A Comparative Study of Methods for Hypermedia Development*

Nora Koch[1]

Ludwig-Maximilians-Universität München
Institute of Computer Science
Oettingenstr. 67
D-80538 München
Tel. +49 89 2178 2151
Fax +49 89 2178 2152
kochn@informatik.uni-muenchen.de

Abstract

This paper presents a comparative study of the most relevant methodologies for hypermedia and Web development published in the last few years. Most of these methods focus on the design of hypermedia applications; only a few cover more aspects of the life cycle, such as requirements capture, implementation and/or testing. One common characteristic is the separation of the domain analysis from the specification of the navigation space structure as well as from the design of the user interface. A brief description of each of these methodologies is given as well as a set of comparisons. We compare the main concepts, the phases o steps of the model, CASE Tool support as well as the modelling technique, notation and graphical representation. A list of the most relevant characteristics of the Unified Process is given as well comparing it to hypermedia development methodologies.

Keywords

Hypermedia Development Methods, Hypermedia Systems, Hypermedia Design, Web applications, Unified Process.

1 Introduction

Hypermedia development is a new and still evolving discipline. We are at the beginning of a large process of learning how to develop large hypermedia applications. Hypermedia applications for the Web or CD-ROM are mostly developed ad hoc, evolving usually from small to large applications and becoming very soon unmaintainable. Some guidelines or tools are beginning to appear assisting the hypermedia developer. But these current practices fail when used to develop large-scale applications for the same reasons as such a development fails in other areas of software development. These reasons are lack of planning and inappropriate techniques, processes and methodologies.

The development of hypermedia systems differs from the developing process of traditional software in several dimensions. People with very different skills are involved in the process, such as authors, layout designers, programmers, multimedia experts, marketing specialists. The role of the users is augmented and makes it more difficult to capture the structure of the domain. The non-linearity of the hyperdocuments as well as the possibility to connect easily to other hypermedia applications increments the complexity and risk of "lost in the hyperspace".

---

* Technical Report 9905, Ludwig-Maximilians-Universität München, November 1999.

[1] also working at FAST e. V. (Research Institute for Applied Software Technology), Arabellastr. 17, D-81925 München, Germany, koch@fast.de.

Hypermedia development has to take into account aesthetic and cognitive aspects as well, that traditional software engineering environments do not support (Nanard and Nanard, 1995). It tends to be more fine grained, the process more incremental and iterative, and the maintenance phase plays a more important role in the life-cycle of hypermedia applications than in the life-cycle of other systems.

Hypermedia applications that are the results of ad hoc implementations tend to be confusing for the users and difficult to maintain for the authors. Thus, engineering of hypermedia applications is a critical factor playing an important role in maintenance of future hypermedia systems.

In the last few years many development methods have been defined. They have similarities and differences. The purpose of this paper is to compare them, finding out what are the similarities and what the differences. These methods for the development of hypermedia systems propose a different number of steps and activities, some of them focus only on the design; others focus on the complete development of hypermedia applications. They prescribe different techniques and/or notations to be used in the development process. Tools have been implemented that support the development process that some of these methods propose.

This work is structured as follows. Section 2 briefly describes eleven different methodologies. Section 3 outlines the current notations used in hypermedia design. Section 4 presents a comparison of the hypermedia development methods based on their process steps, concepts, notations, techniques and tools. Section 5 compares the Unified Process with some of these hypermedia development methods. Section 6 gives some conclusions.

2 Methods for Hypermedia Systems Development

To analyse and compare methods for the development of hypermedia systems, we need first of all, a more precise definition of the notion of systems development method (methodology). It is often very vaguely defined, not only because of the ambiguity of the concept of *method* and *methodology*[2], but also because of the difficult to give a precise definition of *system* and *system development* (Iivari and Maansaari, 1998).

Avison and Fitzgerald (1995) define *method for system development* as “a set of phases which guide the developers in their choice of techniques that might be appropriate at each stage of the project”. These techniques also have to help them to plan, manage, control and evaluate information systems projects. Palvia and Nosek (1993) instead give the following definition: A methodology is a "an organised and systematic approach to systems life cycle or its parts. It will specify the individual tasks and their sequences". Another problem is the distinction between method and technique. Pavia and Nosek (1993) defines technique as accomplishing a task in the systems life cycle. The result of applying a technique are certain outcomes (deliverables).

