whether there is a movie on the clipboard. These operations are also used by the above mentioned `movieEditor`.

4.2.9 Movies and resources

Movies are stored in what is known as *resources* in files. Without going into a discussion on Macintosh resources and their many uses, it should be mentioned that MacEnv includes a `resource` class to deal with these. Hejmdal adds a new `resource` operation, `getMovie`, which returns the movie in the resource, provided that it is a movie resource.

4.3 Discussion

This section discusses some aspects of the design of Hejmdal and argues that the object-oriented model is simpler and clearer than the procedural model of QuickTime. It also briefly describes the experiences with using Hejmdal for different purposes.

Hejmdal is easier to use than QuickTime, especially for interaction. For example, the programmer using QuickTime for displaying a movie has two choices. He or she can specify in each program the response to a double-click on a movie's display, even if the response is always the same: to start playing the movie. The alternative choice is using a movie controller, which takes care of all events (mouse-clicks, etc.), but requires changing the event loop of the program. In the case of adding elements of multimedia to an existing program, this may be disadvantageous. In Hejmdal the result of double-clicking on a movie's display is specified only once, and every programmer can use it without thinking about it. At the same time this supports user interface consistency across QuickTime applications. If not otherwise specified, the same user action produces the same result each time: the result specified in Apple's user interface guidelines. This holds true for movie fields with and without the movie controller, and for movie editors. This is so mainly because in the object-oriented model it is natural and easy to specify within a class of interface objects the responses to different user actions. In a procedural model, it is more complicated.

Hejmdal (but not QuickTime) provides a clear distinction between a field in a window and the contents of that field. `MovieField` is the field, and a `movie` can be displayed and edited in it. In this respect too, Hejmdal is consistent with MacEnv; in MacEnv a `text` can be displayed and edited in a `textField`.

Similarly a picture (in PICT format) can be shown in a `pictureField`.

Hejmdal uses QuickTime's IDS and names for movies in a file, thereby avoiding the need for sequential reading and writing of movies in a file. This also has the advantage of making a clear distinction between the operation of saving a new movie (addMovie operation) and that of overwriting an existing one (updateMovie operation).

Clarity is enhanced by using local `poster` and `preview` objects in a movie, which are accessible from outside the movie through their specified interface. At the same time, this interface protects the internals of the poster and preview objects from improper access.

It is concluded from the above that the object-oriented model of Hejmdal is simple and clear compared to the procedural model of QuickTime, and yet more powerful. The power and simplicity of Hejmdal can also be illustrated bv the code example given earlier in the chapter.

4.3.1 Use of Hejmdal

Hejmdal has been used to include movies as one of the media in the Macintosh version of DEVISE Hypermedia system (DHM) beside text and graphic.[13] This is one example of using Hejmdal to add elements of multimedia to an existing application. The original work was done by Kaj Grønbæk and the author. Later, while the author was in USA, three students built a larger extension to implement time-based anchors in movie nodes in DHM. (A time-based anchor in a movie is defined by a start time and a duration.) Except for an initial introduction to Hejmdal, they used it without any assistance beside the documentation. They reported that they had no problems using Hejmdal for the task.

In a multimedia course in the Institute of Information and Media Science at Aarhus University, the author converted a multimedia presentation from SuperCard to BETA using Hejmdal. The report written for the course concluded that Hejmdal was well suited for assembling such presentations. The

[13] On Devise Hypermedia, see for instance Kaj Grønbæk, Jens A. Hem, Ole L. Madsen & Lennert Sloth[48]: Designing Dexter-based Cooperative Hypermedia Systems. In Hypertext '93 Proceedings. Fifth ACM Conference on Hypertext Proceedings. ACM 1993. Pages 25–38.

BETA implementation gave efficiency improvements which made the presentation look nicer. Using the persistent store in the Mjølner BETA system a database was built that was structured better than the database in the SuperCard implementation. This database contained pointers to the relevant QuickTime movies in the form of file names.

The uses of Hejmdal reported above suggest that Hejmdal is useful both for adding more media to existing computer programs and for assembling presentations from scratch, as long as the material is available as QuickTime movies.

4.4 Further requirements and future work

In the introduction, page 26, it was argued that interactive multimedia can be more useful than non-interactive multimedia. Since *interactive* computer-based multimedia are often needed, the most important future development of Hejmdal will be that of including different kinds of interaction. Hejmdal today supports two kinds of interaction: interactive movie playback and interactive movie editing (cut-copy-paste). Some ideas for different kinds of interaction are described in this section.

Hejmdal should be an object-oriented model of all of QuickTime. As described in this chapter, it models the central part of QuickTime, the movie toolbox. The further requirements are briefly sketched below.

Having Hejmdal cover the functionality of QuickTime would mean including (most importantly) the following kinds of interaction:

- Interaction through pointing to specific (possibly moving) objects in the movie. This could be realized through the dynamic definition of areas in the movie that are sensitive to mouse clicks. Fortunately, QuickTime 2.0 supports this through inclusion of hit tracks in a movie, so it should be practicable to include it in Hejmdal.
- Interactive capture of video and sound from external sources (e.g. video camera, video cassette player, laser disc player and microphone).
- Interactive panning and zooming of the video or animation track(s)

of a movie, while the whole picture is not visible at the same time. The scroller class of MacEnv combined with QuickTime's possibility of scaling the image could realize this.

As described in subsection 4.1.1 QuickTime, page 51, QuickTime, besides the movie toolbox, supports image compression and the addition of components. Hejmdal covers the functionality of the movie toolbox, and as it is developed it should cover image compression and components too. This implies that object-oriented models will be designed for the *compression manager* of QuickTime, and for the *component manager* including each of the seven possible component types.

MacEnv has a successor called *Lidskjalv* or *GUIEnv*, a family of class libraries for building graphical user interfaces across different platforms: Macintosh, Windows and *X*. Hejmdal should be ported to Lidskjalv and should be implemented on other platforms where QuickTime is available: Windows and possibly the Indy.

4.4.1 On the fly editing

Interactive editing while the movie is playing should be enhanced, so that it is not limited to the cut-copy-paste editing possibilities already in Hejmdal. Rather, it should be possible to fade the movie in and out, make special effects when changing from one movie to another, and use filters. It should be possible to define and realize all of this while the movie is playing. At a click of a button, the current frame should dissolve or fade to an arbitrary frame in the same or another movie, or the new movie should slide in from the side, or the old one out.[14]

In implementing this, one could take advantage of the mechanism in QuickTime to set the degree of transparency of a single video track in a movie. In the following it is assumed that each QuickTime movie to be played consists of a single video track. This assumption makes the explanation easier. The idea obviously also works with more tracks.

First a new movie is created. This will be the one that is actually *playing* all

[14]The movie editing program Premiere from Adobe can use many such special effects when making transitions from one clip to another; but not on the fly.

the time. To show other movies, the tracks from them are copied into this one. First the track is copied from the movie to be shown first; or if it is not to play from the beginning, only the segment from the starting point to the end is copied.

When a jump using a cross dissolve is wanted, the new track (could be the same as the old track) is copied into the playing movie. Only the segment from where watching should start to the end is copied. It is given an offset equal to the time of the playing movie so it starts immediately. The new track is layered behind the old one. Then gradually, while the movie is playing, the old track is made increasingly transparent, to make the new one fade in. When it is completely visible, the old track can be deleted. (Alternatively, the new track can be layered in front of the old one and made completely transparent from the start, then gradually less transparent.)

QuickTime might have performance problems, given it is treated as sketched. It cannot be known in advance whether poor playback quality due to efficiency problems will result. Finding out requires trying it.

Sliding movies in or out can be done in a similar way; instead of gradually adjusting transparency, location and clipping is gradually changed.

Sound tracks and other kinds of tracks introduce a new level of complication, but are probably manageable. The old sound track is faded out by repeatedly decreasing the volume, and at the same time the new one is faded in.

4.4.2 Defining interaction in the documents

A movie is conceptually linear in time, and the natural way to play it will be from the beginning to the end. A consequence is that the movie cannot itself include interaction. Hejmdal will allow the above kinds of interaction to be defined in *application program* manipulating movies.

However, a standard multimedia architecture should allow interaction to be defined in the *documents*. For the kind of multimedia presentations used for instance in museums and education, this will be the natural place to define it, since these are sometimes closer related to traditional media than to traditional computer application programs. You may also say that an interactive multimedia document resembles a program more than a traditional docu-

ment. Hence the program displaying the multimedia document gets the role of an interpreter.

Fortunately, standard document architectures that include interaction exist. Among them are HyTime[15] and MHEG. The latter is still under development.

4.5 Conclusion

In the preceding sections a suggestion has been given of an object-oriented platform or class library for manipulating multimedia on a computer. The platform has good support for playback of multimedia consisting of still pictures, videos, animations and sounds. It has some support for editing, but only little support for the creation of new multimedia documents. Hejmdal can be used for a range of applications, from those adding a few elements of animation or sound to an existing application to the 'pure' multimedia application programs. Hejmdal has the advantage of using a proposed standard document architecture, the QuickTime movie.

It has been pointed out that interaction is very useful in multimedia, so a platform for working with multimedia should support the creation of interactive multimedia. It has further been argued that it should be possible to define interaction in the multimedia documents, not only in the application programs using them. However, QuickTime movies lack this possibility.

The object-oriented model of Hejmdal is found to be simple, clear and powerful, without loss of flexibility compared to the underlying QuickTime.

Further development of Hejmdal will enhance the possibilities of defining interaction with multimedia. Furthermore, development of support for import and compression of image and sound data will make it reilistic to use Hejmdal for the *creation* of multimedia presentations, including creation of the material.

[15] On HyTime see SGML SIGhyper Newsletter[98]. An Occasional Publication of the Standard Generalized Markup Language (SGML) Users' Group's Special Interest Group on Hypertext and Multimedia (SIGhyper). Volume 1, Number 1, October 1991.

Chapter 5

Theory of Narration

The purpose of this chapter is to lay the ground for the exploration of Petri nets as a tool for interactive story telling presented in the following chapters. The basis has the form of narratology, a theory of narration. The purpose is to get an understanding of what a narrative (a story) is, which in turn will be used to develop a model of a narrative of sufficient quality to be used in the work in following chapters. Especially in the context of story construction, which is the subject of the exploration in the following chapters, it is highly relevant to look at concepts and tools developed to *analyse* narratives. It is assumed that the same concepts and tools are useful in *synthesis and construction* of stories, and thus can serve as a good inspiration for computerized tool support. In addition, they can provide an *understanding* of what a story is and the requirements for tools for building computer-based stories.

Many theories of narration exist, most of them in the form of general literature theories. This chapter briefly sketches five common theories. It argues why the New Criticism is chosen as the theory on which the work with Petri nets for elastic story telling is based. Thereafter, the extended layer model from New Criticism is presented in greater detail.

5.1 Choice of theory

Some of the better known literature theories are:

1. New Criticism
2. Structuralist narratology and semiotics
3. Marxist literature criticism
4. Impressionist literature criticism
5. Deconstruction

5.1.1 New Criticism

New Criticism, or Anglo-Saxon New Criticism,[1] holds that a literary work is an autonomous unit and that it would thus be misleading to supplement text reading, e.g., with studies of the life of the author or the historic background of the text.[2] New Criticism rejects the idea that *form* and *content* should be distinct notions and considers them inextricably connected. Instead it uses the pair of *material* and *structure*. According to New Criticism, literature is a branch of art the material of which is language. The use of language in literature is claimed to be fundamentally different from its everyday use. New

[1]New Criticism in its narrow sense is a movement in the Southern States of the USA after World War I, though its members polemicize against each other and disclaim connection with the movement. In its broader sense, New Criticism is a current evolving along with modem literature, receiving impulses from many sides, most notably in USA and England, and having consequences all over the Western world. The first main work of the development of New Criticism was 'The Sacred Wood' from 1920, by American-born English poet Thomas Stearns Eliot (1888-1965)[39]. The last one was 'Theory of Literature' from 1949, although Rene Wellek[113] (born 1903) did not consider himself belonging to New Criticism. Other main characters of New Criticism are Ivor Armstrong Richards (1893–1979), William Empson (1906–1984), John Crowe Ransom (1888–1974), Allen Tate (1899–1979), Cleanth Brooks (born 1906), Richard Palmer Blackmur (1904–1965), Kenneth Burke (born 1897) and possibly Yvor Winters (1900-1968). Source: Johan Fjord Jensen: Den ny kritik. Berlingske Forlag 1962.2nd printing Munksgards forlag 1966. Reprinted with an epilogue by the author: Kimzere 1989. (In Danish.)[61] Other references: T.S. Eliot: The sacred wood. Essays on poetry and criticism. 1920. 7th edition, reprinted: Methuen 1960.[39] Rene Wellek & Austin Warren: Theory of Literature. Third Edition. ('New revised edition'.) A Harvest Book. Harcourt, Brace & World, Inc.1962.[113]

[2]Finn Brandt-Pedersen and Anni Bonn-Paulsen: Metode bogen. Analysemetoder til litteizre tekster. Nogleforlaget Aps. 1980. (In Danish.)[24] New Criticism is treated on pages 9–34.

Criticism says that the literary work has its own life, also in the experience of its creator[3]. The literary work is concerned with universal human problems like life, death, love, hate and deceit. The quality of a work lies in being convincing (credible) to the reader. There is one correct reading of a work, but no authority to decide which. The reader can only rely on his or her own judgement[4]. Finn Brandt-Pedersen and Anni Rønn-Poulsen[5] give two text models belonging to New Criticism; the general *layer model* and the *extended layer model* for epic texts. The layer model defines eight layers in a text, starting with the graphical layer and ending with the statement of the text.[6] The extended layer model is concerned with five layers: composition, narrator issues, fiction, language issues, and finally summary and interpretation. The extended layer model will be further discussed in section 5.3, pages 79–88.

[3]This is expressed in many places. An example is this quotation about what T.S. Eliot calls 'automatic writing': ' ... ; but he to whom it happens assuredly has the sensation of being a vehicle rather than a maker.' From T.S. Eliot: Selected Essays. Third enlarged edition April 1951. Faber and Faber Limited. Reprinted July 1972. Page 405.[38]

[4]In other words 'Thereby the work, the text itself, becomes the highest authority, the final court of apical in all questions regarding the correct understanding of the text.' 'That the text is the highest authority can also be expressed as the reader is the highest authority. Each individual reader. One cannot experience art on others' authority.' (Quotations from Finn Brandt-Pedersen: Tekstlæsning. Gyldendal 1967. 5th printing 1974. Pages 7 and 8–9. My translations from Danish.)[23]

[5]Previously referenced work, [24].

[6]The number and kinds of layers are discussed by Finn Brandt-Pedersen [24], pages 17–23.

5.1.2 Structuralist narratology

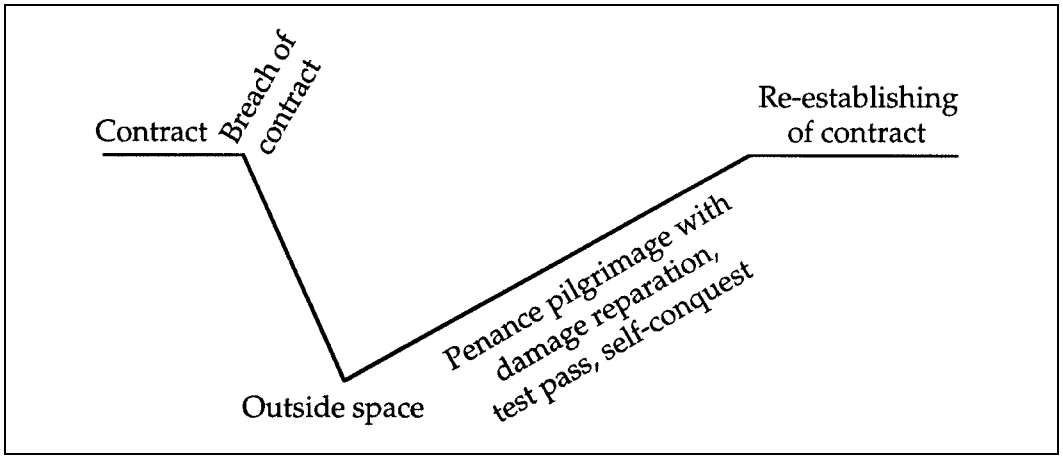


Figure 5.1: Example of a model from structuralist narratology: the contract model.[7] When breaching the contract (which may be informal), the main character is expelled from society into the outside space. Here, rules are different; magic may take place, for example. Through a long and cumbersome process (often through three tests; the qualifying, the decisive and the glorifying test) the main character shows that he (she?) deserves re-admittance into society (often as a hero). Hereby the contract is finally re-established.[8]

Structuralist narratology[9] and semiotics are two out of several subdivisions of French structuralist research[10]. Structuralist narratology gives three models

[7]The figure is translated and redrawn from Finn Brandt-Pedersen and Anni Rønn-Paulsen[24], page 49.

[8]Finn Brandt-Pedersen and Anni Rønn-Paulsen[24], page 49–50. Algirdas Julien Greimas & Joseph Courtés:[46] Semiotik. Sprogteoretisk ordbog. Aarhus universitetsforlag 1988. Danish version of Semiotique. Dictionnaire raisonné de la théorie du language. Hachette 1979. Entries narrativt skema, kontrakt, prøve, kvalificerende prøve, decisiv prøve and glorificerende prøve (narrative schema, contract, test, qualifying test, decisive test and glorifying test).

[9]Structuralist narratology was developed in France in the 1960s. It has two origins: first *linguistics*, founded by Ferdinand Saussure (1857–1915), next the *Russian formalism*, especially the work done by Vladimir Propp at the end of the 1920s. Structuralism as such originates as a language theory from the beginning of the 20th century, according to Pi1 Dahlerup: Dekonstruktion — 90'ernes litteraturteori. Gyldendal 1991. (In Danish.) Page 27.[32]

[10]Some prominent structuralist researchers in France in the 1960s are Claude Lévi-

of a story: the static *actant model*, the dynamic *contract model* and the static *s* or *square model* containing the dynamic *butterfly route*[11]. As an example, the contract model is sketched in the figure on the previous page.

5.1.3 Marxist literature criticism

Marxist literature criticism[12] is a sub-area of Marxist research. According to Marxism, nothing should be seen separately from its social and historical connections. In Marxism, there really is only one research field, namely the entire society in its historical development. A key point in Marxism is that the ruling thoughts are the ruling classes' thoughts; the proletariat's consciousness is dominated by norms that serve the rulers' interests, not their own. This phenomenon is known as *false consciousness*. Literature can on one hand cement a false or on the other hand promote a true consciousness, an awakening. A number of models and tools are used by Marxists in literature criticism. As an example, György Lukács (1888-1971) developed the concept of *realism*, which denotes a positive quality of a literary work. The criterion is whether the work blurs or promotes an understanding of society. There are two requirements: First, there is the requirement of broadness and totality of society. Rather than describing only situations that may be more interesting, the text should include all of work life, leisure time, family life, etc. Second, the main characters' personalities should be influenced by essential contemporary problems in society. Fulfilling these requirements and making talented observations will lead a writer to the victory of realism, György Lukács writes[13]. Realism is in some sense in opposition to naturalism; naturalism shows detached images of the world while realism shows coherence.

Strauss, Roland Barthes and Algirdas Julien Greimas. John Chr. Jørgensen also regards New Criticism as a kind of 'formal structuralism'. John Chr. Jørgensen: Litterær metodelære. Metoder i dansk litteraturforskning efter 1870. 2. udgave (2nd edition). Borgen 1974.[63]

[11]Finn Brandt-Pedersen and Anni Rønn-Poulsen, [24], pages 35–69. Algirdas Julien Greimas and Joseph Courtés, [46].

[12]Finn Brandt-Pedersen and Anni Rønn-Paulsen [24], pages 71–111.

[13]Finn Brandt Pedersen and Anni Rønn-Paulsen,[24], page 85.

5.1.4 Impressionist literature criticism

Impressionist literature criticism is the lack of method or model, the spontaneous evaluation. The individual experience of whether the text touches or fascinates and the associations that the reader derives from the text are essential.[14]

5.1.5 Deconstruction

Deconstruction, taken in a broad sense, has four points[15]:

1. The meaning of the text is not *in* the text, but only results from the reader working with it[16]. The literary experience is a reader's meeting with him or herself, not with something new and different. The reader is present in the text in many ways, ranging from explicit address to for instance comparisons with something the reader (supposedly) knows[17]. The reader in the text is, however, different from the physical reader, just as the narrator in the text is not the same as the author.

2. Deconstruction gets its name from deconstructing the unambiguous contrasts held by structuralism (briefly presented above). Deconstruction claims that such contrasts are artificially bound together by suppressing incidental circumstances that do not fit with the contrast[18]. In reality, each side of a contrast holds the other side inside itself.

3. Paul de Man[19] worked with a deconstructive theory of rhetoric and

[14]Finn Brandt-Pedersen and Anni Rønn-Paulsen, previously referenced work[24], pages 113-121.

[15]The account of deconstruction is based entirely on Pil Dahlerup, previously referenced work[32], from which other references have been taken.

[16]Jane P. Tompkins (editor): Reader-Response Criticism.[105] John Hopkins University Press, Baltimore and London (1980) 1984.

[17]Gerald Prince: Introduction to the Study of the Narratee. In Jane P. Tompkins[105], previously referenced work.

[18]Jacques Derrida: De la grammatologie I: L'ecriture avant la lettre. Collection critique 1979 (original version 1967)[35].

[19]Paul de Man 1919–1983, former nazi, is the thinker of the Yale school.

with allegories.[20] He deconstructs romanticism by claiming it is filled with allegories rather than symbols, as it is usually said to be[21].

4. There are a number of feminist decontructionist movements. Broadly taken, they tend to move from feminist research to gender research, from empiricism to theory and from reality to semiotics, especially the signs for man and woman[22].

5.1.6 Choice of theory

Among the above theories, New Criticism is chosen as the theory to be used in the following work with interactive stories, for a number of reasons:

1. While most of the theories are general literature theories that do not concern themselves specifically with narration, New Criticism contains a text model especially suited for epic, that is narrative texts; the extended layer model.

2. While most of the theories reach beyond the narrative itself (e.g., Marxism into society as a whole), New Criticism explicitly regards a text as an autonomous unit. In this way, it suits well the computer scientific tradition of studying objects separately.

3. Several of the concepts in the extended layer model seem naturally applicable, not only to analysis of stories, but also to story construction, which is essential here.

Many multimedia systems (e.g., museum systems) do refer beyond themselves. This is not relevant, though. New Criticism will be used for the *narrative* aspects of multimedia systems, which supposedly are the same even if the multimedia system also is *more* than a piece of fiction.

[20] A symbol has a double meaning: a concrete meaning and a symbolic or metaphorical meaning. An allegory has only the latter. A more precise definition of a symbol is given on page 85.

[21] Paul de Man: The Rhetorics of Romanticism. Columbia University Press 1984[79].

[22] Pil Dahlerup[32].

5.2 New Criticism

The basis and ideas of New Criticism were presented above. This section and the next present New Criticism's tools for analysis and evaluation.

For New Criticism, *close reading* of the literary work is a key idea, as opposed to a study of the author's letters, contemporary cultural movements or other objects. The literary quality depends on the consequence with which the details have been organized into the whole and the purpose of the text. The purpose is the establishment of a statement. Two aspects are evaluated:

- the essentialness of the statement that the text formulates; and
- the precise and consequent use of means of expression to establish that statement; the *aesthetic* evaluation.

Finn Brandt-Pedersen argues that there is a close relation between the two[23].

The layer model is used to analyse the organization of a text[24]. The model consists of eight layers. On each layer, the organization of the entire text can be studied. The lavers are:

1. The graphical layer: visual appearance, for instance size of paragraphs.
2. The sound layer: metrically organized rhythm, rhymes, onomatopoeia.
3. The semantic unit layer: the single words and short word connections, including metaphorical meanings, associations, leitmotifs and symbols.
4. The syntactical layer: lengths and complexities of sentences, whether sentences are complete or incomplete, and closer to written or spoken language.
5. The plane structure layer: whether longer sections of the text (possibly the entire text) can be read on two or more planes, e.g., a real and a symbolic plane.

[23]Finn Brandt-Pedersen Tekstlæsning. Gyldendal 1967 (5th printing 1974), pages 85–96[23].
[24]Finn Brandt-Pedersen; the referenced work[23].

6. The big elements layer: how the details combine to greater units such as persons, environments, phases of action and problem-presentations.

7. The narrator attitude layer.

8. The statement of the text.

In analysis, it is considered a good idea to start out from the obvious observations, e.g., where the text surprises or breaches norms and at sudden changes, for example, in style. The observations made on each layer should all contribute to the final layer, the statement.

5.3 The extended layer model

The extended layer model supplements the layer model. It contains five layers. This section presents these layers. The presentation will be based on Finn Brandt-Pedersen and Anni Rønn-Paulsen[25]. Since the extended layer model was developed for analysis of non-interactive, spoken or written *textual* stories, its application to interactive and non-textual stories, especially stories in stored, computer-based multimedia, will be discussed.[26]

Where appropriate, how to interpret the concepts in this new context will be discussed. The quality of the resulting implicit model of non-textual stories lies not so much in being true to New Criticism as in being useful in the work on elastic stories presented in the following chapters.

The five layers of the extended layer model are:

1. Composition

[25]Previously referenced work, pages 23 and 155–169[24].

[26]Christian Metz the founder of film semiotics, would have claimed that one cannot take a model from one medium and apply it to works in another medium. However, in this chapter this is found a perfectly meanngful thing to do. Christian Metz's warning is taken only to mean that one cannot expect that the resulting model covers the new medium well. For this reason, it will later be supplemented with observations from elastic stories and with considerations about the specific multimedia user interface in which the realization of elastic stories is envisioned. For a presentation of Christian Metz's ideas in Danish, see Palle Schantz Lauridsen: Christian Metz' filmsemiotik[69]. In Sekvens — filmvidenskabelig årbog 1984. Københavns Universitet. Institut for Filmvidenskab.

Sections, sequences, suspense curves.

2. Narrator issues

   (a) Type: first person narrator
   'visible' third person narrator
   concealed third person narrator

   (b) Point of view, placing, personal technique.

   (c) Forms of presentation

   - scenic: dialogue, situation narrative, compressed narrative (with time sequence);
   - panoramic: compressed narrative (without time sequence), description, information, commentary, reflection.

   (d) Narrated time/narrative time.

   (e) Postulate/demonstration.

3. Fiction

   Characters, environments, problem-presentation.

4. Language issues

   Characterization of persons or environments through language. Symbols, images (metaphors), leitmotifs, etc.

5. Summary and interpretation.

The concepts will be presented in the same order as in the above model, with a few exceptions. Explanations of the concepts of the model will be provided where deemed necessary, that is under items 1, 2 and 4. Thus, the model summary above serves as a rough 'table of contents' for this section.

5.3.1 Composition

According to the extended layer model, a story consists of *sections* (item 1. in the model). This holds true for both textual and non-textual stories. For instance, a movie may consist of sequences that are built from scenes which

are composed from shots. In non-interactive stories, the sections are ordered linearly, which is often not true for interactive stories.

Sequences are found in many media (texts, plays, films, slide shows, sounds) while others lack them (pictures, sculptures). Interactive media often have other ordering schemes besides linear sequences.

It is an open question whether it is possible and feasible to build *suspense curves* in interactive media. Jørgen Bang argues that the great potential of interactive computer fiction is in what he calls *circular narration*, in which the suspense curve has many tops rather than one climax[27]. For discussions of suspense curves, Jørgen Bang refers to Birgitte Hesselaa[28] and Peter Harms Larsen[29].

5.3.2 Narrator issues

The narrator of a story (item 2.a.; not to be confused with the author) can be either a first person or a third person narrator. A first person narrator is a narrator who is at the same time one of the persons in the story. A third person narrator is not present *in* the story. A third person narrator can be 'visible' or concealed. A 'visible' third person narrator is a third person narrator who reasons about the events in the story, e.g., by comments, evaluation or explanations. (However, the narrator is not so visible as the persons in the story.)

A 'visible' narrator (whether first or third person) is much less common in movies and cartoons than in textual stories. It may not become widespread in computer-based multimedia either. On the other hand, in non-textual media, non-fiction may need a 'visible' narrator more than fiction, as a way to introduce text in a natural way.[30]

[27]Jørgen Bang: The meaning of plot and narrative. In Peter Bogh Andersen, Berit Holmqvist & Jens F. Jensen (editors): The computer as medium. Cambridge University Press 1993[9]. Pages 209–221.

[28]Birgitte Hesselaa: Kan detektiver synge? In Kritik 85, pages 41–58. Gyldendal 1988.[56] (In Danish.)

[29]Peter Harms Larsen: Faktion - som udtryksmiddel. Danskkererforeningen and Forlaget Amanda 1990.[68] (In Danish.)

[30]For a contemporary example of a first person narrator in a non-fiction cartoon, see Claus Deleuran: Illustreret Danmarkshistorie for Folket. Volume 1–8. Ekstrabladets forlag

In non-textual media, there is no technical difference between a 'visible' narrator and any other person in the story: any person visible or audible may, depending on the contents of the media, serve as a narrator. Therefore, there is no need for separate tool support for first and 'visible' third person narrators.

The narrator's *placing* (item 2.b.) is the place in the story from which he or she reports. In movies, the narrator's placing is roughly the same as the placing of the camera(s) and microphone(s). As in textual stories, the placing may be with one of the characters. Two common examples:

- The camera may look over the head or shoulder of a person to show the viewers what that person sees, thereby taking the place of that person.[31]
- Using the 180° rule, one shot may look at a person, the next shot show what that person sees.[32]

The *point of view* is more subjective. Typically, the point of view will, if present, be either with one of the persons in the story or with the narrator (if 'visible'). One possible way to establish a point of view is through *personal technique,* see below. (Also the placing may of course be with one of the persons.)

Personal technique is used when a narrator, without directly stating it, conveys a person's *experience* of the events of the story rather than an objective

1988- 1994[34]. (In Danish.) (This narrator is depicted on the front page of the thesis.) Microsoft Bob may be seen as an example of a computer-based first person narrator.

[31] Dan Nissen and Anne Jerslev[88] write in a film analysis: 'The camera tilts to Paul, and the angle of view of the room, seeing Johnny and Hans, tends to be as Paul sees it.' Translation by Helen Gray and the thesis author from Dan Nissen & Anne Jerslev: Film- og TV-analyse. In Ib Brokner Christiansen & Else Lutzhoft (editors)[88]: Billeder i bevægelse. Fakta og fortolkninger. Foreningen af filmlærere i gymnasiet og HF and Dansklaererforeningen (FFS) 1984. Page 162.

[32] " ... the rules for doing so [identifying the spectator with the camera] have been assimilated to the dominant conventions of filmmaking to such an extent that they appear natural and inevitable. Among these unquestioned assumptions are the following: ... 2) the 180° rule, ensuring that the spectator always finds the same characters in the same part of the screen, i.e., matching 'screen space and narrative space' ... " Robert Lapsley and Michael Westlake: Film Theory: An Introduction. Manchester University Press 1988[67]. Pages 140–41.

account. Example: 'That darn door bell rang again. Was his life a farce? Was he to be split between two eternally ringing bells? The door bell and the telephone? ... '[33]. Personal technique differs from a 'visible' third person narrator in that the 'visible' narrator conveys his or her own reflections and opinions, not those of one of the characters in the story. In other words, with a 'visible' narrator the point of view is with the narrator rather than with one of the persons. The two techniques can of course be combined, in which case the point of view will shift back and forth.

In movies, a point of view can for instance be established using what is known as a *subjective camera*. Rather than showing a realistic image of something, a subjective camera shows a distorted image of what things look like through some person's eyes or in that person's mind[34]. E.g., an object that frightens the person may be shown bigger than life. A similar use in still and moving pictures in multimedia is imaginable. One may say that subjective camera mimics personal technique.

Narrated time (item 2.d.) is the time that passes *in* the story. Narrating time is the time it takes to narrate the story.

The following paragraphs go through the different *forms of presentation* (item 2.c.) as though they were clearly distinct. In real stories, mixtures and middle forms often occur.

In a written text, a *dialogue* is usually between two persons, and no other information than the content of the lines[35] is given (not even the names; it is tacitly assumed that every second line belongs to each person). In other media, more than two persons can be involved in a conversation, face expressions can be given, etc. In a movie, cutting back and forth between the persons as they speak is often used. In a dialogue or conversation, narrated

[33]Translation by the thesis author's from Finn Brandt-Pedersen and Anni Rønn-Paulsen[24], reviously referenced work.

[34]'We not only see what the character sees, but also *how* he sees it.' (Emphasis is original.) Edward Branigan: Point of View in the Cinema. A Theory of Narration and Subjectivity in Classical Film. Foreword by David Bordwell. Mouton Publishers 1984[25]. Page 6.

[35]In this and the next chapters 'line' consistently refers to a person saying a coherent text. Hence 'line' is sometimes used where one would expect 'text', 'line of text' or even 'speech'. 'Line' is used to distinguish from 'speech', which is used to refer to the *sound* of the line. Obviously, the line in this sense can be longer than one text line.

time and narrating time are nearly if not completely identical.

Situation narrative is a sequence of a narrative in which a series of events is described in detail, such that the reader is to some extent able to visualize it, almost like a sequence of a film. Narrated and narrating time are close to being identical, or at least there is an illusion that they are.

In a *compressed narrative with time sequence*, a sequence of events is summarized without all the details. The events are reported at a much faster pace than they happened. In a radio play, movie or multimedia system, a similar effect can be obtained using frequent cutting.

The forms of presentation presented so far (dialogue, situation narrative and compressed narrative with time sequence) are referred to as *scenic*, since they can be directly staged as scenes in a play or a movie. As an exception, the narrative of a stream of consciousness is also referred to as scenic. The scenic forms of presentation constitute the *epic* genre in its strict sense. All of them (or variants of them) are commonly found in media other than text (e.g., film) and are clearly suited for multimedia.

The non-scenic forms of presentation are presented in the following. They are collectively referred to as *panoramic* (second bullet in item 2.c.).

Here is an example of compressed narrative without time sequence: 'The following weeks, Søren spent the afternoon and evening at Kirsten's place nearly every day. Sometimes they did their homework, sometimes they just had tea. The few moments he still had to be with his parents, they asked no unpleasant questions.'[36] Presumably, a compressed narrative without time sequence is well suited for telling about situations that last a while and events that are repeated in narrated time. The very different time specifications in the example are typical for this form of presentation.

Description, information, commentary and reflection are different kinds of insertions from the narrator. All of them break the time sequence of the narrative.

Description may be said to be often replaced by still or moving pictures of persons, places, etc. in non-text media, including multimedia.

[36]Translated from Finn Brandt-Pedersen and Anni Rønn-Paulsen[24], previously referenced work, page 162.

Except for this substitute for description, the panoramic forms of presentation in media other than text require the narrator to become more or less 'visible', which happens less often in non-text media. They can of course be used within spoken or written text which may be part of the presentation. For example, one person may be the author's mouthpiece and give the author's description, information, commentary or reflection in his or her lines. Note that strictly speaking, such lines would still count as situation narrative, not as a panoramic form of presentation.

5.3.3 Language issues

Persons or environments can be characterized through language (item 4). A person can be characterized either through his or her own language or through the language used by the narrator when speaking about the person in question. In movies, a person or his or her relation to the surroundings can be characterized through camera angles. In operas, persons are sometimes characterized through the musical style[37]. These and possibly other options are available in multimedia.

A symbol is to be read on two planes: a real plane, and a symbolic plane where it serves as a metaphor for something else. A symbol is recurrent in the entire text or a larger section of it. For instance, if a lover in a love story grows roses, the statements in the text about why the roses sometimes flourish and sometimes do not, may be intended to be read as statements about the love affair. Symbols are used in both textual and non-textual media, e.g., movies. For example, in some movies the weather is used as a symbol in a number of scenes' sunshine may symbolize joy, rain sadness (tears), thunder anger or a threat, and so on. It seems, however, that symbols in movies are often local to a single scene, contrary to the definition just given.[38]

An image is similar to a symbol; but it is not to be taken literally on the real plane. Here is an example: 'It is irresponsible to keep silent in a furiously

[37]The opera Fidelio by Ludwig van Beethoven is the first known example of this.

[38]For instance in a scene in the movie 'The Piano', the husband is lurking outside the door while his wife is with the neighbour. A dog comes and licks his hand. This can be interpreted as a symbol of what the neighbour is doing under the wife's skirts. No symbolic value of the dog has been noticed elsewhere in the movie. Hence this seems to be a 'local symbol'.

boiling world where whistle after whistle dies down'.[39] While there were roses being grown in the love story and rain falling in the movie, there are no kettle whistles in this quotation. They are mere metaphors for alarms. An image, as opposed to a symbol, may be purely local. The use of images in pictorial media is rare, but is sometimes seen. The same is expected for multimedia. An example of an image in a movie will be mentioned. In the Swedish movie "Jag är med barn" (I am pregnant, c. 1980), an iron bar door in a jail closes in front of the main character when he promises to move in together with his girlfriend, who has just become pregnant. This is an image of cohabitation as a prison.[40] The image recurs with some elaboration throughout the movie. With some uses of images, it is made explicit that the image is not to be taken literally. One might expect that this would have to be the case always, or the image would be taken as a symbol instead. If the two examples were to be taken at face value, then there would be kettle whistles in the former and a jail cell in the latter. When they are not perceived this way, it is because it does not make sense in the context to take them literally; so the reader, listener or viewer is forced to find another interpretation (or ignore the phenomenon).

A leitmotif[41] is a recurrent motif used to symbolize for instance a person, a setting (atmosphere) or an event[42]. Leitmotifs were first discussed with music. They are also used for instance in movies[43]. In television, the introductory sequence to each part of a series, especially a signature tune, can be a leitmotif. The difference between a leitmotif and a symbol is that the symbol bears some resemblance to or similarity with that which it symbolizes. In a multimedia system, a visible object or a sound, e.g., a piece of music, could serve as a leitmotif.

[39]Benny Andersen here quoted from Finn Brandt-Pedersen and Anni Rønn-Paulsen, previously referenced work[24], page 155, my translation.

[40]A Humprey Bogart-like figure also appears, which suggests that the idea has been taken in part from Woody Allen's film 'Play It Again, Sam' (1973); but 'Jag är med barn' takes it somewhat further.

[41]German for 'leading motif'.

[42]Gyldendals Tibinds Leksikon[49], sjette bind (volume 6), kve-mum. Gyldendal 1978. (In Danish.), entry on ledemotiv, page 98.

[43]Dan Nissen and Anne Jerslev [88] in the previously quoted work, page 187, give an example where the same sounds (a clock, birds and the wind) are used as symbols in some parts of a movie, and as a leitmotif in other parts of it. A visual leitmotif is imaginable; but no example has been found in the study.

5.4 Discussion

This chapter, after giving sketches of five common literature theories has argued why the extended layer model from the New Criticism has been chosen as the model of a story on which the work with elastic stories is to be based. That model has been explained with emphasis on the composition and the narrator and language issues. The application of the model to stories in media other than text has been found meaningful. Such an application has, however, turned out not always to be non-trivial and has therefore been discussed where it was relevant. The resulting 'extended layer model for non-textual stories' describes the structure of such stories.

A distinction has to be made between two kinds of structure: Some of the structure is explicit in the story, other structure is not. Going back to the layer model (not the extended layer model), the first layers (the graphical layer, the syntactical layer and partly the sound layer) are immediately visible in the text. The later layers (especially the plane structure layer, the big elements layer, the narrator attitude layer and the statement) are only accessible through understanding the content of the text. In the extended layer model, the two kinds of structure are mixed. Many of the concepts are concerned with the kind of structure that is only recognizable through the content of the story.[44]

The intention of the following chapters is to provide tool support for the two kinds of structure in two different ways: the tool should directly support the building of the immediately visible structure. It should also support the building of the content of the story, but should leave the responsibility for the *structure* in that content to the author.

The extended layer model for non-textual stories lays a basis for the work in the next chapter, in which a new model specifically covering *elastic* stories in *computer-based multimedia* will be built. That model will rely on the extended layer model as presented here. The understanding gathered in the work presented in this chapter will influence the new model, where it will appear as requirements for tool support for elastic stories. The new model will explicitly cover those concepts in the extended layer model that are

[44] A semiotician would say that some of the structure is in the *expression* of the text, other structure is in its *content*.

concerned with the immediately visible structure of an elastic story. It will in turn be used as a basis for developing tool support for elastic story writing, in the form of Petri nets.

Chapter 6

A Model of Elastic Stories

This chapter describes the kinds of stories intended to be told using Coloured Petri Nets. It does so by building a model of elastic stories in multimedia, consisting of a number of concepts. This model forms the basis of the work presented in the following chapter, in which each concept will be translated to a segment of a Petri net.

First, section 6.1 explains what an elastic story is.

For the sake of the study, only one style of multimedia interface is under consideration for elastic story telling. This multimedia interface has a big, scrollable background picture with moveable objects on it, with additional windows for pictures and video, and includes sound. It is hoped that the results obtained will be extendible to multimedia with other interfaces. The user interface is presented in section 6.2.

The model of elastic stories in computer-based multimedia follows in section 6.3. The set of concepts making up the model is based on the theory presented in the previous chapter, on the idea of elastic systems, and on the style of user interface used in the work. It is intended that the set of concepts in the model should cover what can be conceived as the basic elements of elastic stories today. One should bear two things in mind: First, the model is restricted to the style of interface described here. Second, future multimedia authors may, and probably will, invent new concepts.

6.1 Elastic stories

Intuitively, an elastic story is an interactive story in which the reader can try to influence the course of events, without any guarantee that he or she will succeed every time. The harder she or he tries, the greater is the probability that the user will succeed. Figuratively, it should be like pulling a rubber band. Elasticity is probably even more prominent from the author's point of view. The author may for instance—still figuratively—tap the user's shoulder and say 'look here, I have got something to show you'. The user may follow ('let the elastic pull him or her') or instead look at something else ('pull in another direction'). To write an elastic story, the author should put an effort into creating an interesting story, which among other things requires some structure[1].

A system that is used to tell an elastic story is a kind of *elastic system*[2]. An elastic computer system in turn is a kind of *elastic medium*. Elastic systems were invented by Peter Bøgh Andersen in his quest for computer rhetoric and aesthetics.[3] Elastic systems form a middle ground between *user-controlled* and *developer-controlled* systems, the two paradigms traditionally used in multimedia and other computer systems.

A user-controlled system is a passive system; nothing happens until the user does something. Examples include traditional database systems, hypermedia

[1]Peter Bøgh Andersen, Jens W. Johansen, Jakob A. Mikkelsen & Morten Sams: Interaktive tekster. In an anthology from Odense Universitetsforlag, in press. (In Danish.)[10]

[2]Three references Peter Bøgh Andersen: Vector Spaces as the Basic Part of Interactive Systems: Towards a Computer Semiotics. In Patricia Baird (editor): Hypermedia, Volume 4, number 1,1992. Taylor Graham.[8] Peter Bøgh Andersen: Katastrophen und Computer. In Roland Posner (journal editor), Martin Warnke & Peter Bøgh Andersen (guest editors): Zeitschrift für Semiotik, Band 16, Heft 1–2. Stauffenburg verlag 1994. Pages 29–50.[5] Peter Bøgh Andersen, Jens W. Johansen, Jakob A. Mikkelsen & Morten Sams, previously referenced work[10].

[3]Peter Bøgh Andersen attributes this use of the term 'elastic' to Hans Peter Brøndmo and Glorianna Davenport. However, the referenced paper by the latter authors does not define the term 'elastic'. The system it describes, the Elastic Charles, is not elastic in the sense in which the term is used here. Peter Bøgh Andersen: Vector Spaces, previously referenced work, subsection 4.4, page 74.[8] Hans Peter Brøndmo & Glorianna Davenport: Creating and viewing the *Elastic Charles*: a hypermedia journal. In Ray McAleese and Catherine Green (editors): Hypertext: State of the Art. Papers from UK Human Interface Interactive Learning Systems SIG conference on hypertext, Hypertext II, University of York, 1989. Intellect, Oxford, England, 1990.[26]

systems, operating system shells and modern GUI programs and tools. Using a user-controlled system can be effective if the user has a specific purpose. If not, such systems are usually boring, and there is a great risk of becoming 'lost in hyperspace', except in very small systems[4].

As extreme examples, a lump of clay and a pile of blank sheets are media that are user-controlled to an extent that makes them uninteresting in themselves (the term 'user-controlled' is used for lack of a better one to denote the opposition to 'author-controlled', even though the author is also a kind of user).

As the other extreme, traditional slide shows and movie films and their computerized counterparts are examples of developer-controlled systems. Developer-controlled systems usually are not interactive. They *can* be, for example certain courseware (programmed teaching) and question-and-answer interface[5]. In developer-controlled systems, it is relatively easy for the author to use timing as an effect and build suspense curves, thus adding to the attraction of the system.

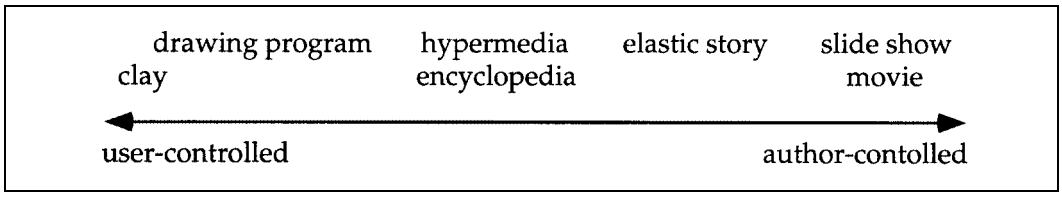


Figure 6.1: Elastic media fill the gap between user-controlled and author-controlled media. Putting the different media on a scale like this is of course an oversimplification. Firstly, users can exercise different *kinds* of control over different media. Hence it is usually open to interpretation which of two media (for instance a drawing program and a lump of clay) is more user-controlled. Secondly, the same medium (especially a computer program) may behave in a more user-controlled way at one time and a more author-controlled way at another time.

[4]About being lost in hyperspace, see, for instance, Edwards, Deborah M., & Lynda Hardman: 'Lost in Hyperspace': Cognitive Mapping and Navigation in a Hypertext Environment. In Ray McAleese (editor): Hypertext: theory into practice, Ablex Publishing Corporation, New Jersey, and intellect books, Oxford, 1989. Pages 105–125.[37]

[5]Question-and-answer interfaces are defined in Peter Beyer et al.: Brugervenlige EDB-systemer. Teknisk Forlag A/S 1988. (In Danish.) Pages 73 and 77.[17]

Elastic systems give both the author and the user some control, but neither of them unconstrained power over the course of events. One aspect of elastic systems, as a contrast to 'rigid' or 'hard' author-controlled systems, is that it should not be possible for the user to pull so hard that the system breaks down and gives no (or decidedly erroneous) results. Any user input should be interpreted as well as possible.

