Using a Hypermedia System for Systems Engineering

Jörg M. Haake, Ajit Bapat, Thomas Knopik
GMD-IPSI, Dolivostraße 15, D–64293 Darmstadt,
e-mail: haake@darmstadt.gmd.de

Published in *Proceedings of the East–West International Conference on Multimedia, Hypermedia, and Virtual Reality (MHVR'94)*, Moscow, Russia, Sept. 14–16, 1994, pp. 63–68

1 Introduction

Today's complex technical systems like e.g. cars, aircraft, or power stations have growing requirements with regard to their security, environmental and social compatibility. Therefore, a planned system should be validated before its realization and (mass) production. A second aspect are the increasing demands by legal regulations (e.g. based on ISO 9000). In order to accomplish such a validation task the presentation of the information on the technical system has to meet various demands:

- Different views to the system.
  Different groups of participants of the validation process have different knowledge of the system's functionality and also have different information requirements of the system's properties (e.g. customer, test engineer, neighbour of a plant).
- Assisting the validation process.
  Because of the growing complexity of technical systems keeping track of everything is not easy. Therefore, the validation should be aided by a guided exploration of the available system information. Furthermore, there should be a way to run experiments concerning the system's functions (i.e. simulations), including exceptional system states.
- Feedback for the modelling process.
  Results and findings of the validation should have repercussions on the modelling process. To achieve this, some means of conveying critics and suggestions for improvement has to be provided.

Today, system representation consists of large amounts of textual information enriched by graphics or CAD modelling data. Some special parts of the representation — coded in formal languages — allow for simulations. The connections between different parts of the representation are often implicit and informal. In addition, expensive physical prototypes are used sometimes. One property of the process of creating systems representation is its collaborative nature. In complex technical systems, a large number of people is involved and often enough groups work on shared documents. Thus, the need for coordination arises.

In order to overcome these problems the MuSE[1] project[2] [Lux 93] aims at providing a computer-based environment for the engineering of such technical systems. This environment comprises both the validation of the system model and the support of the modelling techniques required by the designer.

In the basic architecture of the MuSE environment the technical system is represented by a model. Specific views of the model are provided by interfaces for validation and modelling. The interfaces act as filters to the model, thus showing information that is relevant to the respective views. Via these interfaces the validators and the designers can perform their tasks, i.e. examining resp. constructing/changing the model of the specified system.

2 The role of hypermedia in MuSE

As described above, the model that is to be created and maintained in the MuSE environment consists of a large number of pieces of information. These units of information are of different kinds of media. E.g., documentation may be provided as an SGML-text, specifications may come along as descriptions in some formal language, simulations may be documented by video sequences, possibly combined with audio annotations, etc. Various relationships may exist between these units: variants to a design proposal, part-of hierarchies, used-by hierarchies, references between models and simulation results, references between different parts of the documentation, references from component parts to associated documents, and many more.

Thus, the MuSE model is made up of two constituent parts: units of information on the one hand and relationships or references on the other hand. Together, these two elements form an information network. Because of the multimedia aspect of the information units and since the references may not only occur between two units but also from within one unit into another (e.g. from a comment in a program's source code into an associated text document exactly to that part of text which is related to the comment) hypermedia suggests itself as an appropriate interface between the MuSE model and the users (system validators and designers) [Schütt et al. 93].

---

1. Multimediale Systementwicklung (Multimedia Systems Engineering)
2. The MuSE project is sponsored by the *Deutsche Forschungsgemeinschaft* (DFG), grant number HE 1170/5–1.

We can identify the following major requirements for a hypermedia system in MuSE:

- The whole spectrum of different media used in MuSE (as described above) has to be made available by the hypermedia environment.
- The organization of the model's information has to be supported by the environment. This calls for the ability to both express explicit relationships (e.g. references between two documents) and cluster information in subnets (e.g. part-of hierarchy).
- Apart from representing the existing information, the MuSE environment calls for an integration of the various tools used in a systems engineering environment, e.g. CAD tools, editors, compilers for different programming languages, etc. These tools not only change the existing information but also produce new units of information which need to be stored and whose relationships to the existing information have to be represented. Thus, the hypermedia environment has to be able to integrate information from various heterogeneous sources (like e.g. application programs).
- Furthermore, in order to support the specific tasks of modelling or validation (as mentioned in section 1) corresponding task-specific views need to be provided on the complex hypermedia network.
- Modelling and validation is an iterative process where the findings of the validators lead to feedback for the modelers who then have their new model validated once more. Therefore, appropriate ways to support this process have to be found.