One scope problem is to determine which aspects have to be covered by a methodology. Rumbaugh (1995) proposes that a method should include four components:

- a set of modelling concepts to capture semantic knowledge about the problem and its solution,
- a set of views and notations for presenting the underlying modelling information,
- a step-by-step iterative process for constructing models and implementations of them,
- a collection of hints and rules of thumb for performing development.

[2] Method and methodology are used as synonyms in this work.

Henderson-Sellers (1995), in contrast, suggests a more extensive list of aspects that have to be covered by a methodology. Tthere are the following nine constituents:

- a full life-cycle process,
- a full set of concepts and models that are internally self-consistent,
- a collection of rules and guidelines,
- a full description of deliverables,
- a workable notation,
- a set of metrics, together with advice on quality, standards and test strategies,
- guidelines for project management,
- advice for library management and reuse, and
- identification of organisational rules.

Most of the major methodologies developed for hypermedia systems only partially cover the life cycle of hypermedia systems focusing on the design of these systems - according Rumbaugh's definition. Only HFPM (Olsina, 1998) and the engineering approach of Lowe and Hall (1999) cover the whole development process following Henderson-Sellers proposal.

Basically, two techniques are applied, if any, in hypermedia design: entity-relationship and object-oriented techniques. HDM and RMM are based on the E-R model. In contrast, EORM, OOHDM, SOHDM and WSDM adopt object-oriented approaches. Other methods mainly focus on the analysis step, like RNA or go beyond the design and implementation, like HFPM, describing the process covering the whole life cycle of hypermedia applications. In addition to analysis, design and implementation, HFPM includes project management as well as feasibility studies, deployment, maintenance and/or quality control. HFPM and the OO/Pattern Approach suggest the use of pattern for the navigational and user interface design. The hypermedia engineering model presented by Lowe and Hall (it is not named) goes further, proposing the creation of a reference model for hypermedia development processes.

In the remainder of this Section we give a brief description of the methods mentioned above.

2.1 HDM: Hypermedia Design Method

The Hypermedia Design Model (HDM) is one of the first methods that has been developed to define the structure and interaction in hypermedia applications (Garzotto, Paolini and Schwabe, 1993). It is based on the E-R methodology, but extends the concept of entity and introduces new primitives as units (corresponding to "nodes") and links. HDM entities have an inner structure and have associated a browsing semantics to them, i.e. a specification of how navigation may be performed and how information is visualised. An entity is a hierarchy of components and components are made of units. Three types of links are defined: *structural, perspective* and *application links*. Structural links connect components; perspective links units. These links can be automatically derived from the structure of the entities. Application links are defined by the author connecting components and entities of the same or different type.

Two different groups of entities exit in HDM: the application entities described above and the so called "*outlines*" that allow the access to the application entities offering entry points to start navigation and possibility to locate and select entities. These outlines or access structures are ordered trees of components. The application entities constitute the hyperbase.

Garzotto, Mainetti and Paollini (1995) distinguish the following dimensions in the analysis of hypermedia applications: content, structure, presentation, dynamics and interaction. The content addresses the pieces of information while the structure is the content's organisation. The presentation

defines how the application content and functions are shown to the users. The interaction for HDM is the dynamic functionality operated on presentational elements. In other methods interaction is considered as part of the dynamics and presentation as it is a combinations of both factors. The outlines defined by HDM are *index links, guided tours* and *collection links*. Index links connect the collection node to each member of the collection. A guided tour link connects the collection's nodes in a linear sequence with each member connected to the next and previous one. In circular collections the last member connects to the first. Collection links are index or guided tour links that allow for traversing the nodes of a collection.

To support the presentational design HDM defines two concepts: *slot* and *frame.* A slot is an atomic piece of information. They can be simple or complex, such as a video synchronised with sound. Slots are composed in frames. A frame is a presentation unit, i.e. what is shown to the user.

This methodology distinguishes between authoring-in-the-large and authoring-in-the-small. The first one identifies the entities, components and units while the second one fills these units with content. HDM specifies the structure of the hyperbase, as authoring-in-the-small is not within its scope.

2.2 RMM: Relationship Management Methodology

The Relationship Management Methodology (RMM) addresses the design and construction of hypermedia applications defining for this purpose a process of seven steps (Isakowitz, Stohr and Balasubramanian, 1995). These steps are: *entity-relationship design, slice design, navigational design, user interface design, protocol conversion design, run-time behaviour,* and *construction and testing*. An application design is represented with RMDM (Relationship Management Data Model) based on the entity-relationship model and HDM. RMM is at the same time a top down and a bottom up approach.