Peter Bøgh Andersen writes about an elastic relationship between user and systems designer:

> " ... They [interactive media] do have an artistic form, but this form is elastic. It is designed to be manipulated within the limits set by the designer.
>
> One of the benefits of the vector concept is that it allows the designer to work with a continuous scale of elasticity, ranging from a 'hard' form where the author is in control (e.g. the Mystery of the Razor of the museum system), to a 'soft' form where the reader rules supreme (the browsing part). However, it is the fine nuances in between that are most interesting in interactive systems (the History of Interpretation).
>
> The systems we have designed by means of the vector concept feel elastic in a very concrete way, so the vectors can probably be seen as one way of realizing this aesthetics of elasticity."[6]

Later he adds:

> "The notion of elasticity applies to interaction in general: ... The [the interactive system's] form should denote exactly that which yields to or resists user interaction. An elastic relationship between user and designer is relevant in all teaching systems and process control systems. In both cases, there is an agent that sometimes should be allowed to control user's options: the author of a teaching system, because he may want to present longer coherent information to the user, and the author of the process

[6] Peter Bøgh Andersen: Vector Spaces, previously referenced work, subsection 4.4, pages 74–75.[8]

control system, because it needs to send a warning about a critical state."[7]

It should be noted that although the author and the user of an elastic story both experience that they are sharing power, it is the author who decides how much power to give to the user. A good author should know how much and which power to give the user for the user to have a good experience.

Elastic systems are not well explored, so it remains to be seen whether they give the best (or the worst) of both worlds or something entirely new. It can be argued that if computer-based multimedia contains a potential for something radically new (as is often claimed), the new is to be found in the area of elastic systems; purely user-controlled and purely author-controlled systems will tend to be mere repetitions of the kinds of systems we already know.

6.2 User interface

In the study of Petri Nets for interactive story telling, concern is restricted to the user interface described in this section. The interface is one that is easy for the reader (user) to use without prerequisites; typically one that would be used in a museum setting where users cannot be expected to be trained in computer use in advance and are not motivated to spend much time learning it. Most of the ideas used in the interface originate from the Wodan's Eye project, though some of them were later changed in that project. Also some of them have been changed here, for instance to make them more general.

The basic element of the user interface is a big background picture, which is typically considerably larger than the computer monitor.[8] The background picture is provided by the author. Part of the background picture is visible

[7]Peter Bøgh Andersen: Katastrophen und Computer, previously referenced work, pages 29-30. The translation from German is inspired by a draft in English made available from Peter Bøgh Andersen.[5]

[8]Examples where this has been used are found in Peter Bøgh Andersen: Vector Spaces, previously referenced work[8], and in Bjørn Laursen: Tegning og Kognition. VENUS Report No. 6. Department of Information and Media Science, University of Aarhus, 1990.[70]

through a window on the screen. The user can navigate (pan) freely over the picture.[9] Panning can also happen from program control, as we shall see later.

In the background picture a number of characters and objects are visible. More precisely, they are visible if within the window. The user can move characters and objects across the background at any time. They can also be moved from program control.

Each character or object can have a number of postures if the author provides a number of different drawings of the same character or object. At most one of them can be visible at a time. The character or object can also be made invisible from program control. The author specifies the character or object's initial posture, and whether it is initially visible.

A number of windows containing still pictures can be opened from program control. They will typically hold close-ups of items from the background. This is useful for studying details of items.

Similarly, a number of windows showing live video segments with or without sound can be opened. When no video is playing in a window, it is closed (not visible).

Finally, sounds can be played through the computer's speaker. A number of sounds can be played at the same time.

The following conditions can be tested from within a program:

- whether a specified object or area of the background picture is visible in the window. The object may be a character or a moveable object. The area can be any fixed part of the background picture (e.g., a house or a road).
- whether a set of specified objects and areas are all visible and beside each other (e.g., whether the maximum distance between any two of them is below some threshold). The objects and areas are the same as in the previous item.

(The latter is a generalization of the former.)

[9] E.g. using a spotlight, see Peter Bøgh Andersen: Vector Spaces, previously referenced work.[8]

6.2.1 An action

In what follows, an *action* is taken to mean an atomic action performed on the user interface from program control. All the actions are what in chapter 4 were called time-based, which implies that they each have a duration. Possible actions are:

- Moving an actor (a character or an object) on the background picture. The author provides a target co-ordinate set for the movement, either relative to the background or relative to the current position of the same or another actor. The movement is animated, so the user experiences a linear sliding from the original to the target position.
- Moving the user's view of the big background picture. This is also known as a panning. As with moving an actor, a linear sliding to the new position is performed. The author provides either a target coordinate set or the name of an actor that is to become visible by the panning. Panning can be regarded as moving the user's eye over the background picture.
- Playing a designated sound segment (a speech, an effect sound or some music).
- Playing a video sequence in a separate window.
- Displaying a picture in the close-up window for an amount of time. Displaying a picture can be seen as a special case of playing a video sequence in which the video has only one frame in it and thus does not move.
- Changing the posture of a character or object. The change can be fading or animated. The author decides the duration. If a sudden change is desired, a zero duration can be specified. Animation requires that intermediate postures are provided.
- Making a character or object visible or invisible. Again, a fading can be used, and the duration can be zero if desired.

When talking about the author above, this can refer to any persons involved in the development of a multimedia program.

6.3 Concepts: story structure and requirements for tools

This section defines a model of elastic stories in multimedia as consisting of the following concepts:

- An event (not related to a MacEnv event described in section 4.1.2)
- A thread or story line
- Parallelism
- Synchronization
  - Inter-event synchronization, including a generalization
  - Sub-event synchronization
  - A resumption
  - Intra-event synchronization
- A branching and a connection
  - A fork (a kind of branching)
  - A join (the corresponding kind of connection)
  - A choice (another kind of branching)
  - A merging (the corresponding kind of connection)
- Non-determinism
- A pause

The reader will notice that not all the elements of multimedia narratives as found in the extended layer model have direct counterparts in the above model. One should remember that many of the concepts in the extended layer model, including for instance all the language issues, depend entirely on the *content* of the presentation, and hence are the sole responsibility of the author. Multimedia authoring tools should support the building of the content that the author wants to create rather than giving direct support for individual language issues. Therefore the latter are not explicit parts of the model presented in this section.

6.3.1 An event

An event is a set of actions that are meant to happen at the same time. Popularly speaking, an event is a way to order actions *beside* each other in time. When an event happens, *all* the actions happen. They will normally be executed simultaneously. (We shall see later that this is no requirement.) Since actions take time, the execution of an event extends over some amount of time.

Example: One line of a story can be represented by an event consisting of two actions: moving the view to the character speaking so she or he is visible (if she or he was not already), and at the same time playing the relevant speech.

6.3.2 A thread

According to the extended layer model, a story is composed of sections and sequences. Sequences are realized through threads, also called story lines. A thread is a linear sequence of events and other threads making up a piece of a story. It follows that threads can be nested. A system of nested threads can model an arbitrarily deep hierarchy of sections, subsections and sequences. A thread in which the events are lines[10] may model a dialogue. Similarly, the other scenic forms of presentation in the extended layer model can be modelled by threads. Popularly speaking, a thread is a way to order events *after* each other in time.

Example: If a coherent discourse in a story is not affected by interaction, it will be naturally represented as one thread, one sequence of events.

6.3.3 Parallelism

It was noted in subsection 5.3.1 (page 80) that in interactive media, sections need not be ordered linearly. To overcome linear ordering and to be able to build interesting plots, an author may let several threads take place *in paral-*

[10]Lines or speeches or text. This chapter reserves 'speech' for describing the sound of a line; therefore 'lines' is used here.

lel. Parallel threads form the antithesis to linear ordering in that the former may exist with no ordering between them at all (the following subsections will, however, present ways to introduce a degree of ordering).

Also in linear stories, sequences may happen in parallel in narrated time, though they are usually told in some order in narrating time. In linear stories, the author specifies the exact order of the sequences in narrating time, the exact points where cutting occurs, etc. Specifying the story in parallel threads relieves the author from having to specify when the switching (cross-cutting) occurs; the system (often helped by user interaction) takes care of this.

Examples: A story may switch back and forth between describing the actions of the villain and the actions of the victim; actions that happen in parallel until the two finally meet. By modelling the story as two parallel threads, the author is relieved from specifying when the cross-cutting is to happen.

As a different example, in a multimedia system it may also simply be that there is more than one story to tell. Let us assume there is a story to tell about each of a set of the objects on the background. Each story may be started when the object becomes visible in the window to the background picture. When a number of stories have been started, they go on in parallel.

6.3.4 Synchronization

When dealing with parallel threads, the author will sometimes want to impose weaker or stronger *synchronization* between the threads; in other cases she or he will not want to synchronize them at all. In case synchronization is needed, it may be done between events between actions from different events, or inside one event. The three kinds of synchronization are described in the next three subsections, where examples are also given. The different kinds of synchronization can be combined in the same story.

6.3.5 Inter-event synchronization

Synchronization on the coarse level, between events, is called inter-event synchronization. It should be possible to specify that an event only happens

after another specified event has happened, or only after a specified number out of a set of specified events have happened.

Examples: The author may prohibit the villain from entering the victim's house until he or she has reached a certain point in her thread.

A good way to introduce abstract ideas is by means of examples. A multimedia author may take advantage of this fact for instance in the following way: After the user has met three examples of items that the Vikings bought or sold (or both), the system may offer some general information on trade in the Viking age. After four or five examples have been met, some more general information may be added. In general, if a presentation contains a number of examples of some general idea, the system may introduce the abstraction explicitly after the user has met some of the examples. An event serving this purpose is called a *generalization*. Inter-event synchronization makes sure that the generalization is activated only after the specified number of *examples* have been met.

6.3.6 Sub-event synchronization

It should be possible for the author to specify synchronization between actions of different events. In general, the author can specify classes of actions that cannot occur concurrently; then the system should ensure that only one action from each such class is executed at a time.

Examples: It may be desirable to avoid playing speeches from multiple concurrent events on top of each other, since the result could be incomprehensible. Sub-event synchronization could also be used to avoid playing two pieces of music at the same time, since this would sound ugly. An author may for instance specify that at any given time, at most one speech, at most one piece of music, and any number of effect sounds can be played.

Also two or more actions moving the same object or character over the background should be synchronized using sub-event synchronization. (Moving *different* objects can happen concurrently.) The same holds true for parings, since the background cannot be paced in two different directions at the same time.

6.3.7 A resumption

A way is needed to help the user to return mentally to a thread once it has been delayed by sub-event or inter-event synchronization with events in other parallel threads. This is done using a supplementary thread, called a resumption; this is only executed if a certain amount of time has elapsed after the previous event in the original thread. If no or little time has elapsed (below some threshold), the resumption is skipped.

One may ask where to insert resumptions in threads. Having them everywhere is clearly impractical; for instance, there is hardly any point in having resumptions inside resumptions. A first answer is that an author may insert them where threads are interrupted in practice and the thread is not easily picked up after the interrupt. Subsection 11.1.1 discusses more radical solutions to the question.

Example: Here is a simple dialogue that is coherent if told without interrupts:

A: How are your swords?

B: They are the best swords money can buy.

A: Where do you get them from?

B: We buy them in the Netherlands.

However, consider the case where other threads take over the user's awareness in the middle. In this case, two lines can be added to the dialogue using a resumption, e.g., like this:

A: How are your swords?

B: They are the best swords money can buy.

(Twenty seconds of events from other threads, not related to swords)

A: Those swords ...

B: Yes?

A: Where do vou get them from?

B: We buy them in the Netherlands.

6.3.8 Intra-event synchronization

The introduction of sub-event synchronization raises tne question: given that one action of an event (say, the speech) is delayed by the sub-event synchronization mechanism, should the other actions of the same event await it? Ideally, the author should be allowed to specify whether she or he requires all the actions to start at the same time or not. Such a requirement, if specified, imposes synchronization on the fine level, inside one event. It is therefore called intra-event synchronization.

Examples: In an event consisting of playing a speech and showing a close-up of the object described by the speech, the author may not want the close-up to appear before the speech starts. As a different example, in an event consisting of panning to a person and playing a speech by that person, the author may prefer that the panning starts as early as possible, even though the speech may be delayed by sub-event synchronization with other speeches.

6.3.9 A branching

In subsection 5.3.1 on composition (page 80), it was noted that interactive media often have other ordering schemes besides linear sequences. Thus, an interactive story can *branch* in two ways. The two kinds of branching are called a *fork* and a *choice*, respectively. They are described in the following subsections, where examples are also given. After a branching, the resulting threads may later meet again and be connected into one.

6.3.10 A fork

A fork is the kind of branching where one story line or thread of the story branches into two or more parallel story lines or threads. You may say that a fork is a kind of branching where *all* directions are taken.

Example: Consider the old story about the three sons who leave the home in three different directions. In the beginning the sons are at home, and the story is most conveniently told as one thread. At the point where they leave, a branching happens into three threads.

6.3.11 A join

A join is the opposite of a fork; a set of specified parallel threads are merged into one. The execution of the one thread after the join can only continue when all the threads are completed up to the join. Popularly speaking, the threads 'wait for each other'. Note that since a set of threads can exist from the start of a story, a join is possible between threads that do not stem from a previous fork.

Examples: If the three sons mentioned in the previous subsection meet again, a join happens. The remainder of the story cannot start until all three threads have arrived at the join. As another example, when the villain and the victim mentioned under parallelism meet, a join happens between threads that may have been separate from the outset.

6.3.12 A choice

At a certain point, there is a set of possible directions the story can take. Typically, the choice of direction will depend on user activity, for instance whether some object in the interface is visible in the view or not. As was the case with a fork, the branching may be described as one story line branching into two or more. The difference is that the choice is the kind of branching where exactly one direction is taken[11]. If exactly two threads result from a choice, they can be talked about as *alternative* threads.

Example: Assume that the author has three different stories to tell about three different objects located in different places in the background picture. Also assume that one person occurs in all three stories, so that it is not meaningful to tell more than one of them at a time. The author would realize this by building the three stories into one that starts with a choice among the three threads. The choice of thread could depend on which of the three objects was (or first became) visible in the interface. In this way, the user would effectively be allowed to decide.

[11] In an early example of an interactive story, 'Dage med Diam', the interaction is realised by user-controlled choices. (This interactive story is published in a book, not in computerized form.) Svend Åge Madsen: Dage med Diam *eller* Livet om natten. Gyldendal 1972.[78] (In Danish.)

6.3.13 A merging

After a choice, the threads may lead into completely different directions and thus to different endings of the story. Alternatively, the choice may be local and the story continue as one thread after a while. For the purpose of the latter possibility, a *merging* is needed. As a middle ground, some of the threads resulting from the choice may merge, while others stay separate. It may also be that a merging between threads not stemming from a previous choice could be meaningful. A merging is different from a join in that no waiting takes place.

Examples: In the above example, the three threads may merge again in the end and then return control to the choice at the beginning, realizing a loop where one of the stories is told in each cycle.

As a different example, say that the plot of a story (or just the plot of one of the threads of the story) requires that a person gets killed, but it does not matter whether he or she gets shot or stabbed. If both versions are interesting, the author may provide a choice between them. After the person is killed, the two resulting threads are merged back into one, so the story can go on in the same way in both cases.

6.3.14 Non-determinism

To enhance the non-linearity of interactive media discussed earlier, the author may make a choice *non-deterministic*. Using non-deterministic branching, if a story is told more than once, the outcome may be different, even if user actions are the same every time. A random choice realizes this.

Examples: In the above loop, say that two of the three relevant objects can be visible at the same time. Then there would be two threads to choose from. A random choice can solve the conflict. As another example, in Wodan's Eye, a multimedia system in the Viking age museum in Ribe[12], variants of the

[12]Wodan's Eye is a multimedia system recently installed in the Viking age museum in Ribe, Denmark. The Wodan's Eye project is described in two reports: Helle Juel Andersson, Lars Andersen, Berit Holmqvist, Bjørn Laursen, Peter Bøgh Andersen & Stig Jensen: Ødins 0je. Rapport 1. Department of Information and Media Science, Aarhus University and The Antiquarian Collection in Ribe. (Undated, approximately 1993.)[13]

same story are built using choices and subsequent mergings. A person may for instance choose to accept or reject Christianity in different variants of the same story. Non-determinism may be a way to choose different variants each time the story is told.

6.3.15 A pause

A *pause* in a thread can encourage user activity. The first time the user sees an object, something is told about it, but not all there is to tell. Every time the user returns and looks at the object, new information is given[13]. A way is needed to specify that an event in a thread can only take place after some interface object has been invisible and becomes visible.

A pause can also be less than this, by only waiting for an object to be visible *or* invisible. More precisely, three kinds of pauses exist:

1. A pause that blocks a thread until some object has been invisible and becomes visible. 'Visible' means inside the user's view of the background.
2. A pause that blocks a thread until some object is invisible. If it is already invisible, no blocking happens.
3. A pause that blocks a thread until some object is visible. If the object is already visible, no blocking happens.

A pause of the first kind is equivalent to the sequence of one pause of the second kind and one of the third kind. Since the first kind is expected to be the common one, it is convenient to regard this as one pause, not two.

Example: In the above loop, one of the threads may be concerned with some swords visible in a Viking ship. To encourage the user to explore other parts

Helle Juel Andersson, Lars Andersen, Berit Holmqvist, Bjørn Laursen, Peter Bøgh Andersen, Stig Jensen, Thomas Østergaard, Evert B. Hassink & Steffen Sacher: Odins Øje. Rapport 2. Department of Information and Media Science, Aarhus University and The Antiquarian Collection in Ribe. (Undated, approximately 1994.) [14](The reports contain contributions in Swedish, Danish and English.)

[13]Peter Bøgh Andersen, Jens W. Johansen, Jakob A. Mikkelsen & Morten Sams,[10] previously referenced work.

of the ship too, the sword story may only continue if the user moves away from the swords and back at intervals dictated by the author. More precisely, the author inserts pauses at specific points in the thread. Events after a pause may add to and elaborate on information given before the pause.

6.4 Discussion and summary

6.4.1 Comparison with programming terms

For a reader with a background in programming, a comparison with programming language concepts may clarify some of the concepts defined in the previous subsections, or just reassure the reader that he or she has understood them the way they are intended.

You may say that an event corresponds to a statement in a computer program. A story line or thread corresponds to a subroutine containing a sequence of statements in an ordinary program. Having threads inside other threads is a kind of modularization. Parallelism is the same as in concurrent programming. There is however one difference: In most concurrent programming languages, there is only one thread executing from the start. An interactive story can conceptually contain a set of threads all active from the start. A fork and a join are similar to respectively a fork and a join in a parallel program[14]. A choice is like a case branching in a programming language. Thus, conditional execution is possible using a choice between one non-empty and one empty thread.

6.4.2 Summary

This chapter first presented the concept of an elastic story. It then presented the style of user interface for elastic stories that will be used in the work in this and the following chapters. The greater part of the chapter was the definition of a set of concepts making up a model of elastic stories in the

[14]See for instance David A. Watt [112] (with contributions by Willis Findlay and John Hughes): Programking Language Concepts and Paradigms. Prentice Hall. Chapter nine: Concurrency, by William Findlay. Page 170.

described interface. This model is based on three pillars: the extended layer model from the previous chapter, the idea of elastic stories, and the user interface.

Chapter 7

Elastic Stories in Petri Nets

This chapter develops translations of the concepts from the previous chapter into Petri net representations. The reader will remember that chapter 6 developed a set of concepts to cover elastic stories in a certain style of multimedia user interface. That set of concepts relied on the user interface, on the idea of an elastic story and on the extended layer model, as applied to non-textual stories in chapter 5. The purpose of translating to Petri nets is two-fold: to give a more precise semantics of the concepts and to present the idea that Petri nets are well suited for implementing them. The former is done here. Complete implementation, however, requires an underlying system that executes the Petri net, including handling the multimedia interface. Today programs exist that can execute Petri nets; but they do not have multimedia capabilities. This chapter demonstrates how to uses the program Design/CPN from Meta Software to implement and run elastic stories using Coloured Petri Nets (CPN), but currently without multimedia. Design/CPN also has limited facilities for input. It is hoped that these limitations will be removed in the future. No problems are foreseen in developing Petri nets programs that can handle a multimedia user interface required to realize the ideas developed in this chapter.

This chapter and the next assume familiarity with Coloured Petri Nets at least corresponding to chapters 1 and 6 in the first volume of Kurt Jensen's work 'Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical

Use.'[1]

In some cases, the translation of a concept to a Petri nets is straightforward and therefore presented in a terse manner. Other cases need a more detailed explanation.

The presentation is illustrated by examples of Petri nets throughout In those examples, the nets look as they would in a real situation, except that guards and code regions are most often given in pseudo-code. For implementation using Design/CPN, the guards and code regions would have to be written in CPN/ML, an extension to Standard ML.

Many of the concepts do not require that the Petri nets used are *Coloured* Petri Nets. In other words, most of the tokens, places and arcs in the nets have type *unit*, the type of the token that carries no data (the 'plain' or 'colourless' token, popularly speaking). In the Petri net diagrams shown in this chapter, arc inscriptions that evaluate to one token of type unit will be left out.

In the Petri nets, ML functions are used freely in code regions, guards and arc expressions. Some of these functions are not functions in the mathematical sense of the word, yielding at most one output for a given input (e.g., ML functions that deal with the user interface or real time). This means that the nets do not satisfy the formal definition of Petri net[2]. This is not a problem since the intention of using Petri nets is not the application of the formal analysis tools offered by formal Petri nets.

The concepts will be presented in a bottom-up order, starting with an action and event. The rest of the concepts will be presented in the same order as in the previous chapter, except that intra-event synchronization is presented before resumptions.

[1]Kurt Jensen: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume I: Basic Concepts. EATCS Monographs on Theoretical Computer Science. Springer-Verlag 1992[62].

[2]E.g., Kurt Jensen, [62], pages 70 and 107-108.

7.1 An action

Two transitions in a Petri net are used for dealing with an action: one for starting it, and one marking its termination. The figure on the next page shows the scheme.

A token on the place 'Playing' in the middle represents the action being executed. The transaction starting the action puts a token on the place. It has a code region with some code to initiate the execution of the action. The transaction representing the termination of the action removes the token again. It has a guard attached to it preventing it from firing before the action is finished.

There is a deliberate asymmetry in the scheme: the code in the transition starting the action is in a *code region*, while the code of the transition representing its termination is in a *guard*. This is due to the conditions: the net controls when the action starts. It does not control its duration, hence not when it stops (in which case a code region with a stop-playing statement would have been required).

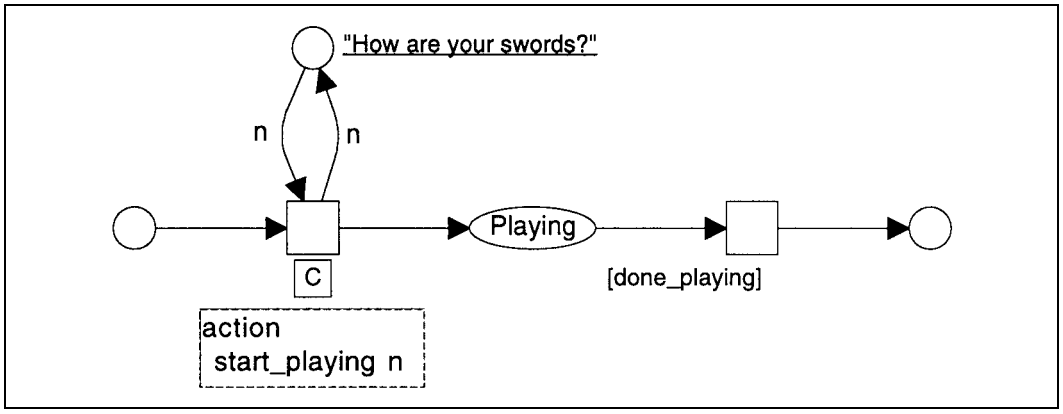


Figure 7.1: Executing an action, e.g., playing a sound, is done using two transitions with a place between them in the Petri net.[3]

[3]The program Design/CPN has been used to produce the Petri net diagrams used as figures in chapters 7–9 and appendix A. Colour sets are left out from the diagrams; it should be possible to infer them from the context. Hence, the topmost place in this figure has a colour set of sound names, while the other places have type unit. As mentioned, where no arc inscriptions are present, one token from the colour set unit is assumed.

Often the transition starting the action, and possibly sometimes the one representing its termination, will need some data about the action, e.g., the name of the sound to be played, or the name of the actor to be moved and its new co-ordinates. The author provides such data as colours on a token on a separate place. In the figure the topmost place is used for this. The data is given in the form of an initial marking of that place. The initial marking shown in the figure could be the name of a speech. me transaction starting the action takes the coloured token and returns it to the same place for future use. In case the termination transition needs data, it is passed on the token on the Playing place.

Section 7.6 will elaborate on the implementation just given of an action.

7.2 An event

The behaviour of an event is described on a separate subpage in a hierarchical Petri net. The subpage contains representations of the actions contained in the event. Since the actions are to be executed in parallel, a transition (labelled 'Distribute' in the figure) distributes one token to each action. In the end, one transition (labelled 'Collect') takes the tokens coming out of each transaction and marks the conclusion of the entire event. An example is shown on the next page.

Often the author will want the actions of the event to be synchronized. Section 7.7 will introduce intra-event synchronization into the implementation of an event just shown.

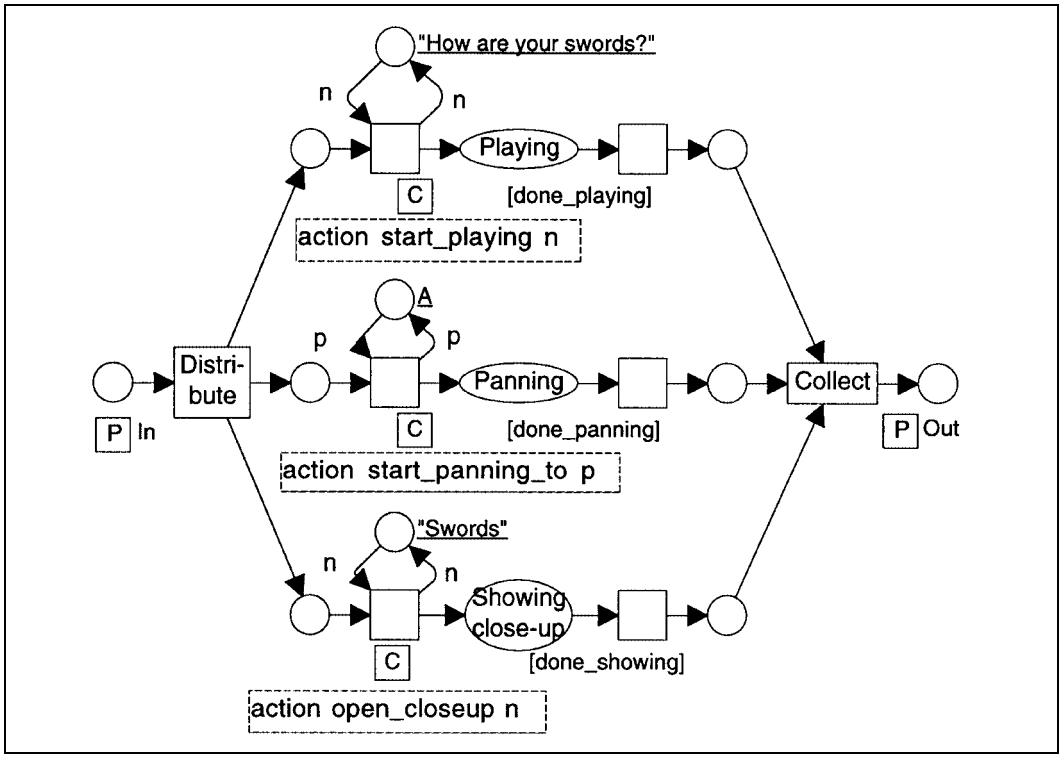


Figure 7.2: An event consisting of three actions: playing a speech (top), panning to person A (middle) and a showing a close-up picture (bottom).[4]

[4]In the figure it is assumed that the close-up picture shown, like the other actions, has a built-in duration, so it makes sense to test whether it is 'done'. This conforms with the earlier note that showing a picture can be seen as a special case of showing a digital video, which obviously has a duration. In case the assumption breaks, the author will have to specify a duration for which to show the picture. This is done in two steps: (1) The guard is changed from [done_showing] to a condition specifying that the appropriate time has elapsed. This involves using the tod() function to generate a 'time stamp' on the token placed on the place 'Showing close-up'. The tod() function Ed time stamps are presented in section 7.8. (2) A code region is added to the same transition to actively close (hide) the picture.

7.3 A thread

A thread is represented by a linear sequence of sub-pages, separated by places, as shown in the topmost figure on the next page[5] This marking is not present if the thread is part of another thread. The label 'HS' (hierarchy substitution). Each subpage can be an event, as described above, or contain its own thread. In the example in figure 7.3, the underlined text W under the first place denotes the initial marking of the net: one token is placed on that place so the thread executes from the start.[6] under each box means that the box stands for a subpage rather than a transition.

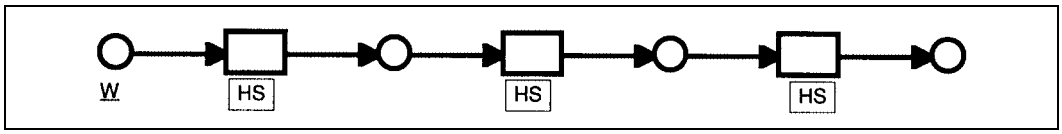


Figure 7.3: A thread is represented by a linear sequence of sub-pages.

7.4 Parallelism

Parallel threads are represented by different sequences of events. If no synchronization is required, the threads are completely independent:

[5]Throughout the chapter, threads are drawn with thick lies, everything else with thin lines.

[6]W is really a shorthand for 1'W: the multiset having exactly one element, W, in it. The declaration color E = with W is assumed, that is, W denotes a token of type unit.

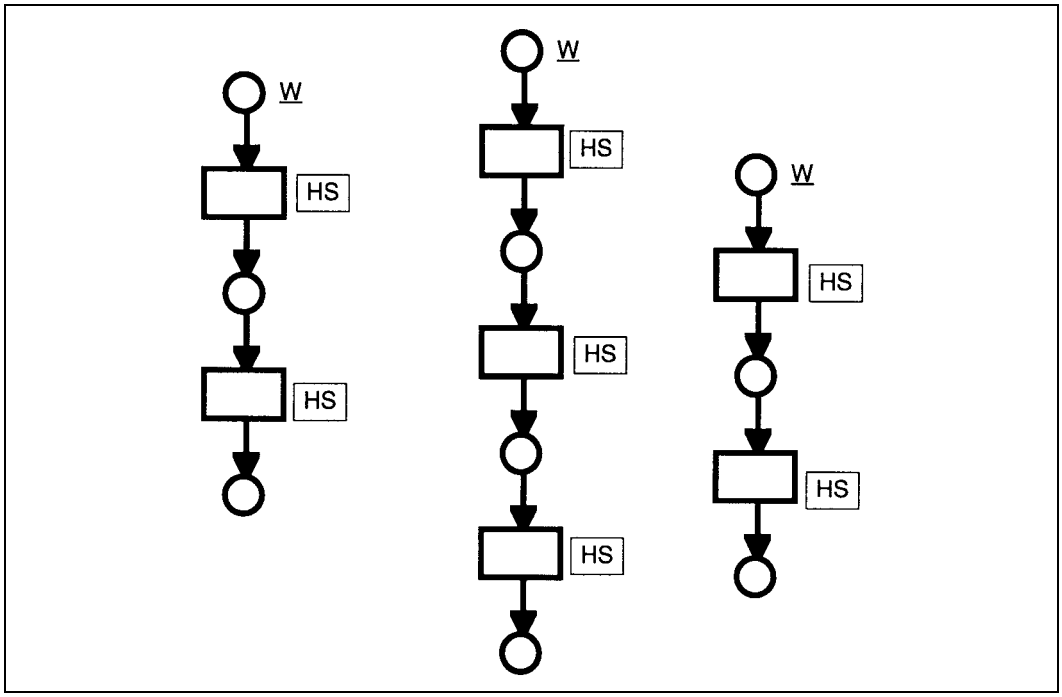


Figure 7.4: Three parallel threads.

Most often in a practical situation, each thread would be drawn on its own page (often a prime page), to enhance modularity.

7.5 Inter-event synchronization

This section describes how to implement inter-event synchronization, including generalizations.

As an introduction, an over-simplified solution to inter-event synchronization is given first in the figure on the previous page: a place is inserted between the two events to be synchronized. The event to occur first puts a token on that place. The event to occur afterwards takes the token and puts it back. In this way, if the thread containing the second event is executed more than once, it is still recorded that the first event has happened, so the thread can continue. The inserted place is called a *synchronization place* in the following. Intuitively, one might expect this solution to be sufficient: as long

as there is no token on the inserted place, the second event cannot occur.

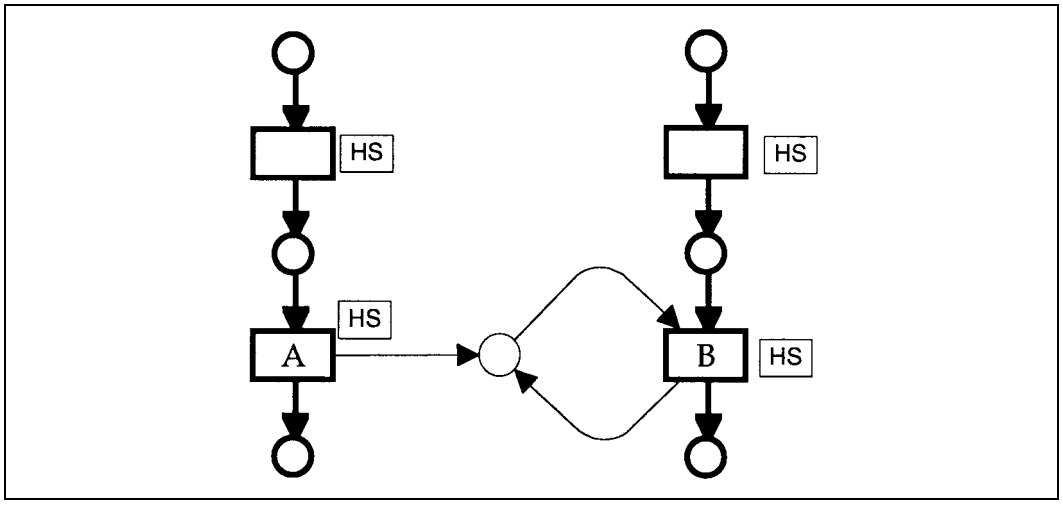


Figure 7.5: The intuitive idea used for inter-event synchronization: a synchronization place is inserted between events A and B so that event B can only occur after event A has occurred. The idea is further developed in figures 7.6–7.8.

A number of changes are made to the scheme. The scheme would require changes to be made inside events A and B. To avoid this, extra transitions are inserted. One transition is inserted after event A to put a token on the synchronization place. To avoid blurring the picture of the original thread with these synchronization details, event A and the inserted transition are moved to a separate subpage (called A' in figure 7.6 and shown in figure 7.7). Similarly, on a separate subpage B', a transition is inserted before event B to inspect the token on the synchronization place. Finally, the synchronization place is replaced by a global fusion set. This makes the same conceptual place accessible from within both subpages A' and B'. In figures 7.7 and 7.8, the members of the fusion set are marked FG for 'fusion global'.

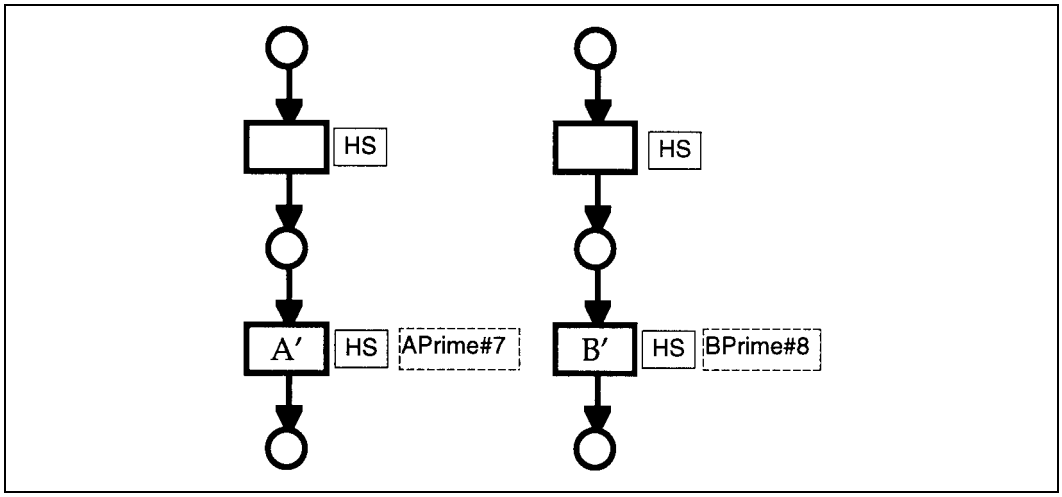


Figure 7.6: Inter-event synchronization: In the original threads, the events A and B are replaced by subpages A' and B'. The contents of A' and B' are shown in figures 7.7 and 7.8, respectively.

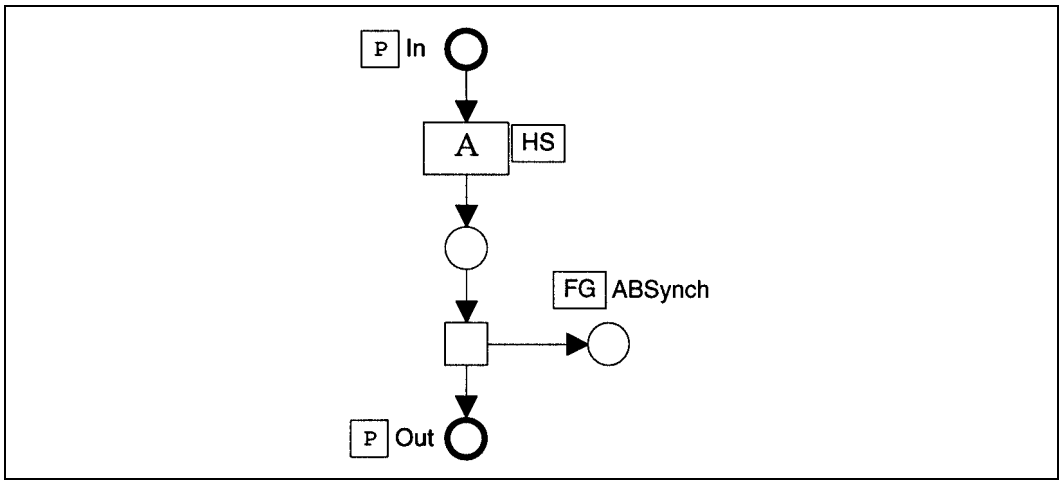


Figure 7.7: The contents of A' from figure 7.6: after event A a transition is inserted that puts a token on the synchronization place. The synchronization place is a a global fusion place (ABSynch in the example).

In case the author will not allow B to happen more than once after A has occurred only once, the transition preceding B can 'keep' the token instead of returning it to the synchronization place ABSynch, as shown in figure 7.9.

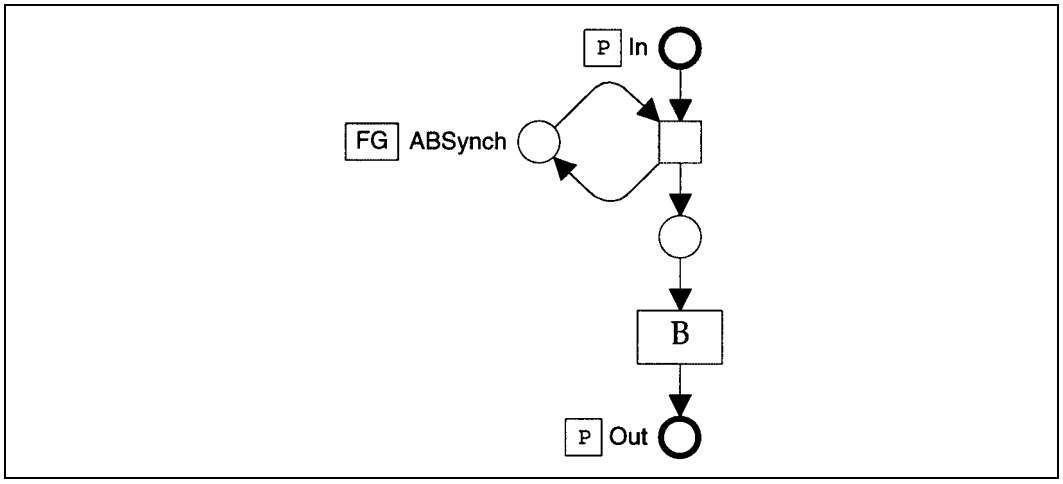


Figure 7.8: The contents of B' from figure 7.6: a transition inserted before B takes a token from the synchronization place. If no token is present, the thread is blocked.

This option represents a higher degree of flexibility in the Petri net than in the concept of synchronization as presented earlier.

It may be obvious that if a number of events (say, B1, B2, and B3) are each allowed to occur after A has occurred, each of them can be handled as B in the foregoing. The case where B is only allowed to happen after a number of other events (say, A1, A2 and A3) have all occurred is treated as a generalization, see below. If B is allowed after some event(s), *and* some event(s) are only allowed after textsfB, transitions should be inserted both before and after B in B', to handle all the synchronization.

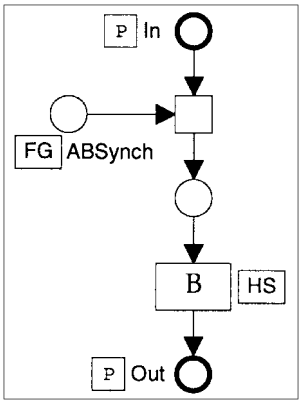


Figure 7.9: Alternative contents of B' which only allows B to occur once after each time A has occurred.

7.5.1 A generalization

The solution to a generalization is presented in the two figures 7.10 and 7.11. The idea is the same as used for other interevent synchronization above: a global fusion place is used for the synchronization.

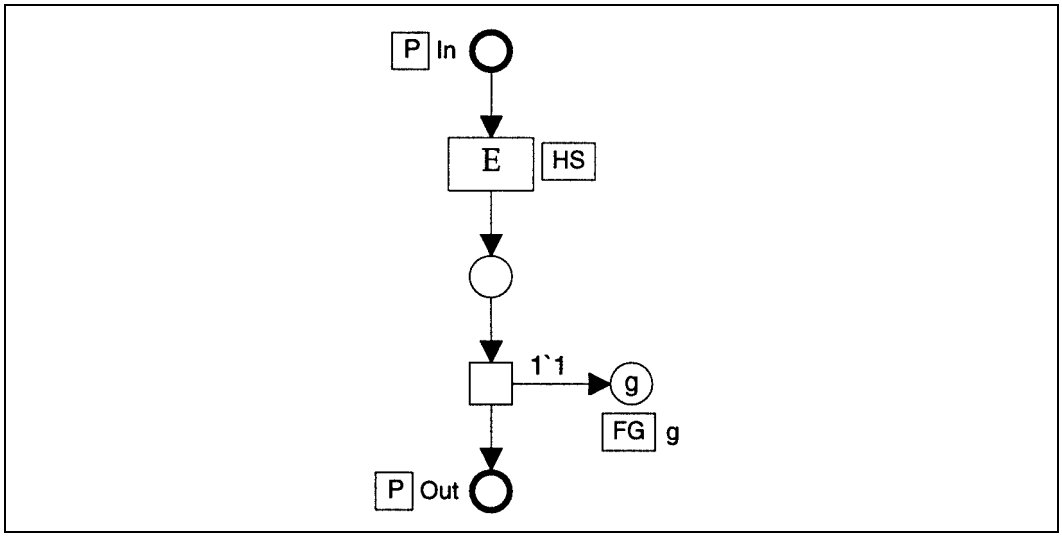


Figure 7.10: Each example used in a generalization is moved to a separate subpage on which it is succeeded by a transition that places a coloured token on the synchronization place, the colour representing the example that has just been presented.

Each example is handled in the same manner as the event A above, with one difference: a *coloured* token is placed on the synchronization place (g in the figures), each example using a unique colour. The synchronization place has the colour set

```
color allExamples = int with 1..max declare ms;
```

The 'declare ms' in the declaration has the effect of creating a multiset with the same name as the colour set, having one instance of each colour in it. Each generalization takes a multiset of tokens from that place, as shown in figure 7.11. A generalization to occur after, say, two examples is given this guard [requireDiff 2 ex] where ex is the multiset of tokens taken from the synchronization place and the function requireDiff is declared like this:

```
fun requireDiff n ex = (size ex >= n)
                       andalso (ex <<= allExamples);
```

<<= denotes the subset relation between multisets. The guard given above ensures that the generalization only happens if ex has at least two elements

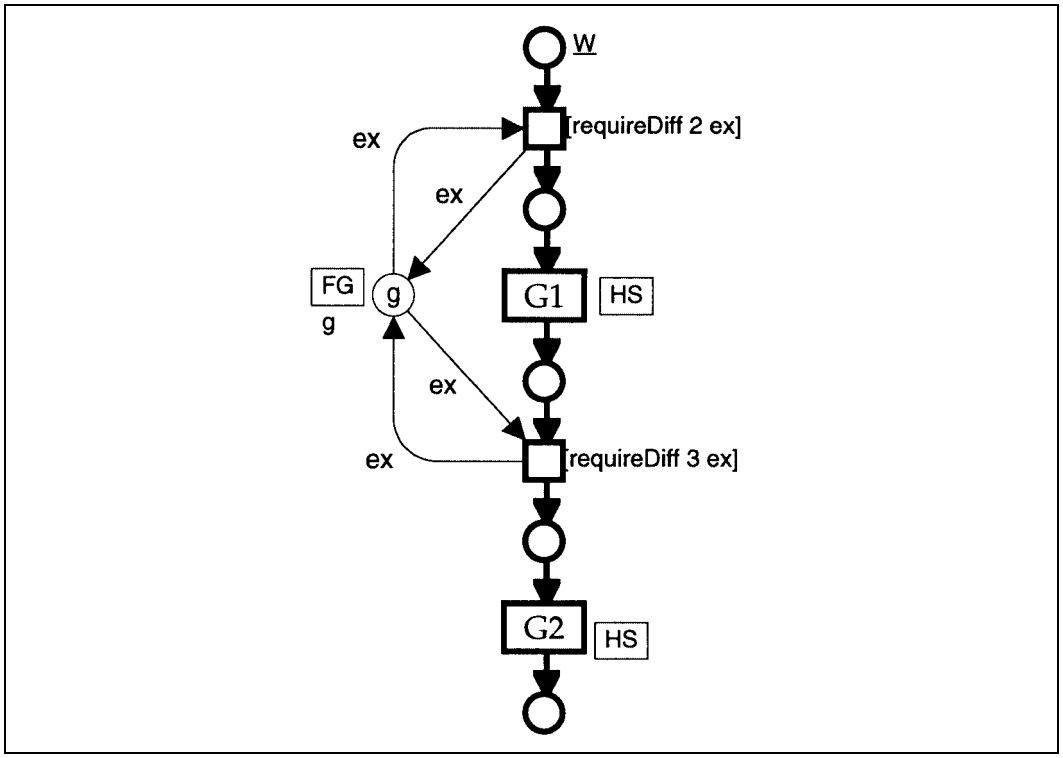


Figure 7.11: A generalization G1 over a number of examples can only be triggered after at least two of the examples have occurred, its continuation G2 not until three of them have. The generalization is an example of inter-event synchronization.

in it and the elements are different, and hence two different examples have occurred. Similarly, the second generalization in the figure (G2) needs a multiset with at least three different elements.