In order to develop such a hypermedia-based system several issues arise:

- How to represent the multitude of information?
- How to realize different, task-specific views?
- How to integrate external tools?
- How to (re-)integrate data produced by external tools?
- How to assist validation and how to provide feedback for the modelling process?

As an implementation basis for the MuSE hypermedia system the cooperative hypermedia authoring environment SEPIA [Streitz et al. 92] was chosen. Originally, SEPIA was designed to support hypermedia authoring and hyperdocument production. Because of its generic concepts, it was possible to tailor and extend SEPIA to comply with the requirements of MuSE. In the following section we will describe this tailored SEPIA — MuSE–SEPIA for short — and how it addresses the above issues.

3 MuSE–SEPIA

The MuSE hypermedia system is based on the cooperative hypermedia authoring environment SEPIA which has been developed at GMD–IPSI with the aim to support the production of hypermedia documents. By extending SEPIA with MuSE-specific hypermedia object types (nodes and links), a connection of the hypermedia server to the VODAK/VML object-oriented multimedia database management system [Klas et al. 93] and the integration of MuSE tools (e.g. a simulator program, specific editors and compilers) we developed MuSE–SEPIA, a cooperative hypermedia modelling and validation environment (cf. Figure 1). In the centre of the system architecture are the components that are used to specify the technical system. The communication component connects the hyperdocument database to specification components. MuSE–SEPIA as the hypermedia authoring environment forms the homogeneous interface that resides at the top. The left part of figure 1 shows the integration of external tools which will be described below.

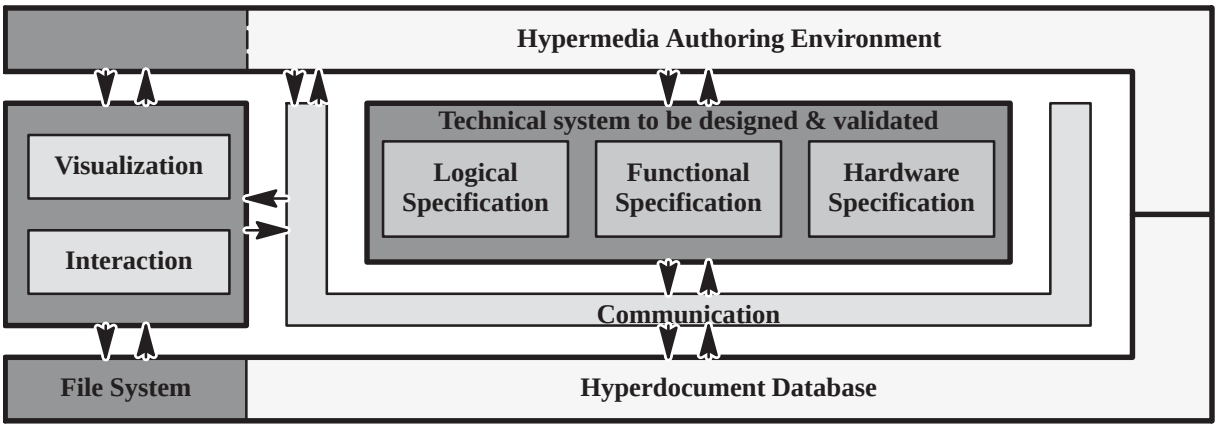

Figure 1: Architecture of the MuSE–System

Since tasks like modelling and validating a technical system are highly collaborative, a supporting system definitely needs to enable the cooperation of the various users. The authoring environment SEPIA already supports collaboration [Haake & Neuwirth 93, Haake & Wilson 92, Streitz et al. 92]. The cooperative capabilities of SEPIA were adopted by MuSE–SEPIA, but since it would certainly go beyond the scope of this paper, the different cooperative aspects of the system are not covered here.

In the following, we will briefly describe how this environment presents itself to the user. After that, we will show how the requirements mentioned in section 2 are met and how the issues that arose from the requirements are answered by MuSE–SEPIA.

