The Development of Multimedia Hypermedia Applications as Evolutionary, Prototyping-Based Requirements Engineering

by

Karin Sequerra-Breitman*
(sequerra@cos.ufrj.br)

Daniel M. Berry
(dberry@cs.technion.ac.il)

Faculty of Computer Science
Technion
Haifa 32000, Israel

Abstract

It is hypothesized (1) that the problem in building multimedia hypermedia applications is that of understanding what is needed, i.e., of understanding the requirements and (2) that the development of multimedia hypermedia is essentially the engineering of requirements for one particular class of highly dynamic software-based computing systems. Moreover, because of the stability of the libraries for multimedia and hypermedia systems, the development of a prototype for validation of multimedia hypermedia requirements yields a system which is essentially of production quality. The hypothesis is supported by comparing properties of multimedia hypermedia and software systems and development in general and comparing the Hiper Autor method for multimedia hypermedia development with the spiral model for software development. Rigorous testing of the hypothesis is left for future work.

* visiting from COPPE Sistemas, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil

1 Introduction

The history of this paper is instructive about its thesis. Berry has been working in requirements engineering for software-based computing systems [Berry83, Berry95]. Sequerra has been working in the problems of building multimedia hypermedia for educational purposes [Sequerra93, Sequerra94, Sequerra95]. As a result of their joint discussions about their work, it became clear to them that there is a lot of similarity in the processes of devising requirements for general software-based systems and for multimedia hypermedia. In both of these areas, there is a two-way communication problem between the clients, who understand their domain but not multimedia hypermedia or software, and the developers, who understand multimedia hypermedia or software but not any client's domain.

Sequerra had devised a method, Hiper Autor, to develop educational multimedia hypermedia applications. For multimedia hypermedia, the programming libraries for using the various media are well developed, so that once it is understood what to do, the programming of a particular multimedia hypermedium application is quite straightforward, much the same as generating applications in domains for which there are well-developed application generators. An examination of the Hiper Autor method shows a strong resemblance to a spiral model, evolutionary approach to engineer requirements for software-based systems, particularly since so little of the effort of the Hiper Autor method is focused on implementation.

Thus, it is hypothesized that *the* problem in building multimedia hypermedia applications is that of understanding what is needed, i.e., of understanding the requirements. Moreover, because of the stability of the libraries, unlike in some other problem areas, the development of a prototype for validation of multimedia hypermedia requirements yields a system which is essentially of production quality.

The rest of this paper attempts to support this hypothesis by showing

1. how development of multimedia hypermedia, in general, is essentially the engineering of requirements for one particular class of highly dynamic software-based computing systems and

2. how the Hiper Autor multimedia hypermedia development method resembles the spiral model [Boehm88], a requirements engineering process for software-based systems.

2 Multimedia Hypermedia Systems

Multimedia systems can be defined as those which combine in one application, different kinds of media, e.g., text, images, video, sound, etc. A *Hypertext* is a system in which

- the text of a document is divided into named *nodes*, each containing some of the text,

- it is possible to address individual *items* inside nodes, where an item may be an individual character, word, phrase, etc., and

- it is possible to create links, each of which points from an item in a node to another node or to another item in the same or another node.

Hypermedia are a generalization of hypertext in which the contents of nodes are not restricted to being text. *Multimedia hypermedia* are hypermedia systems in which the contents of nodes can be in any media supported by current multimedia system technology.

3 Roles in Multimedia Hypermedia Development

In a multimedia hypermedium, the *domain* is the subject of the contents of the hypermedium. That is, if an educational hypermedium about the art of Michelangelo is being developed, then the domain is Michelangelo and it includes art, history, biography, the topics about which he made art, etc.

As with most software development, various kinds of experts are needed to help specify and construct multimedia hypermedia. These include experts about

1. the domain,

2. multimedia,

3. hypermedia, and

4. software development.

Typically, the client provides the domain experts, and the developers provide all the rest, possibly via subcontractors for the various of the multimedia for which the developing company does not have experts in its employ. For an educational hypermedium in which knowledge from many disciplines will be incorporated, there may be many domain experts, each an expert on one or more subject areas in a very wide domain. As in many requirements gathering situations, the developer-supplied experts may be represented by a single analyst, although in many cases, more than one analyst participates, and some of them may have expertise in the media.

4 Multimedia Hypermedia and Software

This section compares and contrasts properties of multimedia hypermedia and general software-based systems that have an impact on their requirements engineering and development.

For both multimedia hypermedia and software in general, as for other artifacts with artistic aspects, e.g., commercial art, movies and videos, buildings, etc., the client typically has only a vague idea of what he or she wants and does not have a full understanding of what is possible and impossible with the media or technology. The designer is typically an expert in the technology or medium but does not have a full understanding of the client's needs or domain. For both, there is a two-way communication problem.

Moreover, for both, what needs to be discovered by the designer is not entirely in the minds of the client [Goguen94]. There are things that need to be considered that are only in the environment and social situation of the client. There are tacit assumptions that will not be expressed unless someone happens to think of them or a situation using them happens to arise during the requirements gathering and design processes.