The scheme obviously generalizes to any number of subsequent generalizations (not only two as shown here).

At any time, the place g contains as many tokens of each colour as there have occurred instances of the corresponding examples. This is ensured by connecting an arc from the transition inserted after each example to a place in the same fusion set. A transition before each generalization takes from g a multiset of different tokens corresponding to the number of examples the

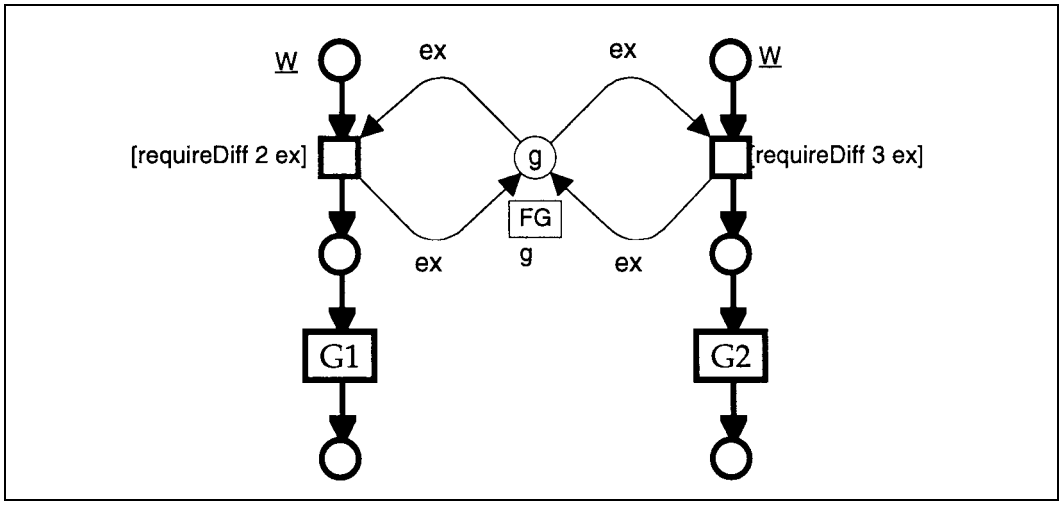


Figure 7.12: Generalizations G1 and G2 can occur in any order.[7]

author requires before triggering that generalization. It returns the same multiset to g.[8]

If the order of generalizations is of no significance, they need not be part of the same thread, as they were in figure 7.11. The diagram in the figure on the previous page shows how the thread can be split.

An ML programmer will of course be able to specify other and more complex conditions on the set of examples to occur before a given generalization. This might be useful, for instance if some of the examples say more than others. As a different example, if the author assumes that repetition promotes understanding, he or she may not want to require that the examples be all different. In that case, he or she can change the guards on generalizations to [size ex $>= n$], where n is the number of examples required. The result is that each example is counted as many times as it has occurred.

[7]In figure 7.12 G2 can occur before G1, since it is not guaranteed that G1 will happen immediately once there are two different tokens on the place g.

[8]Returning the tokens is only necessary in the case of more than one generalization using the same place, as in figure 7.11. However, for the sake of modifiability, it is recommended to make it a habit to do so.

7.6 Sub-event synchronization

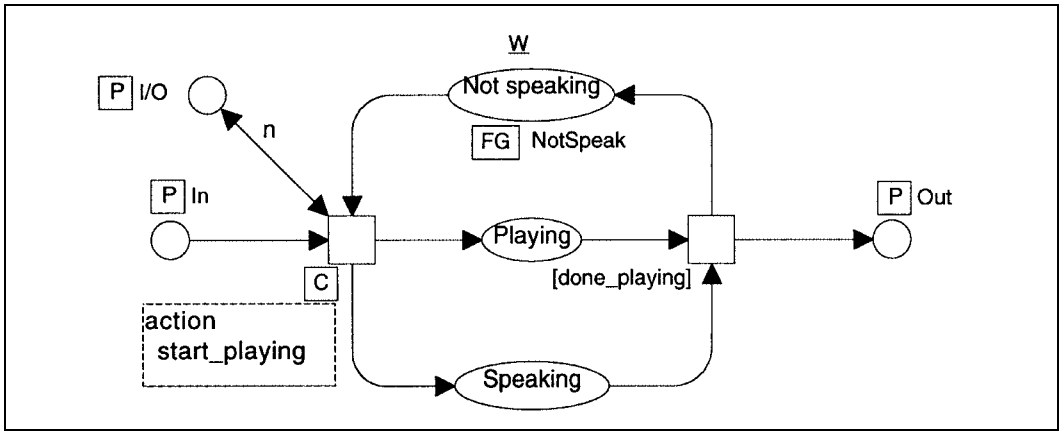


Figure 7.13: A 'speak module'. A fusion place with initially one token on it ensures that only one speech is played at a time.

To prevent more than one speech from playing at a time, the action handling scheme presented in section 7.1 and figure 7.1 is extended as shown in the next figure. The 'speak module' plays the role of a monitor in concurrent programming. It is conveniently located on its own page, as indicated by the port nodes to the left and right in the figure. The speak module would be used as shown in figure 7.14. In figure 7.13, the idea is that a global fusion set NotSpeak contains one tokeninitially, representing that no one is speaking. Metaphorically, one can imagine a resource, a single speech channel represented by a token. One can then think of the place as the shelf where the speech channel belongs when not in use. To play a speech, the module takes that token (the speech channel resource) and moves it onto the place 'Speaking', thus ensuring that no one else can start speaking. On completion of the speech, the token is returned, thus allowing another speech to start. For the time being, the NotSpeak fusion set could be a *page* fusion set. We shall see in the next section why a *global* fusion set is preferable. (The place 'Speaking' is not technically necessary, but is included for readability and consistence with the metaphor.)

To indicate which speech is to be played, the author provides on a separate place a token the colour of which is the name of the sound (e.g., file name, as a text string).

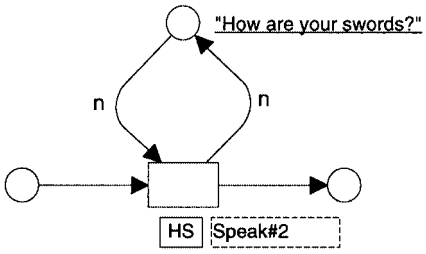


Figure 7.14: Use of the speak module in figure 7.13 from within an event is straightforward. The name of the speech is provided as the colour of the token on the top place.

The author builds a duplicate of the speak module for each class of actions that should not or cannot happen simultaneously (music playing, panning, actor movement, etc.). The modules work independently, so actions using different modules (e.g., a speech and a piece of music) can play simultaneously. Actions that use the same module cannot (e.g., two pieces of music). For actions not involved in sub-event synchronization (e.g., effect sounds) the scheme in figure 7.1 is used unaltered, which will allow more than one such action to happen at a time.

7.7 Intra-event synchronization

Ensuring that a set of actions are executed at the same time requires one transition with a code region that starts all the actions. In figure 7.15 on the next page is depicted an event with three synchronized actions: one is playing a speech, one is panning and one shows a close-up. In this example, all three actions are made to happen at the same time. The principle could also be used to synchronize only some of the actions in the same event. The author can even build groups of actions, where actions in the same group are synchronized with each other, but not with actions in other groups or outside the groups. This is done with one starting transition for each group plus one starting transition for each action not belonging to a group.

It should be mentioned that there is no guarantee against starvation. Say that some events in the system pan, but do not speak, while others speak, but do not pan. A combination of such events may keep the event shown here from ever starting execution. In an elastic story, this is probably no big

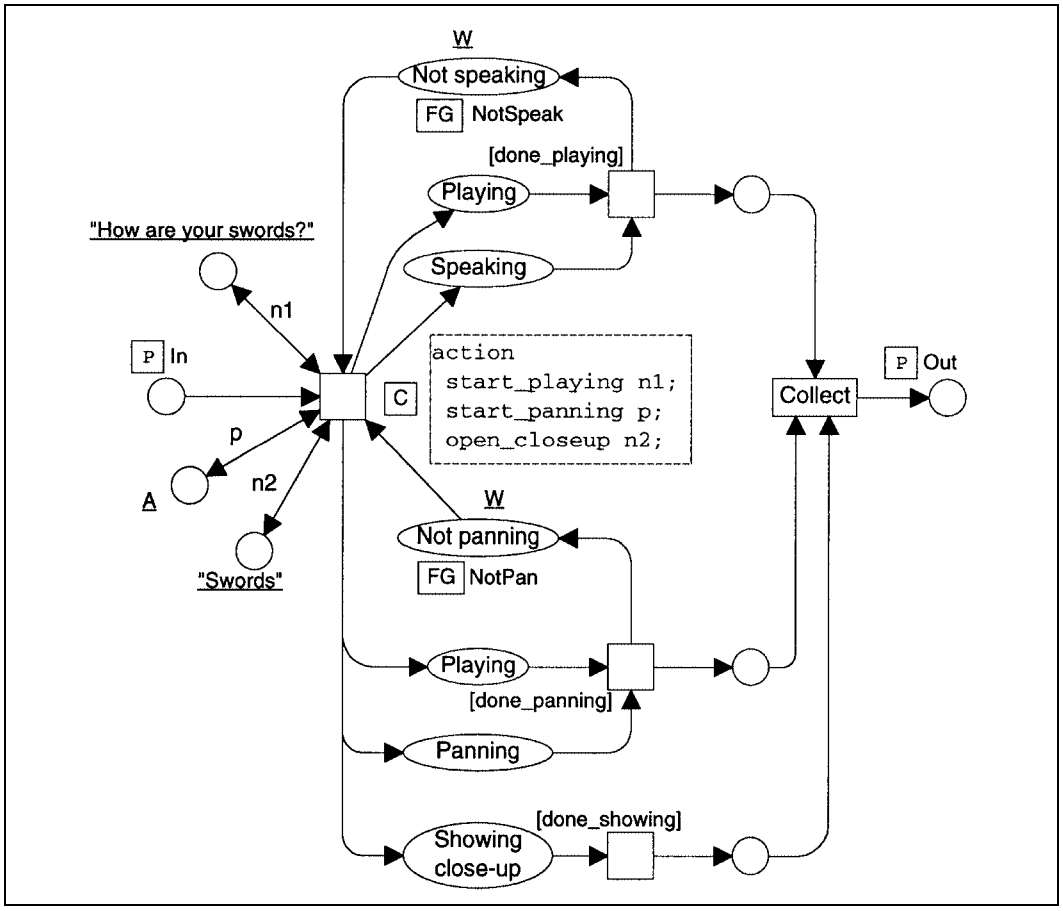


Figure 7.15: An event with intra-event synchronization. The code region of a single transition starts all the actions.

problem, assuming that the events that do get executed are also interesting for the user.

In the figure, there is no 'dispatcher' transition distributing tokens to each action, corresponding to the distribute transition of figure 7.2 (page 111). When *all* actions are synchronized to start simultaneously, that transition would have no job to do and is therefore omitted. The collect transition (taking the token coming out of each action and marking the conclusion of the entire event) is still present.

7.8 A resumption

The way to build a resumption in Petri nets differs slightly depending on whether the purpose is modelling or execution. In this section it is assumed that the purpose is the implementation of the elastic story in a computer system.

The next figure shows how a resumption is built into a thread. The example is the one given in subsection 6.3.7. The resumption starts and ends on the place in the original thread where the resumption is needed.

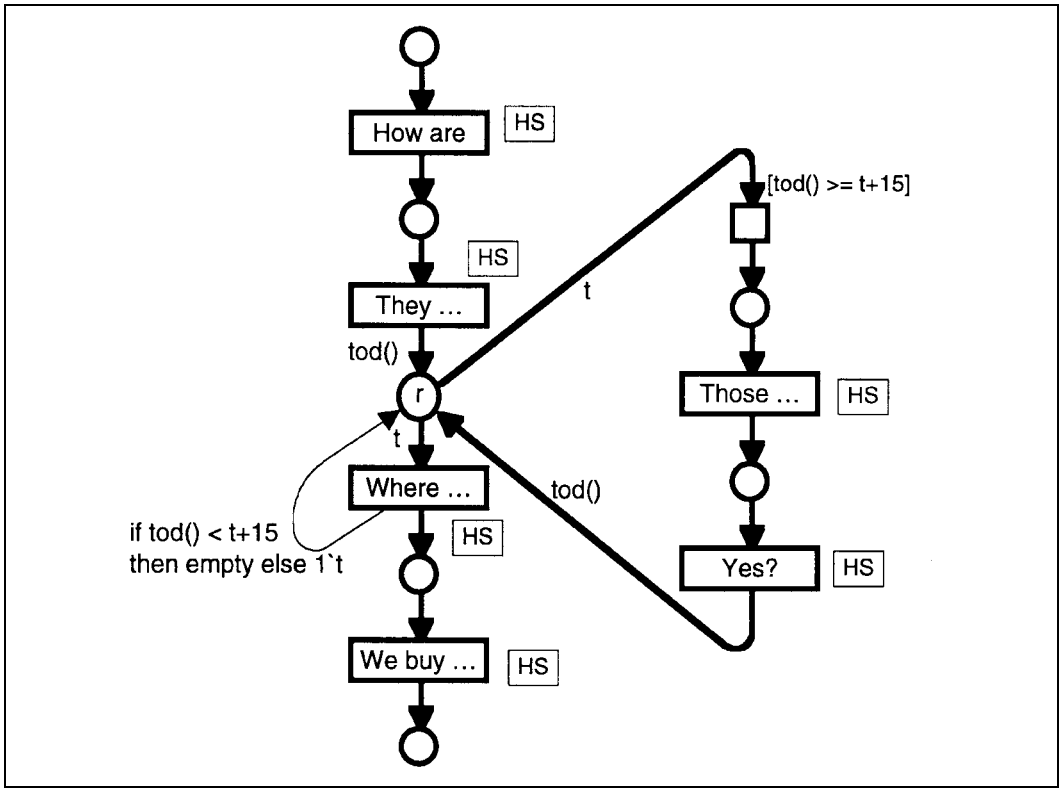


Figure 7.16: The flow of a resumption. The resumption (the two events to the right) is only executed when needed.

In this way, the thread can be executed with or without the resumption, depending on the time constraints described in subsection 6.3.7. In the example, a threshold of 15 seconds is chosen. The author can substitute any

positive number instead of 15.

The resumption brings into the narrative an element of real time; it is activated when some amount of real time elapses without the thread advancing. To deal with real time, the function tod in CPN ML is used. tod (abbreviation of 'time of day') returns the number of seconds elapsed since 0.00 GMT January 1, 1970, according to the operating system clock on the host computer.[9] (The inaccuracy arising from the fact that only whole seconds are counted, is probably acceptable.)

One use of tod is to produce on the place r a token with a 'time stamp': a token the colour of which is a time value (an integer) denoting the time when the token was produced by a transition, i.e. the time when the previous event was completed ('They ... ' in the example).

Most of the arc inscriptions in the figure are on arcs connected to substitution transitions. Such inscriptions have no formal semantics. To have effect, they are assigned to the corresponding arcs on the subpages (from and to port nodes). They are only repeated for readability here on the page containing the substitution transitions.

The way to ensure that the first event of the resumption is only executed if the specified number of seconds have elapsed, is shown in figure 7.16: a transition with a guard requiring the time now to be greater than 15 seconds after the completion of the previous event. It is worth noting that with Design/CPN, a guard is only evaluated when the content of an input place is changed, i.e. not necessarily immediately before the transition fires. If the guard evaluates to true, there is no problem; more time elapsing before the transition fires will not make the guard false, i.e. the transition can only fire when it should. However, if the guard evaluates to false (a probable outcome immediately after the preceding event), the lack of re-evaluation might prevent the transition from firing even after so much time has elapsed that the guard is true. The solution to this problem will be presented shortly.

[9] Readers who know about *Petri nets with time* may wonder if these are not the solution. For implementation purposes they are not, since there is no correspondence between real time and the model time (or simulated time) used in Petri nets with time. For specification, Petri nets with time lack the possibility of stating that a transition must fire before a certain time (if at all). For specification, the Hierarchical Time Stream Petri Net formalism is a good candidate; see section 9.3, from page 163.

To ensure that the first transition after the resumption ('Where ... ' in the example) is only executed if within the time threshold, a more complicated construction is necessary. It would *not* suffice to insert a transition with the guard [tod() < t+l5] (the logical negation of the guard used before), for two reasons:

1. As just mentioned, a guard is only evaluated when the content of an input place is changed. The guard is likely to evaluate to true, after which arbitrarily long time can pass before the transition fires.

2. After the firing of the guarded transition, arbitrarily long time can elapse before the event 'Where ... ' happens.

Instead, the problem is solved inside the first event after the resumption ('Where ... '). Figure 7.17 shows a possible implementation of 'Where ... '.

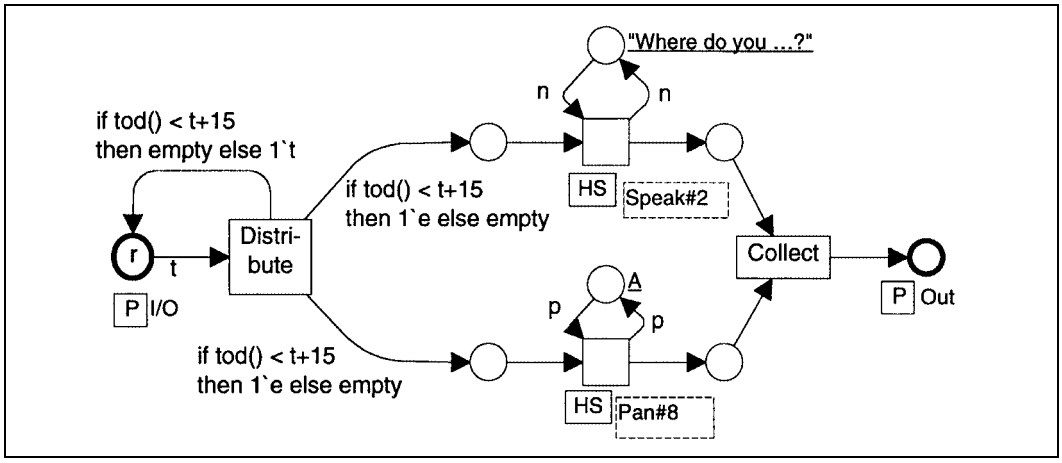


Figure 7.17: The first event after the resumption. If the time now is more than a specified amount (here 15 seconds) later than the time when the previous event was completed, the time stamped token is returned to the place r and nothing else happens.

In the figure, t is a time value, n is the name of a speech, and p designates an object in the background picture: the object to be made visible by the panning. The output arcs of the first transition (here the Distribute transition) have inscriptions. In case the event can start (the time now is within 15 seconds from the previous event), the Distribute transition behaves exactly

as described earlier: it delivers tokens to its output places to the right of it. In the opposite case (too long time has elapsed), it does not deliver any tokens to the rest of the event, but instead returns the input token to the place r without changing the time stamp. This in turn provokes a re-evaluation of the resumption's guard [tod() >= t+15], which now evaluates to true, so the resumption can be executed instead.[10] Strictly speaking, the solution presented so far does not guarantee that the user experiences the event starting within 15 seconds from either the previous event or the resumption. This is so because some time may elapse after the Distribute transition has fired before the speech or panning starts. The solution to this problem is the application of intra-event synchronization as presented in the previous section. Thus the transition starting all the actions in figure 7.15 is to be modified so that if too much time has elapsed, it does nothing except return the time stamped token to the input place. This is a little more tricky than with the Distribute transition because this transition also has other input arcs. The code region cannot directly calculate the inputs, since it would then be impossible to determine whether the transition was enabled before it had fired. Instead, the n1, p and n2 arcs can be left unchanged, since taking a token and putting it back does no harm. For the tokens from the places NotSpeak and NotPan, these are taken unconditionally, but arcs are added to put them back on the same places if too much time has elapsed. The if statement in the code region calculates the output on these extra arcs. In this way, it is ensured that it is consistent both with the other output arcs and with any action of the transition. If not too much time has elapsed, tokens are delivered to the places Speaking and Panning as before. The above implementation may loop. If the token returned after the resumption grows more than 15 seconds old, the resumption will be executed again. This may be desirable to some extent, but probably not indefinitely. The looping could easily be avoided by a small change in the scheme. The cost would be that more than 15 seconds might elapse between the completion of the last event of the resumption and the begin-ning of the following event ('Where ... '), in which case the point in having the resumption would be partially missed. Therefore, the possibility of looping is preferred. The looping reflects a realistic dialogue: a person may repeatedly try to return to a subject, but never

[10] It would have been safer and more readable to have the Distribute transition calculate all its outputs in a code region rather than on each output arc separately. In this case, the if statement would occur only once instead of three times.

succeed for long enough for the treatment of that subject to continue.

7.9 A fork

A fork is realized by a transition with two or more output arcs from it. When firing, the transition produces two or more tokens, one for each thread active after the fork. See the figure to the left.

The fork transition plays a role quite similar to the initial 'distribute' transition of an event.

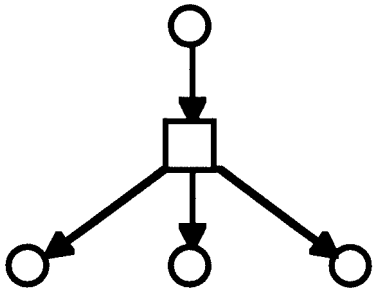


Figure 7.18: A fork is realized by two or more output arcs from a transition.

7.10 A join

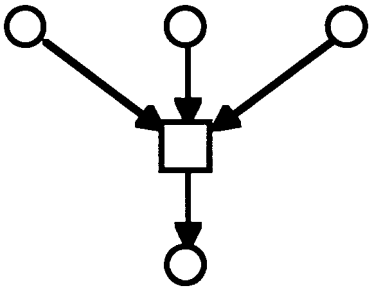


Figure 7.19: A join: two or more input arcs to a transition.

Perhaps not surprisingly, a join is realized by a transition with more than one *input* arc to it, as shown to the right.

Since the transition cannot fire until there is a token on each input place, the resulting thread (under the transition in the figure) does not continue until all threads are complete up to the join. The threads 'wait for each other'.

The join transition has a role similar to the final 'collect' transition of an event.

7.11 A choice and a merging

A choice and a merging are established by connecting more than one output and input arc respectively, to a *place*. Extra transitions are added to the choice to carry guards. The figure below shows a choice (top) and a merging (bottom). The choice depicted is the one from the example in subsection 6.3.12.

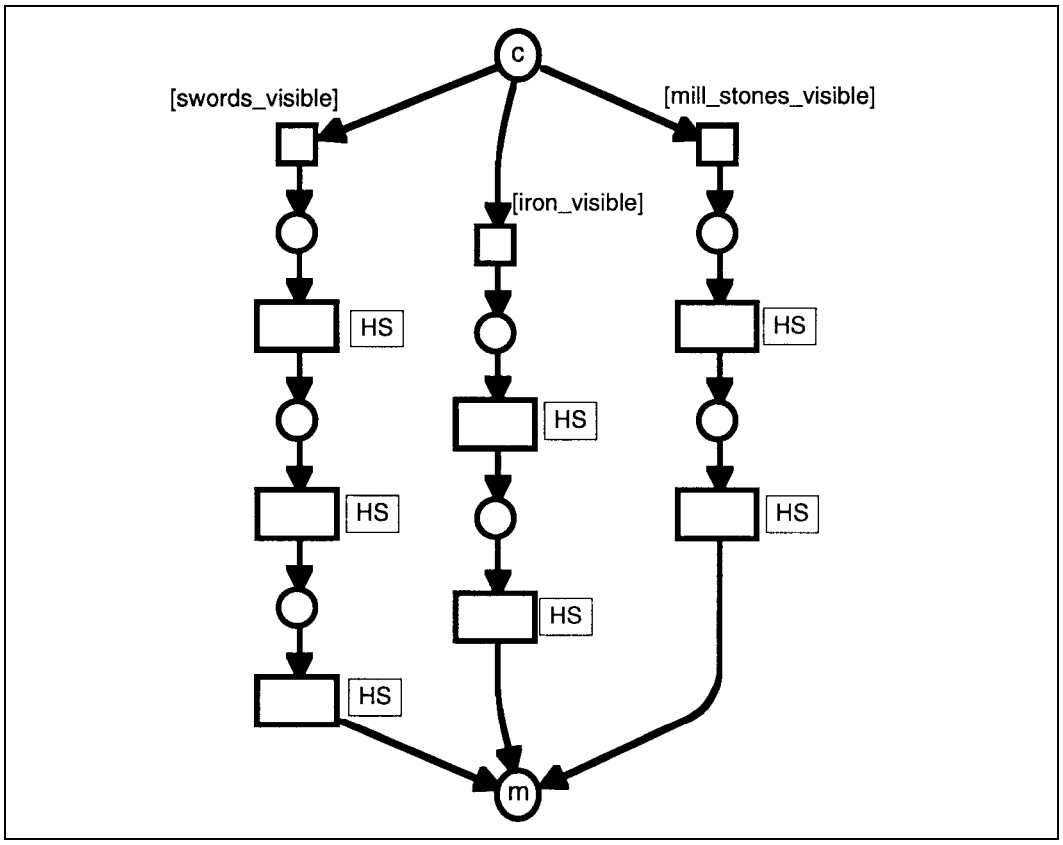


Figure 7.20: A choice and a subsequent merging. A choice, as opposed to a fork, is realized by several output arcs from a *place*. A similar difference exists between a join and a merging.

If exactly one of the guards is true, that determines which thread is chosen. So if the author wants a deterministic choice, she or he will obviously have to construct the guards in a way so that only one is true at a time. If *none* of

the guarded transitions are enabled (none of the guards are true) the story will not continue until one of the guards becomes true. If the author wants to avoid such blocking, she or he will make sure that there is always one true guard.[11]

If only one thread leads to the place c in figure 7.20, and that thread is only executed once, only one token ends up on the place. Once one of the threads leading on from the place starts executing, that token is taken. It is thereby ensured that only one of the threads is chosen.

If all the threads resulting from the choice are eventually merged (as in the example in figure 7.20), and execution is not blocked forever at a point in the chosen thread, exactly one token will eventually reach the merging place (m in the figure).[12]

All the threads need not be involved in the subsequent merging. There may be no merging at all; or only some of the threads may be merged; or there may be a number of subsequent mergings, gradually merging some or all the threads into one. Figure 7.21 shows a different example from the previous one. (Guards are given as '[G11]', etc.)

7.12 Non-determinism

It should be clear from the previous section that a non-deterministic choice is one where more than one guard can be true at a time. In that case, the Petri net formalism does not define which of the transitions will fire. The Design/CPN program used in the experiments described in chapter 8 makes a choice at random in such situations.

[11] An obvious way to ensure this will be to include an 'otherwise' guard: one that is the negation of the disjunction of the other guards, e.g. [not (go orelse g2 orelse g3)].

[12] This argument depends on the events behaving according to their definition. It is possible to build a Petri net where the argument does not hold.

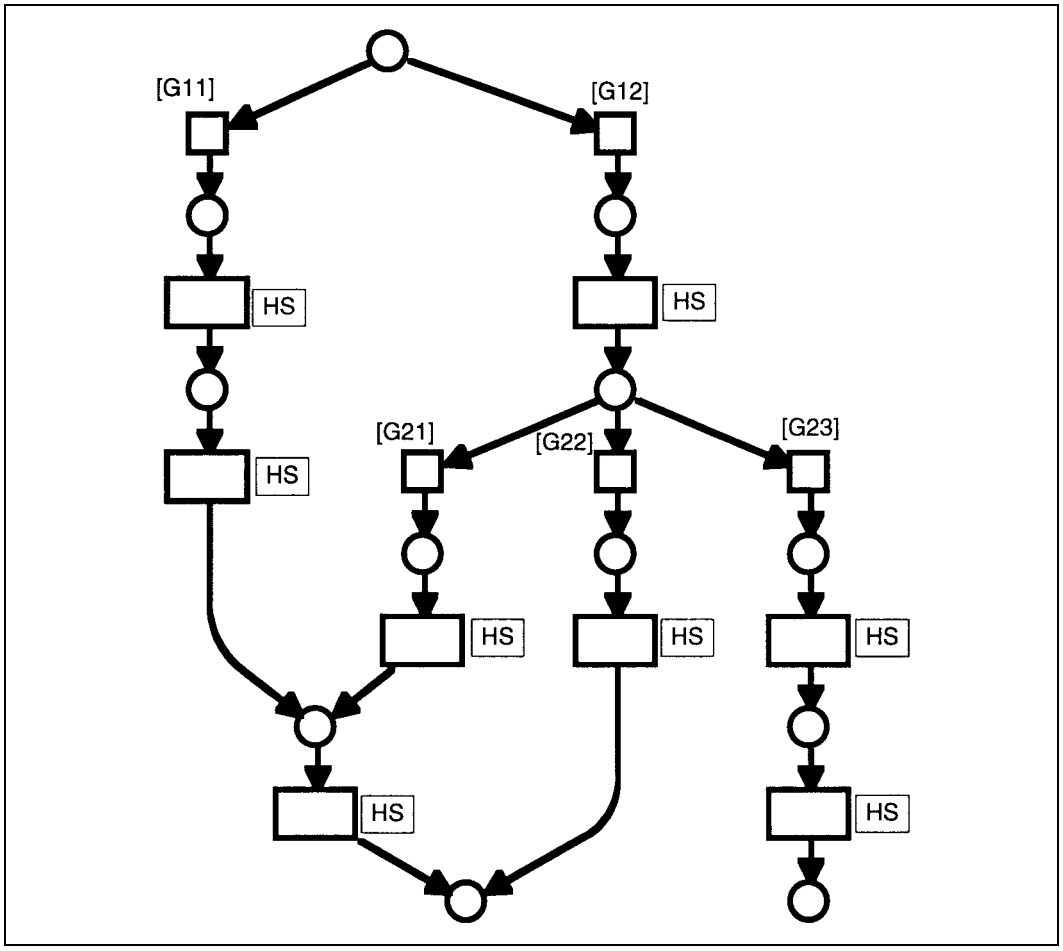


Figure 7.21: Two choices and two mergings.

7.13 A pause

A pause of the kind that blocks a thread until some object has been invisible and becomes visible, is realized by two subsequent transitions with logically opposite guards. See the figure. The example shown is taken from subsection 6.3.15.

A pause that waits for an object to become invisible or visible consists of only one of the two transitions shown in the figure.

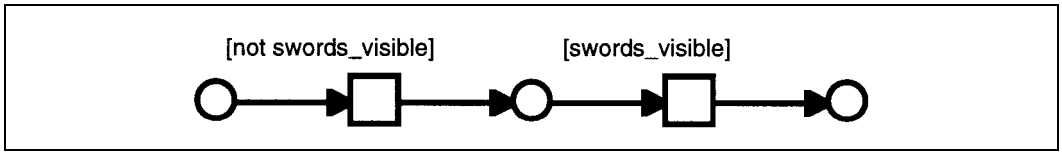


Figure 7.22: A pause is two guarded transitions.

7.14 Conclusion

This chapter has presented the Petri net descriptions of the concepts developed in the previous chapter as a model of elastic multimedia stories. Most of the concepts have very simple and straightforward Petri net implementations: an event, a thread, parallelism, a fork, a join, a choice, a merging, non-determinism and a pause. The implementations of an action and the different kinds of synchronization are slightly less simple, yet still not complicated. Only the implementation of a resumption turned out to be awkward.

The representations of some of the concepts, especially the resumption but the intra-event synchronization too, are probably so complicated that most multimedia authors with no background in programming will not want to deal with them. At the same time, all the concepts (including the basic structure of a resumption on the top-most level) are probably simple enough that many multimedia authors will not have any considerable difficulty learning to work with them, as long as they are not required to work out all the details.

To conclude, Petri nets have proven well suited for describing the concepts developed in the previous chapter as a model of elastic multimedia stories.

In the introduction (chapter 1), two ways to go from here were mentioned:

1. Petri nets may be used in an informal way in a multimedia project and a Petri net programmer be hired to formalize and refine them into nets that work as the formal descriptions Petri nets are intended to be.

2. A syntactic layer may be defined on top of the Petri nets that is easier to use and specifically targeted towards describing elastic stories. In this case, the Petri net constructs given in this chapter can be used to give a precise semantics for the new syntactic layer.

Considering the foregoing, both of these look promising.

This chapter has concluded the theoretical part of the work on story telling. In chapters 5 through 7, four models of stories have been presented, each one building on the previous one:

1. The extended layer model of a story forms the basis of chapter 5.

2. In the presentation of the extended layer model in chapter 5, it was discussed how it applies or can be modified to apply to non-textual stories, especially elastic and other interactive stories in computer-based multimedia. This implicitly described what can be called the extended layer model for interactive computer-based multimedia.

3. Based on and inspired in part by the extended layer model for interactive computer-based multimedia, a model of *elastic* stories in computer-based multimedia, in the form of a set of concepts covering such stories, was developed and presented in chapter 6.

4. Finally, here in chapter 7 it was shown how an elastic story described in this model could be modelled in Petri nets. This means that (a subset of) the Petri net formalism can be a model of elastic stories.

The next chapter presents some empirical work; it will build and run small elastic stories in Petri nets using the implementations developed in this chapter.

Chapter 8

Experiments with Elastic Stories in Coloured Petri Nets

This chapter presents the experiments conducted with Coloured Petri Nets for elastic story telling.

The purposes of the experiments are:

1. To demonstrate that Petri nets used for interactive story telling work as described in the previous chapter.

2. To gain a first experience with building interactive stories in Petri nets. To find out how well and how naturally they support the process.

3. To further evaluate the need for a syntactic ('sugar') layer on top of the Petri nets.

Three stories are described in Petri nets in this chapter:

1. A narrative from Wodan's Eye[1], called 'Kristendom' (Christianity). This narrative illustrates and tests the use of a thread, a choice and a merging, and non-determinism.

[1]For references on the Wodan's Eye project, see footnote 13 on page 103,

2. A narrative inspired by Wodan's Eye, called 'Swords, Iron and Millstones'. This narrative uses parallelism, fork and join, a pause and a generalization.

3. A second version of the same narrative, including only swords and iron, to investigate the use of the three different kinds of synchronization.

In the next section, some general information is given about the setting in which the experiments were carried out. After that, each experiment is given its own section. With the first experiment, a summary of the process of building the Petri net is given, along with some observations about how the tools supported the process. With the following stories, such observations are only made where they contribute something new. A final section discusses what can be learnt from the experiments.

8.1 Setting

The program Design/CPN for Macintosh computers has been used for the experiments. Design/CPN supports the construction, modification, syntax check, interactive and automatic simulation, and formal analysis of Coloured Petri Nets. All except the formal analysis tools in Design/CPN are used in the experiments.

Design/CPN has the graphics capabilities needed for Petri nets. Unfortunately, it does not have multimedia capabilities. Therefore, all output from the experiments is text. Also, because of Design/CPN's limited interaction capabilities, the user input has a different format from the description in section 6.2 on user interface (pages 93–94). With each experiment, ways of overcoming the limitations are discussed, so the experiments are valuable despite the altered user interface.

8.2 Thread, choice, merging and non-determinism

This section describes how a narrative named 'Kristendom' from the Wodan's Eye project was transformed into a Coloured Petri Net, and the results of that experiment.

8.2.1 The story

The narrative is taken from a draft written by Helle Juel Andersson. The draft was made available in the form of a word processor document with up to four text columns to describe up to four alternative threads in the story. The printed pages of the document were glued together into a 70 centimetres long piece to make the overall linear structure of the narrative clearer. The story is in Danish.[2] The corresponding Coloured Petri Net and sample output from it are shown in appendix A, pages 221–225.

A summary of the story will be given. Torsten is the main character. At the beginning of the story, king Harold Bluetooth orders the Danes to become Christians. After a choice, Torsten either rejects Christianity in one thread, or accepts it in the other. If he rejects Christianity, he fights king Harold, either in an open fight or by trying to kill him in a fire (new choice). In either case he fails, and is sentenced at the thingstead (moot) to be either executed or outlawed. If Torsten accepts Christianity, he chooses between marking by the sign of the cross (a precursor for baptism) and baptism proper. The two threads from the last choice merge, and Snorri tells about the law imposed by the king. In the end, all the threads merge, and Snorri talks about outlawry.

8.2.2 Input and output

This subsection discusses how input and output were designed to ensure full value of the experiment in spite of them being different from their planned

[2]Since the draft is written in *old* Danish to resemble the way people talked in the Viking age, an attempt to give it a fair interpretation in English was given up early. However, appendix A gives a translation into English of the output from the net.

format in the multimedia system.

In the imagined multimedia system, user input to the story would consist of panning over the background picture, which makes different persons and objects visible and invisible, possibly combined with moving persons and objects, for instance to make persons meet each other. User actions can occur in parallel with transitions in the Petri net. Design/CPN does not support this style of input. Fortunately, a more primitive fashion was found to be sufficient for the experiment. On a page, ellipses are drawn that represent the characters of the story. At user-controlled choices, the thread brings the window containing this page forward and awaits the user selecting (clicking on) one of the ellipses. The guards in the branching test which character (ellipse) was selected. The idea is that selecting a character corresponds to stating that the character would be visible in the planned multimedia interface (possibly together with other characters). This gives the user as much control over the story as the multimedia version would. The exact conditions (guards) used with choices were invented for the experiment, since they were not given in the draft.

Output from the narrative consists mainly of lines. In a multimedia system, these should be played as sound, possibly combined with other effects, for example as discussed in subsection 6.3.1 page 97. For the experiment, pure text output is chosen. Before the story starts, the Petri net creates a new net page containing a text field for the output. (This allows the output from several runs to be easily compared.) Each event in the story (line or other event) is represented by a single transition appending an appropriate text line to the text field on the designated page. The text is scrolled so that the newly appended text always appears at the bottom of the page. In this way, it is easy for the user to follow the story in the text field. Representing each event by a single transition rather than a subpage implies that the execution of each event does not take any considerable duration. Since the subject of this experiment is the higher-level concepts of the thread, the choice and the merging and non-determinism, this limitation has no implications for the value of the experiment. In a few cases, the text output was combined with a panning to and marking of characters speaking in the input window[3]. However, this was hardly noticed because the text field was much more con-

[3]This was done using the ShowActor function declared at the bottom of the global declaration node shown on page 209.

spicuous. (The fact that Design/CPN blinks the text field at each change probably contributed to this.) Furthermore, marking the character speaking only repeated information already given in the text rather than adding something new. It was found that this extra feature added only little to the value of the experiment. In any case, it constituted a poor substitute for the true multimedia user interface described earlier. Therefore, it was not explored further.

8.2.3 The process

It was noted that the structure of the story was evident from the draft. It was clear that the Petri net would directly reflect this structure. Some ten events (transitions) were drawn first, including places and arcs to connect them. In this process, the first choices were built too. Then modularization was commenced. For instance, the page which now appears as 'Intro#6' was separated from what is now page 'Kristen#5'. See the top of figure A.5, page 213. Design/CPN gives good support for modularization after the net or part of it has been constructed. During the construction of the rest of the net, modularization was done in parallel with drawing the net. After most of the net had been constructed, colour sets were added to places and inscriptions to arcs.

The colour set assigned to most places was the type unit (the type of the token carrying no data), and the most common arc inscription was a token of this type. Since this is so in typical elastic stories, it would be practical for the building of these if places had this type as their default colour set when they were created, and arcs a default inscription evaluating to one such token.

After colour sets and arc inscriptions had been added, it was time for the first trial run. Code regions for writing text to the report window were added later. It turned out to be necessary to add an initialization transition outside the story to set up the report page. With Design/CPN's modularization facilities, this was an easy task. The result is shown in figure A.4, page 209. Finally, guards and interaction were added at choices. Adding these at the end seems to model the way the story was originally written, since the draft did not give them.

8.2.4 Style

Deliberate variations in style in the Petri net were introduced as part of the experiment. The two most prominent examples are discussed in this subsection.

In figure A.5, page 213, different ways to modularize were tried for the two alternative threads. In retrospect, the modularization to the left in the figure, in which the entire thread between the choice and the merging is described on a separate net page, is preferred.

In figure A.6, page 213, among other places, each transition representing a line, is named with the beginning of the text in boldface. In most other places, there are no transition names. Since the entire lines of the transitions are visible in the mentioned figure too, the names seem superfluous if not disturbing. However, in a final multimedia system, the lines are to be represented on subpages. They will be stored as digital sound, not text. Under such circumstances, the names in boldface will probably be helpful.

8.2.5 Results

Text output from one run of the Coloured Petri Net is given on pages 221–225 of appendix A, with English translation. The net works as expected: it produces a coherent narrative following one of the tracks through the original elastic story. Because of an error[4] in Design/CPN, the Danish letters æ, ø and å, and é with an accent (é) do not come through in the text output.

Non-determinism works. In two successive runs, Torsten was selected on the scene at each choice. In the first run, the story took the track to the right at the first choice, on page 'Kristen#5', then to the right again on page 'Torstens#15', so Torsten ended up being baptized. In the second run, the story again took to the right the first time, but then to the left on page 'Torstens#15', so Torsten only got marked by the sign of the cross this time.

On the other hand, in this elastic story the user can control all the choices. For instance, to have Torsten reject Christianity, meet the king in an open

[4]This may have been originally introduced as a feature: characters outside the seven bits ASCII values are removed to avoid unexpected effects of such characters.

fight, and be sentenced to outlawry, the user can select Fortaeller (narrator), Gisle and Bisp (Bishop) Leofdag at successive choices. This has been tried, and worked as described. The sample output given in appendix A stems from this run. (Of course, the normal user does not know in advance the outcome of each selection.)

The net described shows that the concepts of a thread, a choice, a merging and non-determinism can be brought to work in a Petri net in a practical situation. A thorough testing of the Petri net designed to find all or most possible errors, has *not* been conducted. The Petri net has proved a convenient tool for describing the narrative. Design/CPN supported the bottom-up development of the first part well: the net was drawn first, then separated into different pages. It supported just as well the top-down development of the rest of the net: each page was created before it was filled. In a stepwise manner, the net was refined with colour sets and arc inscriptions, code regions and finally guards. It is believed that this procedure resembles the way the story was originally written. It is noted that when it comes to the dynamic splitting of threads into alternative threads by creation of choices and mergings, Petri nets and Design/CPN offer greater flexibility than a word processor. The Petri net formalism lends itself well to the authoring process in which the structure of the story (with choices and mergings) is created before the guards (conditions) to be used with the mergings are specified. It furthermore offers a natural way to describe guards, while in a word processor, the author would be forced into to inventing one. It is concluded that Design/CPN supports the authoring process well. It might even have been an advantage to use it for the Kristendom story in Wodan's Eye, even though the story was later to be manually translated into a different notation. In this case, the author would only have had to learn the basic building blocks: places, transitions, arcs and pages. She could have left out the details and more complex parts, like the interaction and the programming of guards.

8.3 Parallelism, fork, join, pause and generalization

Parallelism is tried in a second elastic story, Swords, Iron and Millstones.

8.3.1 The story

The inspiration for the story comes from the Wodan's Eye project. The scene of the story is the interior of a merchant's ship. In the multimedia system this scene would of course be depicted in the background picture. Different merchandise is visible in different places in the ship: swords, raw iron and millstones. (In Wodan's Eye there are more than three different pieces of merchandise on board the ship.) There is a thread to be told about the swords once the user navigates to their location on the background picture. Similarly, there are threads about the raw iron and the millstones. These three threads are allowed to take place in parallel, since no thread requires knowledge from any of the others. However, as in the example of subsection 6.3.15, the entire sword thread should not pass by once the user finds the swords. Rather, to encourage exploration, only part of it is told at first. To get the next piece of it, the user has to navigate away from the swords and back to them (possibly over to the millstones or the iron, but this is not a requirement). The same strategy is used in the iron and millstone threads. In the present form, each piece of each thread is simply a text string that identifies it. This is boring as a story, but good enough for testing the implementation. In this story, the text strings are in English.

To try a fork and a join, the story is given an introduction to happen before the parallel threads start, and a conclusion to take place once they all have ended. After the introduction, a fork generates the three threads. In the end, the three threads meet in a join, after which the conclusion takes place. Pauses are used to block the continuation of each thread after each part of it. The net also contains a thread of two generalizations. The first generalization is triggered when the story has talked about two different pieces of merchandise, the second one after three different pieces.

The Coloured Petri Net implementing 'Swords, Iron and Millstones' is given on pages 228–240 of appendix A.

8.3.2 Input and output

As in the previous story, output is written to a text field in a designated report window.

Handling of user input is a bit more subtle, but has been carefully designed to gain full value of the experiment in spite of this detail being different from the imagined multimedia system. The input style used in this experiment is the same as in the previous one. More precisely, input consists of selecting among four circles, Swords, Iron, Millstones and None, in a window modelling the background picture with the ship in the imagined multimedia system. 'None' corresponds to navigating away from either of the three items without navigating to any of the other ones. The selected item is recorded on a token on the global fusion place 'LastSel'. Since three threads depend on the scrolling of the background, it was decided not to build user input into any of the threads. Instead, a separate loop in the net repeatedly takes the token from the place 'LastSel' and asks the user to select again (figure A.37, page 238). Finally, the join transition removes the token to stop the loop. This scheme allows an implementation of a pause that closely models the Petri net implementation given earlier (figure 7.22 page 130): the thread is first blocked until a condition is met (an object is invisible), then until the opposite condition is met (the object is visible). The conditions are under user control. Because of the way Petri nets work, the user cannot choose freely *when* to 'navigate' to a new piece of merchandise. The Petri net decides itself whether to fire a transition representing a part of the story or the transition asking the user for input. However, since only one piece of information is given before the user is required to move away from the item, this will never mean that any thread runs for longer than it can under the multimedia interface described earlier. In other words, it does not subtract from the user's power over the course of events.

8.3.3 Results

Here is a sample run of the Petri net. Lines preceded by 'Output:' contain what the Petri net wrote to the report window. Lines preceded by 'Input:' give the item in the ship window on which the user clicked.

Output: Introduction
Input: Swords, Swords, Swords
Output: Swords, first info
Input: Iron, Iron, Iron
Output: Iron, first info
Output: Two kinds of merchandise mentioned
Input: Millstones, Millstones, Millstones
Output: Millstones, first info
Output: Three kinds of merchandise mentioned
Input: Iron, Iron
Output: Iron, second info
Input: Iron, Swords, Swords, Swords
Output: Swords, second info
Input: Iron, Iron
Output: Iron, third info
Input: Iron, Millstones, Millstones, Millstones
Output: Millstones, second info
Input: Iron, Swords, Swords
Output: Swords, third info
Input: Iron, Millstones, Millstones, Millstones
Output: Millstones, third info
Input: Iron
Output: Conclusion

In the current form, the net asks for user input relatively frequently. In the above run, an attempt to compensate for this has been made in repeating the clicks on the same merchandise, modelling the situation where the user does not navigate to another part of the ship. In the multimedia interface described earlier, there will be no problem, since selection is replaced by navigation over the background picture outside the Petri net's control.