During the E-R design step entities, attributes and relationships are identified which will become nodes and links in the resulting hypermedia application. The second step, slice design, involves grouping entity attributes for presentation. *Slices* are "presentation units" which appear as pages of hypermedia applications. Separation of contents and presentational aspects is not fulfilled in this step. The navigational design is the step for the identification of the navigation paths. RMM specifies navigation through *access primitives*: *link, grouping* (menus), *index, guided tour* and *indexed guided tour*. The conversion protocol design converts components of RMM into physical objects in the target hypermedia application. The step user interface design involves the design of the screen layouts of every diagram element including access primitives, links, anchors, indexes and general navigational aids. The techniques proposed for the user interface design by Balasubramanian, Bieber and Isakowitz (1996) are construction of mock-ups and prototyping.

A Relationship Management Case Tool - RMCase - has been designed (Diaz, Isakowitz, Maiorana and Gilabert, 1995) to support the development of hypermedia WWW applications. It supports the RMM methodological stages via development of contexts, one for each stage. Design objects are shared among different contexts. The transition between contexts is possible through navigation.

2.3 EORM: Enhanced Object Relationship Methodology

The Enhanced Object-Relationship Model (EORM) is defined as an iterative process concentrating on the enrichment of the object-oriented modeling by the representation of relations between objects (links) as objects. According to Lange (1996), this has the following advantages: relations become semantically rich as they are extensible constructs, they can participate in other relations and they can be part of reusable libraries. This method proposes the construction of a prototype of the user interface at an early stage during design.

The method is based on three *frameworks*: *class*, *composition* and *GUI*. The class framework consists of a reusable library of class definitions. To identify classes for an application EORM follows standard object-oriented techniques. EORM distinguishes two types of object-oriented relationships: *generalisation relationships* and *user-defined relationships*. Whereas the first ones have predefined semantics associated to them; the second ones rely completely on the user specification.

The composition framework consists of a reusable library of link class definitions that enable users to reuse already developed link classes and extend them when necessary by using inheritance. The semantics of the basic link classes is the following:

- *simpleLink:* is the root link class that provides basic interlinking capabilities, including functions for creation, deletion and traversing.
- *navigationalLink:* provides traversal mechanisms for hypermedia links, including storage of creation time and history information (backtrack). It inherits from simpleLink.
- *nodeToNode:* is a link that inherits from navigationalLink providing an object-to-object hypermedia link functionality.
- *spanToNode:* inherits from navigationalLink. It links the content of an object to another object.
- *structureLink:* is a child of simpleLink and the root of the structural links. It is inserted after creation into the structural context.
- *setLink:* is a structureLink that provides access to an object in an unordered collection of objects.
- *listLink:* is a structureLink that provides access to an object in an ordered collection of objects.

The last step of this method is the design of the GUI application using elements of the GUI Framework. It proposes to determine the windows of the domain and which presentation has to appear in each window, to obtain presentations from attributes and operations of classes and determine how functionality is assigned to window menus.

The ONTOS Studio tool was developed to support the hypermodelling process with EORM. It utilises an interactive graphical user interface that generates a C++ implementation of the hypermodel. It is based on the ONTOS database.

2.4 OOHDM: Object-Oriented Hypermedia Design Method

The Object-Oriented Design Method (OOHDM) developed by Rossi and Schwabe (1996, 1998) comprises the following four activities:

- *conceptual modeling*,
- *navigational design*,
- *abstract interface design*, and
- *implementation*.

These activities are performed in a mix of incremental, iterative and prototype-based development style. Object-oriented models are built in each step improving the models designed in previous iterations.

The *conceptual model* of the application is represented with a class diagram. This method sees an application as a view over the conceptual model. Classes of these views are called navigational classes. The concept of *navigational context* is introduced to describe the navigational structure. It is a powerful concept that allows different groupings of navigational objects for the purpose of navigating

them in different contexts. The access to these navigational elements is modelled with *access structures*, such as *indexes* and *guided tours*. Different types of indexes and navigational contexts are defined in the navigational design. A special notation is used for the representation of the navigational context schema. In addition, InContext classes are defined to enrich navigational objects allowing them to look different, present different attributes (including anchors) as well as different behaviour (methods) when navigated within a particular context.