In all of this similarity, as observed by Sequerra, there is a key factor that separates software and hypermedia on one hand from all of the others. Commercial art, movies and videos, buildings, etc. do not change once they are published or constructed. However, both software and hypermedia undergo changes by virtue of their use, a use that changes the situation of their application in a way that changes their requirements. In other words, hypermedia have all the earmarks of E-type software systems [Lehman80]. They can get to be very large in order to cover vast amounts of material. Their very use leads to discovery of new connections that have to formalized in the form of links. Their very use leads also to pushing the envelope of the scope of the material as new related topics are discovered. Just adding all these new links and topic nodes begins to degrade the structure of the hypermedia and increases the chance that users get lost in hyperspace and the managers lose track of the contents of the hypermedia in a big messy bowl of spaghetti. It is necessary to undergo systematic evolutionary maintenance in order to add the new links and nodes in a manner that preserves the structure of the hypermedia and sometimes it is necessary to restructure the hypermedia to accommodate the changes in a coherent fashion. If one were to replace "hypermedia" with "software", "nodes" with "procedures", "contents" with "lines of code", and "links" with "invocations and accesses", the above can be read as a description of a typical large evolving E-type software system.

It is well-known that software development is labor intensive rather than technology intensive* [Wegner84]. As a software development exercise, it is the observation of these authors that development of multimedia hypermedia is more client intensive than most other software systems, which tend to be more developer intensive. For a multimedia hypermedium, the client stays with and participates in the development far longer into the lifecycle and in a greater percentage of the lifecycle than for other software, e.g., operating systems, spread sheets, formatters, process controllers, air traffic controllers, etc., in which the developers tend to go off and work by themselves once they believe that they understand the requirements.

Finally, Vincent Quint, in an editorial for an issue of the journal *Electronic Publishing—Origination, Dissemination, and Design* devoted to active documents, observes this same essential similarity between programs and documents [Quint94].

5 Hiper Autor Development Method

The Hiper Autor development method helps the educational multimedia hypermedia development team organize and discipline both its thought and its work. Hiper Autor provides specific tools and supports techniques for each phase of the hypermedium development process. Hiper Autor is composed of two graphical languages and specific documentation forms. Both the languages and the forms are used by multimedia hypermedia programming experts during successive interactions with the domain experts to organize the interaction and

* In 1984, Wegner stated that in its youth, software development is labor intensive, and expressed the hope that it will become more capital intensive as tools are applied and reuse is practiced more universally. Sadly, even though it is now ten years later, software development is still highly labor intensive. In a sense, it is inevitable and expected also that multimedia hypermedia development remain labor intensive, despite the availability of well-developed libraries and development toolkits. After stripping away the difficulties overcome by the use of the libraries and tool kits, one is left with only the problem of understanding the domain of the hypermedium. No amount of technology and reuse will help here, because this understanding requires delving into clients' minds and social situations and charting new territory for which there is nothing yet to reuse!

to document the product.

The Hiper Autor development process follows the spiral lifecycle model, and evolves successive versions of the product in successive sweeps of requirements specification and design. The phases of the method, and thus of each sweep, are initial requirements elicitation, elaboration of the global content scheme, elaboration of the node network, evaluation of the node network, detailed design specification, implementation of the next version, and evaluation of the version. Figure 1 shows the spiral model adapted to multimedia hypermedia development.