Using the Petri net it is possible to get more than one piece of the same thread in a row, even without resorting to clicking on 'None'. Here is an example:

Output: Introduction
Input: Swords, Iron
Output: Swords, first info
Input: Swords
Output: Iron, first info
Input: Iron, Iron
Output: Two kinds of merchandise mentioned
Input: Iron, Iron, Iron, Millstones, Iron
Output: Millstones, first info
Input: Millstones, Millstones
Output: Three kinds of merchandise mentioned
Output: Iron, second info
Input: Iron, Iron, Iron, Swords, Iron, Iron
Output: Iron, third info
Input: Millstones, Millstones, Millstones
Output: Millstones, second info
Input: Iron, Iron, Iron, Millstones, Millstones
Output: Millstones, third info
Input: Iron, Swords, Swords, Swords
Output: Swords, second info
Input: Iron, Iron, Swords, Swords
Output: Swords, third info
Input: Iron
Output: Conclusion

The last half of the output from this run consists of two pieces in a row from the iron thread, then two pieces from the millstone thread, finally two pieces from the sword thread before the conclusion. This can happen because clicking on an item does not guarantee that a piece of the corresponding thread is told. This is due to the way Petri nets work, and is in good keeping with the idea of an elastic story.

8.4 Synchronization and resumption

A second version of the elastic story about swords, iron and millstones is used for the last experiment, an experiment with two kinds of synchronization

(intra-event and sub-event synchronization) and with resumptions. The third kind of synchronization, inter-event synchronization, was tried in the form of generalizations in the previous experiment.

8.4.1 The story

The new version of the story has only two threads; the millstone thread is omitted. Each thread consists of two events, numbered 1 and 2, plus a resumption consisting of one event to be inserted between the two if too long time elapses between them. This sums up to six events. Each event consists of two actions: playing a speech and panning to the person speaking. In two of the events, intra-event synchronization is used to force the two actions to start at the same time. Sub-event synchronization was used throughout to ensure that only one speech could occur at the time, and only one panning could occur at a time.

8.4.2 Input and output

Since user input is sufficiently explored in the first two experiments, there is no interaction in this version of the story. Output from the experiment consists of trace statements from many of the transitions, so it can be known when actions start and finish.

8.4.3 Results

Twenty-five runs were made. Output from one run was:

```
Start swords 1
Speak done swords 1
Speak start Iron 1
Pan done swords 1
Pan start Iron
Speak done
Speak start Swords 2
Pan done
```

Pan start Swords
Speak done
Pan done
Pan start Iron
Speak start Iron resumption
Speak done
Pan done
Start iron 2
Speak done iron 2
Pan done iron 2

Seven of the runs gave output identical to the listing above. It is not easy to get an overview of the run from it. For this reason, a graphical presentation of the same information is given in the figure on the next page. The figure shows the order of speech and pan actions.

The *times* actions started and ended have not been recorded.[5] Only the order is shown in the figure.

Some of the runs are described in appendix A, using the same graphical notation for the output. There was surprisingly little variation in the results. With one exception, the same events occurred in the same order in all the runs. With interaction added, much more variation is expected.

[5]This might have been done either with a watch while the story was running, or by having the transitions time-stamp the trace output.

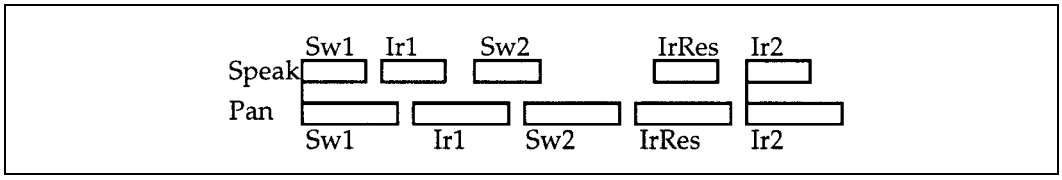


Figure 8.1: The order of speech and pan actions in a sample run of the second version of the 'Swords and Iron' story. The time progresses from left to right. The figure shows the order of the starts and ends of speeches and pannings. 'SW' refers to the sword thread, 'Ir' to the iron thread. 'IrRes' means the event in the resumption of the iron thread. There is no 'scale'; no information about the duration of actions or spaces between them should be inferred. The vertical lines connecting pairs of actions (for instance, the speech and panning of the event 'Sw1') denote that intra-event synchronization was used to make the two actions start at the same time.

The exceptional run is graphically presented here:

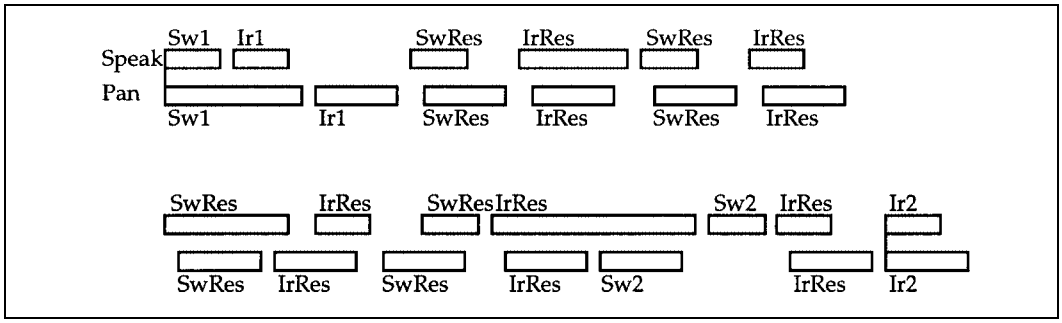


Figure 8.2: In one run, the two resumptions were repeated four and five times respectively.

The almost endless repetitions of the two resumptions are conspicuous. The viewer may take them as a joke at first. When they continue, he or she will probably think that the system is in error and it has entered an infinite loop. At that point the system surprises again in that it finally escapes the loop.

It was noted that in the above run and one other run, the line 'Start iron 2' was missing from the output. To understand how this can happen, see the transition that produced this line in all other runs in figure A.50, page 251. The arc inscriptions of the transition are evaluated at the time when it is

decided that the transition can fire. The code region is not executed until it actually fires. Then, due to the elapsed real time, the condition in the if statement may evaluate to false, so the WriteHistory function is not called. As discussed in the second last paragraph of section 7.8 (page 126), the way to repair the situation is to have the code region calculate all the inputs and outputs of the transition.

Sub-event synchronization works; in all the runs, after an action is started, there is a 'Speak done' or a 'Pan done' respectively each time before the next action of the same kind starts. This fact implies that no two speeches nor two pannings overlap.

Resumptions seem to work as expected. A resumption is never executed immediately after the first event in the same thread. If the thread is interrupted by the other thread, the resumption sometimes occurs —presumably if the duration of the interrupt is long enough.

8.5 Summary of experiments

In the experiments, each concept from section 6.3 has been implemented in an elastic story in a Petri net. One error in the realization of a resumption was found, and a correction was shown. Apart from this, the nets behaved correctly. This confirms that elastic stories as presented in chapter 6 can be described as Petri nets.

It has been argued that Petri nets fit the authoring process well. Apparently the steps involved in building the Kristendom Petri net were performed in an order that resembled the way the original elastic story had been written. It has been observed that Petri nets lend themselves to different working styles such as top-down and bottom-up, and to incremental development. It has been suggested that Petri nets would even be a flexible tool for the multimedia author who does not want to deal with the intricate details of Petri nets such as the programming of guards. To conclude, the experiments indicate that the Petri net formalism is a convenient tool for the author of elastic stories in multimedia.

The Petri nets resulting from the experiments seem to confirm what was suggested in the conclusion of the previous chapter. On the one hand, the

greater part of the diagrams are so simple that a multimedia author could build them without any training in the formal side of Petri nets. On the other hand, the diagrams contain a few intimate details (specifically, guards) that would require either a simpler syntax for the multimedia author to work with, or a specialized Petri net programmer to complete the details of certain diagram pages.

Chapter 9

Related Work on Elastic Story Telling and on Petri Nets

The work on elastic story telling presented in chapters 5–8 was inspired by Peter Bøgh Andersen and his colleagues at Institute of Information and Media Science at Aarhus University. The work of Peter Bøgh Andersen was introduced in section 6.1. More relevant work is summarized in the next section, and comparisons with the work in this thesis are made.

Two groups of researchers have worked with Petri nets in connection with multimedia or hypermedia: David Stotts and Richard Furuta have developed the Trellis hypermedia model. Patrick Sénac, Michel Diaz, Roberto Willrich and Pierre de Saqui-Sannes have developed and worked with Hierarchical Time Stream Petri Nets. Though the works of these authors are not very closely related to the work presented in this thesis, they are treated in the following sections.

9.1 The work of Peter Bøgh Andersen on interactive narratives

This section presents the interactive stories work of Peter Bøgh Andersen and colleagues. Over the years, Peter Bøgh Andersen has worked with different

formal bases for interactive narratives: vectors[1], catastrophe theory[2] and most lately a home-grown system of events that have forces, triggers and operations[3]. This system will be described in more detail shortly.

A key point in Peter Bøgh Andersen's work with interactive stories was first published in 1992:[4]

> 'The basic theory of interactive narratives that emerged during this work is that the author should not see himself as writing a narrative but a world! This world is designed in such a way that there is a good chance for the reader to experience exciting and touching stories if he acts in the world.'

In Wodan's Eye, Peter Bøgh Andersen specified each interactive story by a set of *narratives*[5]. A narrative contains a coherent discourse. Each narrative consists of *events*. At any time during presentation, a set of the events are designated *current events*. The *start events* are the set of events that are current from the start. The narratives in a story are executed in parallel.

Each event is described by means of the following fields:

`Name and pressure`, *A name, A Pressure*
`Triggers` `Forces`, *Force trigger, ... , Force trigger*
`Screen`, *Screen trigger, ... , Screen trigger*
`Success` `Media`, *Media operation, ... , Media operation*
`Forces`, *Force operation, ... , Force operation*
`Continuity`, *Continuity operation, ... , Continuity operation*

[1]Peter Bøgh Andersen, the quoted work[4].

[2]Peter Bøgh Andersen: Katastrophen und Computer. In Roland Posner (journal editor), Martin Wamke & Peter Bøgh Andersen (guest editors): Zeitschrift fur Semiotik, Band 16, Heft 1-2. Stauffenburg verlag 1994. Pages 29-50.[5] (In German.)

[3]Peter Bøgh Andersen: Narratives. Unpublished working paper, September 1994. Part of the paper appears in Peter Bøgh Andersen: Medien — Kunst — Computer/Hyperkult I-IV. In an anthology edited by Wolfgang Coy, Georg-Christoph Tholen and Martin Wamke, forthcoming[7] (in German).

[4]Peter Bøgh Andersen Vector Spaces as the Basic Part of Interactive Systems: Towards a Computer Semiotics. In Patricia Baird (editor): Hypermedia, Volume 4, number 1,1992. Taylor Graham.[8] Page 55.

[5]The description is based on the working paper 'Narratives'[7], see earlier footnote. The final Wodan's Eye system differs from this description on some points.

`Failure` `Media,` *Media operation, ... , Media operation*
`Forces,` *Force operation, ... , Force operation*
`Continuity,` *Continuity operation, ... , Continuity operation*

The name is used for referring to the event from other events. The pressure, also called the force, is used to schedule multiple current events; the one with the highest value is executed. This can be thought of as the most topical or current event. Execution of an event consists of first evaluating all its triggers, then either the success or the failure part, depending on whether all the triggers were true. There are four kinds of triggers:

- Force trigger: whether the force of a specified event is less then, equal to or greater than a specified value. In practice, the event is often `'me'`, the event having the trigger.
- Screen trigger: whether a specified object is visible in the user's view.
- Random trigger: true with the probability $1/n$, where n is supplied by the author.
- Time trigger: whether a specified event has never happened, is under execution, or has completed, or whether it started or ended at least a specified number of seconds ago.

Triggers in the same event are implicitly connected by conjunctions (*'ands'*); there are no *or* or *not* operations.

Possible operations in the success and failure parts are:

- Media operation: show a close-up picture, play a sound or pan to an object. In the last case, it is specified whether the object triggers the relevant screen triggers (that is, triggering can be suppressed).
- Force operation: increase or decrease the force of a specified event with a specified value, or set it to a specified value.
- Continuity operation: add and remove specified events from the set of current events, or replace the entire set with a specified new set. If an event is part of a thread (as defined earlier), a common set of continuity

operations will be `-me, +next` or simply `=next`, where next is the next event in the thread.

Peter Bøgh Andersen demonstrates how to build the following rhetorical structures using events:

- Fork: the same as a fork in chapter 6.
- Choice: the same as a choice in chapter 6.
- Generalization: the same as a generalization in chapter 6.
- Object characterization: The system gives different information about an object each time the user looks at it, as described in subsections 6.3.15 and 8.3.1.
- Object association: A thread starts the first time an object is seen, and continues whenever there is a place in the conversation.
- Parenthesis: A parenthetical event in a thread is only played if the user has the prerequisite in terms of another event. If not, the parenthesis is left out and the thread continues.
- Escalation: The system gives the user gradually stronger hints about doing something; if the user does not do it, the system does it itself in the end.

As an example of an escalation, the sound of a captured slave crying for help invites the user to find the slave and hear his story. If the user ignores the cry, then at some point the system will take the user to where the slave is.

Some of the rhetorical structures require the use of dummy events for holding what corresponds to global variables in their force field. Note that this is the only field that is ever changed during the presentation.

The distribution of events into separate narratives serves as a modularization. This is convenient for the author. For comparison, hierarchical Petri nets allow for a modularization in which each module can again be modularized to any level, which is much more flexible.

The structure of an event with a success and a failure part seems to be a way to have two events in one. In many of Peter Bøgh Andersen's examples, one of the parts is empty. It would probably be a better and more natural design to allow *or* and *not* operations on triggers and do away with the failure part. Also the words 'success' and 'failure' are value-laden in a way that does not always reflect their use. Possibly it would be better to find more neutral words.

The possibility of replacing the entire set of current events in a narrative, is a dangerous convenience. While an event is local in its nature, this facility globally kills all current events. The facility is unnecessary, since replacing the set can be done by removing unwanted events and adding the new ones. Killing the unwanted events by name has the advantage that other threads can later be added to the narrative without risking being killed unintentionally by an event that replaces the set of current events. For example, if the event *culture* is followed by a choice between events *plyndring, langvejs* and *tungeting*[6], Peter Bøgh Andersen prescribes the following continuities in *culture* and *langvejs*:

Culture continuity: = `plyndring langvejs tungeting`
Langvejs continuity: = `kendtil`[7]

The continuity of *langvejs* kills the events plyndring and tungeting so that the choice does not work as a fork. The same effect can be obtained with the following continuities:

Culture continuity: `- me, + plyndring langvejs tungeting`
Langvejs continuity: `- plyndring langvejs tungeting, + kendtil`

This solution is safer, as described.

The global replacement of all current events using the equal sign may be warranted in one case, though. If a story consists of phases, each one with a complicated content, the equal sign can be used to go to the next phase. In this case, killing each event from the earlier phase by name (using the minus sign) could be dangerous; it would be easy to forget one of the events.

[6]For the sake of completeness, the event names translate to plundering, fromfaraway and heavythings, respectively.

[7]Kendtil translates to knowof.

Compared to Peter Bøgh Andersen's notation, Petri nets have the advantage that the structure of the narrative in terms of threads, forks, choices, etc., and the location of each event in this structure, are more clearly visible. The Petri net description is probably more natural in this respect, since an event's position in a thread is more than a local attribute of the event.

Of the seven rhetorical structures used by Peter Bøgh Andersen, chapters 7 and 8 have described how to build the first four in Petri nets (fork, choice, generalization and object characterization). The following paragraphs will sketch the building of the remaining three (object association, parenthesis and escalation).

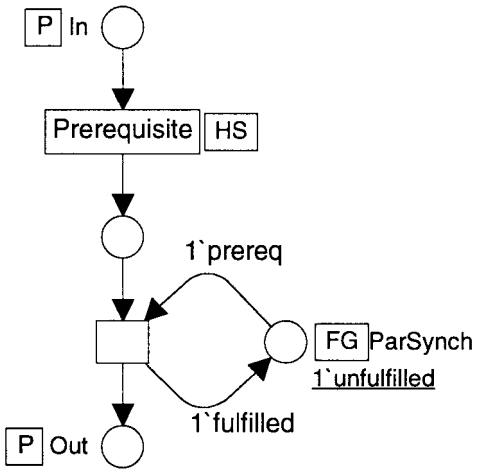


Figure 9.1: The prerequisite of a parenthesis is 'wrapped' on its own page with a transition that changes the marking of the synchronization place to `fulfilled`.

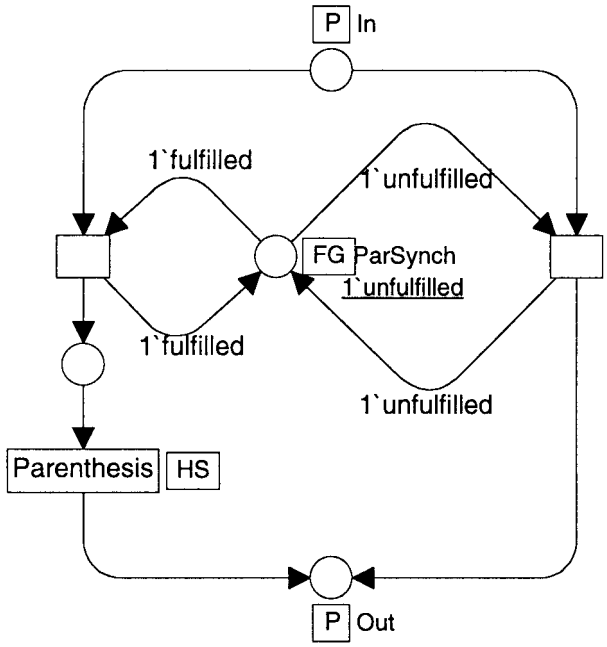


Figure 9.2: The parenthesis is 'wrapped' on a separate page with a choice and two transitions that control the choice. If the prerequisite is not `fulfilled`, the token on the synchronization place is `unfulfilled` and the transition to the left cannot fire. In this situation, the one to the right fires, which makes the thread continue without the parenthesis. By contrast, if the token is `fulfilled`, only the left route can be chosen, including the parenthesis.

It is relatively easy to build an object association in a Petri net: the first event in the thread has a guard that requires the object to be visible. The remaining events do not have guards.

The two figures 9.1 and 9.2 describe a parenthesis, as defined on page 152. The idea is related to the idea behind the implementation of inter-event synchronization. With a parenthesis, the synchronization place always holds one token, the value of the token depending on whether the event containing the prerequisite has vet been played. In the figures, the initial value is `unfulfilled`, which is changed to `fulfilled` when the prerequisite has been met.

An escalation is shown in figure 9.3. The idea is that each hint can run in a loop, and intermediate transitions bring the token on from one hint to the next. The escalation shown has two hints. It can be generalized to any number of hints, as long as each place beside a hint has an output transition that can bring the token to the start of the story (here the slave story) if the slave becomes visible. The story itself is depicted as a single event in the figure; usually it will be a thread consisting of a number of events.

In Wodan's Eye, the user can in some cases interrupt an event after it has started. Say that a thread is concerned with an object on the background.

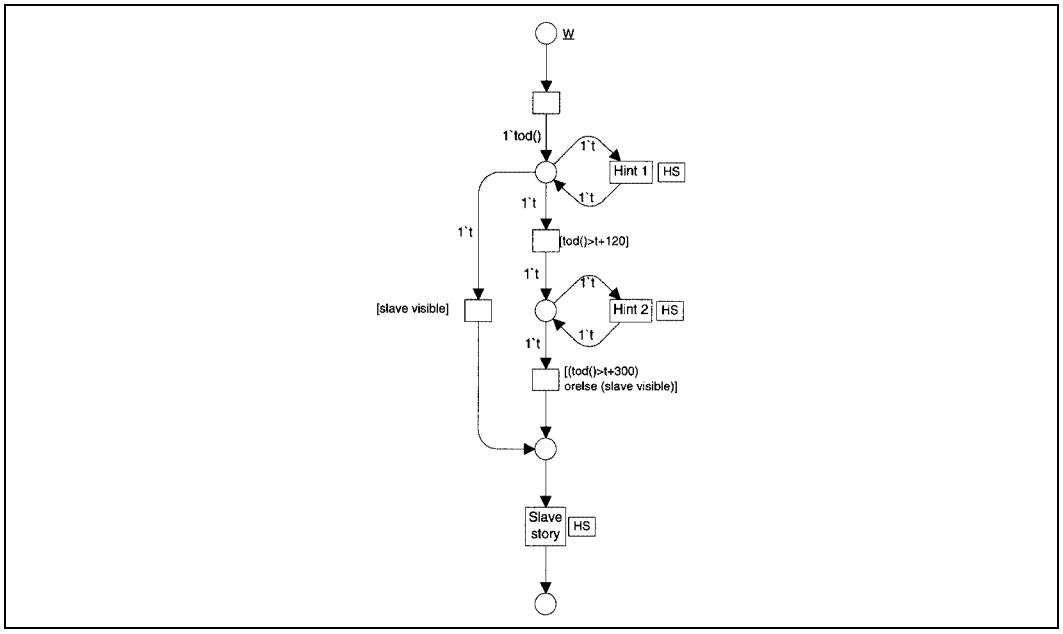


Figure 9.3: An escalation. The transition at the top produces a time stamped token. The first hint can repeat until the user finds the slave, at which point the transition to the left can start the slave story (bottom). If the user does not find the slave within two minutes (120 seconds), a guarded transition brings the time stamped token down to the place beside the second hint, which can now execute repeatedly. When the user sees the slave or after a total of five minutes (300 seconds), the next transition fires and starts the slave story.

The thread will typically begin with a pause waiting for the object to become visible and then a panning *(homing)* that brings the object to the middle of the view (window). If the user disrupts this panning by panning in a different direction, the thread does not start at all, but waits for the object to be visible again later. This functionality goes across the concept of separate events organized in a thread; but it gives the user extra control in a situation where the author will often want to give that to the user.

A second version of an escalation, called a *counterpoint*, is described by Peter Bøgh Andersen, Jens Johansen, Jakob Mikkelsen and Morten Sams in

'Interaktive tekster'.[8] In the beginning of Wodan's Eye, the user's alter ego *Adso* does not understand that the 'barbaric' Vikings can have knowledge and insight. As he learns more about them, he gains more respect. The system keeps a counter (a force in a dummy event) to represent the amount of respect due to the Viking's knowledge. However, with the increasing respect, the Viking's slave gets more aggressive, as do his cries and protests. This can be done in a Coloured Petri Net too. A global variable keeping track of the respect can be kept on a token in a global fusion place.

'Interaktive tekster' describes *interactive texts*. It uses the extended text concept, as it is called: a text is coherent and meaningful collection of signs. 'Text' does not refer to words and characters alone; the text may be multimedia. The text is called interactive only if user actions can influence the course of the text and these actions can be interpreted *within* the universe of the text (e.g., moving the hero to the right in a video game, but not turning to a different page in a book).[9]

The latter condition does not seem a clear one; it is open to interpretation in many cases. As an example, consider a hypertext reader who points to and clicks on an element of the text (be it a word or a picture), which brings the reader to a different section (node) of the hypertext. On one hand, you may argue that pointing is a way to indicate that *that concept* (within the universe of the text) is the one in which the reader is interested and on which she or he wants more detail, so the hypertext is an interactive text. On the other hand, assuming that *clicking on* a symbol is not part of the universe of the text, you may argue that this is merely navigation to a different node, navigation also not being within the universe of the text, so the hypertext is not an interactive text.

[8]Peter Bøgh Andersen Jens W. Johansen, Jakob A. Mikkelsen & Morten Sams: Interaktive tekster. In an anthology from Odense Universitetsforlag, in press.[10] (In Danish.)

[9]A similar definition is found in Peter Bøgh Andersen: Interaktive værker. En katastrofeteoretisk tilgang. In Per Aage Brandt and others (editors): Almen semiotik nr. 5. Tema: Computersemiotik Psykosemiotik. Aarhus Universitetsforlag 1992. Pages 89–112.[4] (In Danish.) The concept defined therein is called an *interactive work*. Peter Bøgh Andersen has touched on the distinction of whether user actions can be interpreted within the world of the text even earlier. In Vector Spaces ([8], page 55) he writes: "Clicking a button is a meta-action that is not part of the story, and even if the button is disguised as a tree in the landscape, it remains just as much 'meta' — who has ever heard of a story in which tapping on a tree is an important and meaningful part of the narrative?"

'Interaktive tekster' describes two ways to build an interactive text: *Disorder in order* and *order in disorder*. Going the former way, the author can start out from order or cosmos in the form of a complete coherent text, and create chaos by introducing interaction and non-determinism. The other way is to start out from a chaotic universe of 'concurrent' events (a 'narrative protoplasm'), and gradually induce local order, e.g., in the form of dialogues, causal relations between events, etc. The paper argues that the latter is the better way for a number of reasons:

1. It resembles all other known creative processes: the artist is giving form to a given substance (be it stone or language).

2. It is more natural than disorder in order. Creating disorder in a finished text is compared to producing a jig saw puzzle. It leaves the reader to reconstruct the correct text for no purpose, since the author could have given it to him or her from the outset.

3. With order in disorder, the experiments are real; the author may not know better than the user in advance what will be the exact outcome of a given user action. It is a way to produce surprises for both designers and users.

The Petri net formalism is flexible enough to allow the author to go either way. It even allows the author to describe the narrative on the right level of order from the outset, rather than work from one extreme (complete chaos or complete order) towards the other. The individual thread is in order locally and can be created so; but parallel threads can exist in a 'disorderly' space with no unnecessary ordering between them. his is probably an even more natural way to describe the interactive narrative than order in disorder. Furthermore, this way of using Petri nets still gives plenty of room for surprises for both author and reader. For example, the repeated resumptions described in figure 8.2 above were a surprise for the author. It should be noted that the above does not necessarily mean that the author gets the amount of chaos right from the start. At some times during authoring, he or she may be inducing more order into the narrative, while trying at other times to increase the amount of chaos.

9.2 Trellis

This section describes a very different related work, the work on *Trellis* by P. David Stotts and Richard Furuta.[10] *Trellis* is a formal model of hypermedia based on Petri nets. In Trellis, each place in the Petri net can be associated with a hypermedia node or *content element*. Each transition is associated with a link. Firing is controlled from a button. The user traverses the link by pressing the button. That is, a link is directed and goes from a number of source nodes to a number of destination nodes. The button controlling link traversal is only enabled when the transition is, that is, when all source nodes of the link are displayed.

[10]See for instance the following two references: P. David Stotts & Richard Furuta: Petri-Net-Based Hypertext: Document Structure with Browsing Semantics. ACM Transactions on Information Systems, Volume 7, Number 1, January 1989,[101] pages 3–29. P. David Stotts & Richard Furuta: The Trellis Hypertext Project. Position paper for Petri net models and CSCW Workshop. In Computer-Supported Cooperative Work, Petri Nets and Related Formalisms. A one-day workshop, Chicago, June 22, 1993, Proceedings,[102] pages 1–12.

Thus, Trellis separates content and link structure. The link structure is defined by the Petri net. The content is defined by a mapping from places to content elements. The mapping may be partial; that is, not all places are required to have a corresponding content element. A content element may be another Trellis hypermedia document, so a hierarchical system of hypermedia documents can be built.

Trellis allows for separate concurrent browsing paths which work much as the threads presented in chapter 6. Concurrent browsing paths are two or more sequences of content elements displayed at the same time. They can exist from the start or be created by a link with more than one destination node. They may or may not be coalesced again in the end, just as threads may or may not meet in a join. David Stotts and Richard Furuta describe one kind of synchronization of concurrent paths: the combination of a join and a new fork. This is a stronger synchronization than inter-event synchronization; it requires both paths to be at a certain point at exactly the same time. Figure 9.4 shows an example of concurrent browsing paths.

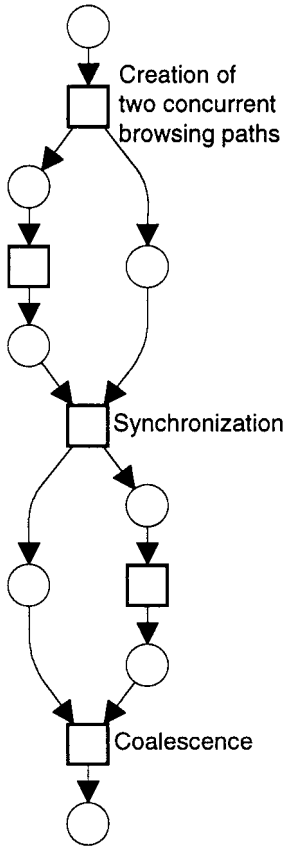


Figure 9.4: Petri net of a hypermedia document with two separate concurrent browsing paths, after David Stotts and Richard Furuta. The example corresponds to an elastic story with two subsequent pairs of parallel threads.

David Stotts and Richard Furuta show how to build tailored versions of the same hypermedia document, with different access restrictions for different readers, by providing different initial markings. A simplified example is given in the figure to the right. Petri net analysis tools make a number of analyses of Trellis hypermedia possible. Among other things, analyses may determine:

1. The maximum number of nodes open at any one time—the *display complexity* of the hypermedia document. (David Stotts and Richard

Furuta show how to calculate the maximum number of marked places over all states. Since not all places need to be associated with content elements, this is not the same as the maximum number of open nodes. However, the algorithm can be easily modified to find the latter.)

2. The reachability of each place, given an initial marking. This can be used to verify that the access restrictions for some users are enforced.

3. Whether the browsing can terminate by reaching a state where no transitions are enabled.

Building a hierarchical system of hypermedia documents, as described above, makes modular analysis possible; each hypermedia document can be analysed separately.

David Stotts and Richard Furuta have constructed two prototype hypermedia systems, α*Trellis* and later χ*Trellis*, both based on Trellis.

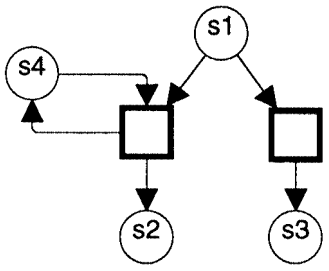


Figure 9.5: Petri net for a hypermedia document with access restrictions. With an initial marking of s1 only, a user can access s1 and s3, but not s2. A user with unlimited access to the document will have an initial marking where both s1 and s4 are marked. (s4 is not mapped to any content element.)

It is interesting that David Stotts and Richard Furuta build the hierarchy by substituting subnets for *places*, while Coloured Petri Nets substitute subnets (pages) for *transitions*. The original hierarchy paper[11] describes both possibilities, though the details of substitution places are more elaborate than in the version used by David Stotts and Richard Furuta.

This difference has implications; a consequent difference between Trellis and the use of Coloured Petri Nets presented in this thesis is that in Trellis, the content is associated with places, while on the higher level of the nets

[11]Peter Huber, Kurt Jensen & Robert M. Shapiro: Hierarchies in Coloured Petri Nets. In G. Rozenberg (editor): Advances in Petri Nets 1990. Lecture Notes in Computer Science vol. 483, Springer Verlag 1991, pages 313–341. Also in Kurt Jensen and G. Rozenberg (editors): High-level Petri Nets. Theory and Application. Springer Verlag 1991[58], pages 215–243.

presented in this thesis, content is associated with (substitution) transitions. The reason for this is the need for a hierarchy: the content is associated with the kind of node that allows substitution. On the bottom level, for instance inside the speak module in figure 7.13, the content is associated with the place in the sense that the speech lasts for as long as the token is on the place 'Playing'.

In traditional hypermedia, links may have two or more endpoints. Each endpoint may serve as source or destination or both. However, David Stotts and Richard Furuta's notion that *all* source nodes of a link must be displayed before the user can follow the link, seems an unnatural and unusual restriction.

This restriction is clearly reflected in the example Petri nets provided by David Stotts and Richard Furuta in their papers; links with more than one source are only used with concurrent browsing paths, where all source nodes will eventually be displayed during browsing. Furthermore, the concurrent paths in the examples are short (one or two content elements) and few, so it is not difficult to find the way to the end. David Stotts and Richard Furuta occasionally link into the middle of a set of concurrent browsing paths; a link links to one node in each path. However, it would be easy for an author doing this to forget one of the paths. If this happens, a reader reaching the point of coalescence will experience a dead end, since not all paths reach that point. Alternatively, an author seeking to avoid such dead ends may provide several destinations for unnecessarily many links, or avoid links with multiple source nodes. The unavoidable result of such strategies will be a mess of too many open nodes at the same time. David Stotts and Richard Furuta sometimes solve this problem with 'done links', that is, 'links' with no destination. Such a 'link' simply removes its source node(s) from the display.

It might be argued that the same problems might occur with the use of Petri nets for elastic stories. However, in the concepts underlying this use, there is nothing that invites for linking into the middle of parallel threads. Furthermore, a new syntactic layer over the Petri nets, as discussed in chapter 11, may provide the ultimate solution by prohibiting such linking completely.

As another point of criticism, it is unnatural and probably unsafe to apply access control to links rather than nodes. Adding new links could easily circumvent access control unintentionally. A Petri net analysis would discover

the hole, as stated above; but it relies on the author remembering to perform the analysis and close-read the result.

A final criticism: Even maps of hypermedia documents can get confused. To judge from the examples provided by David Stotts and Richard Furuta, a Petri net of a hypermedia document with many links very quickly gets almost unreadable. A hierarchy of hypermedia documents may help to some extent, but at the price of limiting the possible links, since a link cannot link nodes in different documents in the hierarchy.

9.3 Hierarchical Time Stream Petri Nets

Patrick Sénac and Michel Diaz have developed Hierarchical Time Stream Petri Nets (HTSPN).[12] HTSPN, an extension to Petri nets, is a formal model of multimedia and hypermedia capturing both media synchronization and linking. (Michel Diaz and Patrick Sénac call linking *logical synchronization* while synchronization is called *temporal synchronization.*) The following presentation is deliberately informal. It is hoped that it gives the impression of a notation the syntax and semantics of which can be fully formalized. The reader seeking the formal presentation of HTSPN should read the works referenced in the footnotes.

HTSPN are hierarchical in the same way as Trellis nets; it can be specified that an entire subnetwork (similar to a page in CPN) is to be substituted for a single place on the higher level. The HTSPN hypermedia model consists of three layers:

- the *link synchronization layer*, describing links;
- the *composite synchronization layer*, describing multimedia svnchronization;
- the *atomic synchronization layer*, describing playback of individual atoms.

[12]The following presentation is mainly based on Patrick Sénac, Roberto Willrich & Michel Diaz: Hypermedia Synchronization Modelling: A Case Study. In Ed-media 95 World Conference on Educational Multimedia and Hypermedia Proceedings.[97] 1995.

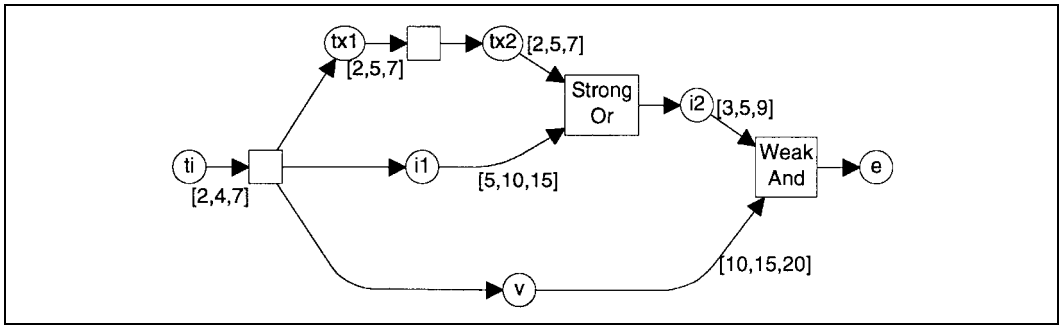


Figure 9.6: Example multimedia presentation from Patrick Sénac and Michel Diaz. ti represents a title. tx1 and tx2 represent successive texts, i1 an image to be shown concurrently with the two texts, i2 another image to follow the first and the texts, and v a voice to accompany the texts and images throughout. The time inscriptions in square brackets give the minimum, the nominal and the maximum duration of the presentation of each element. The two transitions with multiple inputs are assigned the strong-or and weak-and firing rules respectively. These are explained in the text.

Figure 9.6 shows an example of a page from the composite synchronization layer (the middle layer) of a HTSPN.[13] The page represents a non-interactive excerpt (a composite) of a multimedia presentation. First, a title is presented, represented by place ti. The arc leaving ti has the inscription [2,4,7], meaning that the presentation of the title has a nominal duration of four seconds, but can have any duration between two and seven seconds. In this way, the model takes variations in durations or *jitter* into account. Such temporal variations occur if retrieval or playback of the media takes shorter or longer time than expected, a realistic possibility in asynchronous, distributed systems. Variations can also be specified to introduce flexibility in scheduling, or to allow an incomplete or imprecise specification in early stages of a modelling process.[14]

The interval from two to seven seconds after the start of the title is called the *temporal validity* of the output arc from ti. The transition following ti (having only one input arc) is *firable* within this interval, which means that

[13]The example has been taken from Michel Diaz & Patrick Sénac: Time Stream Petri Nets. A Model for Timed Multimedia Information. In Robert Valette (editor): Application and Theory of Petri Nets 1994. 15th International Conference, Proceedings. [96]Springer-Verlag 1994.

[14]Patrick Sénact personal communication.

can only fire within it, and it *must* fire no later than the end of it.

After the title, three items are presented in parallel: a text in tx1, an image in i1, and a voice comment in v. The voice comment lasts for the rest of the presentation. A new text in tx2 replaces the first after nominally five seconds. After another nominal five seconds, a new image in i2 replaces both the text and the first image.

In the example, it is assumed that there is a redundancy between the first image (i1) and the second text (tx2), so when the first of them finishes, they can both leave the scene for the next image. To this end, the transition marking the change (having two input arcs) is associated with a firing rule named 'strong-or'. Under this rule the transition is firable after the first input arc becomes temporally valid and until the temporal validity of any input arc ends. For example, if the arc from tx2 is temporally valid from time 12 to 17 (measured from the beginning of the presentation), and the arc from i1 is from 11 to 21, then the 'strong-or' transition is firable between times 11 and 17. There is one exception, though: a transition is only firable when it is enabled. Therefore, if a 'strong-or' transition becomes enabled only after one of the temporal validity intervals has ended, it must fire immediately; it is only firable in the same moment.

A second firing rule is also used with one of the transitions in the example: the 'weak-and' firing rule. Its semantics are the opposite of 'strong-or'; the transition is firable from the moment all input arcs are temporally valid until the last one ceases to be (on the condition that it is enabled during this interval).

For a transition with exactly one input arc, the firing rule is irrelevant; the transition will be firable during the temporal validity of the arc. For a transition with no input arcs, firability is undefined.

Thus, arcs from places to transitions (but not from transitions to places) in the HTSPN formalism, are *timed arcs*. A timed arc has an inscription of the form [x,n,y], where three numbers in the brackets denote the minimum, nominal and maximum duration, respectively. The nominal duration has no formal semantics and can be replaced by '$\star$' if it is unknown. An untimed arc can be realized using the inscription $[0,^{\star},\infty]$, which is also the default.

Patrick Sénac and Michel Diaz introduce nine different firing rules to handle

the situations where transitions have several input arcs. For each such transition, the author chooses one of the nine firing rules. The nine firing rules form a pattern. For the beginning of the firability interval, the author can choose among three points in time:

1. The time when the first input arc becomes temporally valid.
2. The time when the last input arc becomes temporally valid.
3. The time when a designated master input arc becomes temporally valid.

(Due to the dynamics of the formalism, the order in which arcs become temporally valid is not always statically decidable.) Similarly the author chooses among three points for the end of the firability interval:

1. The time when the first temporal validity interval ends.
2. The time when the last temporal validity interval ends.
3. The time when the temporal validity interval of the designated master input arc ends.

		Firability starts when the following temporal validity interval starts:		
		Earliest	Latest	Master
Firability ends when the following temporal validity interval ends:	Earliest	strong-or	and	strong-master
	Latest	or	weak-and	weak-master
	Master	or-master	and-master	master

Table 9.1. The nine firing rules for HTSPN.

(The order in which temporal validity intervals end is not statically decidable and can be different from the order in which they started.) The combination of a start and an end time defines the firing rule as shown in the table on the previous page.

Two modifications are applied to the firability interval found above:

1. If the transition is not enabled at the start of the firability interval found, the start time is set to the time when the transition becomes enabled. If the transition is never enabled, it is never firable.

2. If the end of the firability interval lies before the beginning, it is set to the same as the beginning. In this case, the firability is a single point in time, at which the transition must fire.

Perhaps having nine firing rules is more than sufficient. Not more than three or four of them are used in the examples presented by Patrick Sénac and colleagues.

The link synchronization layer (the top layer) of a HTSPN specifies hyperlinks in the same way as in Trellis. The time inscriptions in a natural way introduce timed links, an idea already presented by David Stotts and Richard Furuta in 1989.[15] For instance, a sole input arc to a link transition with the arc inscription $[7, *, 30]$ describes a link that is only available after seven seconds and is followed automatically if the user does nothing for 30 seconds.

Patrick Sénac and colleagues do not describe the bottom layer, the atomic synchronization layer in detail.

Ongoing work by Patrick Sénac and colleagues includes further verification and analysis techniques for HTSPN and automatic generation of MHEG documents from HTSPNs.[16]

Patrick Sénac, Roberto Willrich and Michel Diaz describe a case study. One simplified module from a computer aided learning program used by Airbus Training for pilot and maintenance staff training, was modelled in HTSPN.

The following *methodology* was used:

1. use of the HTSPN for logical and temporal synchronization of hypermedia documents;

[15]Petri-Net-Based Hypertext, previously mentioned work, page 26.[58]

[16]Patrick Sénac Pierre de Saqui-Sannes & Roberto Willrich: Hierarchical Time Stream Petri Net: a Model for Hypermedia Systems. In Giorgio De Michelis & Michel Diaz (editors): Application and Theory of Petri Nets 1995. 16th International Conference, Turin, Italy, June 26-30,1995, Proceedings. Lecture Notes in Computer Science vol. 935, Springer Verlag 1995, pages 451–70[97].

2. verification of the modelled multimedia scenarios, in order to check time inconsistencies; also logical properties can be verified; if any errors are found then go to 1.;

3. simulation and analysis of the HTSPN specification to verify the behavioural correctness of the presentation;

4. design of the run-time system, using the validated HTSPN resulting from 3.

Patrick Sénac, Roberto Willrich and Michel Diaz report that HTSPN was sufficiently powerful and expressive for modelling the entire simplified module. It was easy and quick to build the model. HTSPN was used for a full and accurate simulation of the module.

Patrick Sénac, Roberto Willrich and Michel Diaz claim that the methodology can reduce design errors significantly. They do not report any errors caught during the modelling of the training module.

It should be noted that Patrick Senac, Michel Diaz and colleagues take the opposite approach to synchronization than QuickTime, presented in section 4.1.1. QuickTime makes sure that the playback duration is fixed and leaves out some of the data if needed. HTSPN are meant to model the playback of all the data and take into account that the exact duration is not known. HTSPN require that the upper and lower bounds on the duration can be given. It is probably safe to assume that in some cases, such bounds can be given, in others, they cannot. The HTSPN approach has the advantage of being more realistic in a distributed, asynchronous system, while the more perfect synchronization in QuickTime is sometimes desired, for instance for lip synchronization and for dance with music. (Even these would probably allow slight time inaccuracies.[17])

The use of timed arcs as in HTSPN can make the Petri net representation of a resumption, as presented in section 7.8, considerably simpler. In figure 7.16 page 123, the arc from r to the beginning of the resumption is given the inscription [15,15,15]. This serves a double purpose: First the resumption cannot start before 15 seconds have elapsed. Second, after exactly 15 seconds,

[17]For inaccurate lip synchronization, Patrick Sénac refers to Steinmetz: Human perception of multimedia synchronization. IBM technical report No. 43.9310[100].

the token is removed from the place r, making sure that the thread does not continue without the resumption after this time. The arc from r to the following event in the original thread does not need a time inscription, and the changes to that event (shown in figure 7.17) are now superfluous. Two facts should be noted, though:

1. This use of the HTSPN formalism does not use HTSPN's capabilities for imperfect timing (jitter). A simpler Petri net extension than HTSPN could serve the same purpose.

2. For the HTSPN solution to be used for implementation, it is necessary that executable code can be generated from the HTSPN, for instance MHEG code as discussed above.

Table 9.1 may give the impression that the HTSPN formalism is complete in terms of the types of synchronization that can be specified. However, some kinds of synchronization that authors may realistically want cannot be specified in a HTSPN:

1. Two different masters, so that the firability interval is from the beginning of the temporal validity of one master to the end of the temporal validity of the other master.

2. Any reference to the beginning or ending of the second, third, second last, third last, etc., temporal validity interval. For instance, a transition is firable whenever at least two input arcs are temporally valid.

3. Any kind of averaging.

The first of these can be modelled by a 'weak-master' or 'or-master' firing rule in a HTSPN. Each of number 2. and 3. can be modelled by an 'or' firing rule. In each case, a very inaccurate model results; synchronization information is lost.

As Patrick Sénac, Roberto Willrich and Michel Diaz mention, analyses of a HTSPN can catch many synchronization errors in a design. To judge the usefulness of this opportunity, a couple of open questions remain: 1. Do hypermedia designers often make synchronization errors? 2. If they do, are these errors caught as easily with other (and maybe less formal) methods, or do they often pass unnoticed?

9.4 Summary

This chapter has presented works by other authors related to the work presented in the previous chapters on the use of Petri nets for elastic story telling. Many observations were made, only the most important ones being summarized here.

It was observed that Petri nets had no problems modelling the rhetorical structures used by Peter Bøgh Andersen in the Wodan's Eye project.

The Trellis hypermedia model inspired the observation that the content elements of a presentation are associated with the kind of nodes (places or transitions) that allow for hierarchy substitution: places in Trellis, but transitions in CPN. It may or may not be more natural to associate content with places.

As a final observation, the timed arcs and firing rules in HTSPN are useful for multimedia synchronization, not only in traditional hypermedia systems, but also in elastic stories.

Chapter 10

Repertory Grids for Hypermedia Navigation

10.1 Introduction

This chapter describes Talaria[1], a hypermedia training and reference tool for health care providers managing patients with cancer pain, being built at the Fred Hutchinson Cancer Research Center (FHCRC) in Seattle, USA. The work covered in this chapter has been previously published in a conference article[2].