The abstract interface model is the result of the specification of the interface objects the user will perceive. OOHDM uses *Abstract Data Views* (ADVs) to model the static aspects of the user interface (Carneiro, Cowan and Lucena, 1993) while dynamic aspects of the user interface are modelled with a technique based on Statecharts (Harel, 1987).

The OOHDM-Web environment allows for a rapid prototyping of hypermedia applications designed using OOHDM. It requires the conversion of navigation constructs defined by the user, such as navigation classes, navigational contexts, classes for presentations (*InContext* classes) and access structures into tables as it is not an object-oriented approach. Navigation and interface constructs of OOHDM are mapped into a library of functions in the cgi-Lua script language using therefore the Lua-Database. Based on the table it generates pages dynamically. The designer builds page templates with a mix of HTML tags and special commands that are interpreted by the cgi-Lua scripting environment with a set of special functions derived from OOHDM navigation primitives.

2.5 SOHDM: Scenario-based Object-oriented Hypermedia Design Methodology

Another method recently developed by Lee, Lee and Yoo (1998) is the Scenario-based Object-oriented Hypermedia Design Methodology. It consists of six phases*: domain analysis, object modeling, view design, navigation design, implementation design* and *construction*. This methodology has similarities with RMM, OOHDM and EORM. It differs in the use of scenarios, which are described through scenario activity charts that are based on events, activities and activity flows primitives. *Scenarios* are defined in the domain analysis phase and are used for the object modeling. View design consists in determine object-oriented views generated from single object classes or from associations between object classes.

The navigation design uses scenarios to determine access structure nodes. They are defined as menu-like mechanism that enables users to access to other parts of hypermedia documents. The access structures nodes (ASN) together with the object-oriented views are called navigation units. The ASNs are similar to the access primitives of RMM: *grouping* (menu), *index* and *guided tour*. Object-oriented views are categorised into three types: *base view*, *association view* and *collaboration view*. These views are similar to contexts defined in OOHDM. A base view is generated from a single object class. An association view is extracted from an association relationship. Similarly, a collaboration view is generated from a collaboration relationship. The identification of navigation links complete the navigational design.

During the implementation design the user interface, pages and a logical database schema are modelled. This method presents a clear sequence of steps that benefits of the scenarios obtained as results of the analysis phase. It defines its own elaborated graphical notation.

2.6 WSDM: Web Site Design Method

The Web Site Design Method (WSDM) is a user-centred approach that defines the information objects based on the information requirements of the users of a Web application. WSDM proposed by De Troyer and Leune (1997) consists of three main phases: *user modeling, conceptual design* and *implementation design*.

In the user modeling phase the potential users/visitors of the Web site are identified and classified according to their interests and navigation preferences, for example. Starting point is the description of the domain taking into account the user activities. Different perspectives are defined for the user classes; these are different ways classes of users look at the same information and navigate through the information. Conceptual design consists of two steps: object modeling and navigational design. Object modeling is then done in three steps: business object modeling, user object modeling and perspective object modeling.

The navigational model consists of a number of navigation tracks, one for each perspective expressing how users of a particular perspective can navigate through the available information. WSDM describes it in terms of components and links. It distinguishes three types of *components*: *navigation, information* and *external.* Each *navigation track* consists of three layers: context, navigation and information layers. The context layer is the top level of the navigation track starting with a navigation component. The information layer is the bottom level of the navigation track. The navigation layer connects the context layer and the information layer. To access the information intermediate components and links are created, such as indexes.

This kind of navigational design achieves Web applications that have a very hierarchical structure. The implementation design step achieves to create a consistent and efficient look and feel for the conceptual model. Few recommendations are given in this step, such as the use of index pages, information divided into right-sized chunks, use of context and information cues and use of navigational cues.

2.7 RNA: Relationship-Navigational Analysis

The Relationship-Navigational Analysis (RNA) defines a sequence of steps to be used for the development of Web applications, focusing on the analysis (Bieber, Galnares and Lu, 1998). It is specially useful for those Web applications created on the basis of legacy systems. It consists of the following five steps:

- *stakeholder analysis*, which purpose is to identify the application's audience.
- *element analysis.* In this step all elements of interest in the application are listed. This list includes all types of items, documents, forms, mock-ups, etc.
- *relationship and metaknowledge analysis,* which achieves to identify a schema, a process, an operation as well as relationships of different kinds, such as descriptive, parametric, statistical, etc.
- *navigation analysis*. In addition to the relationships identified in the previous steps the access and navigation structures will be included in this step.
- *relationship and metaknowledge implementation analysis*. The developer must decide which of the relationships identified in the third step will be implemented here.