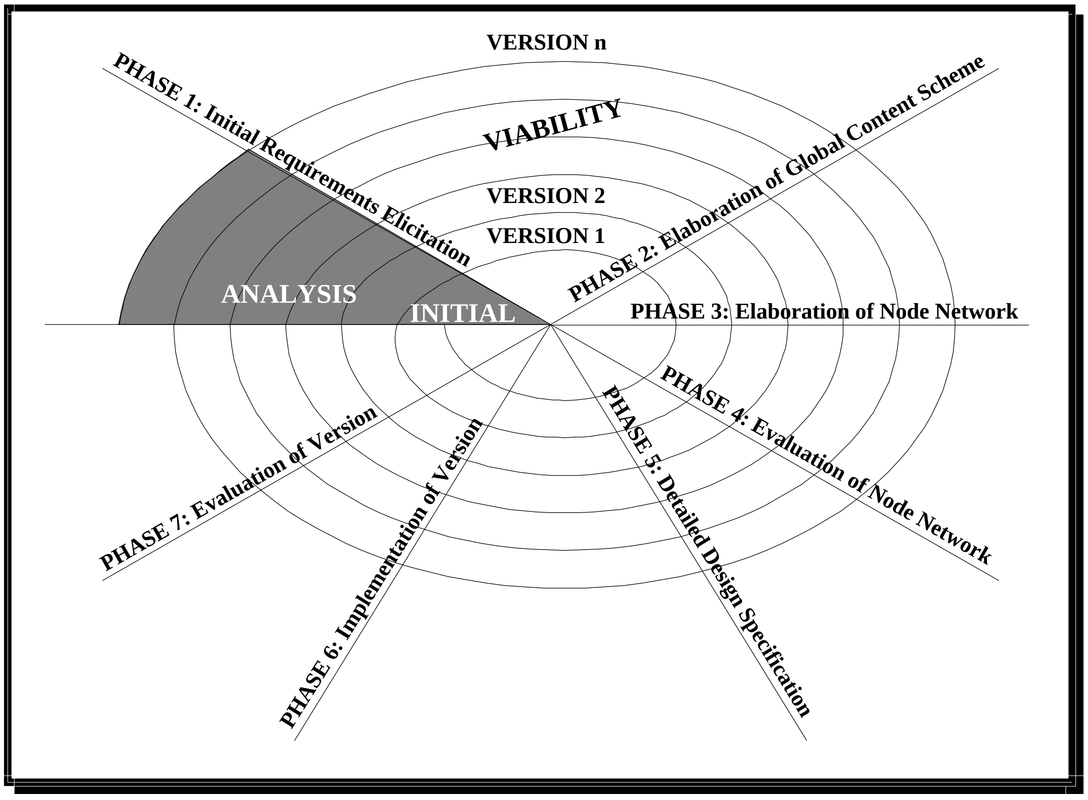

VERSION n
PHASE 1: Initial Requirements Elicitation
VIABILITY
PHASE 2: Elaboration of Global Content Scheme
VERSION 2
VERSION 1
ANALYSIS
INITIAL
PHASE 3: Elaboration of Node Network
PHASE 4: Evaluation of Node Network
PHASE 5: Detailed Design Specification
PHASE 6: Implementation of Version
PHASE 7: Evaluation of Version

Figure 1: Boehm’s Spiral Model Adapted to Hypermedia Development

Each of the phases is described in greater detail in a section of its own.

5.1 Initial Requirements Elicitation

The goals of the initial requirements elicitation phase are to establish the scope of the application’s database and to identify the educational goals. The topics that will be covered in the application must be chosen and the depth to which they will be covered has to be determined. This activity must be carried out by domain experts and at least one analyst. The analyst begins with an elicitation session with the domain experts. Elicitation consists of collecting, validating, and specifying the requirements of the software to be built. If a requirement is considered a necessary condition in order to attain a goal, the first part of requirements specification can be defined as “... [translated from Portuguese] a process in which what must be done is elicited and modeled. This process should deal with various points of view and uses a combination of methods, tools, and actors. The final product of this process is a model from which a document, the requirements specification, can be extracted" [Leite91].

Elicitation encompasses fact collection, validation, and communication; modeling encompasses organization and representation. During elicitation, the *universe* of information to be included in the application is first acquired in order to be modeled in a later phase.

Modeling is responsible for characterizing the results obtained during elicitation. During modeling, *contexts* are defined. Contexts are the partitioning of the database into large groups of related topics. It is considered reasonable to address up to five contexts in a single database, because a greater number would not be easily generated or managed. In the event that a database seems to require more than five contexts, it is suggested to group some of them together into single contexts. In some cases, some contexts may be further partitioned into *sub-contexts*. Both domain experts and analysts are needed in order to determine the best possible consistency-maintaining partitioning of the application database, which also attains the previously determined educational goals. The product of this phase is the *context diagram* (diagrama do contexto) form. The context diagram form consists of two parts; part 1 is the diagram itself in a graphical notation summarized in Figure 2; part 2 is a natural language statement of the objectives of the hypermedium.