The clinical practice guideline on cancer pain relief, released by the Agency for Health Care Policy and Research (AHCPR) in 1994, formally defines much of the content of Talaria[3]. The purpose of the program mirrors that of the practice guideline: to improve the management of pain in patients with cancer by informing physicians, nurses and other health care providers about

[1]The Talaria were the winged sandals of Mercury, messenger to the Gods

[2]David Madigan, C. Richard Chapman, Jonathan Gavrin, Ole Villumsen & John Boose: Repertory Hypergrids: An Application to Clinical Practice Guidelines. In European Conference on Hypermedia Technology 1994 Proceedings. ACM Press 1994.[76] Pages 117–125.

[3]Jacox, A., Carr, D.B., Payne, R., et al. *Management of Cancer Pain. Clinical Practice Guideline* No. 9. AHCPR Publication No. 94-0592. Rockville, MD. Agency for Health Care Policy and Research, U.S. Department of Health and Human Services, Public Health Service, March 1994.[60]

current therapeutic options and principles.

As discussed in the thesis introduction (subsection 1.3.1 pages 32–34), it is often advantageous to use multimedia and hypermedia in combination. For this reason, it has been found interesting to include in the thesis this chapter on hypermedia. Talaria uses both hypermedia and multimedia to overcome the problems associated with booklet-based clinical practice guidelines.

This chapter describes the development of a novel hypermedia linking scheme to meet Talaria's requirements (many other asbects of Talaria, such as its user interface, will be considered in detail during its development at Fred Hutchinson Cancer Research Center over the coming two years). The linking scheme implicitly constructs links between nodes by assigning each node a location in a 'context space'. A node links to another node if they are close in context space.

Focusing on hypermedia linking, this chapter places itself near the user-controlled extreme of the scale shown in figure 6.1 on page 91. Ultimately, Talaria will include more author-controlled parts too, in the form of guided tours, and possibly elastic guided tours.

To evaluate the effectiveness of the approach to linking a protocol analysis was conducted. The results suggest that the linking scheme is effective and overcomes many of the difficulties associated with large hyper-linked documents.

A sketch of the problem domain is given first.

10.1.1 Cancer pain and the AHCPR guideline

Pain is a pernicious force that increasingly threatens the functional capability and psychological well-being of the cancer patient as disease progresses. Because of unrelieved pain, many patients spend the last weeks, months or even years of their lives with needless discomfort and disability[4]. Tragically, the extensive suffering caused by cancer pain is largely unnecessary. Gains in knowledge about pain and its control and technological advances in pain management now enable informed physicians to relieve up to 90% of cancer

[4]Bonica, J.J. "Cancer pain", *The Management of Pain.* Lea and Febiger, Malvem, PA, second edition,[18] 1990.

pain[5]. However, many patients get inadequate relief because of underuse of treatment resources. This largely stems from a lack of knowledge amongst caregivers. Talaria, like the AHCPR guideline on cancer pain, addresses this obstacle.

The AHCPR defines clinical practice guidelines as 'systematically developed statements to assist practitioner and patient decisions about appropriate health care for specific clinical circumstances'. The AHCPR sponsors private sector panels, composed of experts from relevant disciplines, to develop these guidelines to reduce clinical uncertainty and improve patient outcomes. The cancer pain guideline is a 260-page, paperback booklet consisting of text, tables, and figures. It addresses pain assessment, the psychological and physiological impact of cancer pain, interventions for the treatment of cancer pain including pharmacological, psycho-social and procedurally based interventions, and a variety of special topics. The target audience for the guideline extends to patients (both adults and children), patients' families and clinicians at all levels.

Clinical practice guidelines of non-federal origin have proliferated in recent years. The American Medical Association alone offers $1,300$ different guidelines. The AHCPR estimates that over $10,000$ guidelines have been developed in the history of medical practice. While the cancer pain guideline is the immediate focus of this chapter, the development of a general purpose methodology is intended.

[5]Two references Levy M.H. "Pain management in advanced cancer", *Semin Oncol.*, 12, pages 394–410,[71]1985. Portenoy, R.K. "Cancer pain: epidemiology and syndromes", *Cancer.* 63, pages 2298–2307,[91] 1989.

10.1.2 Talaria objective and requirements

Much anecdotal evidence exists that guidelines impact practice patterns minimally[6]. More formal studies such as those of Grilli et al.[7], Lomas et al.[8], and Ford et al.[9] report similar findings. The objective of Talaria is to render guidelines in a more useful form. The paper-based guidelines have many deficiencies and these largely define the requirements for Talaria:[10]

- Booklet guidelines have little depth and provide no support for users in specific specialities who want explanations of concepts, mechanisms and procedures.
- Booklet guidelines cannot demonstrate procedurally based interventions.
- Booklet guidelines lack the facility to implement an instructional strategy and provide no feedback to the reader.
- Booklet guidelines provide minimal support for practical day-to-day problems such as drug dose calculations, or dose conversions across routes of delivery.
- While guidelines in booklet form facilitate distribution, such booklets typically get absorbed into the mound of literature accumulated by every health care provider.

[6]Gardner E "Putting guidelines into practice", *Modern Healthcare*, pages 24–36,[44] 1992.

[7]Grilli R., Alexanian A., Apolone, G., Marsoin, S., Nicolucci, A., Torri, V., Compagnucci, M., and DiMambro, EI et al. "The impact of cancer treatment guidelines on actual practice in Italian general hospitals: The case of ovarian cancer", *Annals of Oncology*, 1, pages 112–118,[47] 1991.

[8]Lomas J Anderson, G., Dominick-Pierre, K., Vayda, E., Enkin, M., and Hannah, W. "Do practice guidelines guide practice: The effect of a consensus statement on the practice of physicians", *New England Journal of Medicine*, 321, pages 1306–1311,[72] 1989.

[9]Ford, L., Hunter, C., Diehr, P., Frelick, R., and Yates, J. "Effects of patient management guidelines on physician practice guidelines: the Community Hospital Oncology Program experience", *Journal of Clinical Pharmacology*, 5, pages 504–511,[41] 1987.

[10]Two references: Cook P. "An encyclopedia publisher's perspective, Interactive Multimedia", Apple Computer, Inc., Microsoft Press,[31] 1988. Madigan, D., and C.R. Chapman: Multimedia tools for cancer pain education. In C. Ghaoui and R. Rada (editors): Medical Multimedia, Intellect 1995,[75] pages 121–136.

- Booklet guidelines provide little motivation for clinicians to learn new information.

- Booklet guidelines provide little support for patient-clinician communication.

Booklet guidelines do, however, provide an effective browsing tool and a key requirement is to emulate this ability.

Hypermedia can support browsing as well as address the above deficiencies. However, in large hypermedia documents, manual creation and maintenance of links prove difficult[11]. Each time the document acquires a new node, the author must examine all the existing nodes to determine what new links are required. Tuning links to optimize performance presents similar difficulties. The cancer pain guideline has 136 sections, 18 tables, and a variety of supporting material. Furthermore, the guideline is evolving: the AHCPR plans to issue new editions every two or three years. A linking scheme is required that does not demand hand-creation and maintenance of each link.

There have been only a few reports on efforts to evaluate the effectiveness of hypermedia applications formally; evaluation is central to the approach presented here.

[11]Six references: Bernstein, M., Bolter, J.D., Joyce, M., and Mylonas, E. "Architectures for volatile hypertext", *Hypertext 91: Proceedings of the Third ACM Conference on Hypertext*, ACM Press, pages 243–260,[16] 1991. Canter, D., Rivers, R., and Storrs, G. "Characterising user navigation through complex data structures", *Behaviours and information Technology*, 4, pages 93–102,[28] 1985. Tang, H., Bardn, R., and Clifton, C. "A new learning environment based on hypertext and its techniques", *Advanced Research on Computers in Education*, R. Lewis and S. Otsuki (editors), North-Holland, pages 119–127,[103] 1991. Lucarella, D., Parisotto, S, and Zanzi, A. "MORE: Multimedia object retrieval environment", *Hypertext '93: Proceedings of the Fifth ACM Conference on Hypertext*, Seattle, pages 39–50,[73] 1993. Tompa, F.W., Blake, G.E., and Raymond, D.R. "Hypertext by link-resolving components", *Hypertext '93: Proceedings of the Fifth ACM Conference on Hypertext*, Seattle, pages 118–130, [104]1993. Utting, K. and Yankelovich, N. "Context and orientation in hypermedia networks", *ACM Transactions on lnformation Systems*, 7, pages 58–84,[108] 1989.

10.1.3 Overview of the chapter

Section 10.2 describes current ideas about the navigation environment as experienced by the user, and guided tours and other learning aids beside hyperlinks. Section 10.3 describes repertory grids and the developed linking scheme. Section 10.4 describes the use of the linking scheme in Talaria. Section 10.5 presents an evaluation of aspects of the scheme. The final section describes current activities and future directions.

10.2 Navigation and the travel metaphor

Talaria's interface design derives principally from the concept of a Learning Support Environment (LSE) developed and implemented by Hammond and co-worker[12]. A similar approach characterizes the Hyperties system of Morariu and Schneiderman[13] and Marchionini and Schneiderman[14]. An LSE provides the learner with a set of tools to support exploration of, or instruction in, some field of knowledge. The tools include both aids for accessing information and for learning[15]. A *travel metaphor* structures the navigation tools and provides the user with an intuitive context mechanism. Each screen display, be it text, animation sequence, image, movie or a combination, represents a place to visit. The various access facilities represent the ways and means of travelling around. Two explicit forms of navigation through the materials reflect the extremes of user-controlled (learner-controlled) and author-controlled access, as these terms were used in section 6.1, pages 90–91.

[12]Hammond N V "Hypermedia and learning: who guides whom?" *Computer assisted learning-ICCAL 89*, H. Maurer (editor), Springer-Verlag, pages 167–181,[52]1989. Hammond, Nick V. and Allinson, Lesley J. "Travels around a learning support environment: rambling, orienteering or tutoring?", *CHI '88 Conference Proceedings: Human Factors in Computer Systems,* Special Issue of the SIGCHI Bulletin, Elliot Soloway, Douglas Frye and Sylvia B. Sheppard (editors), ACM Press, pages 269–273,[53]1988.

[13]Morariu, J. and Schneiderman, B. "Design and research on the interactive encyclopedia system (TIES)" *Proceedings of the 29th Conference of the Association for the Development of Computer-based Instructional Systems,* pages 19–21,[85] 1986.

[14]Marchionini, G. and Schneiderman, B. "Finding facts and browsing knowledge in hypertext systems", IEEE Computer, 21,[80] 1988.

[15]Madigan and Chapman, previously mentioned work (footnote 10, page 174).

The forms are *go-it-alone travel* and *guided tours*[16].

The notion of elasticity, as defined in section 6.1, could be made useful in guided tours too, yielding what might be called 'elastic guided tours'. An elastic guided tour, rather than being a linear tour, would itself be an elastic story, with possibility of parallelism and branchings. If the 'passenger' remained inactive, the tour would find a way through; but if not, it would allow the user to choose a way.

Elasticity could be applied to go-it-alone travel too, thus completely softening the transition between the two extremes. Maybe some links could 'pull more than others'. Also, one link can be followed automatically if the user does nothing for a while. For the latter facility not to be annoying, there would probably have to be a way for the user to specify staying in the same place, so he or she can be allowed to study a node for as long as desired. Peter Bøgh Andersen, David Stotts and Richard Furuta, and Patrick Sénac, Roberto Willrich and Michel Diaz give examples where it would be useful for the system to move along if the user does not do so.[17]

As Halasz[18] indicated, navigational tools alone are not sufficient. The metaphor encompasses a range of access facilities along with the navigation tools: *maps* give users 'bird's eye' views of the material available and they can zoom in on any chosen node. An *index* provides a mechanism for keyword-based access. Ultimately both the maps and the index in Talaria could adapt to individual users' needs. Ichimura and Matsushita[19] suggest that maps may be problematic: it is hard to generate maps that communicate effectively the contents of the nodes. A combination of fisheye views[20] and pop-up node summaries may alleviate this problem. These are considered for implementation in Talaria.

[16]Trigg R "Guided tours and tabletops: Tools for communicating in a hypertext environmint", *ACM Transactions on Office information Systems,* 6, pages 398–414,[106] 1988.

[17]See for instance Patrick Sénac, Roberto Willrich & Michel Diaz: Hypermedia Synchronization Modelling: A Case Study. In Ed-media 95 World Conference on Educational Multimedia and Hypermedia Proceedings.[96] 1995.

[18]Halasz F "Reflections on NoteCards: Seven issues for the next generation of hypermedia systems", *Communications of the ACM,* 31, pages 836–852,[50] 1988.

[19]Ichimura S and Matsushita, Y. "Another dimension to hypermedia access", *Hypertext '93: Proceedings of the Fifth ACM Conference on Hypertext*, Seattle, pages 63–72,[59] 1993.

[20]See, for example Noik E G "Exploring large hyperdocuments: Fisheye views of nested networks", *Hypertext '93: Proceedings of the Fifth ACM Conference on Hypertext,* Seattle, pages 192–199,[89] 1993.

10.3 An implicit linking scheme

Kibby and Mayes[21] suggested a linking scheme that obviates the need for explicit links (although it does not preclude explicit links). The work presented in this chapter has focused on extending their ideas, linking them to recent progress in 'knowledge acquisition', and evaluating their effectiveness. The fundamental idea is to assign independently to each node a location in a high-dimensional 'context space'. Nodes that are close together form a 'neighbourhood' and link implicitly since they share a similar context. The author or the user defines the neighbourhood as the nodes within a certain distance of the current node, or as the nearest n nodes. Talaria currently adopts the latter approach and typically chooses n to be between 10 and 30 to yield a manageable neighbourhood size. Tudhope, Taylor and Beynon-Davies include a review of works on implicit linking in hypermedia based on similarity[22].

The decisive advantage of this scheme is modularity: an author can link a new or modified node to its associated nodes simply by rating it against each of a number of 'traits', thereby eliminating the requirement to examine all existing nodes.

10.3.1 Repertory grids

The context of each node is defined with a high-dimensional *trait vector*. Each element of a trait vector is a number representing the strength of association between the node and the corresponding trait[23]. For example, in the cancer pain guideline, section 3.3.2 (Dosage Titration) rated a 6 on 'Drug' and a 2 on 'Pain assessment' on a scale from 1 to 6. Every node is scored against every trait. Nodes that have similar ratings on a large number of traits will

[21]Kibby, M.R. and Mayes, J.T. "Towards intelligent hypertext", *Hypertext; theory into practice,* R. McAleese (editor), Ablex Publishing Corporation, New Jersey, and intellect books, Oxford, pages 164–172,[65] 1989.

[22]Douglas Tudhope, Carl Taylor & Paul Beynon-Davies: Navigation via Similarity in Hypermedia and Information Retrieval. In Rainer Kuhlen & Marc Rittberger (editors): Hypertext-Information Retrieval-Multimedia (HIM'95). HIM'95 conference proceedings. UVK Universitätsverlag Konstanz GmbH 1995.[107]

[23]Called feature or attribute in Kibby and Mayes, previously mentioned work[65].

be relatively near each other in the space spanned by the traits and will thus be *implicitly linked*. Nodes that have rather different trait vectors will be far apart in this context space, and will not link to each other.

	Sections					
Traits	**2.3.1**	**2.3.2**	**T4**	**2.3.3**	**2.3.4**	**2.3.5**
Pain assessment	2	2	1	1	1	1
Barriers to pain management	1	1	1	1	1	1
Bone	6	5	5	2	2	1
Central nervous system	3	6	6	4	1	1
Skin	1	1	1	1	4	5

Table 10.1. A repertory grid with five traits and six sections of the draft AHCPR cancer pain guideline. Here a 6-level rating scale is in use.

Table 10.1 shows a sample of trait vectors and corresponding traits for six nodes of the AHCPR Cancer Pain Guideline. (Here nodes are equated with sections, tables and figures in the guideline. In the future, it will be desirable to redo the chunking of the guideline into nodes[24].) Boose[25] referred to such a table as a 'repertory grid'. No work is believed to have been previously published on repertory grids in the hypermedia context.

For the rating, Kibby and Mayes[26] suggest a binary scheme; according to them, it is easy for authors to specify. They base their approach on the human memory models of Hintzman[27], who uses a three-point scale (−1 0 1). Waltz and Pollack[28], and later Gallant[29], present an essentially identical approach, but in the context of natural language recognition. They adopted 4-point and 5-point scales respectively. Boose et al.[30] in yet another context discussed the

[24]The chapters and sections of the guideline are written by different authors with different writing styles and different criteria for dividing a section into subsections. Currently node sizes vary between 10 lines and several pages, which is not always ideal. The translation into other media than text is likely to reveal new requirements on chunking.

[25]Boose, J. H. *Expertise Transfer for Expert System Design.*[20] Elsevier, New York, 1986.

[26]Previously referenced work[65].

[27]Hintzman, D. L. "Schema Abstraction in a multiple-trace memory model", *Psych Rev*, 93, 4, pages 411–428,[57] 1986.

[28]Waltz, D. L. and Pollack, J.B. "Massively parallel parsing: a strongly interactive model of natural language interpretation", *Cognitive Science,* 9, pages 51–74, 1985[111].

[29]Gallant, S. I. "A practical approach for representing context and for performing word sense disambiguation using neural networks", *Neural Computation*, 3, pages 293–309,[43] 1991.

[30]Boose, J. H., Shema, D.B., and Bradshaw, J.M. "Recent progress in AQUINAS: a

advantages and disadvantages of various nominal, ordinal, and continuous rating scales and suggested different scales for different traits. Anderson uses a 7-point scale[31]. Initially Talaria used a binary rating scale, but soon adopted a six point rating scale, as shown in table 10.2. The analysis by Madigan, Chapman, Gavrin, Villumsen & Boose suggests that a finer rating scale gives more useful links in the end.

Rating	*Operational definition*
6	Node is precisely concerned with this trait
5	Trait is a secondary topic in the node
4	More than a passing reference; less than a secondary topic
3	Explicit passing reference
2	Implicit passing reference
1	No mention implicit or otherwise

Table 10.2. The 6-level rating system.
A precise definition is critical for consistency in the rating.

The next two subsections discuss the construction of repertory grids.

10.3.2 Triadic elicitation of traits

From where do the traits come? Waltz and Pollack[32] suggest that traits 'should be chosen on the basis of first principles to correspond to the major distinctions humans make about situations in the world.' This rather general advice was found to be difficult to implement in practice. Fortunately, Boose and his colleagues[33] provided a formal methodology and software tools (MacQuinas and Dart) for identifying and analysing traits. They based their

knowledge acquisition workbench", *Knowledge Acquisition*, 1, pages 185–214,[22] 1989.

[31]Anderson, N. "Medical center: a modular hypermedia approach to program design", In: *Sociomedia: Multimedia, Hypermedia, and the Social Construction of Knowledge*, E. Barrett (editor), MIT Press, Cambridge, pages 369–389,[12] 1992.

[32]Previously referenced work[111].

[33]Boose J H "A knowledge acquisition program for expert systems based on personal construct psychology", *International Journal of Man-Machine Studies* 23, pages 495–525,[19] 1985. See also the two previously referenced works by Boose and by Boose, Shema & Bradshaw respectively.

approach on Kelly's personal construct theory[34].

In MacQuinas, an 'expert' first lists the possible solutions to a problem such as a medical diagnosis (the solutions in that case would be diagnostic categories). These correspond to nodes. Next, the expert specifies a collection of traits as follows: MacQuinas presents the solutions three at a time and asks the expert to identify what feature best distinguishes any one solution from the others. Kelly suggested these triads for efficiently identifying minimal sets of traits. Once the expert has identified the traits, he or she rates each solution against each trait to create the repertory grid.

[34]Kelly, G. A. *The Psychology of Personal Constructs*. New York: Norton,[64] 1955.

10.3.3 Grid analysis tools

MacQuinas contains a wealth of tools for analysing repertory grids, including trait implication graphs, trait and node cluster analyses, principal components analysis, hierarchical organization of traits and nodes, 'laddering tools' for expanding and contracting traits, and tools for combining the trait-spaces of multiple experts. Boose et al.[36] describe many of these techniques for analysing repertory grids. Cluster analysis was found to be particularly useful for identifying poorly discriminated nodes and redundant traits.

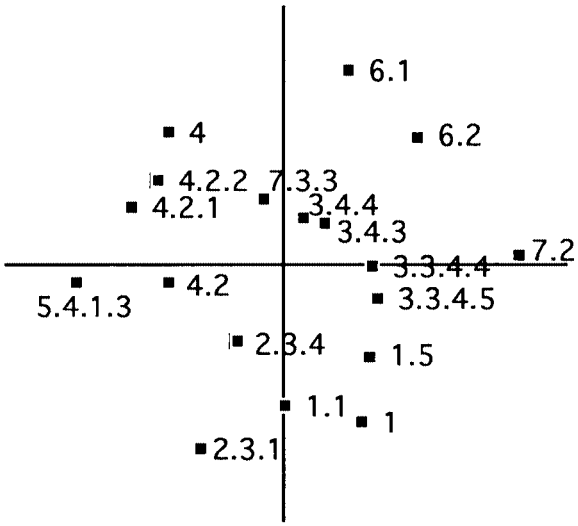


Figure 10.1: [34] MDS 2-Dimensional view of context space. This shows 18 sections from the AHCPR Cancer Pain Guideline. The plot has a similarity with the spatialized text plots of Marshall and Shipman[35].

Multidimensional scaling (MDS) provides an approximate three-dimensional projection of the nodes. Brushing and spinning tools are used to visualize the nodes in 3-D. Figure 10.1 shows a two-dimensional projection of 18 of the sections in the cancer pain guideline. This shows the relative location of the 18 nodes in a two-dimensional projection of context space.

[34] All figures in this chapter are taken from the original ECHT conference paper (hence they use American spelling).

[35] Marshall, C. C. and Shipman, F.M. "Searching for the missing link: Discovering implicit structure in spatial hypertext", *Hypertext '93: Proceedings of the Fifth ACM Conference on Hypertext*, Seattle, pages 217–230,[81] 1993.

[36] Boose, Shema & Bradshaw, previously referenced work[22].

10.4 Implementing the scheme for the cancer pain guideline

Using a single physician expert (JG[37]), a repertory grid for the complete AHCPR guideline with 29 traits and 136 nodes has been constructed. The sections and subsections of the guideline define the nodes.

10.4.1 Traits

Triadic elicitation of traits proceeded as follows: First, the expert selected 20 sections from the guideline to be representative of the material in the guideline. Next, random sets of three were selected from this set of 20 sections. For each triad, a single expert (again JG) identified a trait which 'best distinguished one of these from the other two.' He endeavoured to choose traits relevant for navigation. The procedure continued until five consecutive triads failed to elicit a new trait. At this point 29 distinct traits had resulted. Table 10.3 shows these.

[37]JG is a senior physician with 16 years clinical experience, half of which has focused on cancer pain management.

Surgical	Respiratory
Psychological	Sedative
Education	Ethics
Non-pharmacological management	Diagnosis
Procedural pain	Sleep
Analgesics	Opioid analgesic
Adverse outcome	Non-opioid analgesic
Pain assessment	Family
Barriers to pain management	Regulations
Bone	Demographics
Central nervous system	Anaesthesia
Skin	Indications
Treatment modalities	Mechanisms of pain
Monitoring patients	Mechanisms of pain relief
Drug	

Table 10.3. The 29 traits used for the Cancer Pain Guideline. AHCPR

Some traits are specializations of others. For example, 'Opioid analgesic' is a special case of 'Analgesics'. Currently all traits are treated on an equal footing. Alternative approaches are being investigated.

10.4.2 Rating procedure and grid analysis

The rating of the 136 nodes against the 29 traits required approximately 80 man-hours. To ensure consistency and accuracy, it is essential to have at least two people participating—the expert and a knowledge engineer.

Careful rating proved worthwhile: a randomly selected 20 sections were re-scored more than one month after the initial rating and never differed by more than one on the six-point scale. The key to such consistency is a clearly defined rating scale (table 10.2) and unambiguous trait descriptions. The complete grid is available from the author.

Grid analysis addresses two questions: Are all the elicited traits needed? Are they sufficient? The former question can be addressed with an analysis of the grid as discussed in the subsection 10.3.3 Grid analysis tools above and

subsection 10.5.4 Trait deletion below. However, the latter question requires dynamic analysis: do the links defined by the grid correspond to the links users make when using the guideline to address cancer pain management tasks? Such an analysis is now described.

10.5 Evaluation

10.5.1 Evaluation methodology

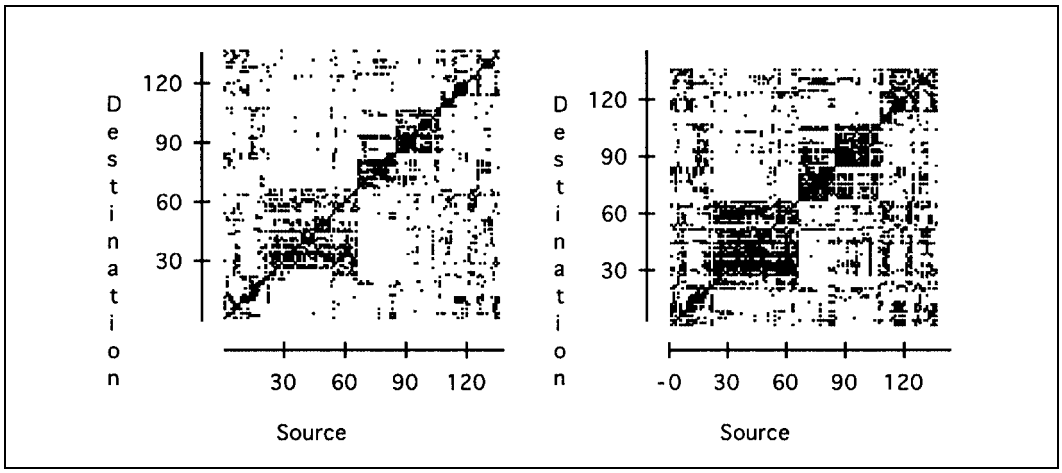


Figure 10.2: Linkplots for the 136 nodes in Talaria. Each dot represents a link. The plot to the left uses a neighbourhood size of 16 nodes while the right plot uses 30 nodes. The nodes are numbered in the order in which they appear in the cancer pain guideline. The rectangular structures in the plots reveal the chapter structure of the book. Note the linking scheme makes many links between nodes in different chapters in the guideline.

From the repertory grid, a matrix of all the distances between any two nodes is produced. The linkplots of Bernstein et al.[38] provide an insightful method for viewing the links implied by this matrix for different neigh-bourhood sizes; see figure 10.2 on the page 185.

[38]Bernstein, Bolter, Joyce and Mylonas [16].

In earlier pilot work, several potential users of Talaria assessed a sample of such distances. These distances were compared with those suggested by the repertory grid. The correspondence was close although the task seemed contrived. Ultimately it is intended to compare competing repertory grids with various distance metrics and trait and node weighting schemes on the basis of user performance in locating information in the hypermedia tool. Nielsen[39] described several hypermedia usability tests. Initially however, rather than confound the linkage analysis with software-specific issues such as interface design, a protocol analysis was conducted[40] using the guideline booklet itself.

Four cancer pain management tasks from the case studies prepared by the Wisconsin State Cancer Pain Initiative were selected. These case studies are prototypical of cancer pain management problems; instructional workshops throughout the USA use them.

Six subjects participated in the analysis: a senior pain service physician, a primary care internist, a senior pain service nurse, a pain service resident physician, a family practitioner, and a paediatric nurse practitioner. The subjects used the guideline booklet to address the four tasks. They were instructed to 'describe as fully as possible your thoughts as you browse the guidelines for the information you seek to address each task.' The subjects' use of the book was recorded throughout. The video recording was used for subsequent clarification. The sessions ranged from 45 to 90 minutes.

When a subject, while addressing a single task, successively visited two sections that both contributed to the task, that was deemed to be a link. This definition of a link is simple and close to being objectively decidable. The subjects visited 24 distinct nodes and made from one to eight links, for a total of 30 links. Among these were 25 distinct links, since some of them were each made by 2 or 3 of the subjects. In the future it is intended to add more open-ended tasks to explore other nodes in the book.

[39]Nielsen, J. *Hypertext and Hypermedia.* New York: Academic Press, 1990.[87]

[40]See for example the references in Boose, J.H. "Knowledge acquisition tools, methods, and mediating representations" H. Motoda, R. Mizoguchi, J. Boose, B. Gaines (editors), *Knowledge Acquisition for Knowledge-Based Systems,* Proceedings from JKAW'90 (The First Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop) IOS Press, Amsterdam, and Ohmsha, Ltd., Tokyo,[21] 1991.

10.5.2 Evaluation of linking scheme

For each of the links made by the subjects, the Euclidean distance from the source of the link to all the nodes in the document was calculated (other distance metrics are discussed below). The rank order of the destination of the link then was calculated. For example, a subject made a link from section 7.4 (Substance Abusers) to section 1.5.1 (Legal Regulation of Opioids). The distances from section 7.4 to all the other nodes were rank ordered. Section 1.5.1 was the fifth nearest. Thus the link from 7.4 to 1.5.1 would be assigned a rank order of five.

Clearly it is preferable to have the destination of each link in the neighbourhood of its source. Equally, to minimize cognitive overload, small neighbourhood sizes are wanted.[41] Taken together, the rank order of the links made by the subjects is required to be as small as possible. Figure 10.3 shows a plot of the rank orders of the 30 links.

As the figure shows, Talaria requires a neighbourhood size of at least 28 to capture all the links made by the subjects in the protocol analysis. A neighbourhood size of 17 captures 24 of the 30 links or 80 % of them[42]. Thus for a range of tasks and users, the repertory grid, with a manageable neighbourhood size, is shown to capture the links made by the subjects. Counting only the 25 distinct links, results are very close to the same; a neighbour-hood size of 17 still captures 80 % of the links, and one of 28 all of them.

Had the links—theoretically—been created at random then with a neighbourhood size of $20, 20/135 \approx 15\%$ of the links could have been expected to be captured, since each node can potentially link to each of the 135 other nodes. So the evaluation clearly shows that link creation is far from random.

If the links made by the subjects are taken to be useful links that should be in Talaria, it is obvious that the above evaluation gives some measure on the

[41] Jeff Conklin Hypertext: An Introduction and Survey, *IEEE Computer*, vol. 20 no. 9, pages 17–41, September 1987[30].

[42] The original conference paper reported that a neighbourhood size of 16 did this. The deviation is due to the fact that two nodes have exactly the same distance from section 3.3.2 of the guideline, the two ranking 16th and 17th in distance. Hence it is not defined which of them would be included in a neighbourhood of size 16. In the thesis, conservative judgements have been made throughout, requiring a neighbourhood big enough to include all nodes that have the same or a smaller distance than the destination node in question.

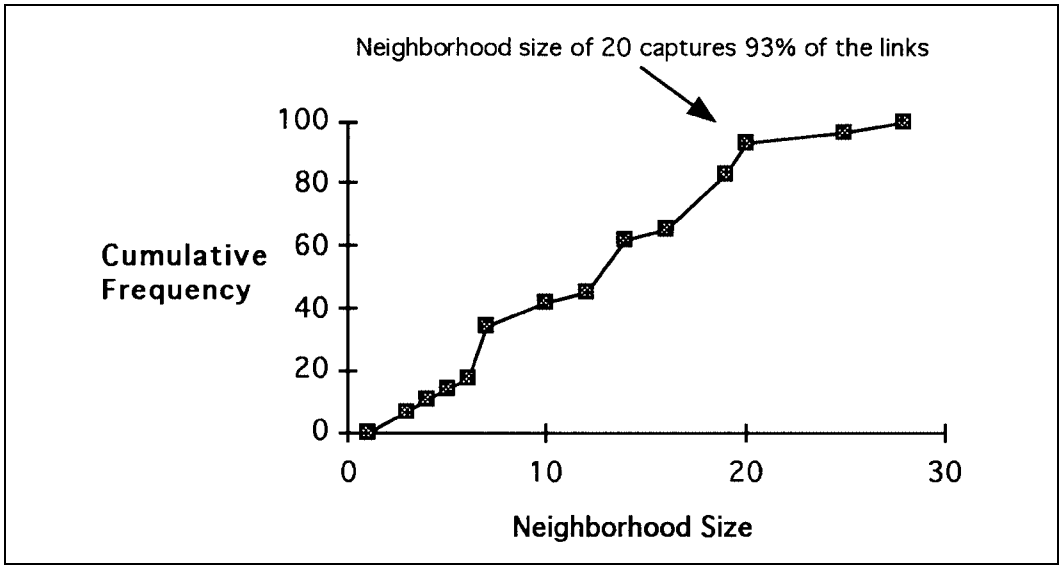


Figure 10.3: Plot of the percentage of the links made by the users in the protocol analysis against neighbourhood size. Ideally, small neighbourhoods would capture most or all of the links made by the subjects. A neighbourhood of size n includes the n nearest nodes.

quality of the linking scheme. Had the linking scheme captured only a few of the links, it would have indicated a poor scheme. In comparing variants of the scheme, this form of evaluation can give a hint about which variant is best. On the other hand, it would probably be a bad idea to maximize uncritically the number of subject-made links captured, by weighting the traits or making other adjustments, since it might be at the cost of eliminating other links that were perhaps just as useful in other situations.

10.5.3 Distance metric evaluation

Euclidean distance represents what is, perhaps, the most natural distance metric. However, Madigan, Chapman, Gavrin, Villumsen & Boose[43] carry out a detailed investigation of six different metrics in addition to Euclidean distance, including two that were invented for Talaria. The results show that Euclidean distance is the best of the seven metrics, while 'American City'

[43]Previously mentioned article.cite76

distance and the Pearson correlation coefficient squared (known as R^2) are also good. American City block distance is defined as the sum of the absolute pairwise differences[44].

Two of the best metrics, American City and Euclidean, are well-known special cases of *p-norms*[45]. In other words, both can be described as taking the absolute pairwise differences to some power p, summing the results, and taking the pth root of the sum, with $p = 1$ for American City distance and $p = 2$ for Euclidean. A p-norm with p lying between 1 and 2 turns out to perform not much differently from Euclidean, except for the maximum (the most distant links): with $p = 1.6$, a neighbourhood size of 23 now captures all the links, which is a noticeable improvement over Euclidean distance's 28.

10.5.4 Trait deletion

It is felt that the trait set used is not ideal. Maybe a user does not desire to navigate from a node concerned with the adverse outcome of one treatment to one on an adverse outcome of a completely different treatment. One might speculate that traits such as Adverse outcome, Treatment modalities, Indications, Anaesthesia and Skin may sometimes lead the user to irrelevant rather than helpful nodes. A completely different question is whether it is advantageous to have traits that are specializations of other traits, as discussed in the last paragraph of subsection 10.4.1.

Some experiments were performed to see how the linking scheme would behave with fewer than all 29 traits.

First, the trait Analgesics was removed since it is nothing but a generic term for two other traits, Opioid analgesic and Non-opioid analgesic, and hence does not seem to provide additional information about the nodes. Performance got slightly worse, measured by the links made by the subjects: it now requires a neighbourhood size of 23 to capture 90 % of the links (using either

[44]Popularly speaking, the American City distance is the travelling distance between two points in an American city, in which the streets are orthogonal, generalized to more than 2 dimensions.

[45]p-norms are treated in many textbooks. See, for instance, page 53 in Gene H. Golub & Charles F. Van Loan: Matrix Computations. Second edition. The Johns Hopkins University Press 1989[45].

Euclidean distance or the p-norm with $p = 1.6$). This may suggest that it is a good idea to keep traits that are generalizations of other traits. Alternatively, the two specializations just need higher weights in the repertory grid. Certainly, no hard conclusions can be drawn from this simple observation.

Instead, four traits were deleted that were suspected of generation of confusing rather than useful links: Adverse outcome, Treatment modalities, Anaesthesia and Indications. An interesting result occurred. Most links got closer. Using the 1.6-norm, the average rank dropped from 11.44 to 10.4. One would immediately consider this an improvement and a confirmation of the suspicion. However, the upper percentiles and the maximum got worse. To capture 90 % of the links, a neighbourhood size of 24 is now needed (23 if using the square of the Pearson correlation coefficient). To conclude, while the four traits may have generated confusing links, some of them must also have helped generate some helpful ones.

Entering the ground of uncritical optimization, it was found that deleting seven of the traits: Adverse outcome, Central nervous system, Skin, Treatment modalities, Family, Demographics and Mechanisms of pain, improves performance. Using the 1.6-norm, a neighbourhood size of 16 now captures all 25 of the links made by the subjects and a neighbourhood size of 13 captures 23 of the links (92 %). As discussed earlier, this result does not necessarily indicate that the mentioned traits should be eliminated. For example, since the guideline makes distinctions based on mechanisms of pain, it is easy to imagine that this trait may be useful in some situations, even though it does not seem that it was in the 4 tasks that the subjects performed. The result does confirm that a critical review of the need for each trait may turn some of them useless and result in better performance of the repertory grid. Testing the grid on a broader range of tasks may clarify the question.

10.6 Discussion and conclusion

10.6.1 Discussion

An implicit linking scheme has been proposed and used to develop a hypermedia implementation of the AHCPR cancer pain guideline. A protocol

analysis suggests that the scheme captures efficiently the links made by users of the guideline. Clinical practice guidelines must be current; hypermedia has much to offer here.

Currently many extensions to the basic scheme are being explored:

- The scheme does not preclude having a small number of additional author- or user-specified explicit links. Users may annotate the guideline too.
- The inter-node distance provides a mechanism for implementing a scaled rather than binary link. Salton et al.[46] make a similar proposal, but in the context of text retrieval. This may implement some elasticity, a little author control: some links may 'pull' more than others. In particular, it may be used as a guide to which link to follow if the user does not choose one, as discussed in section 10.2.
- Chang's HieNet by default creates a link between the smallest pair of nodes in a neighbourhood.[47] This notion of parsimony may be useful in the guideline context.
- Currently the links in Talaria are untyped and connect entire nodes. Generalizations of this may be explored in the future.

10.6.2 Summary

This chapter has sketched a learning support environment based on Hammond and Allinson's travel metaphor. The environment consists of a hypermedia database with map, index, guided tours and go-it-alone hypermedia navigation. Especially in a learning environment, the balance between user and author control may be critical. Therefore, the guided tours may be elastic. As basis for the hypermedia navigation, a novel linking scheme has been developed based on repertory grids. The linking scheme provides for a scaled

[46]Salton, G., Allan, J., and Buckley, C. "Automatic structuring and retrieval of large text files", *Communications of the ACM*, 37, pages 97–108, 1994[93].

[47]Chang, D. T. "HieNet: A user-centered approach for automatic link generation", *Hypertext 93: Proceedins of the Fifih ACM Conference on Hypertext,* Seattle, pages l45–158, 1993.[29]

rather than a binary link, which makes elasticity possible in the navigation too: the closest link is the one that attracts most strongly. An initial evaluation of the linking scheme was performed, using the case of the cancer pain guideline and comparing with the results of a protocol analysis of users using the guideline. Results were satisfactory and suggested that a further refinement of the repertory grid is possible.

Chapter 11

Conclusion

This thesis has explored tools for development of stored, interactive multimedia. General multimedia tool requirements have been identified, and tools and techniques have been developed for specific purposes and developers. The largest part of the thesis is concerned with Petri nets as a tool for building elastic stories in multimedia. Other parts present repertory grids for semi-automatic generation and maintenance of links in hypermedia documents, and Hejmdal, an object-oriented multimedia tool for programmers.

11.1 Petri nets for elastic story telling

The use of Petri nets for elastic story telling has been explored. Elastic systems used for elastic story telling form a soft middle ground between author- controlled and user-controlled systems. An elastic system is neither hard like a book or a sculpture nor completely yielding to the user like wet clay or blank paper.

First, a model of elastic stories was developed, based on the extended layer model from New Criticism. Petri nets are found to be well suited for the elastic story telling, since most of the concepts in the model have very simple and straightforward descriptions in Petri nets. Also the rhetoric structures used by Peter Bøgh Andersen in the Wodan's Eye project are not difficult to mimic in Petri nets. They include a parenthesis, an escalation and a

counterpoint.

Petri nets can be used today to describe elastic stories formally. Once Petri net tools with multimedia capabilities become available, it will also be possible to use them for implementing elastic stories in computers. Work is going on in this direction: As mentioned in section 9.3, Patrick Sénac and colleagues work on automatic generation of MHEG documents from HT-SPNs. In the Devise project, Design/CPN is being integrated with Devise Hypermedia (DHM), which will allow the former to access the latter.

Experiments indicate that Petri nets fit authoring processes well, and that Design/CPN is a flexible and natural tool for authoring of elastic stories. In the future, empirical studies with real multimedia authors should be carried out to determine how well Petri nets suit such authors.

11.1.1 Problems, solutions and further possibilities

A tool built around a new syntactic layer on top of Petri nets may be more realistically usable for authors than Petri nets themselves, especially when it comes to resumptions. Such a tool is discussed in the next subsection. The following subsection discusses the need for giving different priorities to threads or events, as with the force or pressure used by Peter Bøgh Andersen. A further subsection is concerned with the possibility of using Petri nets for describing a wider class of multimedia: fully user-controlled and fully author-controlled multimedia systems, besides elastic systems.

A tool with a new syntactic layer

Most multimedia authors probably will not want to work out the finer details of a Petri net used for an elastic story. One possibility is to create a new syntax on top of the Petri nets. The new notation should reflect the concepts in the model of elstic stories still better than Petri nets. While most concepts have simple and natural Petri net representations, the resumption would benefit from a much simpler representation.

A new syntactic layer should probably still be graphical, to give a clear representation of the structure of an elastic story with forks, choices, etc. A

tool should make it possible to edit and run elastic stories described in the new notation. The Petri net descriptions given in the thesis would serve as definitions of the semantics of the new notation. The execution of elastic stories described in the new syntax could be implemented by translation to Petri nets or by other means.

There are some possible advantages of a new tool desired specifically for authors of elastic stories, in addition to being realistically learnable for such authors:

1. Convenience. For example, when an elastic story is described as a Petri net, the author is to assign a colour set to each place and an arc inscription to each arc in the net. A specialized syntax can relieve the author from this work.

2. Safety. More high-level specification can be allowed, thereby eliminating some error possibilities. For instance, a generalization in a Petri net needs to take a set of tokens from a synchronization place and put the same set of tokens back. The tokens represent the examples that have happened previously and make the generalization meaningful. A guard specifies the number of examples and (usually) that they all need to be different. A new notation may require just the number of examples to be specified, thereby eliminating the possibility of making errors in the guard or forgetting to return the tokens from the generalization. (This increases both convenience and safety.)

3. Clarity. In many instances, extra transitions are needed in a Petri net to accomplish seemingly small effects, specifically in inter-event synchronization and in pauses. A new notation can hide this need from the author. For instance, it may allow (what corresponds to) guards to be applied directly to events.

The design of such a new syntactic layer is not trivial. Specifically, many trade-offs are expected between safety, ease of use, and natural representations of the author's concepts on one hand, and the flexibility and generality offered by Petri nets on the other. As a very simple example, two variants of inter-event synchronization have been presented: one in which the second event can happen arbitrarily many times after the first; and one in which it can only happen as many times as the first. In a notation in which inter-event

synchronization has its own representation, supporting two variants would be an extra complication. It is not clear *a priori* whether the increased flexibility is worth this extra complication.

Many multimedia authors find that it is neither satisfying nor productive to have someone else—a computer programmer—make their ideas reality. While Petri nets may allow authors to work directly on implementation, most authors will probably require a Petri net programmer to complete the nets, which makes experimentation more complicated. A new tool built on top of the nets may allow the author to build and experiment with elastic stories without such assistance.

A tool built around new syntactic layer may not be considered a programmer's tool; it is hoped that it would be easier to use. The syntax would, however, be largely symbolic rather than iconic, as these words were used in chapter 3. There would still be a need for a multimedia author to spend time learning the syntax and semantics before he or she could use the tool.

Finally, if such a syntactic layer turns out to be as easy to use as it is hoped, it will make it easier for a user to 'change sides' and become an author, or just start modifying the story he or she is reading.

Topicality

Although generally the Petri net formalism serves the purpose of describing elastic stories well, two problems with resumptions have been revealed: no method has yet been devised for deciding where to insert resumptions; and resumptions may loop (the latter may not be considered a problem). Ideally, one would not want the resumption to commence until it is known that the thread can continue immediately after the resumption. Peter Bøgh Andersen has solved similar problems by assigning a force or pressure to events: the higher the pressure, the higher is the probability that the event in question is the next event to happen, in case there are multiple current (enabled) events (see page 150). This solution may also be a good one in the model of elastic stories used in this thesis: if a given event makes sure that the next event is the most topical (current) event, the only risk of the thread being interrupted is when the next event is not enabled due to a pause or inter-event synchronization. Thus, resumptions are needed exactly before events after

pauses and before events that can be delayed by inter-event synchronization. Furthermore, if the last event of a resumption makes the next event in the original thread the most topical one, the thread will continue immediately after the resumption. This prevents looping.

Implementing a general topicality (currency) of events in Petri nets is possible, but very cumbersome and awkward. The problem is that in a Petri net with multiple enabled transitions, there is no control of which transition fires first. In other words, there is no way of giving transitions different priority. This means that describing an elastic story with a topicality of each event entirely in Petri nets is probably not worthwhile.

There is another possibility, theoretically. Given that a new notation is built on top of Petri nets as described above, the semantics of this notation could be founded on Petri nets for the most of it, but additionally be required to respect topicality. This means that executing an elastic story requires that the execution conforms with the Petri nets given in chapter 7, but there is an additional requirement that in case of multiple enabled events, the most topical one is executed. Note that the additional requirement does not conflict with the Petri net semantics; it only strengthens it.

As an alternative, topicality can be assigned to entire threads rather than to individual events. This would be more manageable in Petri nets. If this alternative is chosen, a Petri net describing an elastic story would contain a central topicality management module. All threads that use central resources such as the virtual speech channel are controlled by this module, and exactly one is allowed to be active at a time. Additional threads that do not use the central resources and thus do not interfere with resumptions, can be allowed to be active at all times (for example, threads that handle background sounds). When the active thread reaches a point where it cannot continue because of a pause or inter-event synchronization, it gives control back to the topicality manager, which activates a new thread—the one that now has the greatest topicality. The topicality manager thus has a role similar to the process manager in a multitasking system.

Petri nets for a wider class of interactive stories

This thesis has argued that Petri nets are well suited for elastic stories. It may

be interesting to ask how well Petri nets would work for more user-controlled and for more author-controlled systems. Answering this question will give a hint as to whether to use Petri nets for systems containing user-controlled and author-controlled as well as elastic parts.

David Stotts and Richard Furuta have successfully used Petri nets for hypermedia with a greater degree of user control than in elastic stories. A purely author-controlled presentation could be built in a Petri net with only a single thread. This resembles the use of HTSPN on the composite synchronization layer, except for the timing information in HTSPN. These observations suggest that Petri nets easily span the range from user-controlled to author-controlled multimedia.

11.1.2 Future work

The most important future direction of the research on Petri nets for elastic story telling will be the development of a new syntactic layer on top of the Petri nets, as described above.

Another important thing to do is practical testing of Petri nets with real multimedia authors.