RNA only gives some guidelines of the actions to be performed in each step. Neither modelling concepts nor a notation is proposed. Complementary to RNA the DhymE tool was developed to generate automatically links for legacy applications (non-hypermedia applications).

2.8 MacWeb Approach

The MacWeb Hypermedia Development Environment is an engineering environment developed by Nanard and Nanard (1995). They emphasise the importance of the creative aspects in the hypermedia development process: "An important part of hypertext design concerns aesthetic and cognitive aspects that software engineering environments do not support". Therefore, the design activity is performed in a two-dimensional space of *methods steps* and *mental processes*.

The *mental process* includes the steps*: generating material*, *organising and structuring*, *reorganising and updating*, as well as *evaluation*. The *methods steps* are similar to steps in other methods described in this chapter: concepts elicitation, navigation model, abstract interface, implementation model and testing. The designer switches from mental process to methods steps as there are not predefined transition rules between the activities or steps. The designer's strategy may be bottom-up and/or top-down. They assert that this chaotic process has not to be impaired or the author's creativity would be reduced drastically.

MacWeb´s design environment proposes to use object-oriented techniques, such as generalisation and instantiation as well as the well-known technique *light prototyping* used in human-computer interaction (HCI) development. Through prototyping users are enabled to evaluate interface design and hypermedia structure.

MacWeb´s basic hypertext model relies on typed nodes connected by bi-directional typed links. MacWeb has a built-in tool that helps users construct a structure as a semantic network. It provides a few primitive built-in types, such as node, link, script node, group node. Node represents a concept. A link denotes a relationship between two types (nodes). Firing a link to a script node will trigger the execution of its content. A node of type group comprises sets of nodes organised as sub-webs.

2.9 HFPM: Hypermedia Flexible Process Modeling

The Hypermedia Flexible Process Modeling (HFPM) presented by Olsina (1998) is an engineering-based approach, that includes analysis-oriented *descriptive* and *prescriptive* process modeling strategies. It describes existing processes thereby giving guidelines for the planning and managing of a hypermedia project.

HFPM embraces *functional, methodological, informational, behavioural* and *organisational vi*ews or perspectives of the hypermedia development process.

- The *functional view* prescribes a set of phases and activities to perform tasks (e.g. to find users, classes, uses cases, etc.).
- The *methodological view* defines a set of specific process constructors to be applied to the different tasks (e.g. use-case-driven analysis, OOHDM-based conceptual and navigational design, etc). One or more methods are selected to support the tasks of a specific process and one or more tools can support a specific method.
- The *informational view* plans to produce a set of artifacts (i.e. results such as navigational model or physical model) which are required by the tasks.
- The *behavioural view* represents the dynamic of the process model making decisions about sequencing and synchronisation of tasks, iterations, increments, parallelisms, termination conditions, feedback loops, etc.
- The *organisational view* defines aspects such as roles, team organisation, communication mechanisms, groups dynamic, etc.

The list of tasks and suggested subtasks prescribed by HFPM for the development of hypermedia applications is listed below. It includes the following technical, managerial, cognitive and participatory tasks:

- *software requirement modeling*, such as initial survey, use-case modeling, non-functional modeling, glossary construction;
- *project planning*, i.e. analysis and specification of the project plan;
- *conceptual modeling*, that consists of analysis and specification of the problem domain;
- *navigational modeling*, that comprises analysis of intended user's tasks, identification of

navigational classes, specification of navigational schema, specification of navigational transformations;

- *abstract interface modeling,* that refers to analysis of user interface models, specification of interface perceptible objects, events and transformations;
- *design patterns employment*, i.e. use of navigational, architectural and user interface patterns;
- *multimedia data capturing and editing;*
- *physical modeling/integration,* that comprise component employment, rapid-functional prototyping, evolutionary prototyping, sketching or storyboarding, and component integration;
- *validation/verification;*
- *cognitive criteria employment*, such as coherence and orientation criteria employment;
- *quality assurance*, like analysis of quality and improvement strategies, specification of quality plan;
- *project co-ordination and management*, i.e. control and management of process, artifacts and resources; and
- *documentation.*

This is a wide engineering-based approach. It covers all essential phases and activities of a hypermedia project, helps to establish milestones and metrics as well as promoting communication and human understanding. Olsina (1998) presents a conceptual model for HFPM based on a set of key-concepts, such as process itself, task, artifact, resource, agent, role, process constructor, process description, goal, resource and operation.