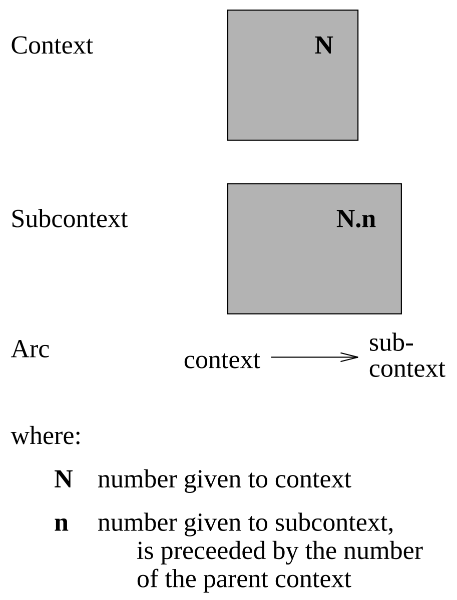
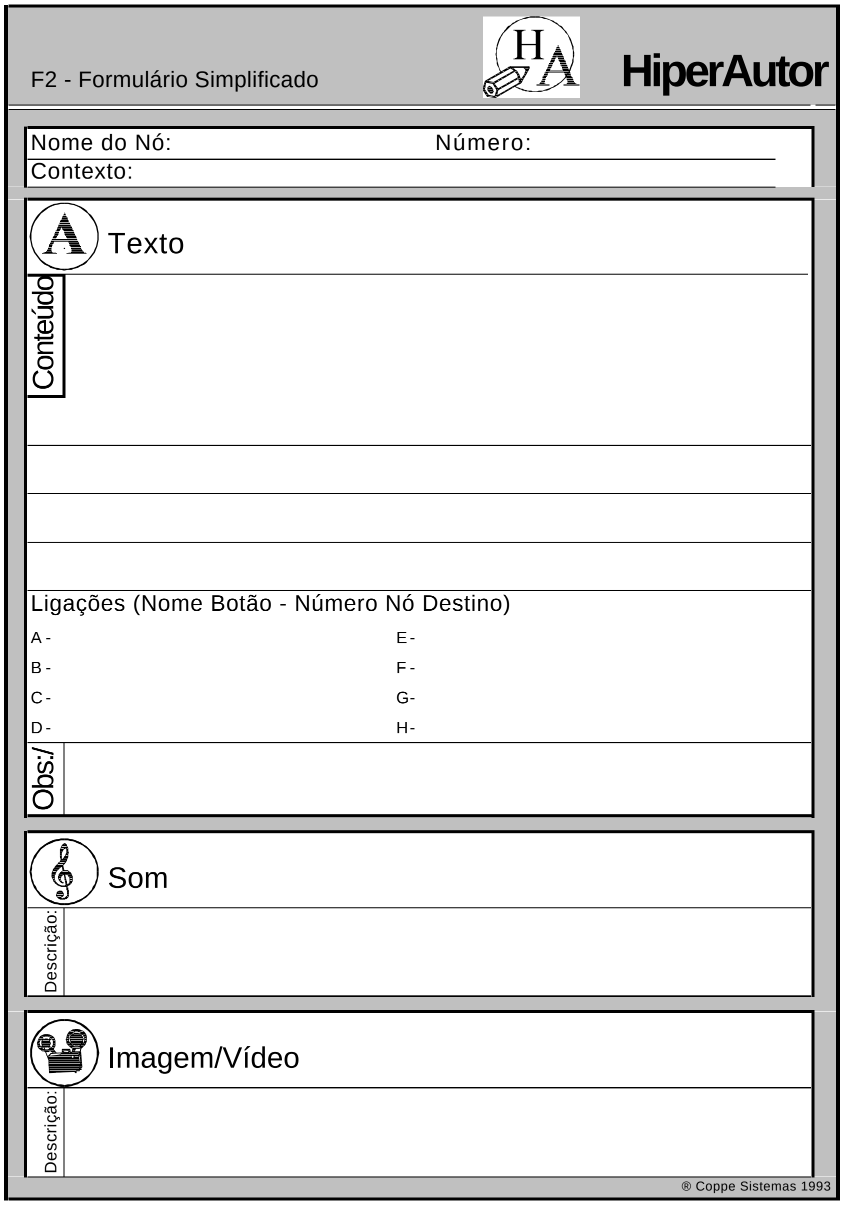

Context
N
Subcontext
N.n
Arc
context
sub-
context
where:
N number given to context
n number given to subcontext,
is preceeded by the number
of the parent context

Figure 2: Graphical Notation for the Context Diagram

5.2 Elaboration of the Global Content Scheme

The goal of this phase is for the analysts and the domain experts to divide the contexts and subcontexts identified during elicitation into *topics*, each of which will be given a *node* in the hypermedium, and then to make a rough draft of the contents of these topic nodes. During this phase, there should be no concern about the format and details of multimedia objects, i.e., images, video, or sound, that will appear in the nodes. The domain experts should offer a general description of what they have in mind and would like to be included. The feasibility of the inclusion of these objects will be determined later.

During this phase, it is not necessary to completely elaborate the contents of the nodes. Instead, it is necessary only to sketch how the chunks of information are going to connected in the hypermedium via links [Gay91] originating at buttons [Berk91]. The analysts meet with the domain experts to identify the links between topics that help promote information accessibility, augment usability, and augment user satisfaction [Glushko89]. The results are documented in the *simplified* form, depicted in Figure 3, one for each node.



F2 - Formulário Simplificado

HiperAutor

Nome do Nó: Número:

Contexto:

Texto

Conteúdo

Ligações (Nome Botão - Número Nó Destino)

A -	E -
B -	F -
C -	G-
D -	H-

Obs:/

Som

Descrição:

Imagem/Vídeo

Descrição:

® Coppe Sistemas 1993

Figure 3: Simplified Form

The rules to fill the simplified form for a node follow.

- Nome/Name: Give a unique and compatible name to the topic at hand in the node.
- Número/Number: Repeat the number of the context or subcontext from which the

node is derived and give it a positive integer extension. This way, each node is guaranteed to have a unique number that can serve as a distinguishing identifier.

- Texto/Text: Give a brief description of the topics to be discussed. The dialogue should be of no concern at this moment.

- Ligações/Links: Determine the links from this node to others, without concern for whether the referred to nodes have already been created. The information given for each link consists of a description, a number, and a destination. The description uniquely identifies the place in the contents of the node, as it is displayed on the screen, where the link's hot spot (button) is located. The number is that of the containing node, extended by a unique number for the link within the node. The destination is denoted by the number of the node that that is referred to by the link; this number is filled in as it becomes known.

- Som/Sound: Give a brief description of the sound (voice or music) that is to be included in the application. It is not necessary at this point to specify the source of the sound or details of its digitization.