The use of Petri nets with other styles of user interface is yet to be explored. Of the concepts in the model of an elastic story in chapter 6, only very few rely directly on the style of user interface presented in the same chapter: pauses and choices rely on the kinds of tests that are possible; the kinds of actions that can be in an event depend on the user interface too. Another style of user interface would still allow pauses, choices and actions. Only the situation waited for in a pause and the ways of specifying the thread followed after a choice would be different, as would the kinds of actions. It would be interesting to explore the use of Petri nets with user interfaces where these three concepts were different; for example, hypermedia-like interfaces where a typical action would be to go to another node.

In the use of CPN presented, all user input is mediated to the net via guards that can be true or false. It will be worth investigating whether in-put on a continuous rather than binary form would be useful. As a simple example of the use of continuous input, the sound volume of the slave's cries (see pages

152—153 and 155—157) could depend on the distance between the user and the slave.

Finally, ways to let the user interrupt the system's pannings and character movements, as in the Wodan's Eye system (see page 157), may be explored.

11.1.3 Petri nets for elastic story telling: Summary

Elastic stories in multimedia may have a great potential still to be explored. Current multimedia authoring tools only indirectly and poorly support the authoring of elastic stories. This thesis argues that Petri nets form a better basis for building interactive stories, since they model the concepts found in elastic stories more directly than other tools. Some practical experiments confirm this, and suggest that Petri nets constitute a natural tool for working with elastic stories. Empiric studies in the form of realistic experiments with real-world multimedia authors using Petri nets would further explore this hvpothesis.

11.2 An object-oriented programmer's platform for multimedia

Hejmdal is an object-oriented class library for the interactive creation, editing and playback of multimedia. Hejmdal was originally developed as a platform to be used for the remainder of the thesis work. It has been used for developing movie nodes in the Macintosh version of DHM. When the latter is integrated with Design/ CPN, it will be possible to use DHM (and thus indirectly Hejmdal) to add multimedia to elastic stories described in Petri nets with Design/CPN, thus overcoming a current limitation in the use of Petri nets for elastic stories. At the same time, Hejmdal may take part in meeting the increasing demand for programmer's tools for multimedia. Hejmdal was also seriously considered for use in the Talaria project.

Hejmdal has good support for playback of multimedia consisting of still pictures, videos, animations and sounds. It has some support for editing, and a little support for the creation of new multimedia documents. Hejmdal

can be used for a range of applications, from those adding a few elements of animation or sound to an existing application to the 'pure' multimedia application programs. Hejmdal is built on QuickTime. It was found that Hejmdal provided a clear, object-oriented model and that it was easier to use than QuickTime, especially for interaction, without loss of power and flexibility.

It has further been argued that it should be possible to define interaction in the multimedia documents, not only in the application programs using them.

11.3 Repertory grids for hypermedia linking

Talaria is a multimedia reference tool on cancer pain management for health care providers, being built at Fred Hutchinson Cancer Research Center.

A novel hypermedia linking scheme based on repertory grids has been developed for Talaria. The scheme performs automatic or implicit linking of nodes in the learning support environment, based on the distance between nodes in a context space. The location of each node in context space is defined by a trait vector assigned to the node by the author or another expert in collaboration with a knowledge engineer. The traits are found by triadic trait elicitation, that is, an expert repeatedly looks at three random nodes and identifies the feature that best distinguishes one of them from the two others. A six-point rating scale for scoring each node against each trait has been invented and described to ensure consistent rating. An evaluation of the linking scheme was performed, based on a protocol analysis. Within a neighbourhood size of 20 nodes, the linking scheme captured 93 % of the links made by the users in the protocol analysis. The evaluation further suggested that the six-point rating scale provided better performance than binary rating, and that refinement of the repertory grid may lead to improved performance.

The notion of elasticity is relevant in the learning support environment. Since the linking scheme is based on distance, it may support elasticity: the closest neighbour may perform the strongest attraction. The learning support envirnment may also contain elastic guided tours.

11.4 General multimedia tool requirements

During the work on Talaria, a number of observations about requirements for multimedia tools were made:

- Editors are needed for each medium used in the multimedia (text, graphics, sound, video, animation).
- Editors should allow the importing (and digitizing, of course) of analogue material, as well as on-line creation and editing of digital material. The editing tools should be separate (allowing you to use only one at a time), yet integrated with each other (sharing the same material) and with the programming environment (if any; allowing the media to be integrated with code).
- Conventional database technology seems insufficient for building a well-structured media database.
- The need for interpretative execution of the program during development is found to be even greater in multimedia than in other fields. At the same time it is advantageous *also* to have a compiler.

Furthermore, the following requirements are found to be little or no different from the requirements for programmer's tools in other fields: strong typing; integration of code and media data; and facilities for structuring of data and code.

The need for programming in multimedia development was investigated. Many multimedia developers seek to avoid or limit programming in the development process. However, based on Paul Brown[27] and indirectly on Charles Saunders Pierce, it was argued that symbolic languages, such as programming languages, press the author to get better acquainted with the computer and the inner workings of it, which promotes a better understanding of its potential and invites for better exploitation of the computer's possibilities and the author's creativity. Here, symbolic languages are seen as a contrast to iconic languages, in which language elements bear some resemblance with that which they signify. The argument of Paul Brown was confirmed by observations from Talaria and other projects. Scripting or programming has its place in most and the most interesting multimedia development projects.

Therefore, the multi-media developer who wants to exploit the computer's potential and his or her own creativity should learn programming.

On this background, Hejmdal is developed as a programmer's tool and is purely symbolic. The other tools studied and developed in the thesis are also rather symbolic than iconic. The repertory grids used in Talaria are purely symbolic. Petri nets are graphical and symbolic; they are traditionally a programmer's tool. A new notation built on top of Petri nets to reflect the model of elastic stories will have to be mostly symbolic. Probably icons can be drawn for threads and forks, but probably not for events, choices, synchronization, pauses and resumptions.

11.5 Summary

The main results of this thesis are:

- General tool requirements for multimedia development correspond to requirements for tools for system development in other fields. In addition, multimedia development puts demands on editors for importing, creation and editing material in the various media used in the multimedia.

- Programming has its place in most multimedia projects, and specifically in the most interesting of them.

- Hejmdal, the object-oriented class library for handling of interactive, time-based multimedia documents in the form of QuickTime movies, has been developed. The object-oriented model of Hejmdal is found to be simple, clear and powerful, without loss of flexibility compared to the underlying QuickTime.

- Petri nets form a better basis for building elastic stories than currently available multimedia tools, as the former model the concepts found in elastic stories more directly than other tools. Petri nets can probably be advantageously used for describing systems in the range from purely user-controlled to purely author-controlled svstems. A new and more author-friendly syntactic layer on top of Petri nets will probably be

still better. The introduction of topicality in elastic stories will also constitute an improvement.

- Repertory grids have been shown to form a workable basis for semi-automatic linking of hypermedia documents.

Appendix A

Petri Net Experiments

This is the 'code appendix'. It contains the Petri nets built during the experiments reported in chapter 8, and in some cases also examples of output from the nets.

A.1 Kristendom (Christianity)

The Petri net

A summary of the elastic story told in this net is given on page 135. Readers who do not understand Danish should be able to follow the tracks by means of the figure captions.

In the net, the interested reader can study the following details: The scene, modelling the background picture with the characters is shown in figure A.3. By clicking on a character, the user models the situation where that character is visible in the view of the background. his gives as much control over the story as the multimedia version would.

Initialization happens in the code region in figure A.4. Here the report page for text output is set up. The WriteHistory function, defined in the global declaration node in figure A.2 and called from events throughout the net, brings this page to the front and appends a line of text to it.

The net is modularized into relatively small unit for readability. In some cases modularization was done after the construction of page contents; for instance, page Intro#6 (figure A.6) was separated from page Kristen#5 (figure A.5) after the net elements on the former page and the surrounding elements on the latter had been drawn. This posed no problems.

[0]Pages in a Coloured Petri Net are designated by page name and page number, separated by '#'. Thus 'Hierarchy#1' denotes the page named 'Hierarchy', having number 1.

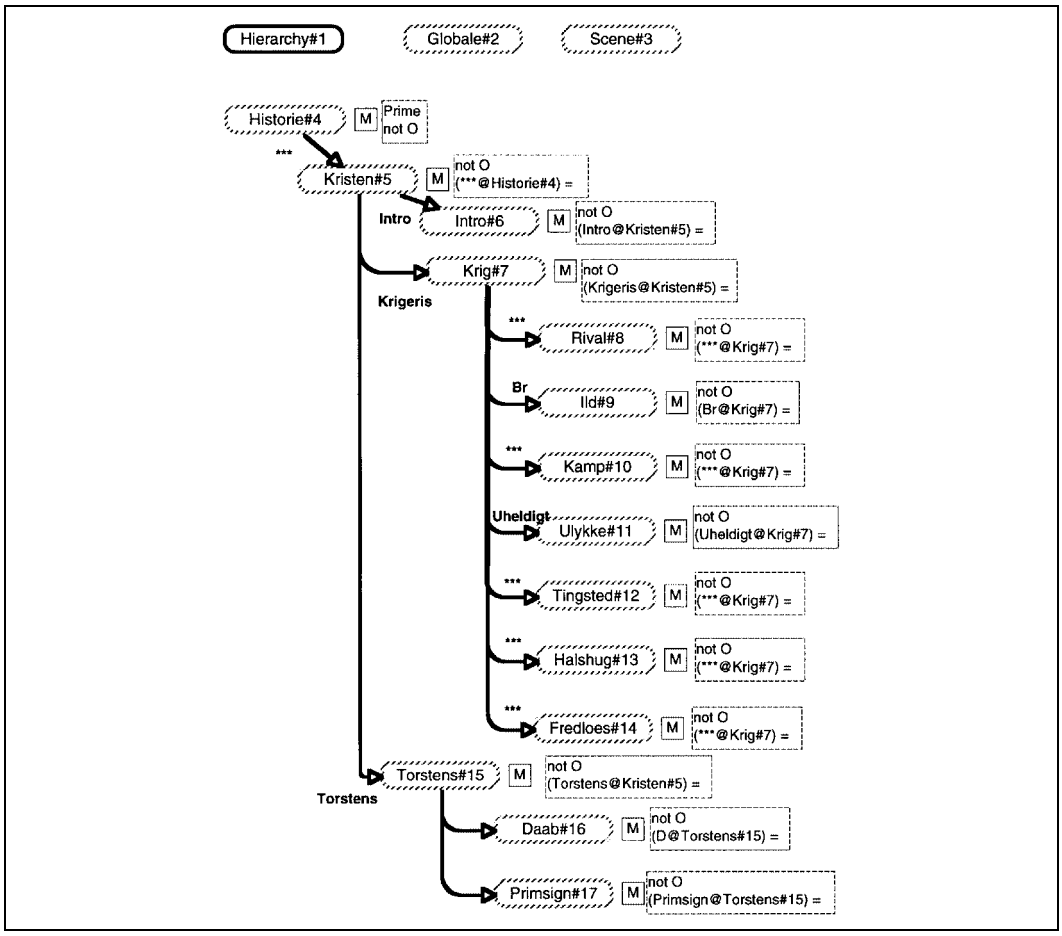
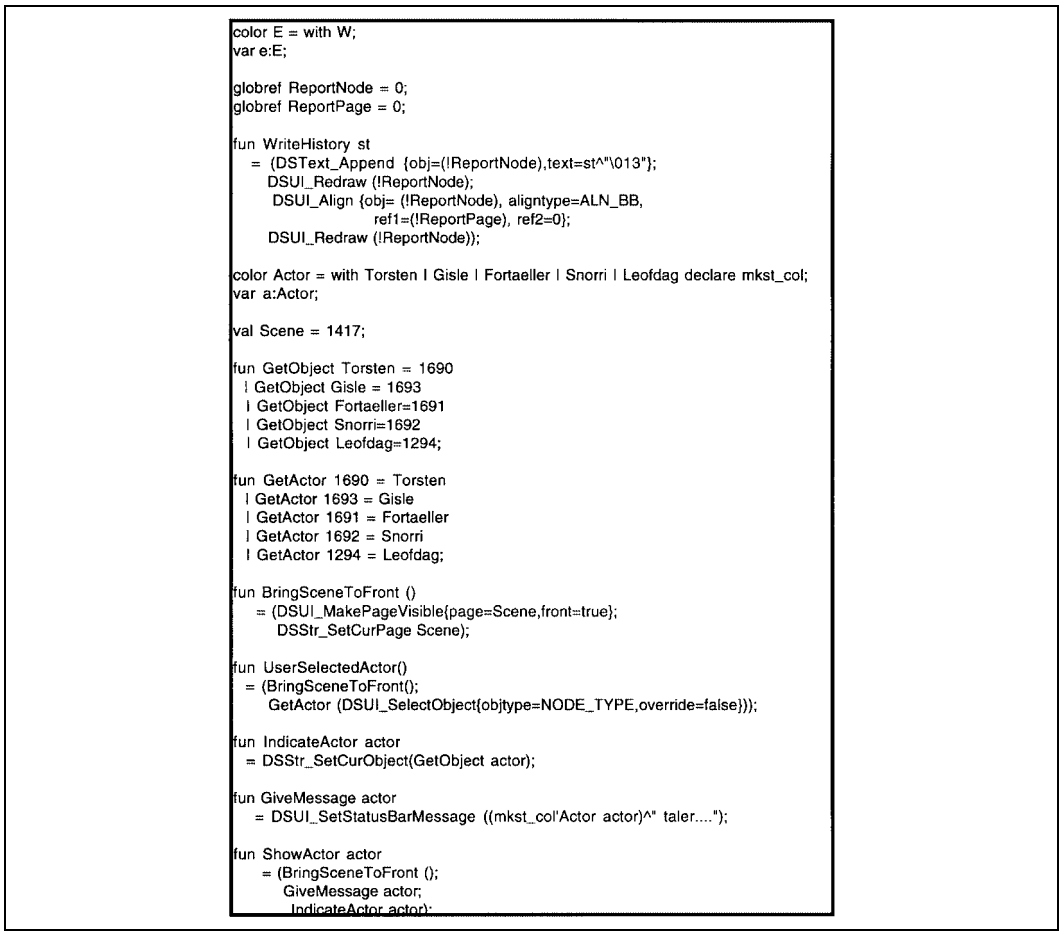


Figure A.1: Page 'Hierarchy#1' [0], the page hierarchy page: overview of the pages in the net.



```
color E = with W;
var e:E;

globref ReportNode = 0;
globref ReportPage = 0;

fun WriteHistory st
   = (DSText_Append {obj=(!ReportNode),text=st^"\013"};
      DSUI_Redraw (!ReportNode);
       DSUI_Align {obj= (!ReportNode), aligntype=ALN_BB,
                   ref1=(!ReportPage), ref2=0};
      DSUI_Redraw (!ReportNode));

color Actor = with Torsten | Gisle | Fortaeller | Snorri | Leofdag declare mkst_col;
var a:Actor;

val Scene = 1417;

fun GetObject Torsten = 1690
  | GetObject Gisle = 1693
  | GetObject Fortaeller=1691
  | GetObject Snorri=1692
  | GetObject Leofdag=1294;

fun GetActor 1690 = Torsten
  | GetActor 1693 = Gisle
  | GetActor 1691 = Fortaeller
  | GetActor 1692 = Snorri
  | GetActor 1294 = Leofdag;

fun BringSceneToFront ()
    = (DSUI_MakePageVisible{page=Scene,front=true};
       DSStr_SetCurPage Scene);

fun UserSelectedActor()
  = (BringSceneToFront();
      GetActor (DSUI_SelectObject{objtype=NODE_TYPE,override=false}));

fun IndicateActor actor
  = DSStr_SetCurObject(GetObject actor);

fun GiveMessage actor
    = DSUI_SetStatusBarMessage ((mkst_col'Actor actor)^" taler....");

fun ShowActor actor
     = (BringSceneToFront ();
        GiveMessage actor;
        IndicateActor actor);
```

Figure A.2: Page 'Globale#2', containing the global declarations.

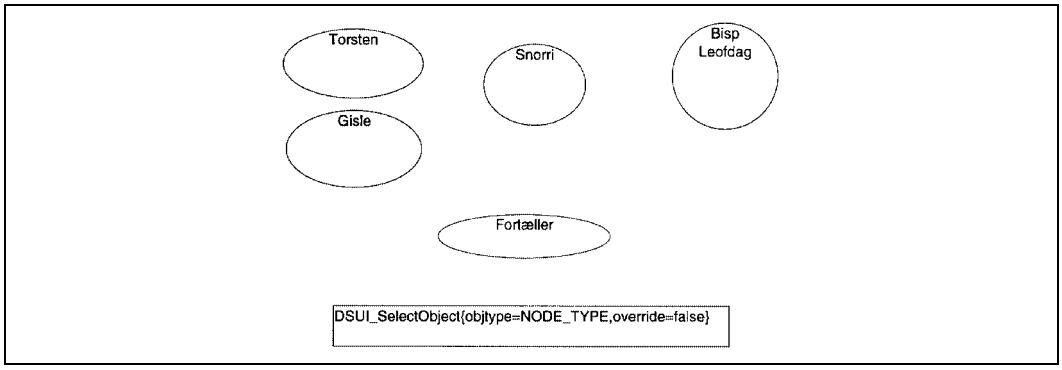


Figure A.3: Page 'Scene#3', the scene with five actors. The scene is used for marking actors when they speak and for user input. The box at the bottom containing SML code is for use during construction and modification of the net.

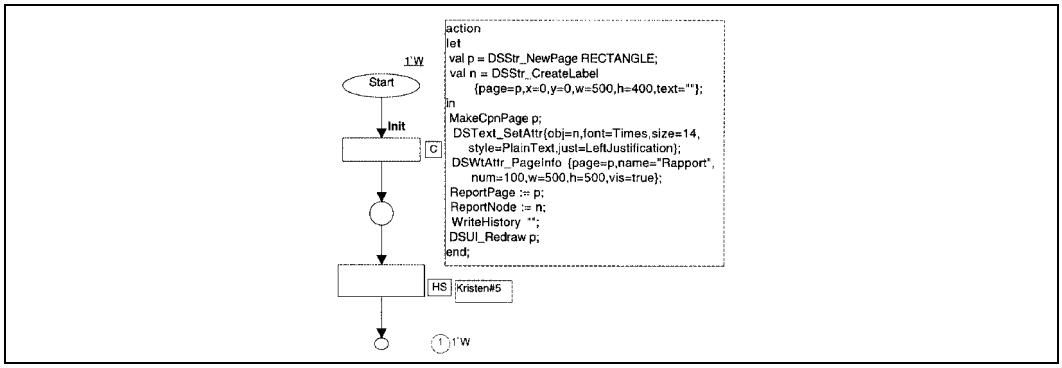


Figure A.4: Page 'Historie#4'. The highest level view of the Petri net; the only prime page of the net. The topmost transition is for initialisation, while the entire story is contained in the subpage at the bottom (page 'Kristen#5'.)

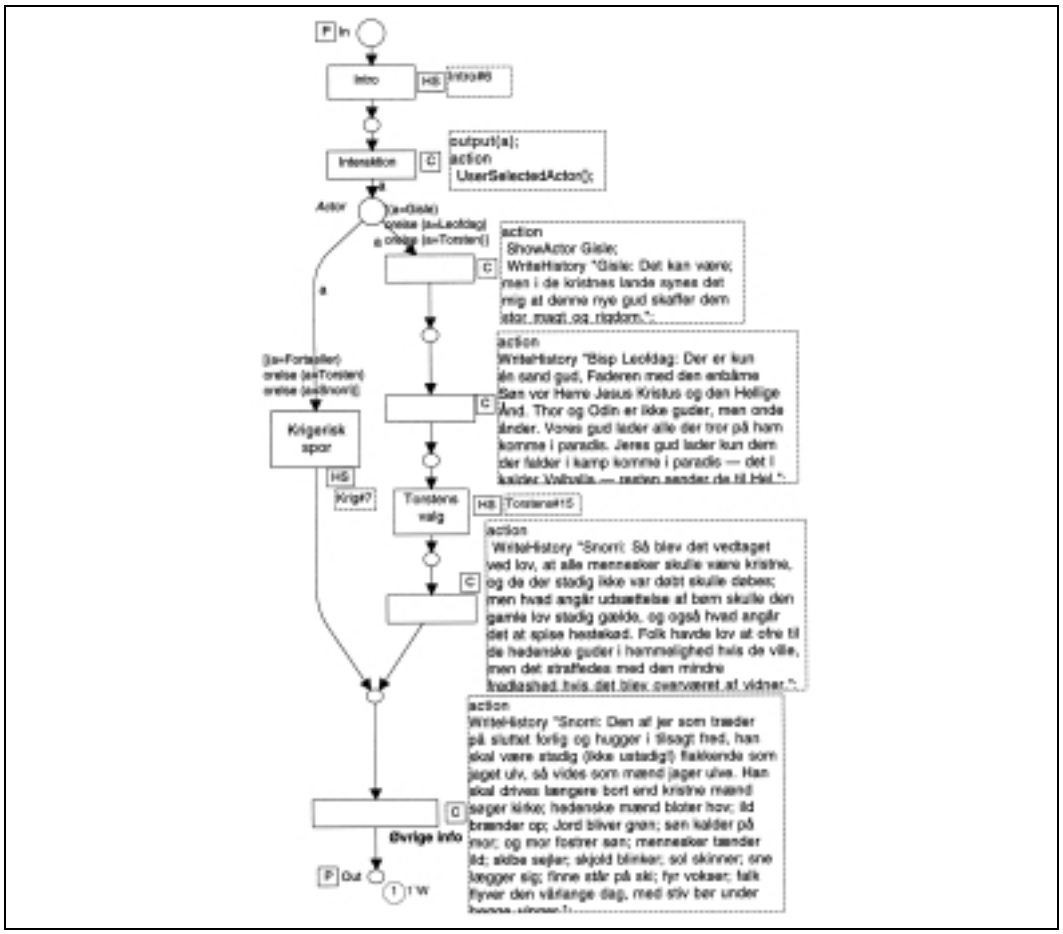


Figure A.5: Page 'Kristen#5'. Overview of the story, with the choice between reject (left) and accept (right) of Christianity. Probably a better modularization would have been obtained if the choice to the right had had its own subpage, as the left one has.

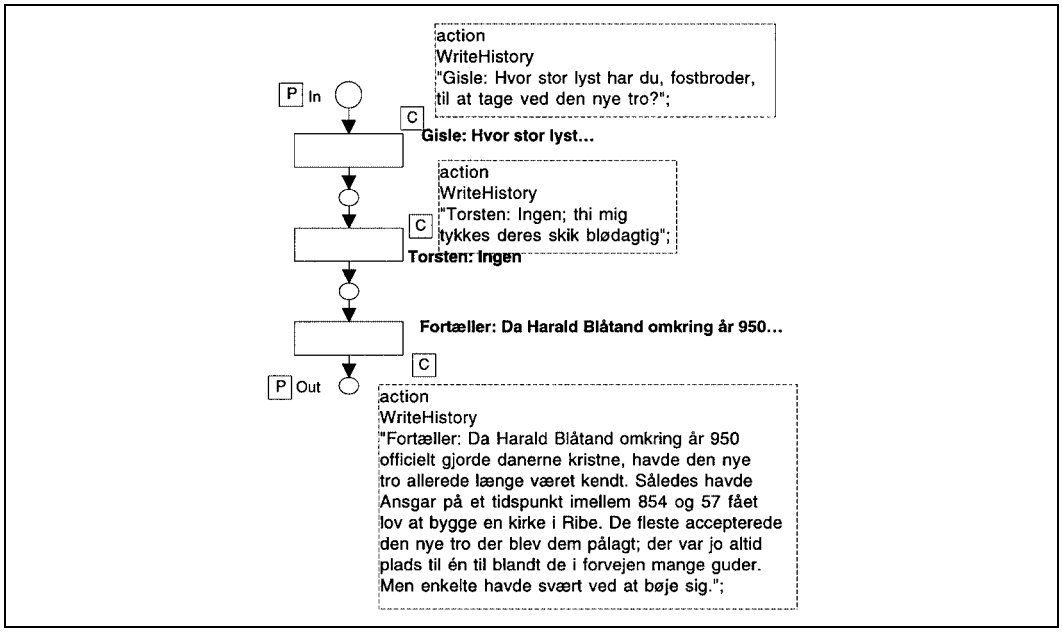


Figure A.6: Page 'Intro#6'. Introduction to the story.

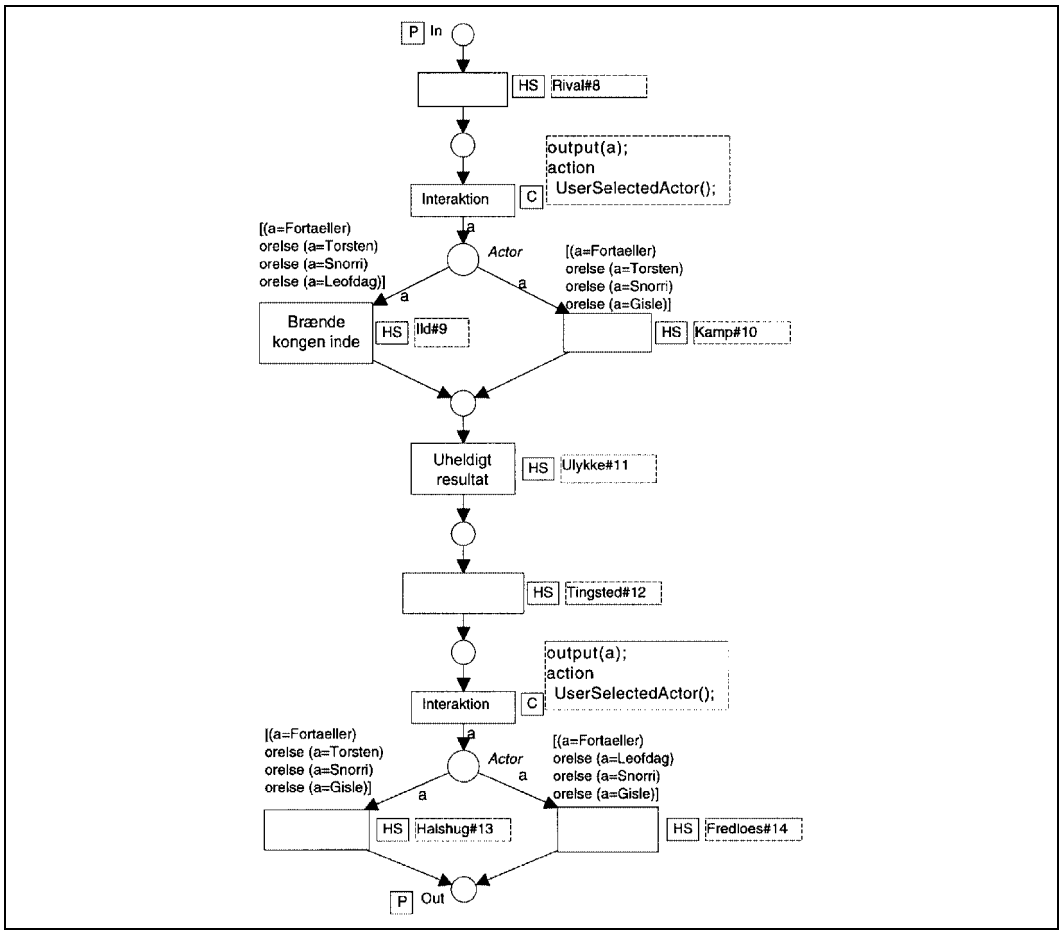


Figure A.7: Page 'Krig#7'. Torsten rejects Christianity and pays the price. The choice at the top is between trying to kill the king in a fire (left) and meeting him in an open fight (right). The choice at the bottom is between execution and outlawry

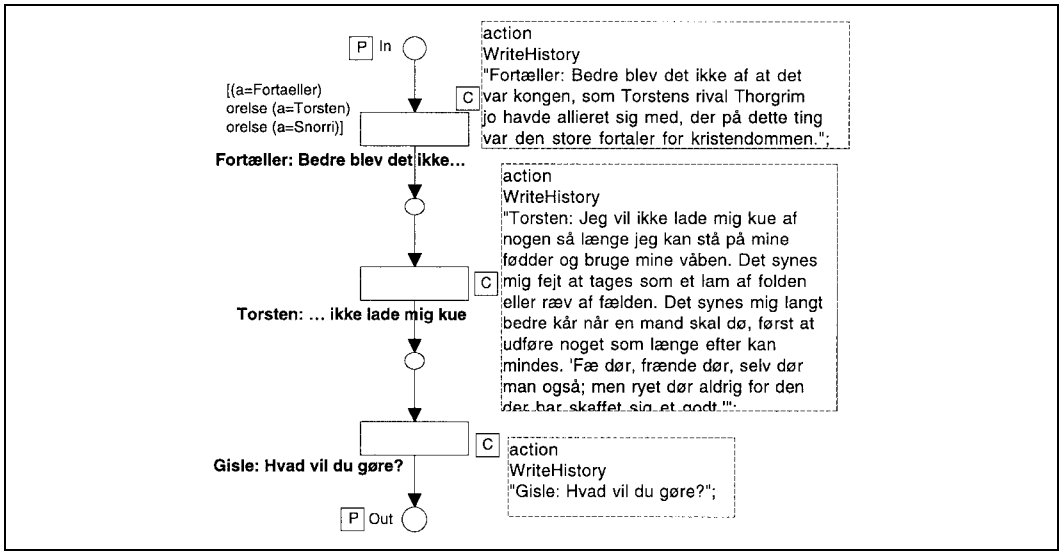


Figure A.8: Page 'Rival#8'.

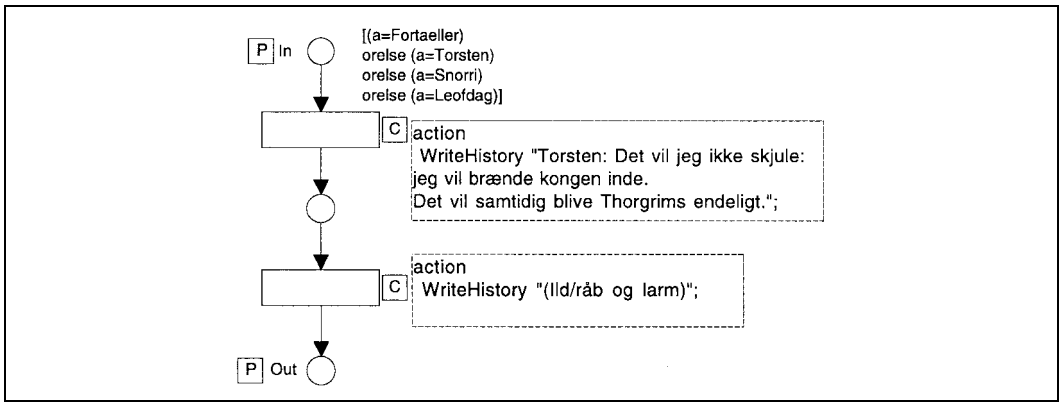


Figure A.9: Page 'Ild#9'. (Ild means fire).

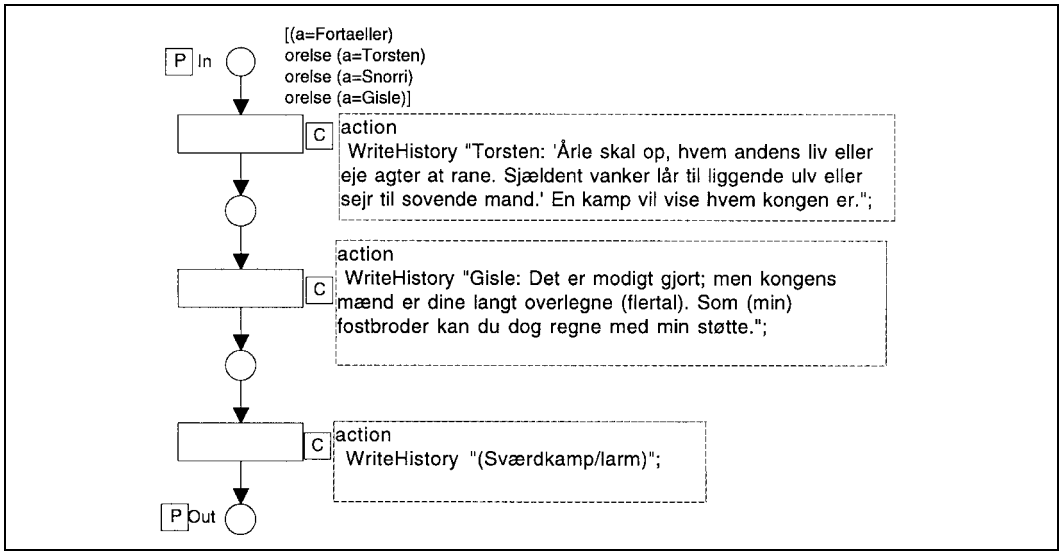


Figure A.10: Page 'Kamp#10' (fight).

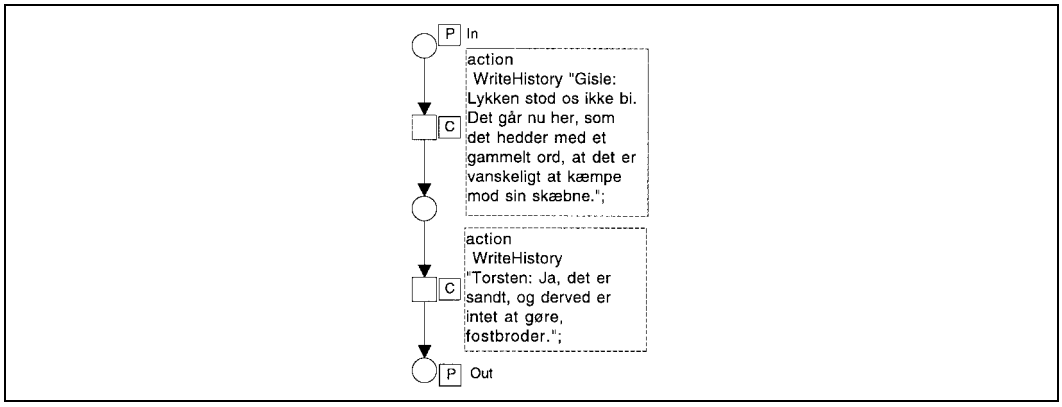


Figure A.11: Page 'Ulykke#11'.

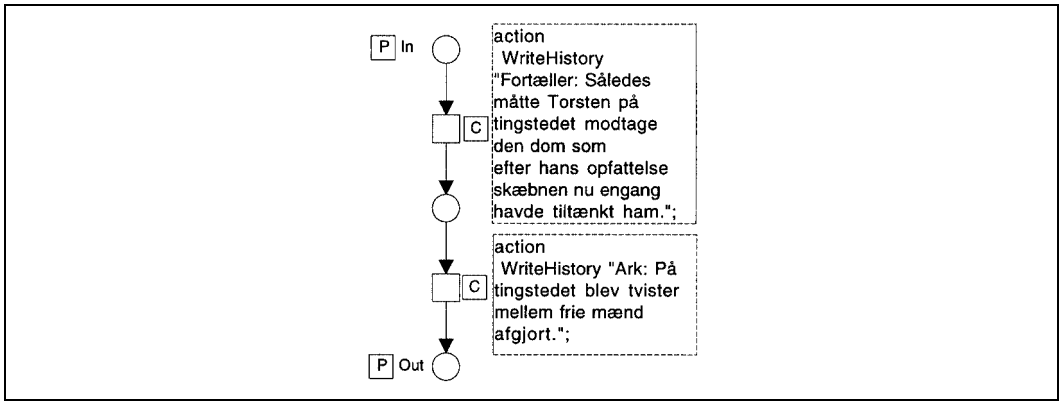


Figure A.12: Page 'Tingsted#12'.

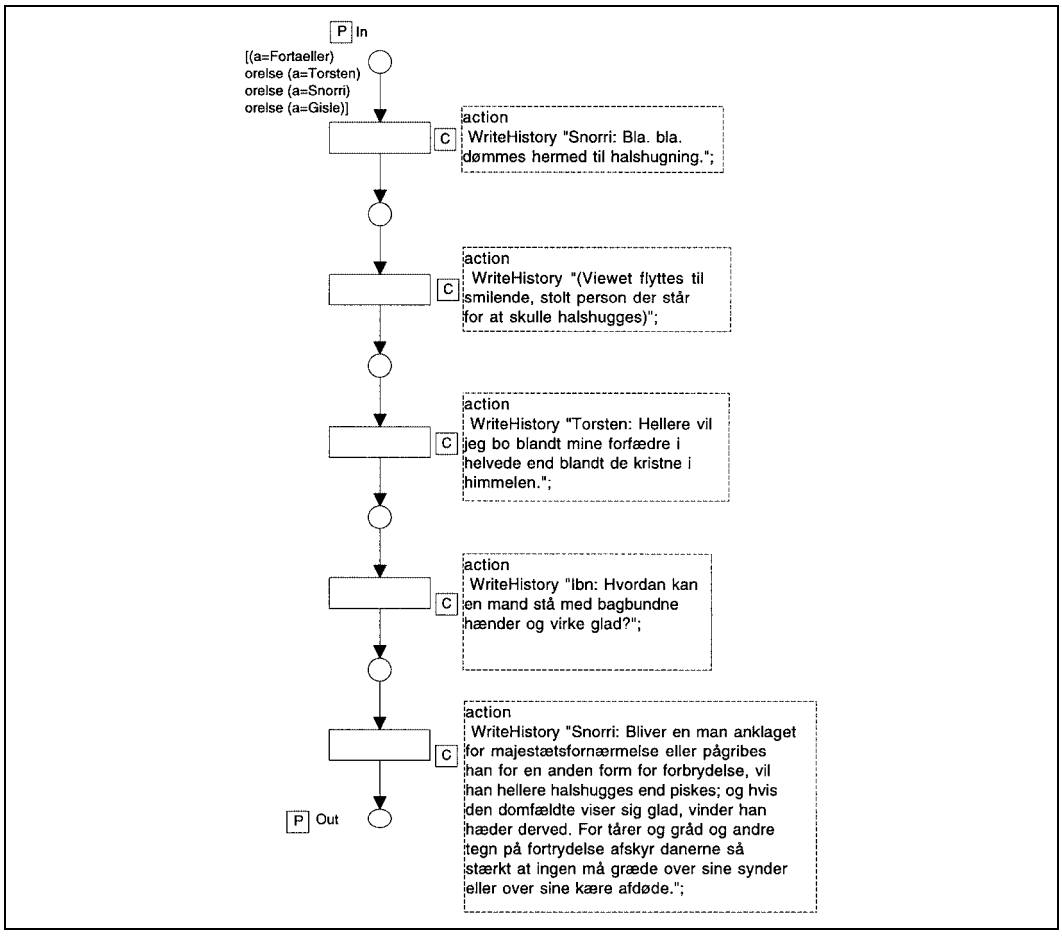


Figure A.13: Page 'Halshug#13' (execution).

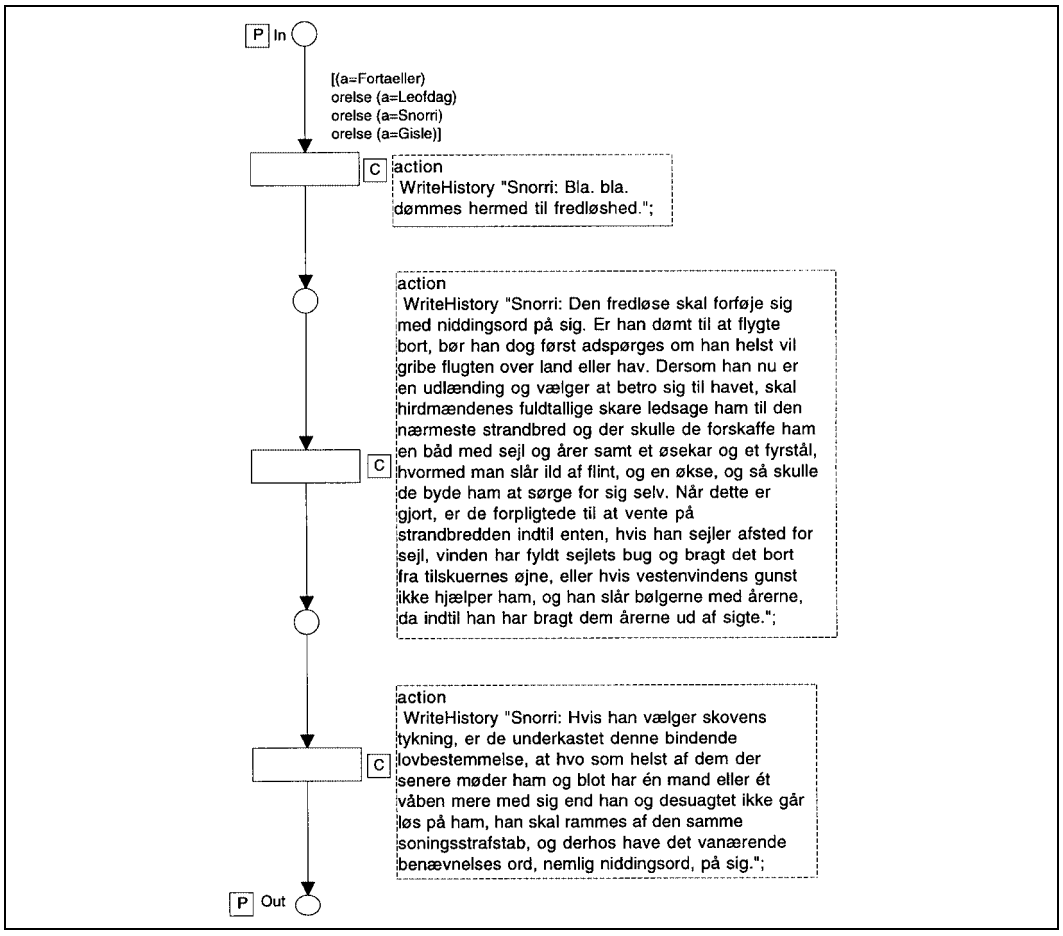


Figure A.14: Page 'Fredloes#14' (outlaw).

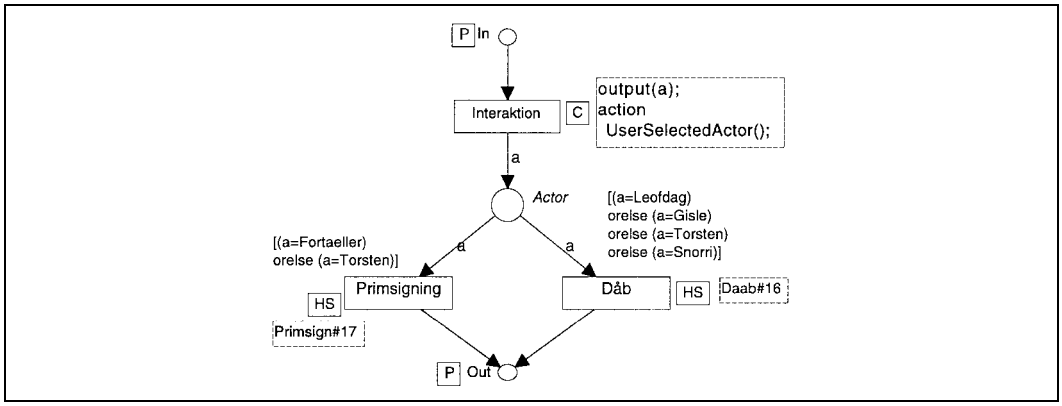


Figure A.15: Page 'Torstens#15'. Torsten accepts Christianity, eighter by being marked by the sign of the cross (left), or by baptism (right).

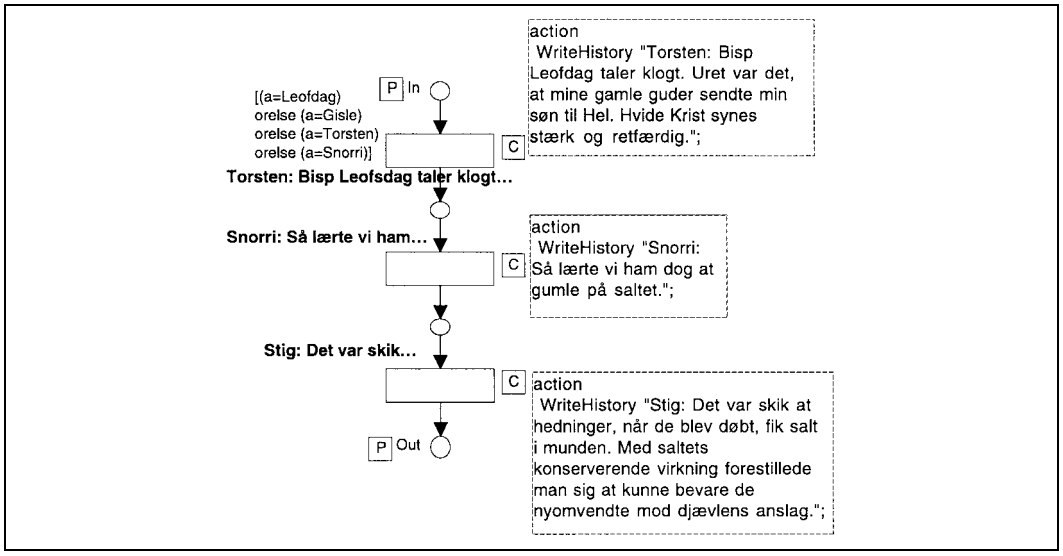


Figure A.16: Page 'Daab#16'. (Dåb means baptism.)

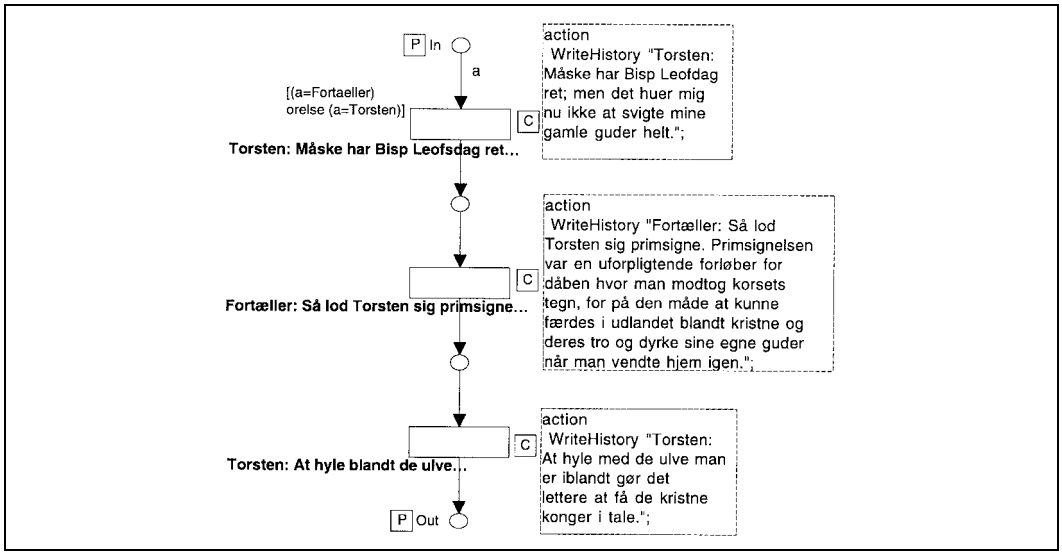


Figure A.17: Page 'Primsign#17'. (The 'primsignelse' was a precursor of baptism in which one was marked by the sign of the cross.