3.1 Description of MuSE–SEPIA

In SEPIA, the process of authoring is supported by the concept of activity spaces. An activity space provides problem-specific objects and operations to facilitate the author's activities when working on the problem. In MuSE, two spaces support the designer and the validator in the corresponding subproblems:

- the modelling space for modelling the technical system
- the validation space for validating the modelled system

For each space, a number of dedicated hypermedia object types (nodes and links) are defined. There are node types and link types common to both spaces (e.g. "specification" node, "visualizes" link), but there are also types which are available in only one of the spaces (e.g. "test protocol" node in the validation space). The various node types are subdivided into composite nodes and atomic nodes. The composite nodes allow for the clustering of subnets. Within a composite node it is again possible to instantiate objects from a predefined set of available node and link types (e.g. composite node "model" may contain the (composite) nodes "variant", "requirement", "regulation" and "information"). To support the iterative design process between modelling and validating, MuSE–SEPIA allows the automatic transfer of design objects between the activity spaces, their re-use, and the indication and control of references between activity spaces.

These concepts are supported by the user interface, where the activity spaces and the contents of composite nodes are presented as graph browsers. Within such a browser, the user can view and rearrange the displayed network, create new nodes and links (restricted to those that are available in the specific browser), inspect and edit node contents or open composite nodes which then again display their subnet in a graph browser. In addition, a "roaming box" in each browser helps to keep an overview of the displayed (sub)net.

Beyond the possibilities for structuring at the network level, MuSE–SEPIA also supports multimedia contents of nodes. All atomic (document) nodes (i.e. those not containing any substructures) may carry multimedia information. In addition to "classic" media like text, graphics, and audio, MuSE–SEPIA was extended to allow atomic nodes also contain videos, specifications in formal languages, CAD data, etc. Furthermore, certain media types are refined, e.g. texts are subdivided into "primitive text", "C text" (i.e. C source code), "configuration text", etc. This refinement was achieved by providing various document node types.

3.2 How to represent the multitude of information

As discussed above (section 2), we can consider the structure of the MuSE information components (objects and references) as a network of hypermedia objects, i.e. a hypernet. Therefore, it is reasonable to use a hypermedia data model for the representation of the MuSE data. SEPIA provides a generic object model with roles for objects, contents and contexts which determine their constraints and behaviour. The notion of roles assists the tailorability of the system: For MuSE, we introduced new content types like "C text" or "CAD data", new object types like "specification" node or "visualizes" link, and new context types for task-specific views like "modelling" and "validation". By using these concepts, we were able to adapt SEPIA's basic functionality for presenting and editing hypermedia structures to the needs of the MuSE environment. The graph browsers, in combination with the clustering of subnets as composite nodes, provide an adequate means to represent and organize the broad spectrum of information in the MuSE environment.

3.3 How to realize different, task-specific views

The realization of different views is achieved by the use of two different activity spaces. These take into account the task-specific needs of the two processes in MuSE — modelling and validation. By concealing individual objects (nodes or links) and by making certain node and link types available only in designated activity spaces or composite nodes the requirement for separate views can be met.

3.4 How to integrate external tools

The first step towards the integration of external tools is the differentiation between nodes and node contents. By tailoring the generic roles for node contents, external applications can be associates with the access of a particular

content. With the invocation of the external tool, MuSE–SEPIA hands over control to the external tool. Upon ending the application, control is returned to MuSE–SEPIA. In the meantime, MuSE–SEPIA marks the content as being used. The interface to an external tool is realized by providing a description of the mapping of abstract functions (e.g. “open”, “play”, “execute”) onto specific functions of the application (e.g. call-syntax). Standard node menus let users invoke the external application accordingly. The left part of Figure 1 shows how external tools interact with the rest of the MuSE environment.