- Imagem/Vídeo/Image/Video: Give a brief description of the images to be included in this node. The possible current nonexistence of the image is not a limiting factor. Images yet to be created can, and should, be specified.

- Observação/Observation: This field can be used to capture any suggestion or comments that any participant thinks fit to include.

The set of filled forms provides a first estimate of the number of nodes of the application to be developed, but do not fully specify the contents of any individual node.

5.3 Elaboration of the Node Network

During the previous phase, the database was divided into smaller partitions, called *nodes*, each of which covers a specific topic [Leite91]. Each node also represents the smallest part of a hyperdocument that can be addressed by a link [Berk91].

The set of simplified forms generated in the previous phase constitutes a draft of the concepts to be fleshed out in the future application, separated into small nodes and related to each other by links. Even with this set, it is difficult to visualize the whole database and the links among the nodes. On the other hand, this visualization is essential to permit evaluating whether the organization of the application will allow attaining the goals established for the product. In order to aid visualization, Hiper Autor models the database with the information contained in the forms. The model provides a graphical representation of the nodes contained in the application as well as the links among them.

Based on the previous forms and further interactions with the domain experts, the analysts prepare a node network in which all navigational options are explicit. As the node network diagram is completed, it becomes clear which nodes have options and which do not because it can be seen how many links leave a node. In building this network, it is essential to keep the defined educational goals in mind. In order to document the node network, Hiper Autor provides a form for the *node network diagram* (diagrama de nòs) whose notation is described in

Figure 4.

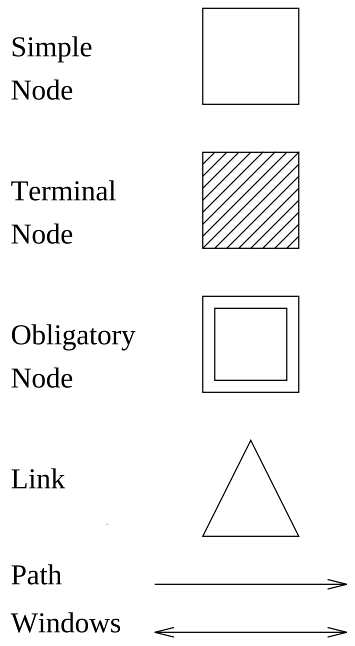


Figure 4: Node Network Diagram Notation

Using the graphical notation, it is possible to represent all the nodes and their links, thus providing a global view of the system. The graphical notation distinguishes three kinds of node, the *simple* node, the *obligatory* node, and the *terminal* node. This classification allows the authors to control movement to and from nodes according to their contents and the educational objectives of the hypermedium. A simple node is one with no restriction on movement to or from the node; it is not even required that the reader visit it. An obligatory node is one which the reader is obliged to visit independently of the chosen path. A terminal node is one from which leaving the system is legal and will not break a stream of thought or interrupt an explanation. Even though the existence of obligatory nodes appears to limit the reader's navigational freedom, precluding some exploration and the serendipity effect, they do guarantee that the application favors attainment of its defined educational objectives. Many authors condemn the creation of networks with nodes linked without any defined criterion. De Young considers arbitrary linking to be dangerous because it is conducive to the construction of "spaghetti type node networks", whose effect is very similar to those of the GOTO statements in programming [DeYoung88].

Currently, the tendency is to employ the metaphor of excursions [Nielsen90], presenting to the user a number of pre-defined paths through the body of information. Using excursions, it can be guaranteed that the user will have orderly access to all essential chunks of information, organized in a clear manner that provides for a good learning environment [Shneiderman91].

5.4 Evaluation of the Design

Once the node diagram is complete, it is necessary to evaluate the design so far. The system analysts, together with the domain experts, evaluate the system as a whole, taking into account all possible paths, the organization of the information, the complexity of each node, and conformance of their classifications to the established educational goals. During this process, one of the analysts is designated to take written notes of the mistakes, incompleteness, and inconsistencies found, together with the modifications desired by the domain experts. Thus, this evaluation can be thought of as an inspection [Fagan76] and would benefit from as much formality as is used in standard software development document inspections. This document, the evaluation report, serves as the basis for revising the system design.

5.5 Detailed Design Specifications

After domain experts have evaluated and approved the node diagram, it is time to begin writing a final draft of the contents of the nodes. Specific attention should be paid to avoid putting an excess of information being put into any one node [Shneiderman91]. According to Wurman, information that has not been digested amounts to no information at all, and it only creates the illusion that the user was capable of accessing it, but in fact he or she did not benefit from it at all. The same holds for information in the database which, if not well organized, will be lost [Wurman89]. In order to minimize the contents of nodes, Ben Shneiderman [Shneiderman91] proposes the following node-writing rules. Let *n* be the node whose contents are being written.

- Instead of discussing a subsidiary topic in *n*, different from the main topic of *n*, one should only allude to the subtopic, designate a sentence that refers to it, and create a link pointing to another node, whose contents discuss the subtopic.