Sample output from the net

The text in this subsection is copied unchanged from the report page generated by the above Petri net. Only formatting has been added.

Gisle: Hvor stor lyst har du, fostbroder, til at tage ved den nye tro?

Torsten: Ingen; thi mig tykkes deres skik bldagtig.

Fortller: Da Harald Bltand omkring r 950 officielt gjorde danerne kristne, havde den nye tro allerede lnge vret kendt. Sledes havde Ansgar p et tidspunkt imellem 854 og 57 fet lov at bygge en kirke i Abe. De fleste accepterede den nye tro der blev dem plagt; der var jo altid plads til n til blandt de i forvejen mange guder.

Men enkelte havde svrt ved at bje sig.

Fortller: Bedre blev det ikke af at det var kongen, som Torstens rival

Thorgrim jo havde allieret sig med, der p dette ting var den store fortaler for kristendommen.

Torsten: Jeg vi1 ikke lade mig kue af nogen s lnge jeg kan st p mine fdder og bruge mine vben. Det synes mig fejt at tages som et lam af folden eller rv af flden. Det synes mig langt bedre kr nr en mand skal d, frst at udfre noget som lnge efter kan mindes. 'F dr, frnde dr, selv dr man ogs; men ryet dr aldrig for den der har skaffet sig et godt.'

Gisle: Hvad vi1 du gre?

Torsten: 'rle skal op, hvem andens liv eller eje agter at rane. Sjldent vanker lr til liggende ulv eller sejr til sovende mand.' En kamp vil vise hvem kongen er.

Gisle: Det er modigt gjort; men kongens mnd er dine langt overlegne. Som min fostbroder kan du dog regne med min sttte.

(Svrdkamp/larm)
Gisle: Lykken stod os ikke bi. Det gr nu her, som det hedder med et gammelt ord, at det er vanskeligt at kmpe mod sin skbne.

Torsten: Ja, det er sandt, og derved er intet at gre, fostbroder.

Fortller: Sledes mtte Torsten p tingstedet modtage den dom som efter hans opfattelse skbnen nu engang havde tiltnkt ham.

Ark: P tingstedet blev tvister mellem frie mnd afgjort.

Snorri: Bla. bla. dmmes hermed til fredlshed.

Snorri: Den fredlse skal forfje sig med niddingsord p sig. Er han dmt til at flygte bort, br han dog frst adsprges om han helst vi1 gribe flugten over land eller hav. Dersom han nu er en udlnding og vlger at betro sig til havet, skal hirdmndenes fuldtallige skare ledsage ham til den nrmeste strandbred og der skulle de forskaffe ham en bd med sejl og rer samt et sekar og et fyrstl, hvormed man slr ild af flint, og en kse, og s skulle de byde ham at srge for sig selv. Nr dette er gjort, er de

forpligtede til at vente p strandbredden indtil enten, hvis han sejler afsted for sejl, vinden har fyldt sejlets bug og bragt det bort fra tilskuernes jne, eller hvis vestenvindens gunst ikke hjlper ham, og han slr blgerne med rerne, da indtil han har bragt dem rerne ud af sigte.

Snorri: Hvis han vlger skovens tykning, er de underkastet denne bindende lovbestemmelse, at hvo som helst af dem der senere mder ham og blot har n mand eller t vben mere med sig end han og desuagtet ikke gr ls p ham, han skal rammes af den samme soningsstrafstab, og derhos have det vanrende benvnelses ord, nemlig niddingsord, p sig.

Snorri: Den af jer som trder p sluttet forlig og hugger i tilsagt fred, han skal vre stadig flakkende som jaget ulv, s vides som mnd jager ulve. Han skal drives lngere bort end kristne mnd sger kirke; hedenske mnd bloter hov; ild brnder op; Jord bliver grn; sn kalder p mor; og mor fostrer sn; mennesker tnder ild; skibe sejler; skjold blinker; sol skinner; sne lgger sig; finne str p ski; fyr vokser; falk flyver den vrlange dag, med stiv br under begge vinger.

English translation of the sample output

Gisle: How much do vou want to follow the new faith, sworn brother[1]?

Torsten: Not at all; I find their customs feeble.

Narrator: By the time that Harold Bluetooth officialy made the Danes Christians in about year 950, Christianity had been known for a long time. Ansgar had been allowed to build a church in Ribe some time between 854 and 857. Most people accepted the new faith — there was always room for another god amongst the many they already had.

But a few people had difficulty submitting.

Narrator: What made it worse was that the king, with whom Torsten's

rival Thorgrim had allied himself, was the greatest advocate of Christianity at this thing[2]

Torsten: No one shall force me to submit as long as I can stand and I have my weapons. I feel it is cowardly to be taken like a lamb from a pen or a fox from a trap. When a man must die, it is better for him to carry out something that can be long remembered. 'Cattle die, kinsmen die, you yourself die; but a good reputation never dies.'

Gisle: What will you do?

Torsten: 'You must rise early to steal someone's life or property. Seldom is there food for a resting wolf or victory for a sleeping man.' A fight will show what kind of man the king is.

Gisle: That is bravely done; but the king's men are far superior to yours. As my sworn brother, though, you can depend on my support.

(Sword fight/noise.)
Gisle: Fortune did not aid us. It now happens as the old saying goes: it's difficult to fight against your fate.

Torsten: Yes, that's true, and there's nothing to be done about that, brother.

Narrator: So Torsten was to receive the sentence which he believed fate had decreed for him, at the thingstead[3].

Ark: At the thingstead, disputes between free men were settled.

Snorri: (Blah-blah) is hereby sentenced to be outlawed.

Snorri: The outlaw shall leave with words of condemnation on him. If he is sentenced to flee, he must choose flight over land or sea. If he is a foreigner and chooses the sea, the full troop of housecarls[4] shall accompany him to the nearest seashore and procure for him a boat with sail and oars, a bailer, a steel to strike a light from flint and an axe. Then they should bid him to take care of himself. When this is done, they should stay on the shore until, if he is using the sails, the

wind has filled the sails and taken them from sight, or if there is no west wind and he rows, until the oars are out of sight.

Snorri: If he chooses the depths of the forest, they are subject to this binding provision, that if anyone should later meet him, even if he has only one man or weapon more than he has and still doesn't attack him, then he shall be given the same punishment and also have the words of condemnation on him.

Snorri: He who tramples on concluded reconciliation and cuts up the promised peace shall roam as unceasingly as the hunted wolf, as widely as men hunt wolves. He shall be driven farther away than Christian men seek church; heathen men offer sacrifice in the high place; fire burns; the Earth becomes green; a son calls for his mother; and a mother breeds a son; a human lights a fire; ships sail; a shield twinkles; the sun shines; snow falls; a Finn skis; a pine tree grows; the falcon flies the long spring day with a stiff, fair wind under its wings.

A.2 Swords, iron and millstones, first version

The following 21 figures show the Coloured Petri Net of the story 'Swords, Iron and Millstones'.

The loop in figure A.2 repeatedly prompts the user for a selection of a visible item—or 'None' if no item is visible. The loop runs from when the fork transition puts a token on the place 'LastSel' in figure A.2 until the join transition in figure A.2 removes it again.

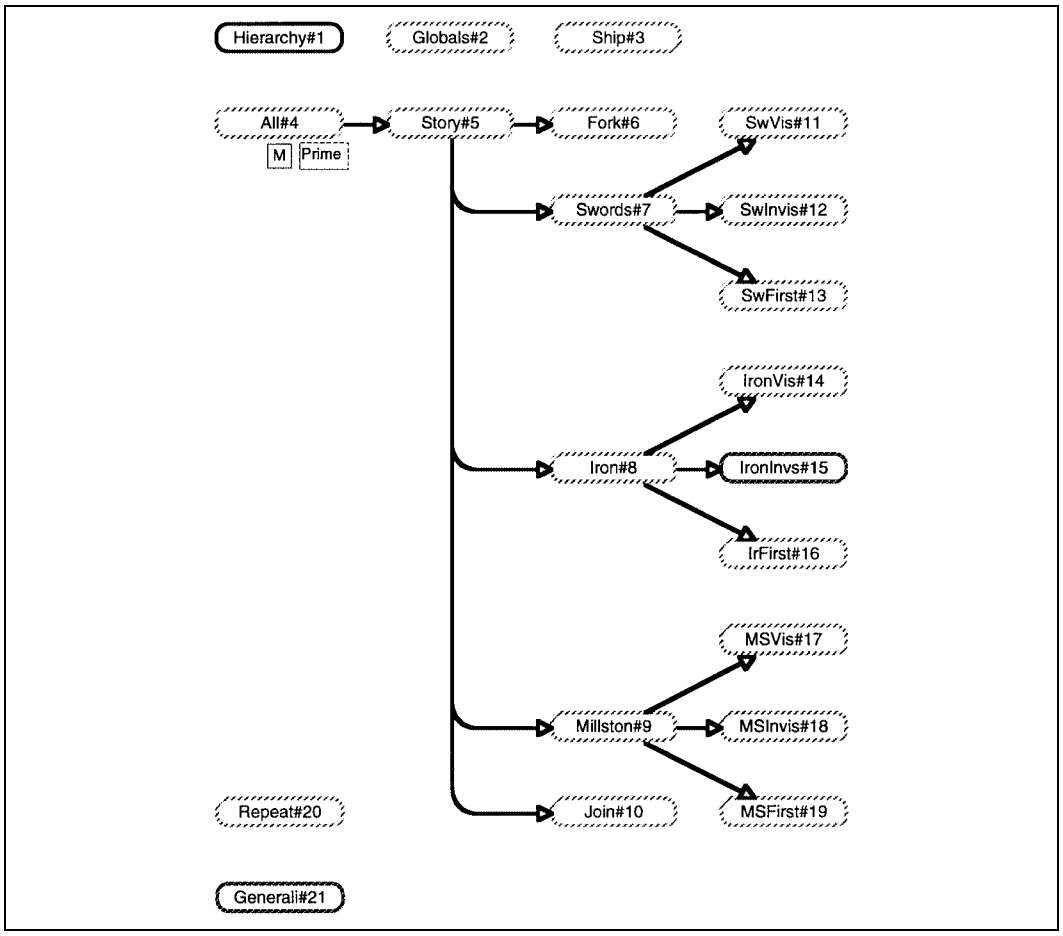
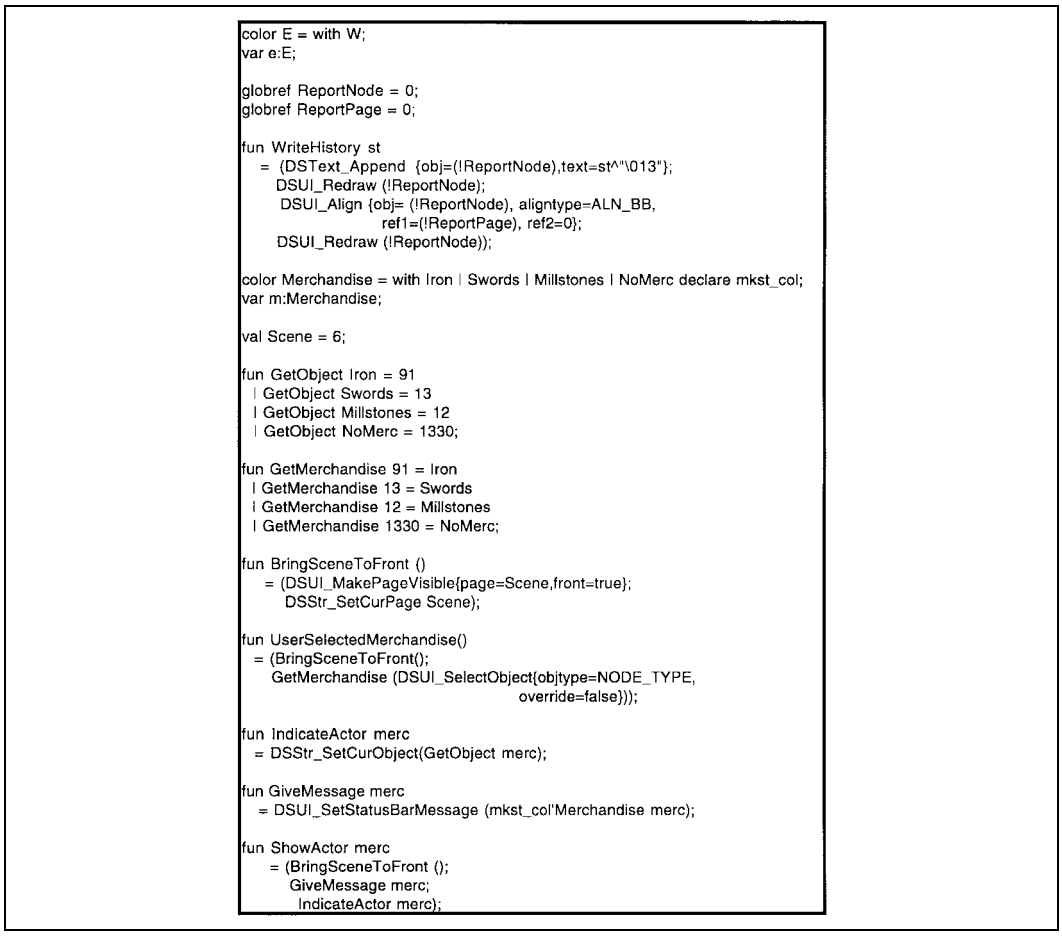


Figure A.18: Hierarchy#1.



```
color E = with W;
var e:E;

globref ReportNode = 0;
globref ReportPage = 0;

fun WriteHistory st
   = (DSText_Append {obj=(!ReportNode),text=st^"\013"};
      DSUI_Redraw (!ReportNode);
       DSUI_Align {obj= (!ReportNode), aligntype=ALN_BB,
                   ref1=(!ReportPage), ref2=0};
      DSUI_Redraw (!ReportNode));

color Merchandise = with Iron | Swords | Millstones | NoMerc declare mkst_col;
var m:Merchandise;

val Scene = 6;

fun GetObject Iron = 91
  | GetObject Swords = 13
  | GetObject Millstones = 12
  | GetObject NoMerc = 1330;

fun GetMerchandise 91 = Iron
  | GetMerchandise 13 = Swords
  | GetMerchandise 12 = Millstones
  | GetMerchandise 1330 = NoMerc;

fun BringSceneToFront ()
    = (DSUI_MakePageVisible{page=Scene,front=true};
       DSStr_SetCurPage Scene);

fun UserSelectedMerchandise()
  = (BringSceneToFront();
     GetMerchandise (DSUI_SelectObject{objtype=NODE_TYPE,
                                       override=false}));

fun IndicateActor merc
  = DSStr_SetCurObject(GetObject merc);

fun GiveMessage merc
   = DSUI_SetStatusBarMessage (mkst_col'Merchandise merc);

fun ShowActor merc
    = (BringSceneToFront ();
       GiveMessage merc;
        IndicateActor merc);
```

Figure A.19: Globals#2.

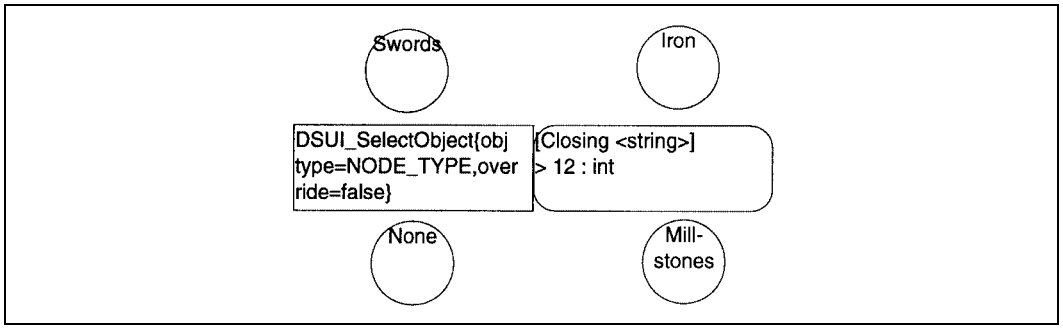


Figure A.20: Ship#3.

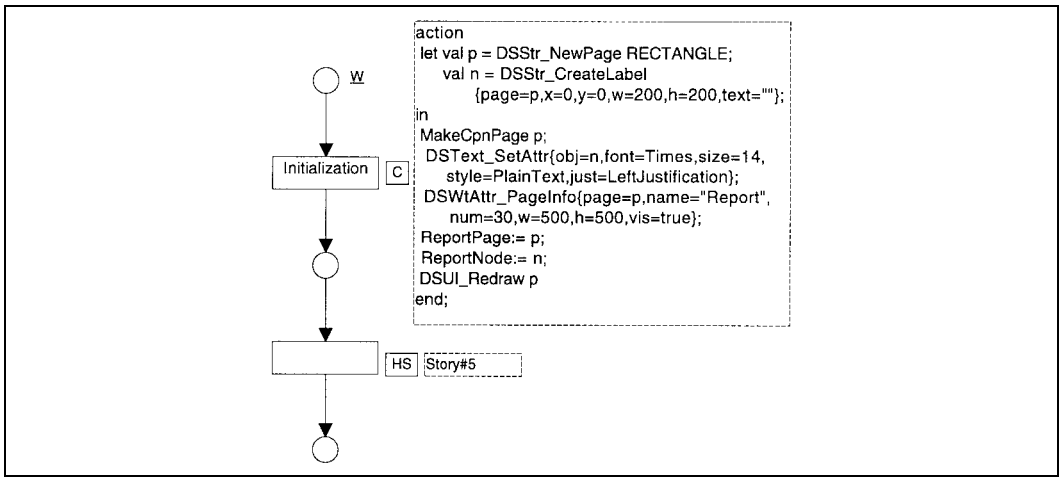


Figure A.21: All#4.

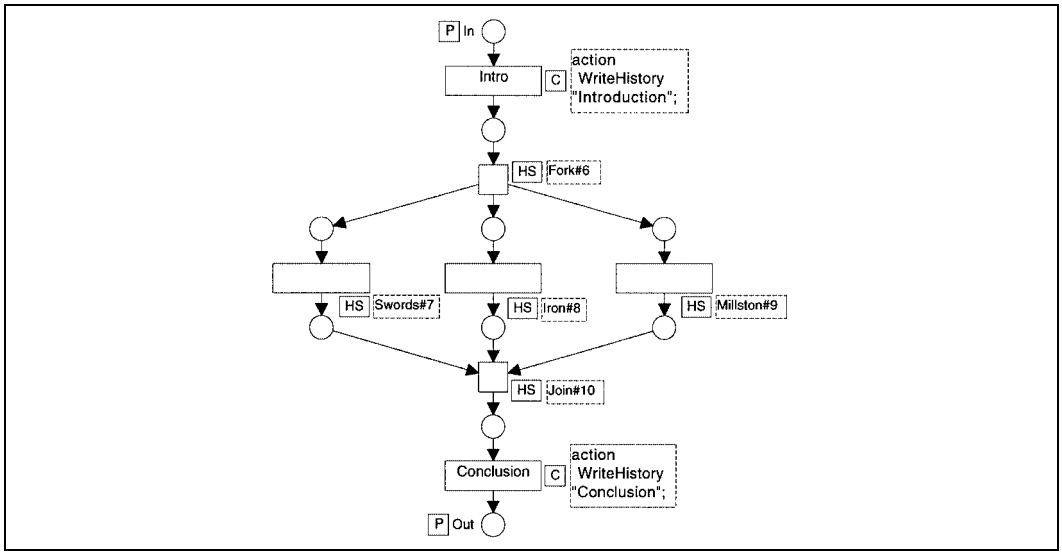


Figure A.22: Story#5.

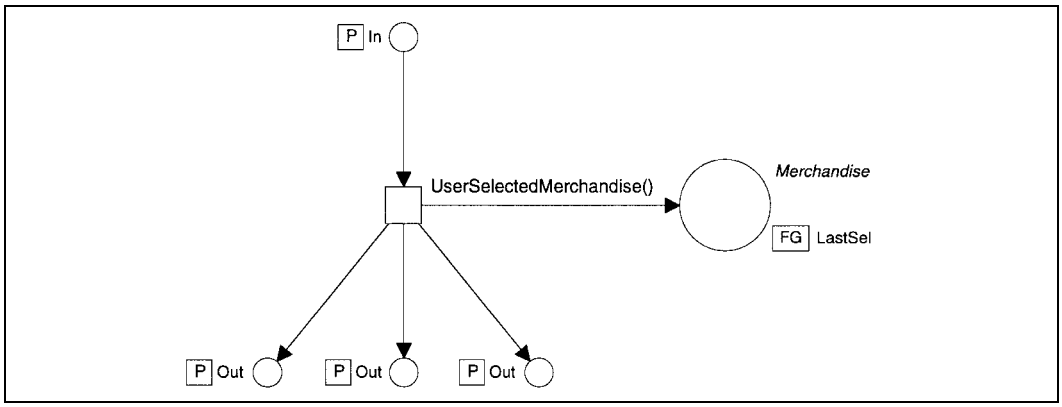


Figure A.23: Fork#6.

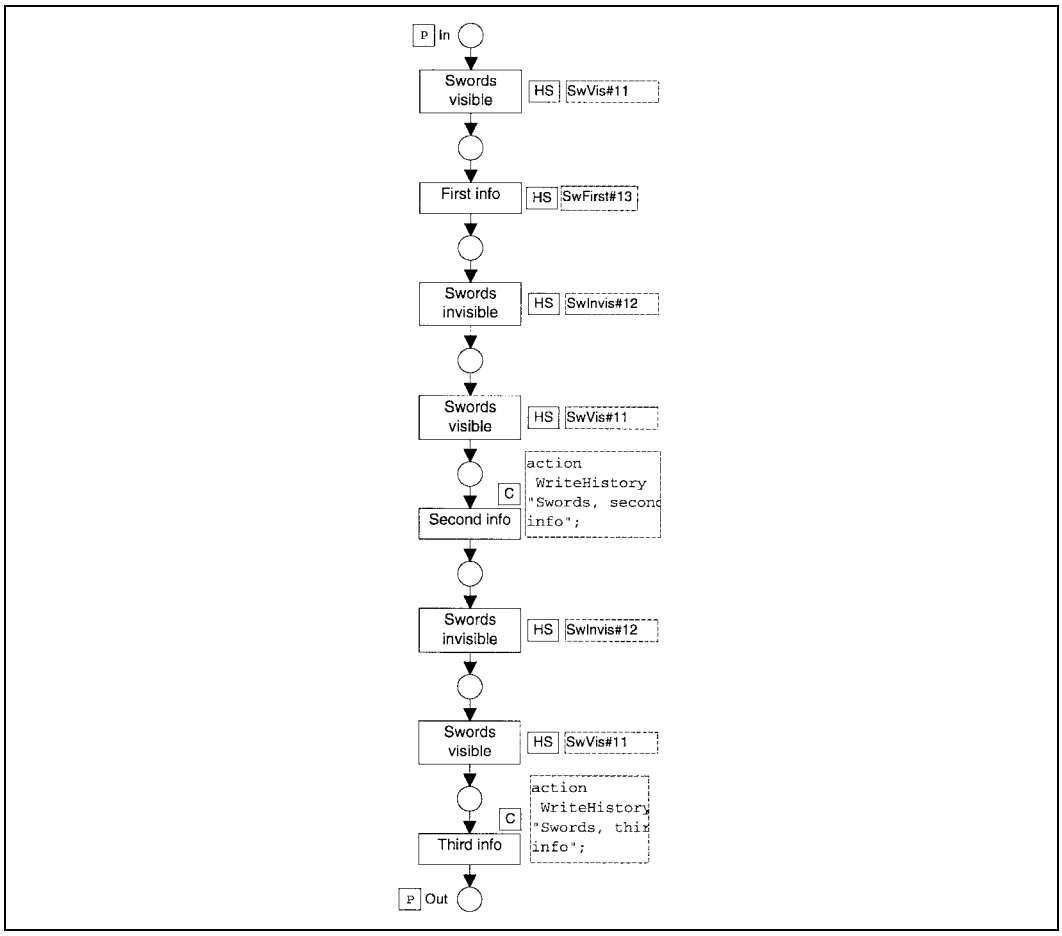


Figure A.24: Swords#7.

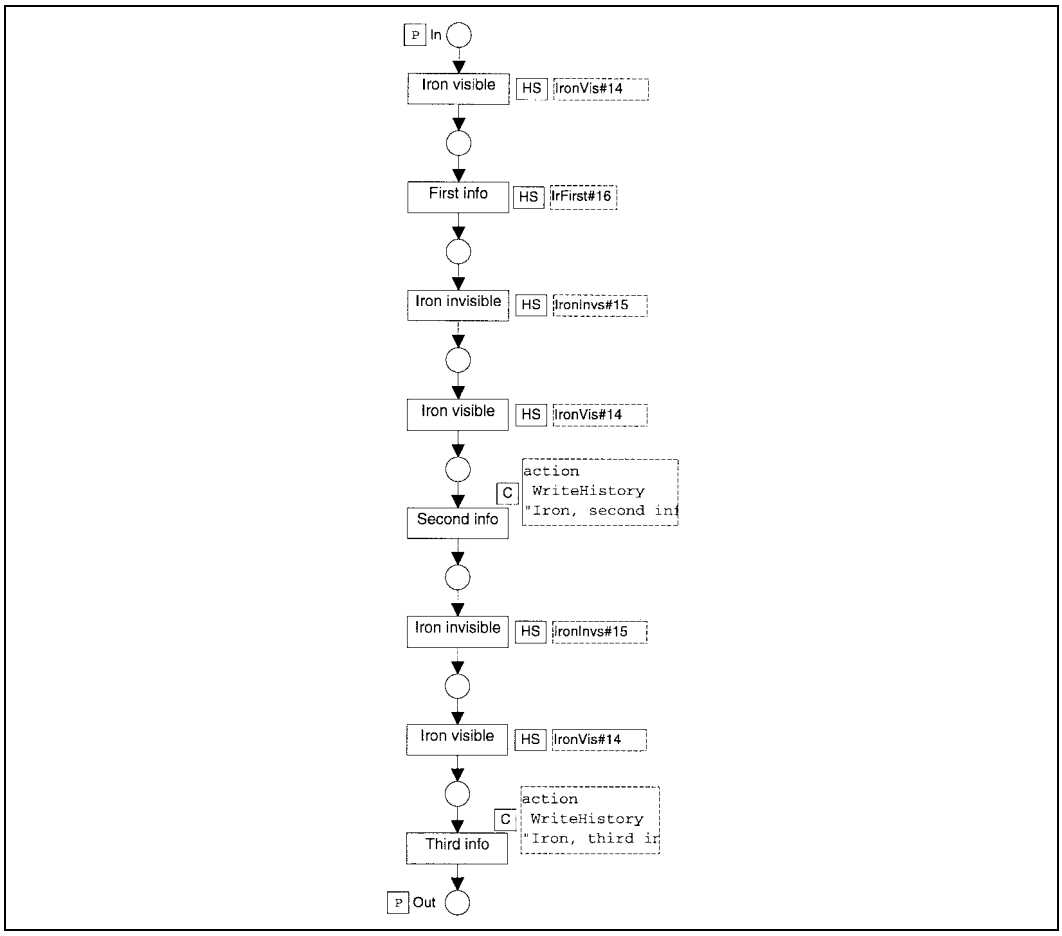


Figure A.25: Iron#8.

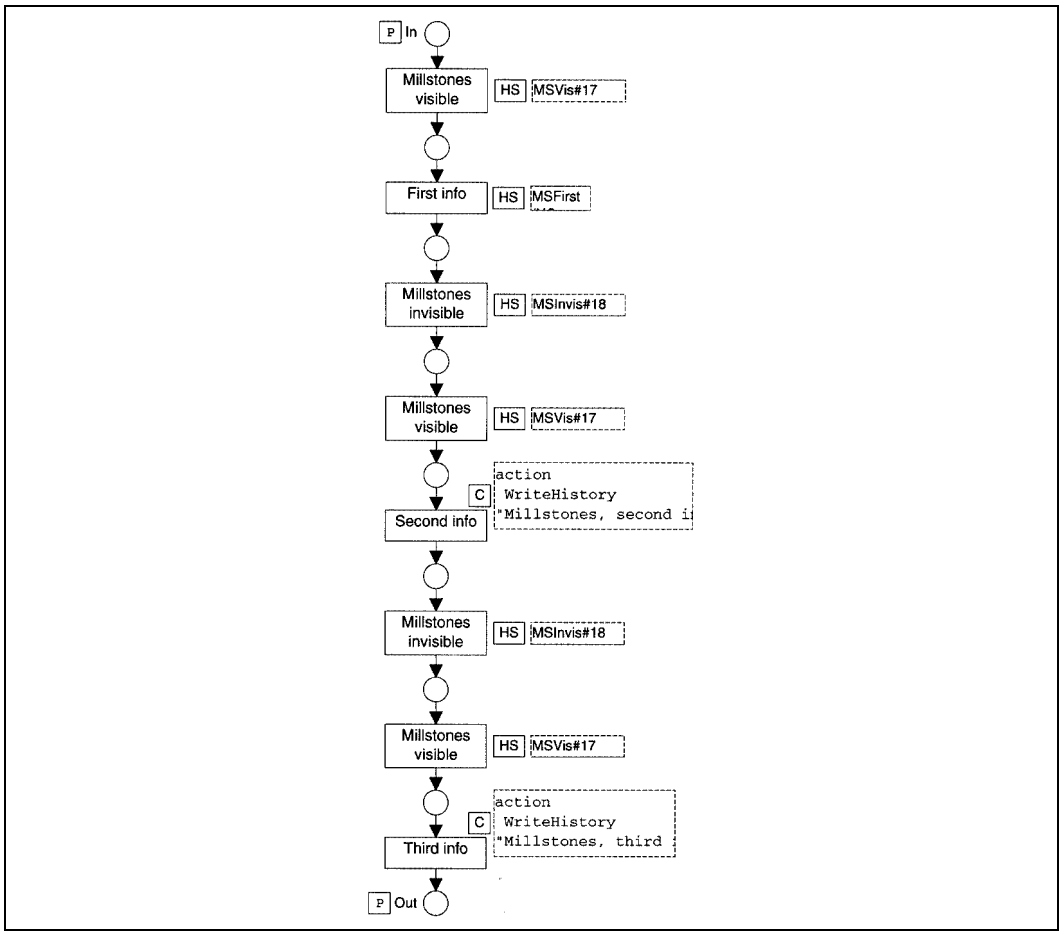


Figure A.26: Millstone#9.

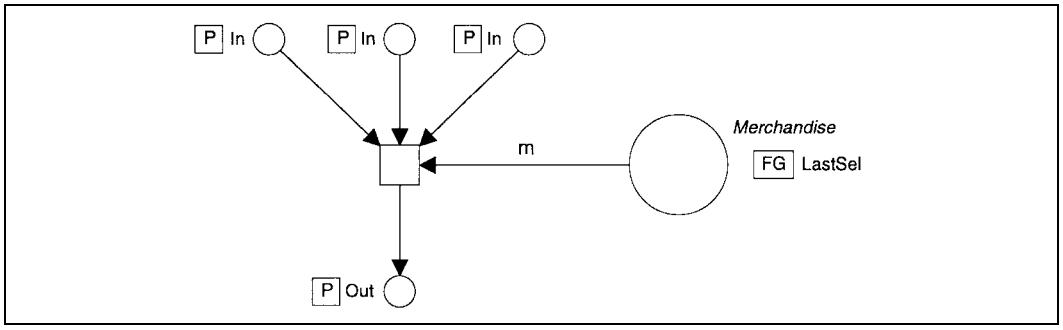


Figure A.27: Join#10.

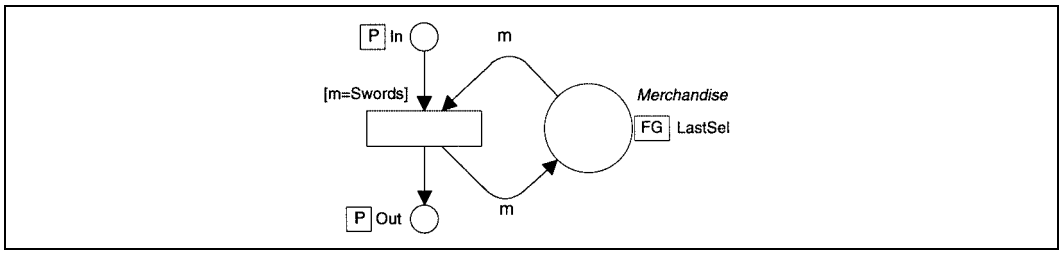


Figure A.28: SwVis#11.

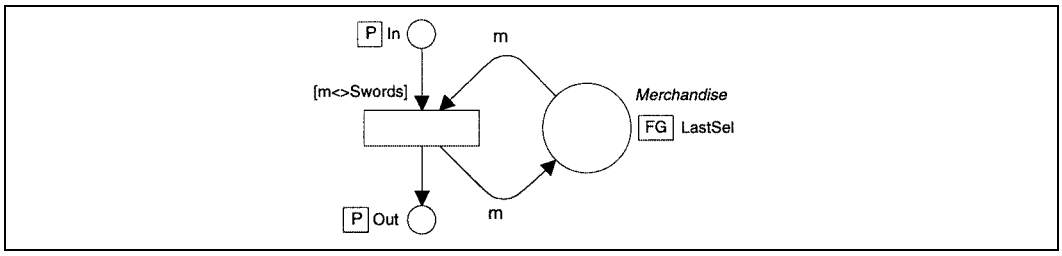


Figure A.29: SwInvis#12.

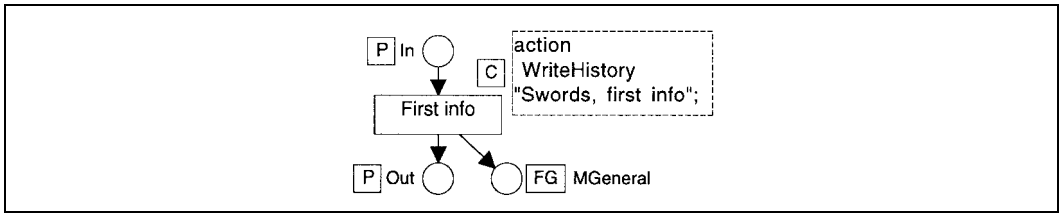


Figure A.30: SwFirst#13.

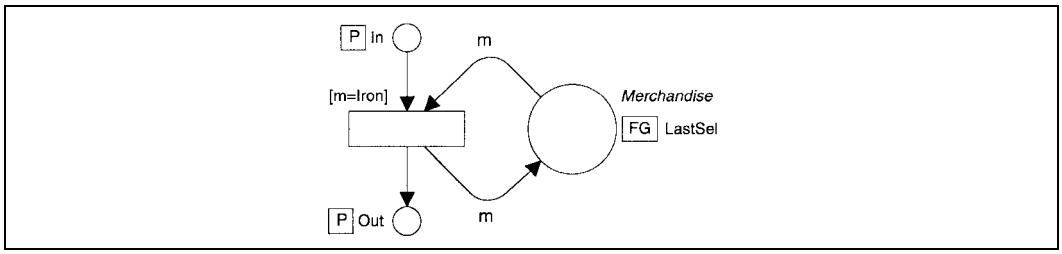


Figure A.31: IronVis#14.

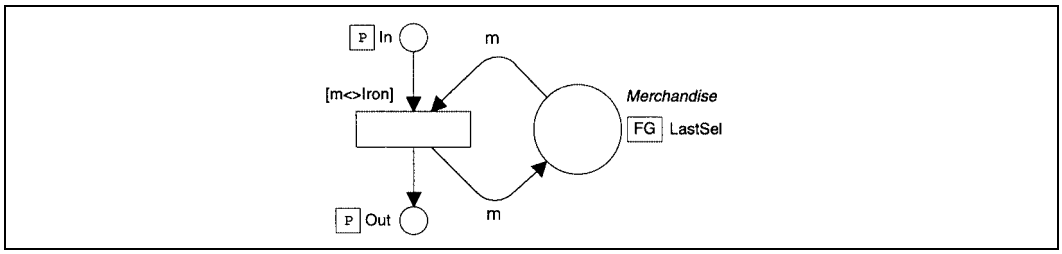


Figure A.32: IronInvs#15.

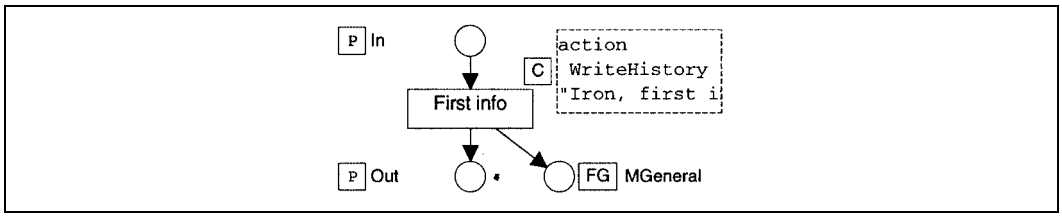


Figure A.33: IrFirst#16.

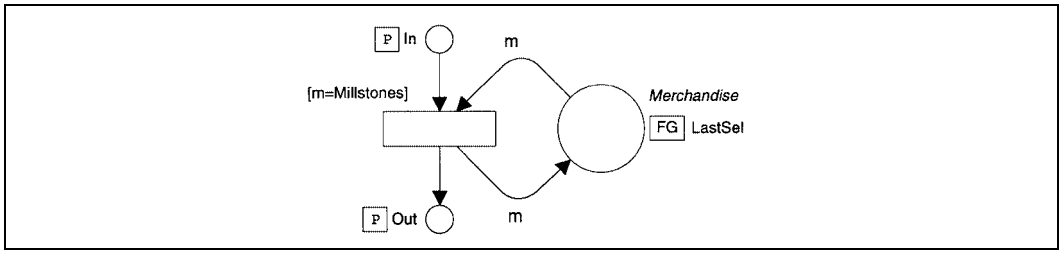


Figure A.34: MSVis#17.

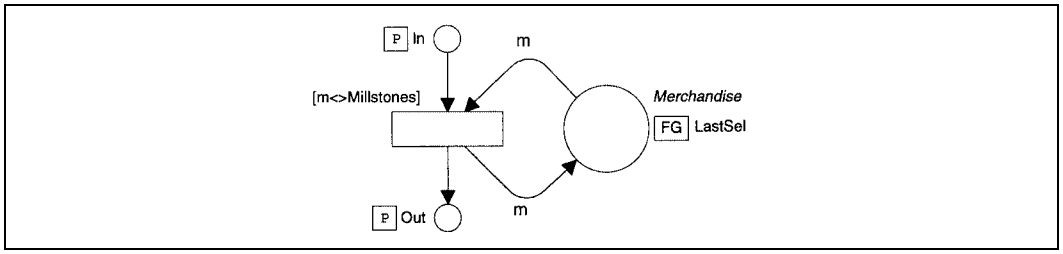


Figure A.35: MSInvis#18.

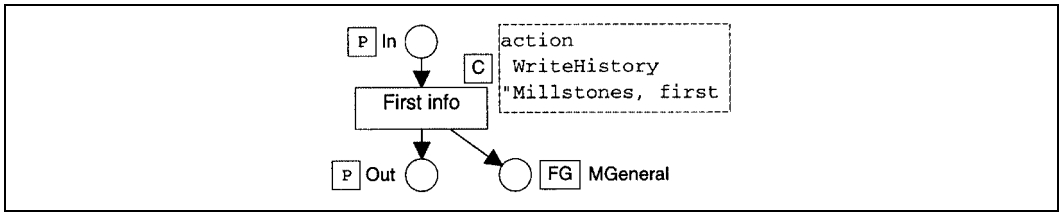


Figure A.36: MSFirst#19.

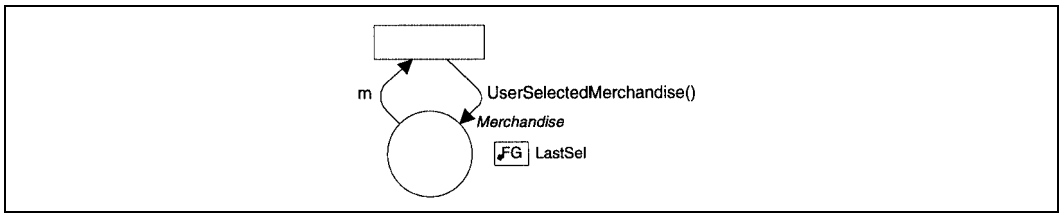


Figure A.37: Repeat#20.

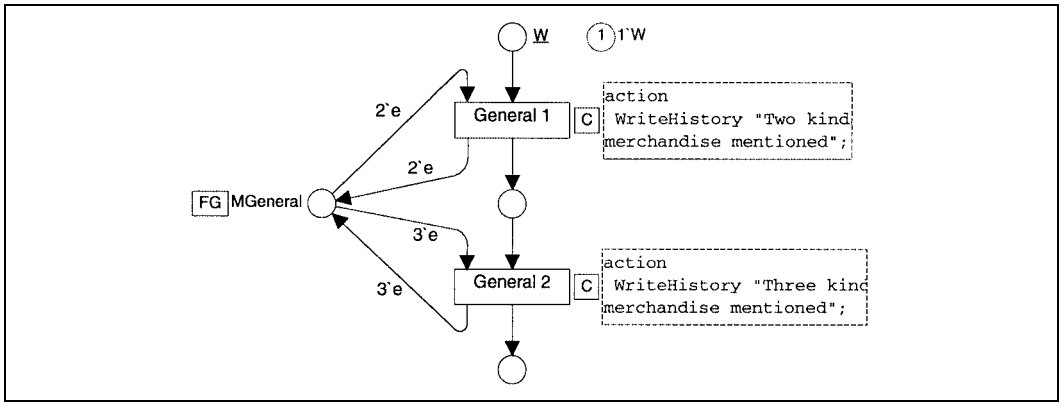


Figure A.38: Generali#21.

A.3 Swords and iron, second version

This section first shows the pages of the Petri net realizing the second version of the swords and iron story. Graphical representations of the output from three of the runs follow.

The global declaration node from the Petri net is too big to fit on a page of this appendix. Instead, the text of the node is given between the figures over this and the next pages.

```
color E = with W;
var e:E;

color realTime = int;
var t: realTime;

globref ReportNode = 0;
globref ReportPage = 0;

fun WriteHistory st
  = (DSText_Append {obj=(!ReportNode),text=stA^"\013"};
       DSUI_Redraw (!ReportNode);
       DSUI_Align {obj= (!ReportNode), aligntype=ALN_BB,
```

```
                        ref1=(!ReportPage), ref2=0};
      DSUI_Redraw (!ReportNode));

color Merchandise = with Iron | Swords | Millstones | NoMerc declare
mkst_col;
var m:Merchandise;

color soundName = string with "  ". . "$\tilde$" and 1-31;
var n: soundName;
```

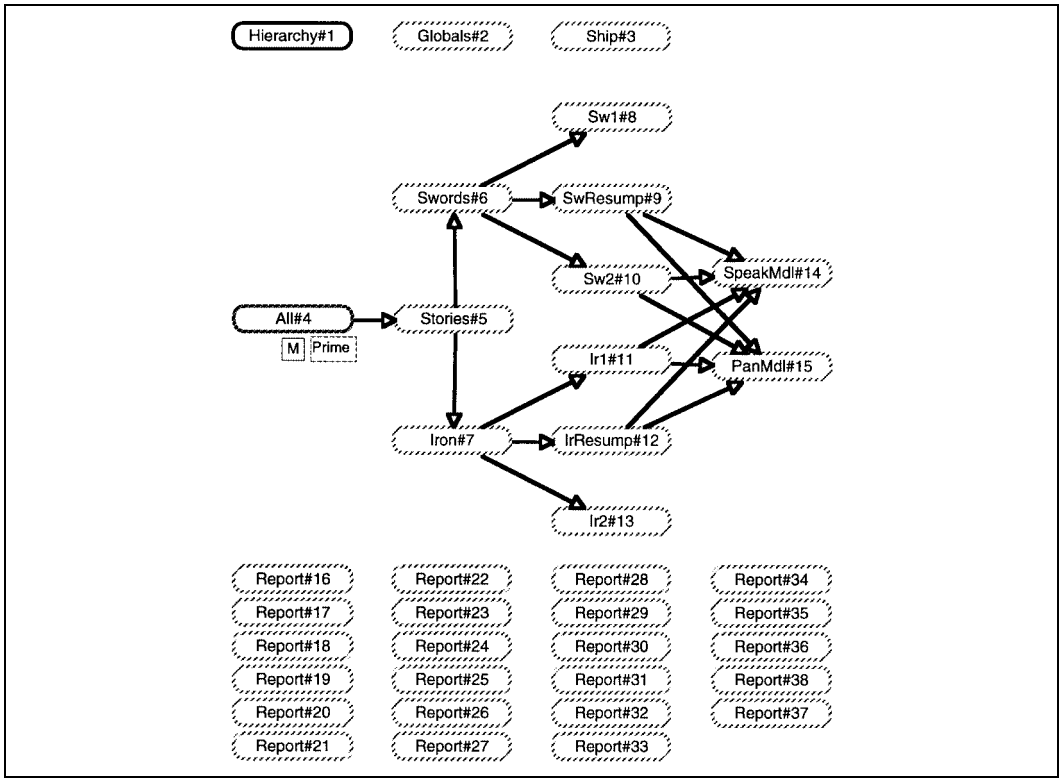


Figure A.39: Hierarchy#1. The pages at the bottom contain text output from 23 of the runs.

```
color duration = int;
color soundspec = record nn:soundName * dd:duration;
var s: soundspee;
```

```
color panSpec = record mm:Merchandise * dd:duration;
var p:panSpec;

val Scene = 6;

fun GetObject Iron = 91
 | GetObject Swords = 13
 | GetObject Millstones = 12
 | GetObject NoMerc = 1330;

fun GetMerchandise 91 = Iron
 | GetMerchandise 13 = Swords
 | GetMerchandise 12 = Millstones
 | GetMerchandise 1330 = NoMerc;

fun BringSceneToFront ()
  = (DSUI_MakePageVisible{page=Scene,front=true};
       DSStr_SetCurPage Scene);

fun UserSelectedMerchandise()
  = (BringSceneToFront();
      GetMerchandise (DSUI_SelectObject{objtype=NODE TYPE,
override=false}));

fun IndicateActor merc
  = DSStr_SetCurObject(GetObject mere);

fun GiveMessage merc
   = DSUI_SetStatusBarMessage (mkst_col'Merchandise mere);

fun ShowActor merc
     = (BringSceneToFront ();
          GiveMessage merc;
          IndicateActor merc);
```

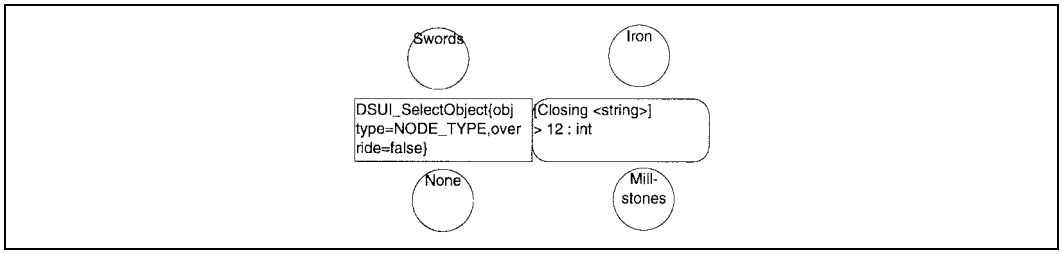


Figure A.40: Ship#3.