3.5 How to (re-)integrate data produced by external tools

Closely intertwined with the integration of external applications is the issue of integrating the data that is produced by these tools. There are two dimensions to be considered:

One aspect is whether the data itself is kept in the database of the hypermedia system or just references are stored in the database while the data is kept in external files. The first solution makes it easier to control the consistency of the data because the database has means for consistency maintenance. The second approach simplifies the access to the data by the external tool since these tools operate on data managed by the file system. For MuSE–SEPIA we have realized both approaches depending on the tool that is associated with the data. E.g., audio data is an integrated data type within the VODAK database system. Therefore, the tool for editing audio data etc. can be accessed directly via the database system. CAD data can only be accessed by an external CAD tool. Therefore the data is kept in external files. A third solution in this context would be to keep all data in the database and to externalize it on demand, i.e. to write the data into a file only shortly before invoking the external tool and re-importing the (changed) data after the application has been ended. Thus, the lack of consistency control can be reduced to a minimum. Using the information of which contents are being used, MuSE–SEPIA can avoid collisions between different users accessing the same content.

The other dimension deals with the hypertext-awareness of the tools used. If an external tool does not support hypertext features like e.g. the insertion of embedded links, there is no way to bypass this drawback. MuSE–SEPIA is not able to extend existing tools. In order to make such features available, the external application has to provide them or the tool has to be an integral part of MuSE–SEPIA. The latter approach was chosen for the text editor within SEPIA, where the standard text editor of the Smalltalk programming environment was extended by features to support embedded links.

3.6 How to assist validation and how to provide feedback for the modelling process

To design a complex technical system is an iterative process that alternates between the modelling and the validation of the system several times before a final state is reached. After an initial modelling and validating phase the work of each phase is influenced by the predecessing task: The validator makes annotations describing needs for changes to which the modeller responds with a (partial) system redesign. Following that, the validator revisits (parts of) the system’s model in order to see in how far his requests have been met. This pattern continues until the validator is finally satisfied and approves the technical system.

These alternating processes are both supported by MuSE–SEPIA. First of all, two different views of the model are provided: one for the system designer(s) and one for the system validator(s). Within these views the special needs of the two different tasks can then be supported appropriately.

For the validation process, MuSE–SEPIA offers a view of the system model as it was designed during the modelling process. Within this view, validators can make annotations to any part of the hyperdocument network, reflecting comments to the system model, e.g. expressing remarks about legal restrictions that are not met by the designed system. On the other hand, the view does not allow any modification of the system model itself, since this is not within the validator’s field of responsibility. In order to see what changes were made to the system since the validator last looked at it, versioning has to be provided. With the integration of a version server [Haake & Haake 93] it will be possible to see how annotations have actually been considered in a redesign of the technical system.

The annotations by the validators are made available for the system designers within their view of the network to let them know what changes are necessary in order to comply with the conditions set up by the validators. This way, the designers get the feedback they need within the iterative process of modelling and validating and can respond adequately to the requests of the validators by making appropriate changes to the technical system.

4 Related work

Related work can be found in two main fields: One is the field of software engineering, mainly focussing on CASE tools. The other is the wide range of hypertext related approaches. In the following, we have compiled a selection of connected work.

The Personal Information Environment (PIE) [Goldstein & Bobrow 84] was designed to create alternative software designs, to examine their properties, and to then choose one alternative as the final production version. In

PIE, software systems are modelled by layered networks. The network represents the sundry objects of a system (modules, procedures, code, comments, etc.) as well as the relationships between these elements. The notion of layered networks makes it possible to represent different versions by layers dominating previous ones. Thus, multiple views of the same document are achieved. The presentation of node contents is done by text browsers which, in addition to the actual node content, also provide information about the nodes location within the network, thus aiding the user's orientation.

DynamicDesign [Bigelow & Riley 87] is a CASE environment for the C programming language. It is based on the Hypertext Abstract Machine (HAM) [Campbell & Goodman 87], which functions as a transaction-based hypertext database server. DynamicDesign allows to edit hypertext objects (nodes and links), to navigate through hypertext graphs, to build hypertext graphs from existing C source code files, and to browse source code, documents and system requirements.

Neptune [Delisle & Schwartz 86] is another hypertext system based on the Hypertext Abstract Machine (HAM). It was designed for the support of large CAD applications. Neptune provides a generic documentation user interface which communicates with the HAM. The interface makes the hypertext information available via three primary kinds of browsers: graph browser to present a view of sub-graphs, document browser to view (parts of) the hierarchical structure of the hyperdocument, and node browsers to support the browsing of individual nodes.