- The same technique should be applied to details. Instead of including the detail in one node, one should make a small reference leading to another node containing the details.

The domain experts should write the final draft of the contents of the nodes, following these rules. This information should be organized in the *detail* form depicted in Figure 5. The analysts should also allow for the possibility that during this process, the domain experts opt for decomposing nodes, resulting in the creation of many new nodes each with less information than the decomposed node. The immediate consequence of this procedure is a revision to the node diagram in order to accommodate the changes.

The rules to fill the detail form follow.

- Nome/Name: The name should be identical to the one used in the simplified form. If the node at hand is a new one, the rules for the naming of nodes given for the filling of the simplified form should be used.

- Número/Number: The number should be identical to the one used in the simplified form. If the node at hand is a new one, the rules for the numbering of nodes given for the filling of the simplified form should be used.

- Conteúdo/Contents: The contents are the final draft of the text to be included in the nodes. The words and expressions that represent buttons should be represented using all caps or some distinctive font or color.

- Alinhamento/Justification: Specify whether the text is left-justified, right-justified, or centered.

- Ligações/Links: Indicate the word, expression, or area that will serve as buttons and the number of the target node.
- Observação/Observation: This field can be used to register any comments relevant to the node.
- Som/Sound: Describe the text or sound to be digitized. Indicate the sound's source and the person who will make the final recording.
- Nome do Arquivo/File name: Give the unique name of the file that stores or will store the representation of the sound.
- Imagem/Vídeo/Image/Video: Describe the image or video to be inserted. In case it already exists in digitized form, specify the source. In case the image or video does not exist, provide the necessary details for its creation.
- Nome do Arquivo/File Name: Give the unique name to the file that stores or will store the representation of the image or video.
- Localização/Location: Give the coordinates of the image or video on the screen if it does not occupy the whole screen.

It is essential that this form is filled taking the node diagram into account because the diagram provides a global view of the nodes in the application.

During the detailed design specification, the software package to be used to implement the application is chosen. For details on the criteria to choose educational hypermedia software packages, refer to Mendonça [Mendonça93].

5.6 Implementation of the First Version

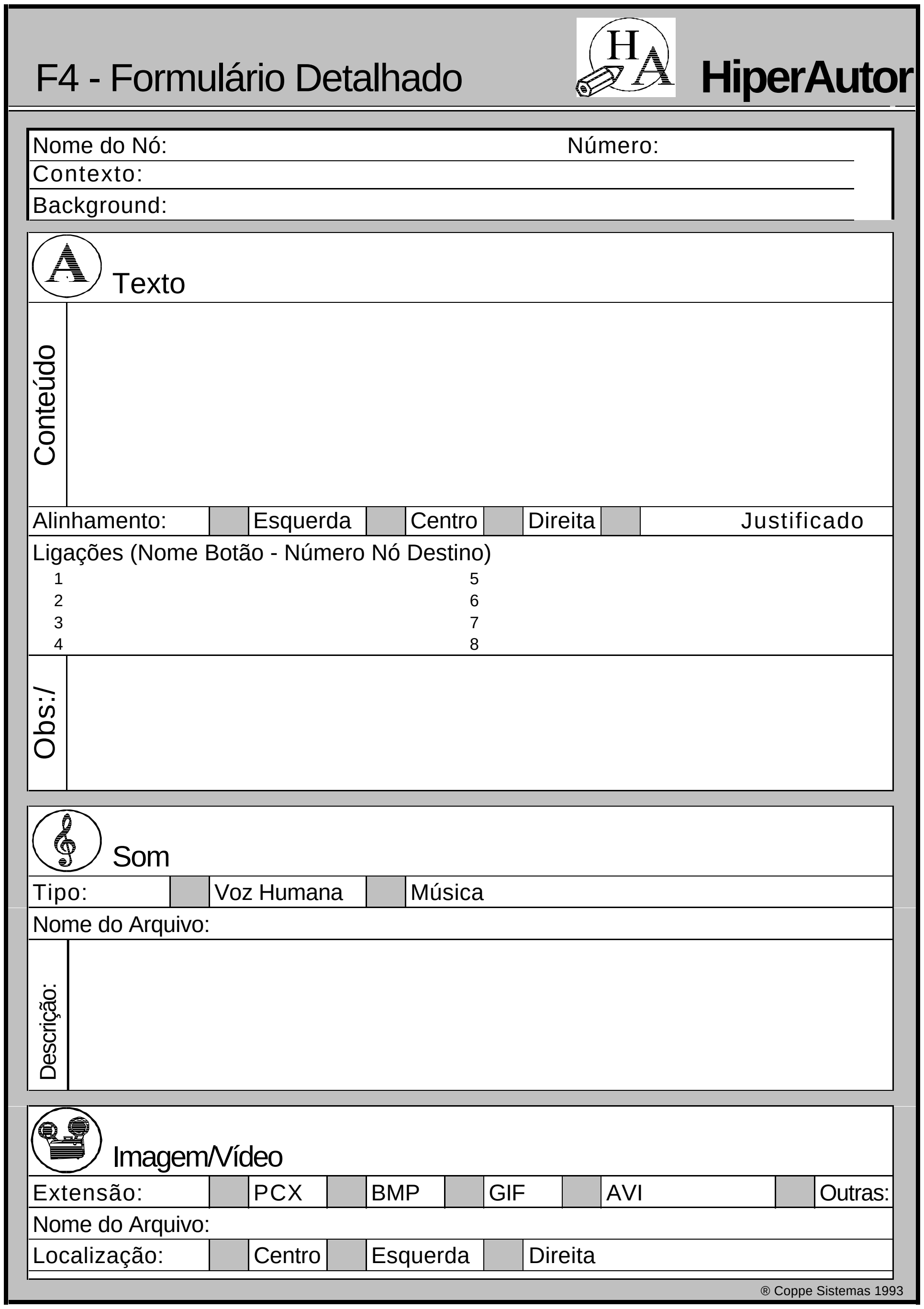
The process of specification and design ends when the revisions of the detailed design specification phase yield mutually consistent detail forms and node network diagram and the implementing hardware and software are chosen.