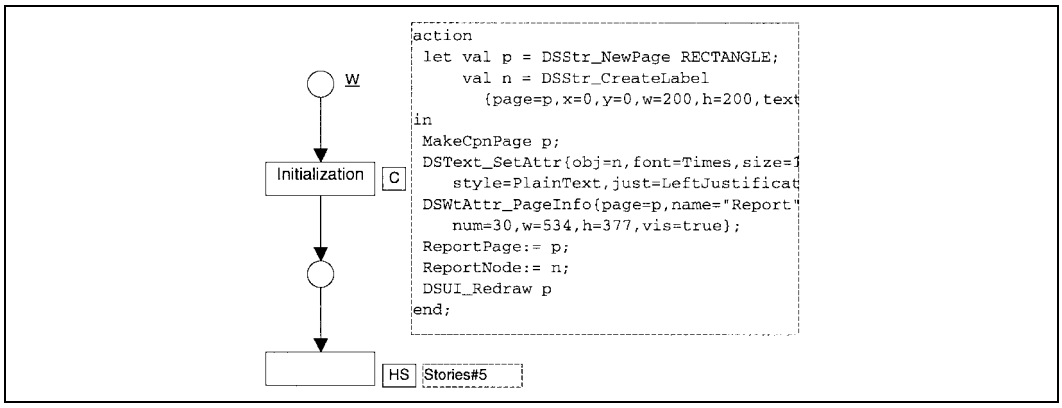


Figure A.41: All#4.

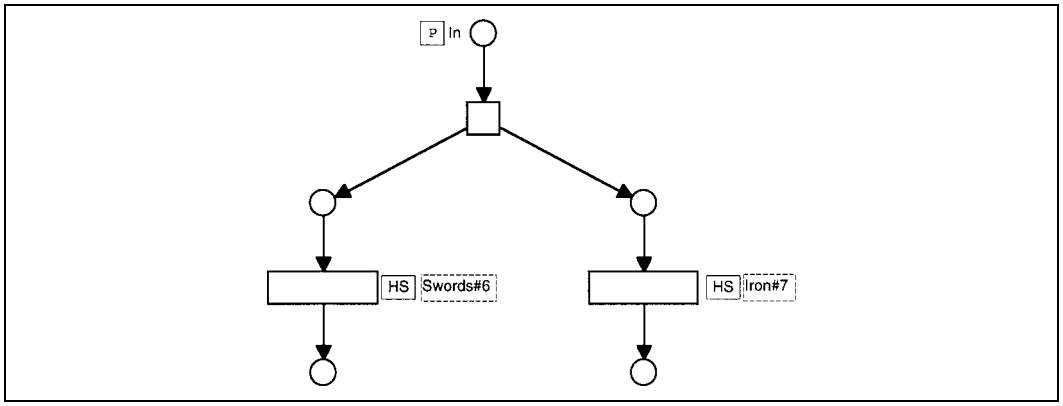


Figure A.42: Stories#5.

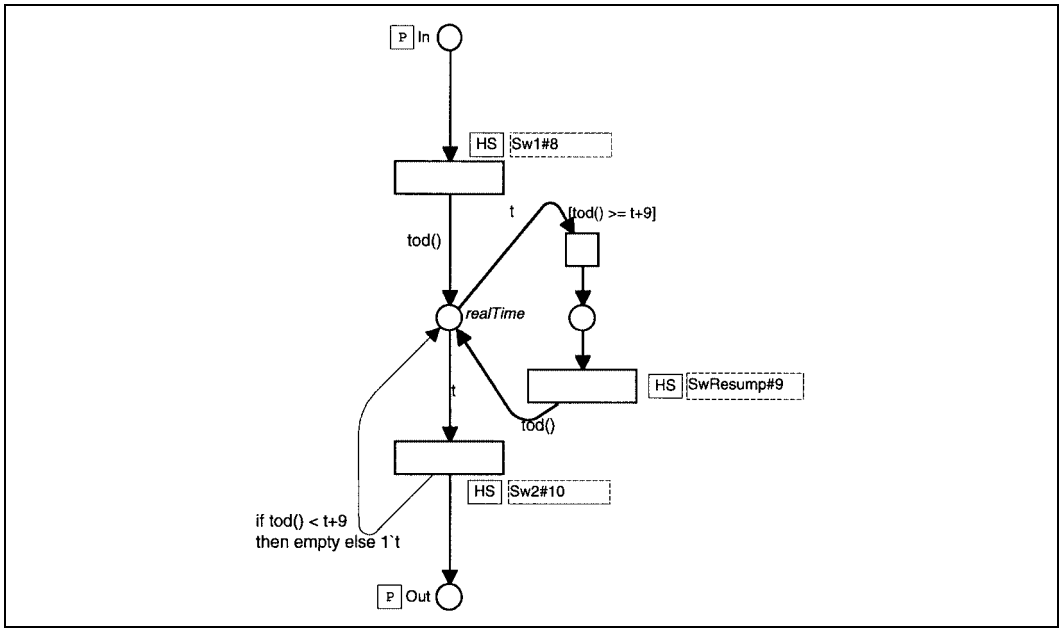


Figure A.43: Swords#6.

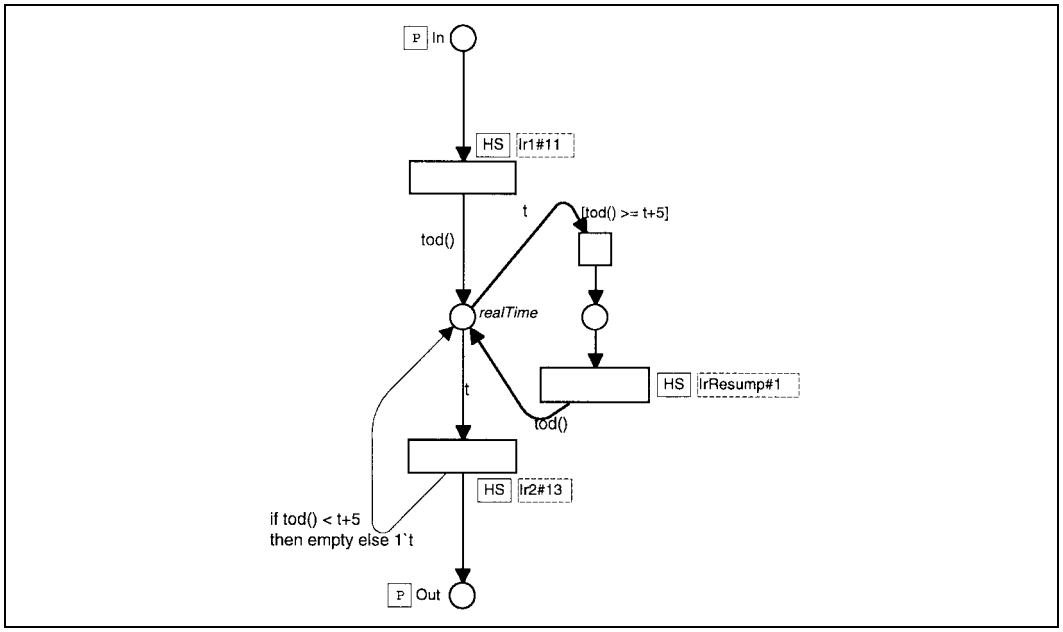


Figure A.44: Iron#7.

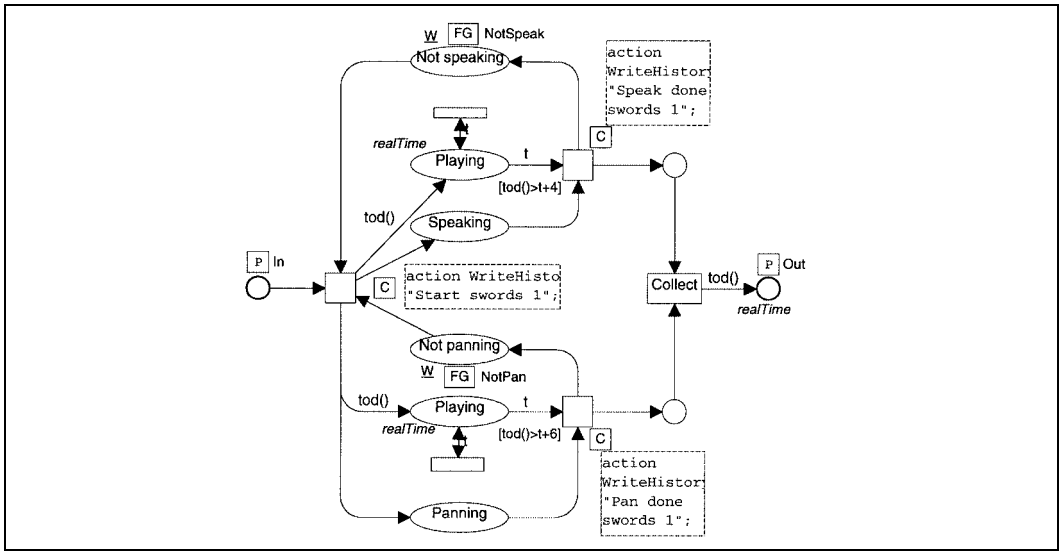


Figure A.45: Sw1#8.

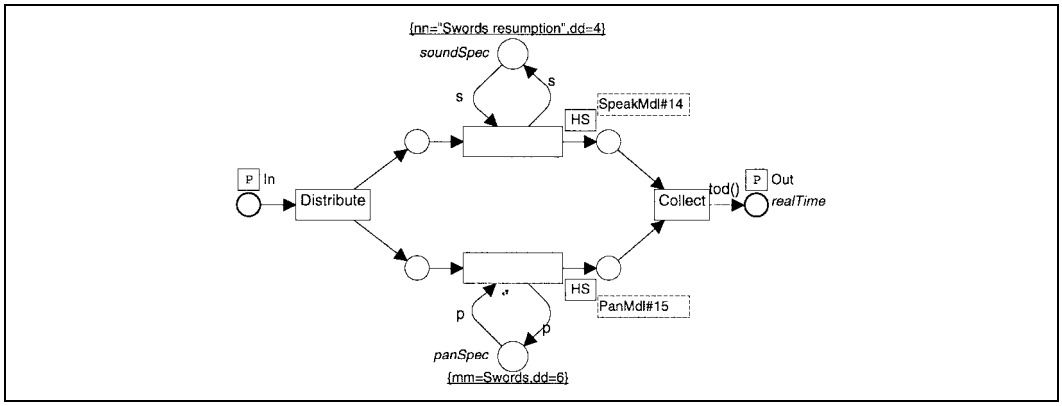


Figure A.46: SwResump#9.

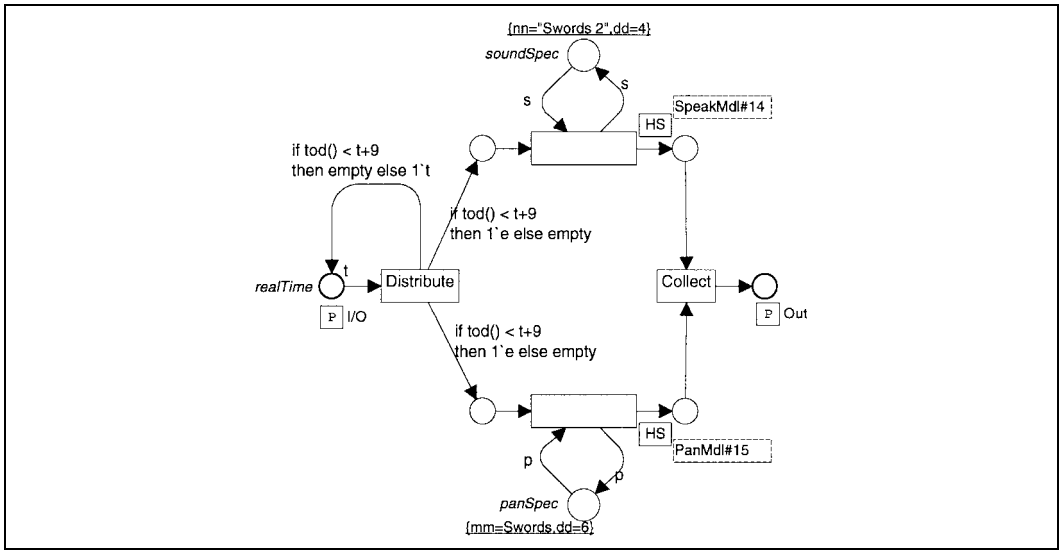


Figure A.47: Sw2#10.

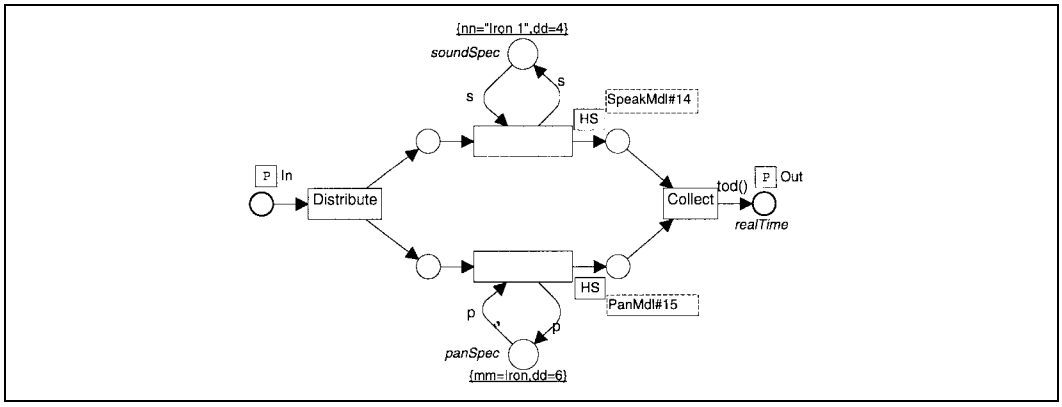


Figure A.48: Ir1#11.

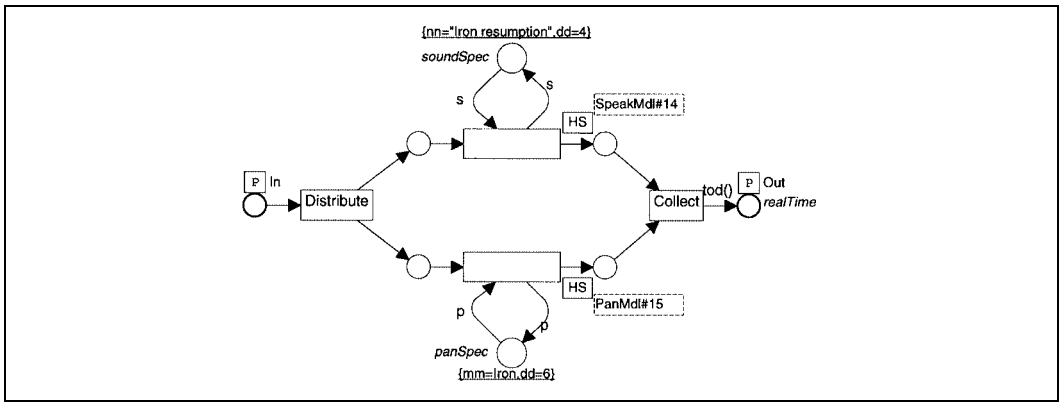


Figure A.49: IrResump#12.

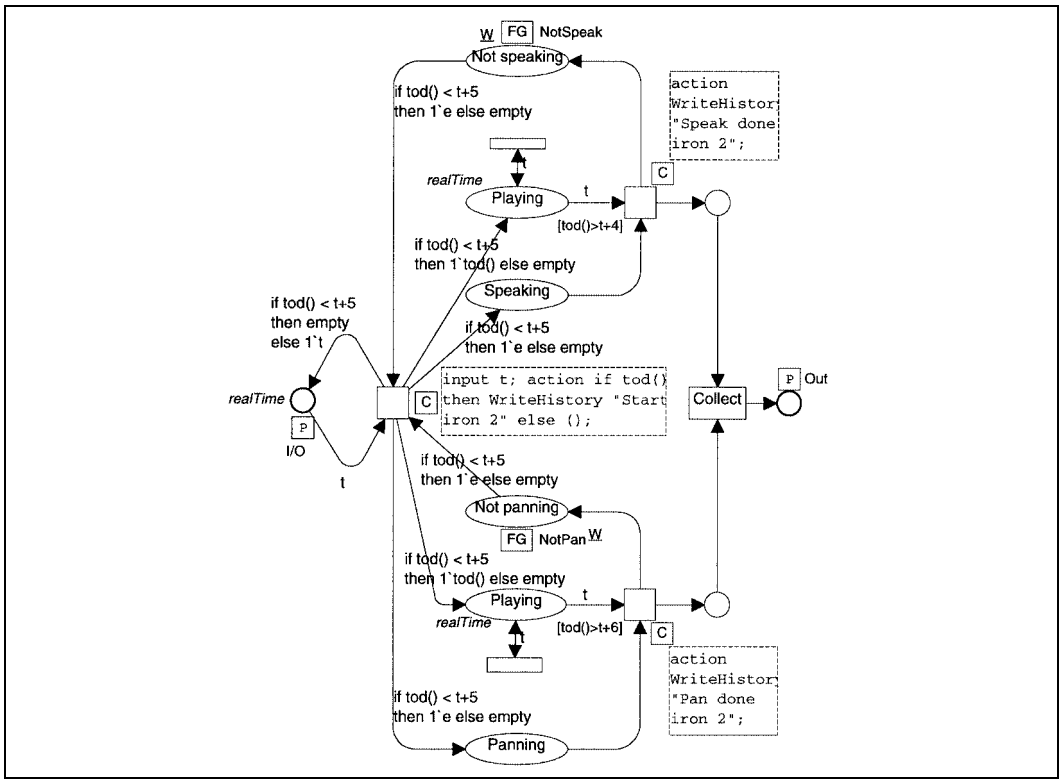


Figure A.50: Ir2#13. This construction turned out to be the culpit when the text line 'Start Iron 2' was missing completely from the output. Instead of the if statements on almost all its input and output arcs, the if statement in the code region should control which tokens are delivered when the transition fires. Tokens from input places can be taken unconditionally; it only requires that output arcs are added to put them back in the case where they should not have been taken.

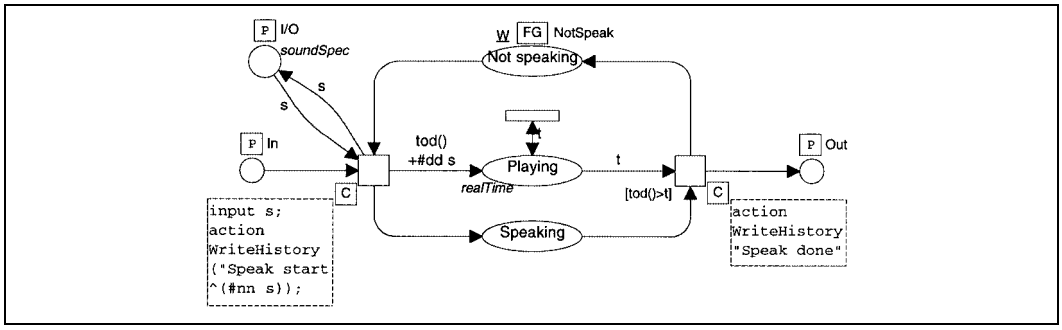


Figure A.51: SpeakMdl#14.

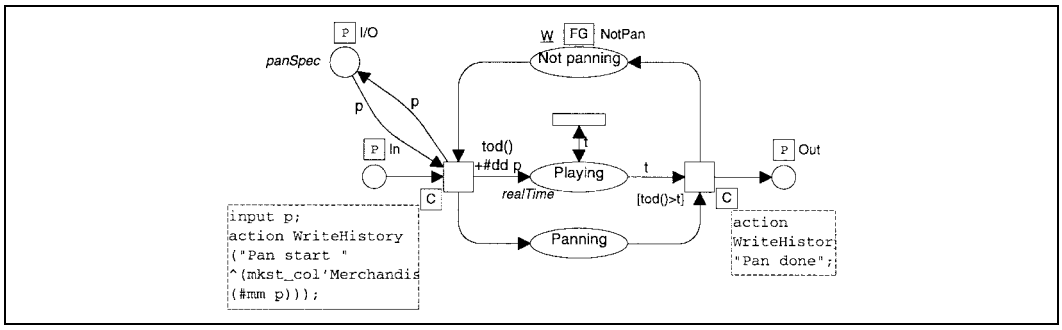


Figure A.52: PanMdl#15.

Graphical representation of output from the net

The figure at the bottom of this page and the two figures on the next page show output from three runs of the Petri net just shown. The way to read the figures is explained in subsection 8.4.3, pages 144–146. The variations between the runs are small.

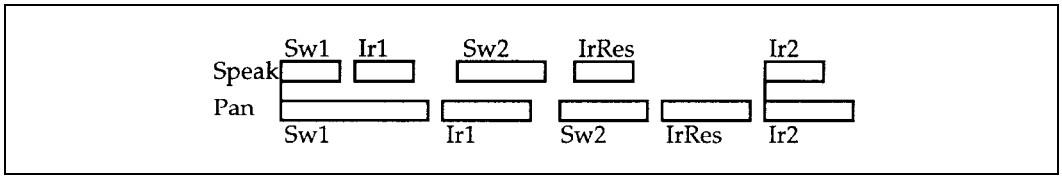


Figure A.53: .

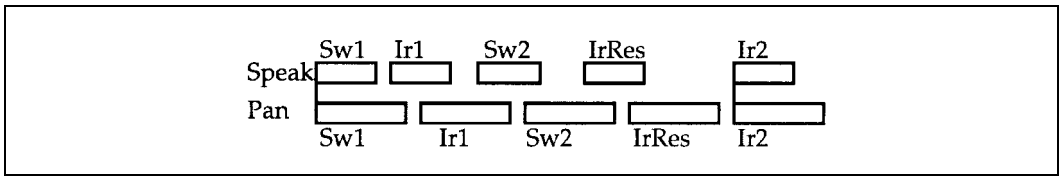


Figure A.54: .

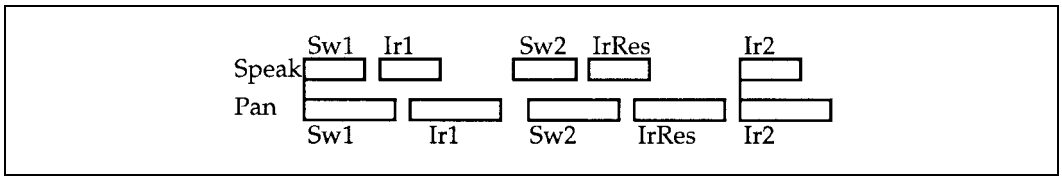


Figure A.55: .

Bibliography

[ ] (In the ordering of authors, ø is alphabetized as o.)

[1] Abildskov, Lars Peter, & Paul Erik Dahl: QuickTime. Unpublished project report, Department of Computer Science, Aarhus University, December 15, 1991. (In Danish.)

[2] Akscyn, Robert M., Donald L. McCracken and Elise A. Yoder: KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations. In Communications of the ACM, vol. 31 No. 7, July 1988.

[3] Andersen, Peter Bøgh: A Theory of Computer Semiotics. Semiotic approaches to construction and assessment of computer systems. Cambridge series on Human-Computer Interaction, Cambridge University Press 1990.

[4] Andersen, Peter Bøgh: Interaktive væker. En katastrofeteoretisk tilgang. In Per Aage Brandt and others (editors): Almen semiotik nr. 5. Tema: Computersemiotik Psykosemiotik. Aarhus Universitetsforlag 1992. Pages 89–112. (In Danish.)

[5] Andersen, Peter Bøgh: Katastrophen und Computer. In Roland Posner (journal editor), Martin Warnke & Peter Bøgh Andersen (guest editors): Zeitschrift für Semiotik, Band 16, Heft 1–2. Stauffenburg verlag 1994. Pages 29–50. (In German.)

[6] Andersen, Peter Bøgh: Medien — Kunst — Computer/Hyperkult I–IV. In an antology edited by Wolfgang Coy, Georg-Christoph Tholen and Martin Warnke, forthcoming (in German).

[7] Andersen, Peter Bøgh: Narratives. Unpublished working paper, September 1994. Part of the paper appears as section 5 in Peter Bøgh Andersen: Medien — Kunst — Computer/ Hyperkult I–IV, see above.

[8] Andersen, Peter Bøgh: Vector Spaces as the Basic Part of Interactive Systems: Towards a Computer Semiotics. In Patricia Baird (editor): Hypermedia, Volume 4, number 1, 1992. Taylor Graham. Pages 53–76.

[9] Andersen, Peter Bøgh, Berit Holmqvist, & Jens F. Jensen (editors): The computer as medium. Cambridge University Press 1993.

[10] Andersen, Peter Bøgh, Jens W. Johansen, Jacob A. Mikkelsen & Morten Sams: Interaktive tekster. In an anthology from Odense Universitetsforlag, in press. (In Danish.)

[11] Andersen, Peter Bøgh & Peter Øhrstrøm: Hypertid. WPCS–92–9, Working Papers in Cognitive Science and HCI, Centre for Cognitive Informatics, Risø National Laboratory, Roskilde University. (In Danish.)

[12] Anderson, N. "Medical center: a modular hypermedia approach to program design", In: *Sociomedia: Multimedia, Hypermedia, and the Social Construction of Knowledge,* E. Barrett (editor), MIT Press, Cambridge, pages 369–389, 1992.

[13] Andersson, Helle Juel, Lars Andersen, Berit Holmqvist, Bjørn Laursen, Peter Bøgh Andersen & Stig Jensen: Odins Øje. Rapport 1. Department of Information and Media Science, Aarhus University and The Antiquarian Collection in Ribe. (Undated, approximately 1993. Partly in Danish, partly in English.)

[14] Andersson, Helle Juel, Lars Andersen, Berit Holmqvist, Bjørn Laursen, Peter Bøgh Andersen, Stig Jensen, Thomas Østergaard, Evert B. Hassink & Steffen Sacher: Odins Øje. Rapport 2. Department of Information and Media Science, Aarhus University and The Antiquarian Collection in Ribe. (Undated, approximately 1994. Contains contributions in Swedish, Danish and English.)

[15] Balasubramanian, V. *Hypermedia issues and Applications, A State-of-the-Art Review,* Independent Research Report as part of Ph.D. Program, Graduate School of Management, Rutgers University, December 1993.

[16] Bernstein, M., J.D. Bolter, M. Joyce, and E. Mylonas "Architectures for volatile hypertext", *Hypertext 91: Proceedings of the Third ACM Conference on Hypertext,* ACM Press, pages 243–260, 1991.

[17] Beyer, Peter, et al.: Brugervenlige EDB-systemer. Teknisk Forlag A/S 1988. (In Danish.)

[18] Bonica, J.J. "Cancer pain", *The Management of Pain.* Lea and Febiger, Malvem, PA, second edition, 1990.

[19] Boose, J.H. "A knowledge acquisition program for expert systems based on personal construct psychology", *International Journal of Man-Machine Studies* 23, pages 495–525, 1985.

[20] Boose, J.H. *Expertise Transfer for Expert System Design.* Elsevier, New York, 1986.

[21] Boose, J.H. "Knowledge acquisition tools, methods, and mediating representations" H. Motoda, R. Mizoguchi, J. Boose, B. Gaines (editors), *Knowledge Acquisition for Knowledge-Based Systems,* Proceedings from JKAW'90 (The First Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop). IOS Press, Amsterdam, and Ohmsha, Ltd., Tokyo, 1991.

[22] Boose, J.H., D.B. Shema and J.M. Bradshaw "Recent progress in AQUINAS: a knowledge acquisition workbench", *Knowledge Acquisition,* 1, pages 185–214, 1989.

[23] Brandt-Pedersen, Finn: Tekstlæsning. Gyldendal 1967. (5th printing 1974. In Danish.)

[24] Brandt-Pedersen, Finn, and Anni Rønn-Paulsen: Metode bogen. Analysemetoder til litterære tekster. Nøgleforlaget Aps. 1980. (In Danish.)

[25] Branigan, Edward: Point of View in the Cinema. A Theory of Narration and Subjectivity in Classical Film. Foreword by David Bordwell. Mouton Publishers 1984.

[26] Brøndmo, Hans Peter, & Glorianna Davenport: Creating and viewing the *Elastic Charles:* a hypermedia journal. In Ray McAleese and Catherine Green (editors): Hypertext: State of the Art. Papers from UK Human Interface Interactive Learning Systems SIG conference on hypertext, Hypertext II, University of York, 1989. Intellect, Oxford, England, 1990.

[27] Brown, Paul: The Ethics and Aesthetics of the Image Interface. Presented at ASIS Mid Year Meeting 1993. Computer Graphics, newsletter of ACM SIGGRAPH, Volume 28, number 1, February 1994, pages 28–30.

[28] Canter, D., R. Rivers and G. Storrs "Characterising user navigation through complex data structures", *Behaviours and information Technology,* 4, pages 93–102, 1985.

[29] Chang, D.T. "HieNet: A user-centered approach for automatic link generation", *Hypertext 93: Proceedings of the Fifth ACM Conference on Hypertext,* Seattle, pages 145–158, 1993.

[30] Conklin, Jeff: Hypertext: An Introduction and Survey, *IEEE Computer,* vol. 20 no. 9, pages 17–41, September 1987.

[31] Cook, P. "An encyclopedia publisher's perspective, Interactive Multimedia", Apple Computer, Inc., Microsoft Press, 1988.

[32] Dahlerup, Pil: Dekonstruktion — 90'ernes litteraturteori. Gyldendal 1991. (In Danish.)

[33] Davenport, Glorianna: Bridging Across Content and Tools. Computer Graphics, newsletter of ACM SIGGRAPH, Volume 28, number 1, February 1994, pages 31–32.

[34] Deleuran, Claus: Illustreret Danmarkshistorie for Folket. Volume 1–8. Ekstrabladets forlag 1988–1994. (In Danish.)

[35] Derrida, Jacques: De la grammatologie I: L'écriture avant la lettre. Collection critique 1979. (Original version 1967.)

English version: Of grammatology. Translated by Gayatri Chakravorty Spivak. The Johns Hopkins University Press 1982.

Danish version: Om grammatologi. Dansk oversættelse ved Lars Bonnevie og Per Aage Brandt. Arena 1970.

[36] Diaz, Michel, & Patrick Sénac: Time Stream Petri Nets. A Model for Timed Multimedia Information. In Robert Valette (editor): Application and Theory of Petri Nets 1994. 15th International Conference, Proceedings. Springer-Verlag 1994.

[37] Edwards, Deborah M., & Lynda Hardman: 'Lost in Hyperspace': Cognitive Mapping and Navigation in a Hypertext Environment. In Ray McAleese (editor): Hypertext: theory into practice, Ablex Publishing Corporation, New Jersey, and intellect books, Oxford, 1989. Pages 105–125.

[38] Eliot, T.S.: Selected Essays. Third enlarged edition April 1951. Faber and Faber Limited. Reprinted July 1972.

[39] Eliot, T.S.: The sacred wood. Essays on poetry and criticism. 1920. 7th edition, reprinted: Methuen 1960.

[40] Ellekilde, H., (after A. Olrik): Heimdal. Entry in Salmonsens konversationsleksikon. Anden udgave, Bind XI: Hasselmus-Hven (2nd edition, volume XI). A/S J.H. Schultz Forlagsboghandel 1921. Pages 149–150. (In Danish.)

[41] Ford, L., C. Hunter, P. Diehr, R. Frelick and J. Yates "Effects of patient management guidelines on physician practice guidelines: the Community Hospital Oncology Program experience", *Journal of Clinical Pharmacology,* 5, pages 504–511, 1987.

[42] Fowler, H.W.: A dictionary of modem English usage. Second edition, revised by Sir Ernest Gowers. Oxford at the Clarendon Press 1965.

[43] Gallant, S.I. "A practical approach for representing context and for performing word sense disambiguation using neural networks", *Neural Computation,* 3, pages 293–309, 1991.

[44] Gardner, E. "Putting guidelines into practice", *Modern Healthcare,* pages 24–36, 1992.

[45] Golub, Gene H., & Charles F. Van Loan: Matrix Computations. Second edition. The Johns Hopkins University Press 1989.

[46] Greimas, Algirdas Julien, & Joseph Courtés: Semiotik. Sprogteoretisk ordbog. Aarhus universitetsforlag 1988. In Danish.

Original French version: Semiotique. Dictionnaire raisonné de la théorie du language. Hachette 1979.

English version: Semiotics and language: an analytical dictionary. Transl. by Larry Crist ... (et al.). Indiana University Press 1982.

Spanish version: Semiotica: diccionario razonado de la teoria del lenguaje. Version espanola de Enrique Ballon Aguirre y Hermis Campodonico Carrion. Gredos 1982.

[47] Grilli, R., A. Alexanian, G. Apolone, S. Marsoin, A. Nicolucci, V. Torri, M. Compagnucci, E. DiMarnbro et al. "The impact of cancer treatment guidelines on actual practice in Italian general hospitals: The case of ovarian cancer", *Annals of Oncology,* 1, pages 112–118,1991.

[48] Grønbæk, Kaj, Jens A. Hem, Ole L. Madsen & Lennert Sloth: Designing Dexter-based Cooperative Hypermedia Systems. In Hypertext '93 Proceedings. Fifth ACM Conference on Hypertext Proceedings. ACM 1993. Pages 25–38.

[49] Gyldendals Tibinds Leksikon, sjette bind (volume 6), kve-mum. Gyldendal 1978. (In Danish.)

[50] Halasz, F. "Reflections on NoteCards: Seven issues for the next generation of hypermedia systems", *Communications of the ACM,* 31, pages 836–852, 1988.

[51] Halasz, F., & M. Schwartz: The Dexter Hypertext Reference Model. In Communications of ACM, vol. 37 no. 2, pages 30–39. 1994.

[52] Hammond, N.V. "Hypermedia and learning: who guides whom?" *Computer assisted learning—ICCAL 89,* H. Maurer (editor), Springer-Verlag, pages 167–181, 1989.

[53] Hammond, Nick V., and Lesley J. Allinson "Travels around a learning support environment: rambling, orienteering or tutoring?", *CHI '88 Conference Proceedings: Human Factors in Computer Systems,* Special Issue of the SIGCHI Bulletin, Elliot Soloway, Douglas Frye and Sylvia B. Sheppard (editors), ACM Press, pages 269–273, 1988.

[54] Hansen, Carsten, Torsten B. Hagemann, Dag A. Helstad: Valhall — et projekt om multi-medier i objektorienteret programmering. Unpublished master's thesis, Department of Computer Science, Arhus University, December 14, 1992. (Partly in Danish, partly in Norwegian.)

[55] Hart, Horace: Hart's Rules for Compositors and Readers at the University Press, Oxford. Thirty-eighth edition, completely revised. Oxford University Press 1978.

[56] Hesselaa, Birgitte: Kan detektiver synge? In Kritik 85, pages 41–58. Gyldendal 1988. (In Danish.)

[57] Hintzman, D.L. "Schema Abstraction in a multiple-trace memory model", *Psych Rev,* 93, 4, pages 411–428, 1986.

[58] Huber, Peter, Kurt Jensen & Robert M. Shapiro: Hierarchies in Coloured Petri Nets. In G. Rozenberg (editor): Advances in Petri Nets 1990. Lecture Notes in Computer Science vol. 483, Springer Verlag 1991, pages 313–341. (Also in Kurt Jensen and G. Rozenberg (editors): High-level Petri Nets. Theory and Application. Springer Verlag 1991, pages 215–243.)

[59] Ichimura, S, and Y. Matsushita "Another dimension to hypermedia access", *Hypertext '93: Proceedings of the Fifth ACM Conference on Hypertext,* Seattle, pages 63–72, 1993.

[60] Jacox, A., D.B. Carr, R. Payne et al. *Management of Cancer Pain. Clinical Practice Guideline* No. 9. AHCPR Publication No. 94-0592. Rockville, MD. Agency for Health Care Policy and Research, U.S. Department of Health and Human Services, Public Health Service, March 1994.

[61] Jensen, Johan Fjord: Den ny kritik. Berlingske Forlag 1962. 2nd printing Munksgårds forlag 1966. Reprinted with an epilogue by the author: Kimære 1989. (In Danish.)

[62] Jensen, Kurt: Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume I: Basic Concepts. EATCS Monographs on Theoretical Computer Science, Springer-Verlag, 1992.

[63] Jørgensen, John Chr.: Litterær metodelære. Metoder i dansk litteraturforskning efter 1870. 2. udgave (2nd edition). Borgen 1974.

[64] Kelly, G.A. *The Psychology of Personal Constructs.* New York: Norton, 1955.

[65] Kibby, M.R., and J.T. Mayes "Towards intelligent hypertext", *Hypertext: theory into practice,* R. McAleese (editor), Ablex Publishing Corporation, New Jersey, and intellect books, Oxford, pages 164–172, 1989.

[66] Kiffer, Mary E. (associate editor): Current Biography. Cumulated Index 1940–1985. The H.W. Wilson Company, New York 1986.

[67] Lapsley, Robert, and Michael Westlake: Film Theory: An Introduction. Manchester University Press 1988. Reprinted approximately 1994.

[68] Larsen, Peter Harms: Faktion — som udtryksmiddel. Dansklærerforeningen and Forlaget Amanda 1990. (In Danish.)

[69] Lauridsen, Palle Schantz: Christian Metz' filmsemiotik. In Sekvens — filmvidenskabelig årbog 1984. Københavns Universitet. Institut for Filmvidenskab. (In Danish.)

[70] Laursen, Bjørn: Tegning og Kognition. VENUS Report No. 6. Department of Information and Media Science, University of Aarhus, 1990.

[71] Levy, M.H. "Pain management in advanced cancer", *Semin Oncol.,* 12, pages 394–410, 1985.

[72] Lomas, J., G. Anderson, K. Dominick-Pierre, E. Vayda, M. Enkin and W. Hannah "Do practice guidelines guide practice: The effect of a consensus statement on the practice of physicians", *New England Journal of Medicine,* 321, pages 1306–1311, 1989.

[73] Lucarella, D., S. Parisotto and A. Zanzi "MORE: Multimedia object retrieval environment", *Hypertext '93: Proceedings of the Fifth ACM Conference on Hypertext,* Seattle, pages 39–50, 1993.

[74] Maartmann-Moe, Erling: Multimedia. Universitetsforlaget, Oslo 1991. (In Norwegian.)

[75] Madigan, D., and C.R. Chapman: Multimedia tools for cancer pain education. In C. Ghaoui and R. Rada (editors): Medical Multimedia, Intellect 1995, pages 121–136.

[76] Madigan, David, C. Richard Chapman, Jonathan Gavrin, Ole Villumsen & John Boose: Repertory Hypergrids: An Application to Clinical Practice Guidelines. In European Conference on Hypermedia Technology 1994 Proceedings. ACM Press 1994. Pages 117–125.

[77] Madsen, Ole Lehrmann, Birger Møller-Pedersen & Kristen Nygaard: Object-Oriented Programming in the BETA Programming Language. Addison-Wesley Publishing Company 1993

[78] Madsen, Svend Åge: Dage med Diam *eller* Livet om natten. Gyldendal 1972. (In Danish. New paperback edition available.)

[79] de Man, Paul: The Rhetorics of Romanticism. Columbia University Press 1984.

[80] Marchionini, G. and B. Schneiderman "Finding facts and browsing knowledge in hypertext systems", *IEEE Computer,* 21, 1988.

[81] Marshall, C.C. and F.M. Shipman "Searching for the missing link: Discovering implict structure in spatial hypertext", *Hypertext '93: Proceedings of the Fifth ACM Conference on Hypertext,* Seattle, pages 217–230, 1993.

[82] Mjølner BETA System. MacEnv Tutorial. Mjølner Informatics Report. MIA 91-18(0.2). March 1992.

[83] Mjølner BETA System. Macintosh Libraries. Mjølner Informatics Report. MIA 90-10(0.6). March 1992.

[84] Monaco, James: How to read a Film. The Art, Technology, Language, History, and Theory of Film and Media. Revised Edition. With diagrams by David Lindroth. Oxford University Press 1981.

[85] Morariu, J., and B. Schneiderman "Design and research on the interactive encyclopedia system (TIES)" *Proceedings of the 29th Conference of the Association for the Development of Computer-based instructional Systems,* pages 19–21, 1986.

[86] Nelson, Theodor H.: Getting It Out of Our System. In G. Schechter (editor): Information Retrieval: A Critical Review. Thompson Books 1967.

[87] Nielsen, J. *Hypertext and Hypermedia.* New York: Academic Press, 1990.

[88] Nissen, Dan, & Anne Jerslev: Film- og TV-analyse. In Ib Brøkner Christiansen & Else Lützhøft (editors): Billeder i bevægelse. Fakta og fortolkninger. Foreningen af filmlærere i gymnasiet og HF and Dansklærerforeningen (FFS) 1984. Pages 145–240. (In Danish.)

[89] Noik, E.G. "Exploring large hyperdocuments: Fisheye views of nested networks", *Hypertext '93: Proceedings of the Fifth ACM Conference on Hypertext,* Seattle, pages 192–199, 1993.

[90] Philips, Rob: Producing Interactive Multimedia Computer-Based Learning Projects. Computer Graphics, newsletter of ACM SIGGRAPH, Volume 28, number 1, February 1994, pages 20–24.

[91] Portenoy, R.K. "Cancer pain: epidemiology and syndromes", *Cancer.* 63, pages 2298–2307, 1989.

[92] QuickTime Developer's Guide. Draft. Developer Technical Publications, Apple Computer, Inc. 1991.

[93] Salton, G., J. Allan and C. Buckley "Automatic structuring and retrieval of large text files", *Communications of the ACM,* 37, pages 97–108, 1994.

[94] Sandvad, Elmer: Object-Oriented Development — Integrating Analysis, Design and Implementation. PB–302. Computer Science Department, Aarhus University 1990.

[95] Schank, Roger, & Chip Cleary: Engines for Education. Lawrence Erlbaum Associates 1995. CD-ROM version for Macintosh available. WWW version: `http://www.ils.nwu.edu/~e_for_e/`.

[96] Sénac, Patrick, Roberto Willrich & Michel Diaz: Hypermedia Synchronization Modelling: A Case Study. In Ed-media 95 World Conference on Educational Multimedia and Hypermedia Proceedings. 1995. A postscript version is available on WWW at `http://hyperg.iicm.tu-graz.ac.at/edmedia_apers_ps`, under Diaz. Proceedings also available on a CD-ROM.

[97] Sénac, Patrick, Pierre de Saqui-Sannes & Roberto Willrich: Hierarchical Time Stream Petri Net: a Model for Hypermedia Systems. In Giorgio De Michelis & Michel Diaz (editors): Application and Theory of Petri Nets 1995. 16th International Conference, Turin, Italy, June 26–30,1995, Proceedings. Lecture Notes in Computer Science vol. 935, Springer Verlag 1995, pages 451–70.

[98] SGML SIGhyper Newsletter. An Occasional Publication of the Standard Generalized Markup Language (SGML) Users' Group's Special Interest Group on Hypertext and Multimedia (SIGhyper). Volume 1, Number 1, October 1991.

[99] Sheplock, G.J., P.S. Thomas and E.M. Camporesi "An interactive computer program for teaching regional anesthesia", *Anesthesiology Review,* 20, pages 53–59, 1993.

[100] Steinmetz: Human perception of multimedia synchronization. IBM technical report No. 43.9310.

[101] Stotts, P. David, & Richard Furuta: Petri-Net-Based Hypertext: Document Structure with Browsing Semantics. ACM Transactions on Information Systems, Volume 7, Number 1, January 1989, pages 3–29.

[102] Stotts, P. David, & Richard Furuta: The Trellis Hypertext Project. Position paper for Petri net models and CSCW Workshop. In Computer-Supported Cooperative Work, Petri Nets and Related Formalisms. A one-day workshop, Chicago, June 22,1993, Proceedings, pages 1–12.

[103] Tang, H., R. Bardn and C. Clifton "A new learning environment based on hypertext and its techniques", *Advanced Research on Computers in Education,* R. Lewis and S. Otsuki (editors), North-Holland, pages 119–127, 1991.

[104] Tompa, F.W., G.E. Blake and D.R. Raymond "Hypertext by link-resolving components", *Hypertext '93: Proceedings of the Fifth ACM Conference on Hypertext,* Seattle, pages 118–130, 1993.

[105] Tompkins, Jane P. (editor): Reader-Response Criticism. John Hopkins University Press, Baltimore and London (1980) 1984.

[106] Trigg, R. "Guided tours and tabletops: Tools for communicating in a hypertext environment", *ACM Transactions on Office Information Systems,* 6, pages 398–414, 1988.

[107] Tudhope, Douglas, Carl Taylor & Paul Beynon-Davies: Navigation via Similarity in Hypermedia and Information Retrieval. In Rainer Kuhlen & Marc Rittberger (editors): Hypertext-Information Retrieval-Multimedia (HIM'95). HIM'95 conference proceedings. UVK Universitätsverlag Konstanz GmbH 1995.

[108] Utting, K. and N. Yankelovich "Context and orientation in hypermedia networks", *ACM Transactions on information Systems,* 7, pages 58–84, 1989.

[109] Vaughan, Tay: Multimedia: Making It Work. Second edition. Macromedia/Osbome McGraw-Hill 1994.

[110] Villumsen, Ole: Hejmdal — an object-oriented platform for working with interactive multimedia. In Third Eurographics Workshop on Object-Oriented Graphics Preprints, Centre universitaire d'informatique, Université de Genève, October 1992, pages 467–481.

[111] Waltz, D.L. and J.B. Pollack "Massively parallel parsing: a strongly interactive model of natural language interpretation", *Cognitive Science,* 9, pages 51–74, 1985.

[112] Watt, David A. (with contributions by William Findlay and John Hughes): Programming Language Concepts and Paradigms. Prentice Hall.

[113] Wellek, René, & Austin Warren: Theory of Literature. Third Edition. ("New revised edition".) A Harvest Book. Harcourt, Brace & World, Inc. 1962.

Danish edition: Rene Wellek & Austin Warren: Litteraturteori. Munksgaard 1964.

[114] Who was who. A cumulated index 1897–1990. A & C Black, London 1991.