KMS (Knowledge Management System, [Akscyn et al. 87]) is designed to organize information management — from individuals to small groups up to whole organizations. A KMS database consists of workspaces (frames) which may contain text, graphics and image items. Each of these items may be linked to another frame or used to invoke a program. Users navigate from frame to frame — displayed in combined browsers/editors — by following the existing links. Items can have programs attached to them, which are executed when the item is selected.

NoteCards [Halasz 87] is an environment primarily designed to assist people in idea processing, starting from a collection of unrelated ideas to an integrated, orderly interpretation of the ideas and their interconnections. The basis of NoteCards is a semantic network of nodes (called notecards) which are connected by typed links. Each notecard can be viewed by means of an editor that also displays the link icons which lead to their destination card(s). Apart from navigating from card to card via links there is also the possibility to use browsers that display a structural diagram of a network of notecards.

Microcosm [Davis et al. 92] is an open hypermedia system. It consists of document viewers, a document control system (DCS), a filter management system (FMS), and filters. The viewers allow users to browse data of different types of media. The DCS is responsible for starting new viewers and informing viewers of documents to be displayed. The FMS provides the message handling functionality. It receives messages from the document viewers (via the DCS) and passes them on to the filters. The filters respond to these methods by taking appropriate actions. One example for such a filter is a linkbase which provides the mechanism to follow links.

Within Sun's Link Service [Pearl 89], link and node data are stored separately. Standalone applications have to be made link-aware by using the Link Service's link library. This library offers a protocol that controls the communication between the applications. This approach can be regarded as a basic mechanism for coupling applications in a software engineering environment.

The Virtual Notebook System (VNS) [Shipman et al. 87] is a hypertext system based on the Sybase relational database system. It is aimed at the acquisition and management of information. By clearly separating the user interface from the data access mechanism for the hypertext, the integration of information resources across hardware and software platforms is aided.

gIBIS [Conklin & Begeman 88] is a hypertext tool that provides a graphical environment for the IBIS (issue based information systems) method. The tool is especially tailored for use with the IBIS method. It supports the three node and nine link types used when applying the method. External data can be referenced by a special node type called "surrogate" where a pointer to the data (usually a pathname to a file) and display program that is to be invoked when accessing the data are stored.

5 Summary and discussion

First, we presented the multimedia systems engineering project MuSE along with its requirements for information representation — including the demand for "openness" in order to integrate external tools. After deciding that hypermedia is an adequate means to meet these requirements, we outlined how the hypermedia system MuSE–SEPIA can satisfy the demands of the MuSE project by answering different issues that arise when deciding to use a hypermedia system for the representation of information in a systems engineering environment. Following that, we gave a brief overview of a selection of related work.

Looking for related work, we had to discover that up to now there is no system available that would cover the requirements of the MuSE project as a whole. Work that comes closest is that in the field of software engineering on the one hand, and hypertext-related approaches on the other. Software engineering-related work mostly focuses

on the support of the software life cycle and the issues of version control. The integration of different media and external applications is no primary subject in this field. The various hypertext-related systems, too, only consider some of the named issues but not the combination of all. For example, the gIBIS tool is aimed at supporting the IBIS methodology and thus only applicable to that very special domain. For our goal of supporting systems engineering, the integration of external data by simple references to files is not satisfactory because of the loss of consistency control.

With MuSE–SEPIA, on the other hand, we have provided a tool that meets all requirements that have been encountered within the MuSE project so far. These include the consistency of the data, particularly the demand for consistency for external data, a common user interface while browsing through the hyperdocument network and an overview function that provides an overall view of the network.

For the future, there are a number of issues to consider for MuSE–SEPIA, e.g. how navigation can be enhanced by retrieval functions, how to integrate hypertext-unaware tools, or how different hypermedia systems can be coupled. The last two examples lead to the general topic of "openness" of hypermedia systems. But instead of restricting openness to single platforms like with Sun's Link Service openness should go across different platforms.

6 Literature

Akscyn, R. M., McCracken, D. L., and Yoder, E. A., KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations. In *Proceedings of the First ACM Workshop on Hypertext (Hypertext '87)*, pages 1 – 20, University of North Carolina at Chapel Hill, November 13 – 15, 1987.

Bigelow, J. and Riley, V., Manipulating Source Code in Dynamic Design. In *Proceedings of the First ACM Workshop on Hypertext (Hypertext '87)*, pages 397 – 408, University of North Carolina at Chapel Hill, November 13 – 15, 1987.