During the implementation of the first version of the product, the abstract specifications are transformed into an operational product. If the all the previous phases were completed properly, implementation should be mechanical and straightforward [Pressman92].

We believe that the documentation generated by using Hiper Autor is sufficient to encourage the implementation quality and, therefore, of the final product. It should be noted, though, that the ideal situation is to have an implementation team familiar with both the hardware and software to be used in the implementation.

5.7 Evaluation of Implementation

The developed product is to be evaluated according to the quality attributes proposed by Mendonça [Mendonça93] and extended for hypermedia systems by Campos [Campos92].



F4 - Formulário Detalhado HiperAutor

Nome do Nó:		Número:
Contexto:		
Background:		

Texto

Conteúdo

Alinhamento:		Esquerda		Centro		Direita		Justificado

Ligações (Nome Botão - Número Nó Destino)

1	5
2	6
3	7
4	8

Obs:/

Som

Tipo:		Voz Humana		Música

Nome do Arquivo:

Descrição:

Imagem/Vídeo

Extensão:		PCX		BMP		GIF		AVI		Outras:

Nome do Arquivo:

Localização:		Centro		Esquerda		Direita

® Coppe Sistemas 1993

Figure 5: Detail Form

5.8 New Versions

Unless the application is a total disaster, the inexorable pressures [Lehman80] for maintenance will set in. Maintenance is the set of activities performed to the software after it is handed to its user [Ghezzi91]. Basically, there are two kinds of maintenance, corrective and evolutionary. Corrective maintenance is modifying the system to fix observed problems, i.e., errors, while evolution is the addition of new nodes to the application. As with other software, maintaining the consistency of the design documents with the software is essential to permit future maintenance. To insure the faithfulness of the design documents, it is suggested that maintenance be carried out by first modifying the forms and graphs and then following through on the method described above, so that it appears that the new implementation were derived from the new forms, rather than having been obtained by modifying the old implementation.

The literature says that the maintenance phase consumes the largest share of software development. Most of its changes are due to mistakes in the requirements specifications [Pressman92, Ghezzi91]. We believe that the use of Hiper Autor may contribute to reducing the maintenance costs.

5.9 Completed Description

This completes the description of the Hiper Autor method. It is now time to turn back to considering the main hypothesis that is the subject of this paper.

6 Support for Hypothesis

Examination of both the general process of multimedia hypermedia construction and the specific Hiper Autor method shows clearly that most of the development of a multimedia hypermedium is focused on dealing with client-or-domain-expert-supplied information and with validating with the client and domain experts the conclusions drawn by the analysts. Consequently, it is fair to say that multimedia hypermedia development is a grand exercise in requirements elicitation, analysis, specification, and validation, i.e., of requirements engineering and that the hypothesis is supported.

One preliminary case study, such as the contents of this paper, is not enough to validate *any* hypothesis [Potts93]. However, enough has been described to make it possible for us and others to conduct more complete and rigorous studies in the future. Moreover, we and others can be on the lookout for counter-examples and variations.

There was another hypothesis stated in passing, which one of the anonymous referees identified as possibly worth examining in detail on its own, namely that libraries for using the various media in multimedia systems are well-developed and stable enough that implementation of multimedia systems are indeed tantamount to application generation. Further examination of this issue is also left for future work.

7 Conclusion

This paper has contended that development of multimedia hypermedia strongly resembles requirements engineering of general software-based systems. It has tried to demonstrate this contention by comparing properties of multimedia hypermedia systems and more general software-based systems and has shown that a specific multimedia hypermedia development method is essentially a spiral model of successive mainly requirements engineering efforts yielding a series of ever increasingly more complete client-validated prototypes. It has also recognized that more studies are needed to properly validate this contention.

What can be derived from this conclusion? It should be that any successful requirements engineering method, such as contextual inquiry [Holtzblatt93] can be applied to the design of multimedia hypermedia. Moreover, it should also be that the process of developing any software system can be reduced to focusing nearly only on requirements if one is able to case and develop the application as a multimedia hypermedium. In other words, reduce the implementation problem to one of thoroughly exercising a well-developed program library or an application generator.

Acknowledgments

The authors thank the anonymous referees for their comments and new ideas.

References

[Berk91] Berk, E., “A Hypertext Glossary,” in *Hypertext and Hypermedia Handbook,* ed. E. Berk, McGraw-Hill, New York, NY (1991).

[Berry83] Berry, D.M. and Berry, O., “The Programmer-Client Interaction in Arriving at Program Specifications: Guidelines and Linguistic Requirements,” in *Proceedings of IFIP TC2 Working Conference on System Description Methodologies*, ed. E. Knuth, Kecskemet, Hungary (1983).

[Berry95] Berry, D.M., “The Importance of Ignorance in Requirements Engineering,” *Journal of Systems and Software* **28**(2), pp.179-184 (1995).

[Boehm88] Boehm, B.W., “A Spiral Model of Software Development and Enhancement,” *IEEE Computer* **21**(5), p.61–72 (1988).

[Campos92] Campos, F.C.A. *et al*, “Autoria e Trabalho Cooperativo: Reflexões sobre um Experiência em Hipertexto,” in *Workshop em Trabalho Cooperativo, Pensamento Crítico e Software Educacional*, COPPE/UFRJ, Rio de Janeiro, Brasil (1992).

[DeYoung88] DeYoung, L., “Linking Considered Harmful,” in *Hypertexts: Concepts, Systems, and Applications*, ed. A. Ritz, N. Streitz, and J. André, Cambridge University Press (1988).

[Fagan76] Fagan, M.E., “Design and Code Inspections to Reduce Errors in Program Development,” *IBM Systems Journal* **15**(3), p.182–211 (1976).

[Gay91] Gay, G., “Structuring Interactive Multimedia Fiction,” in *Hypertext and Hypermedia Handbook*, ed. E. Berk, McGraw-Hill, New York, NY (1991).

[Ghezzi91] Ghezzi, C., Jazayeri, M., and Mandrioli, D., *Fundamentals of Software Engineering*, Prentice Hall, Englewood-Cliffs, NJ (1991).

[Glushko89] Glushko, R., *Hypertext ’89 Course Notes #3: Turning Text into Hypertext*, ACM, Inc. (1989).

[Goguen94] Goguen, J.A., "Requirements Engineering as the Reconciliation of Technical and Social Issues," pp. 165–199 in *Requirements Engineering: Social and Technical Issues*, ed. J.A. Goguen and M. Jirotka, Academic Press (1994).

[Holtzblatt93] Holtzblatt, K. and Jones, S., "Contextual Inquiry: A Participatory Technique for System Design," in *Participatory Design: Principles and Practice,* ed. A. Namioka and D. Schuler, Erlbaum, Hillsdale, NJ (1993).

[Lehman80] Lehman, M.M., "Programs, Life Cycles, and Laws of Software Evolution," *Proceedings of the IEEE* **68**(9), p.1060–1076 (1980).

[Leite91] Leite, J.C.P., "Validação de Requisitos: O Uso de Pontos de Vista," *Revista Brasileiro de Compuatação* **6**(2), p.39–52 (1991).

[Mendonça93] Mendonça, L.F. and Rocha, A.R., "Critérios de Qualidade para Avaliação de Sistemas de Hipertexto," in *Encontro França Brasil de Informática na Educação,* Rio de Janeiro, Brasil (1993).

[Nielsen90] Nielsen, J., *Hypertext and Hypermedia,* Academic Press, New York, NY (1990).

[Potts93] Potts, C., "Software Engineering Research Revisited," *IEEE Software* **10**(5), p.19–28 (1993).

[Pressman92] Pressman, R.S., *Software Engineering: A Practitioner's Approach,* McGraw-Hill, New York, NY (1992).

[Quint94] Quint, V., "Editorial," *Electronic Publishing—Origination, Dissemination, and Design* **7**(2), p.53–54 (1994).

[Sequerra93] Sequerra-Breitman, K., "Hiper Autor: a Method for Specification and Design of Hypermedia Systems," M.Sc. Thesis, COPPE/Federal University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil (1993).

[Sequerra94] Sequerra-Breitman, K., "Authoring Considerations in the Imaginary Museum Project: The Sistine Chapel — A Case Study," in *Proceedings of the ED Media 94*, Vancouver, BC, Canada (1994).

[Sequerra95] Sequerra-Breitman, K., "Using Hypermedia as a Interdisciplinary Learning and Exploration Tool: a Case Study of the Development of Solar," Technical Report, Faculty of Computer Science, Technion, Haifa, Israel (1995).

[Shneiderman91] Shneiderman, B. *et al*, "Editing to Structure a Reader's Experience," in *Hypertext and Hypermedia Handbook*, ed. E. Berk, McGraw-Hill, New York, NY (1991).

[Wegner84] Wegner, P., “Capital-Intensive Software Technology,” *IEEE Software* **1**(3), p.7–45 (1984).

[Wurman89] Wurman, R.S., *Information Anxiety*, Doubleday, New York, NY (1989).