Campbell, B. and Goodman, J. M., HAM: A General Purpose Hypertext Abstract Machine. In *Proceedings of the First ACM Workshop on Hypertext (Hypertext '87)* pages 21 – 32, University of North Carolina at Chapel Hill, November 13 – 15, 1987.

Conklin, J. and Begeman M. L., gIBIS: A Hypertext Tool for Exploratory Policy Discussion. In *Proceedings of the ACM International Conference on Computer-Supported Cooperative Work (CSCW'88)*, pages 140 – 152, Portland, OR, September 26 – 29, 1988.

Davis H., Hall, W., Heath, I., Hill, G., and Wilkins, R., Towards an Integrated Information Environment With Open Hypermedia Systems. In D. Lucarella, J. Nanard, M. Nanard, and P. Paolini, eds., *Proceedings of the 4th ACM Conference on Hypertext (ECHT '92)* pages 181 – 190, Milan, Italy, November 30 – December 4, 1992.

Delisle, N. M. and Schwartz, M. D., Neptune: A Hypertext System for CAD Applications. In Carlo Zaniolo, ed., *Proceedings of the 1986 ACM-SIGMOD International Conference on Management of Data (SIGMOD '86)*, pages 132 – 143, Washington, D.C., May 28 – 30, 1986.

Goldstein, I. and Bobrow, D., A Layered Approach To Software Design. In D. Barstow, H. Shrobe, and E. Sandewell, eds., *Interactive Programming Environments*, pages 387 – 413, McGraw Hill, 1984.

Haake, A. and Haake J. M., Take CoVer: Exploiting version support in cooperative systems. In *Proceedings of the InterCHI'93*, pages 406 – 413, Amsterdam, Netherlands, April 26 – 29, 1993, ACM Press.

Haake, J. M. and Neuwirth, C. M., Collaborative Authoring of Hyperdocuments. In *Proceedings of "Translating and the Computer, 15"*, pages 41 – 58, London, November 17 – 18, 1993.

Haake, J. M. and Wilson, B., Supporting Collaborative Writing of Hyperdocuments in SEPIA. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW '92)*, pages 138 – 146, Toronto, Canada, October 31 – November 4, 1992.

Halasz, F. G., Reflections on Notecards: Seven Issues for the Next Generation of Hypertext Systems. In *Proceedings of the First ACM Workshop on Hypertext (Hypertext '87)*, pages 345 – 365, University of North Carolina at Chapel Hill, November 13 – 15, 1987.

Klas, W., Aberer, K., Neuhold, E., Object-Oriented Modelling for Hypermedia Systems using the VODAK Modelling Language (VML). In *Object-Oriented Database Management Systems, NATO ASI Series*, Springer, Berlin, 1993.

Lux, G., MuSE — A Technical Systems Engineering Environment. Technical University Darmstadt, Department of Computer Science, 1993.

Pearl, A., Sun's Link Service: A Protocol for Open Linking. In *Proceedings of the 2nd ACM Conference on Hypertext (Hypertext '89)*, pages 137 – 146, Pittsburgh, PA, November 5 – 8, 1989.

Schütt, H., Andelfinger, U., Deegner, M., Kühnapfel, B., Henhapl, W., John, W., Lux, G., Wirth, H., MuSE–AG Hypermedia–Einsatz: Endbericht. Technical University Darmstadt, Department of Computer Science, June 1993.

Shipman, F. M., III., Chaney, R. J., and Gorry, G. A., Distributed Hypertext for Collaborative Research: The Virtual Notebook System. In *Proceedings of the 2nd ACM Conference on Hypertext (Hypertext '89)*, pages 129 – 135, Pittsburgh, PA, November 5 – 8, 1989.

Streitz, N., Haake, J., Hannemann, J., Lemke, A., Schuler, W., Schütt, H., and Thüring, M., SEPIA: A Cooperative Hypermedia Authoring Environment. In D. Lucarella, J. Nanard, M. Nanard, and P. Paolini, eds., *Proceedings of the 4th ACM Conference on Hypertext (ECHT '92)*, pages 11 – 22, Milan, Italy, November 30 – December 4, 1992.