2.10 OO/Pattern Approach

The OO/Pattern approach to hypermedia collection design (Thomson, Greer & Cooke, 1998) is similar to HFPM as it proposes to utilise both, an OO-design and the application of patterns for the navigational and presentational design. It differs from HFPM because it does not cover all the life cycle of a hypermedia application, i.e. project management, testing and maintenance aspects are not included. The use of patterns has well-known advantages, such as that the process is well defined, documentation can be reused and maintenance is easy.

This method prescribes the following steps: *use cases, conceptual design, collaboration design, class definition, navigational design* and *implementation*. The innovative aspects of this methods in relation to other methodologies are:

- *Use case analysis* for the different types of users.
- *Collaboration design* based on the defined use cases and the conceptual design.
- *Navigational design* based on patterns.

They describe the layered hierarchy pattern which purpose is to create an intuitive navigation in a naturally hierarchical structured information.

2.11 Lowe-Hall's Engineering Approach

Lowe and Hall (1999) provide a framework for the development process of hypermedia applications. The framework includes *domain analysis, product modelling, process modelling, project modelling, development* and *documentation*. Product modelling consists in choosing a model for the final product. Three different product models are supported by the framework: programming language-based, model-based and information-centred approach.

- In the *programming language-based model* the information and the information structure is embedded into the programming structure.
- In the *screen-based model* pages are manually linked together to obtain the hypermedia application.
- In the *information-centred model* the information is stored in a database.

Process modelling consists of selecting phases, activities and artifacts for the development and establish how these are integrated into the specific development process of an application. Incremental development and prototyping are recommended for the development of hypermedia applications.

The activities that are performed during the development process are*: system analysis, design, production, verification and testing* as well as *maintenance.* A set of development techniques are presented for some activities related to the analysis and to the design of a hypermedia application. For example, it suggests RMM as methodology for the design of the structure.

3 Notations used in Hypermedia Design

Some of the works in the hypermedia modeling field concentrate their attention on the design and the notation, such as the methodologies outlined above: HDM, RMM, OOHDM, SOHDM and WSDM (see Table 2). Other works focus only on notation, like SHDT or the UML extension proposed by Conallen. The one presented by Baumeister, Koch and Mandel (1999) is based on concepts and models defined by already existing methods, but introduces a UML extension – fully compliant with the UML standard for every step.

The Structured Hypermedia Design Technique (SHDT) focuses on an own notation for a set of design primitives, that are used for a static representation of Web applications. These design primitives are: site, diagram, page, layout, form, index, menu, link and dynamic link. Site and diagram allow for grouping of diagrams and pages respectively. The purpose of layout is to assign one or more predefined sets of formatting specifications like headers, footers or background images to pages. Dynamic page generation is represented by templates and dynamic links. SHDT is an implementation-oriented technique as the primitives are derived from the functionality of HTML. The SHDT WebDesigner is a CASE tool that provides support for the design of Web applications. It generates a prototype of the planned Web system based on the SHDT design and on HTML stylesheets performed for the pages.

Conallen (1998) defines a set of UML stereotypes for the Web modeling, but does not present a method for the design of Web applications. It includes stereotypes for components, classes, methods and associations, such as server component, client component, server page, client page, form, frameset, target, server method, client method, links, redirect, build, targeted links and submit. These stereotypes are appropriate to model layout and implementation aspects, but not to define the navigation structure of the Web application. An interesting Web architecture is presented in this work. An Icon file including these stereotypes can be downloaded and be used in the Rational Rose tool.

The work of Baumeister, Koch and Mandel (1999) is a model–based approach whose modeling techniques are UML diagrams and whose graphical representation only uses UML notation. The models built are: the conceptual model, navigational model and presentational model. UML is extended to model the navigation and presentation according to the UML extension mechanisms. These mechanisms are based on the definition of stereotypes and the use of OCL.

The other hypermedia design hypermedia methods use standard notations merely for the conceptual

design. These notations are object-oriented (OO) like OMT or UML, or E-R based. They define their own notation and graphical techniques for the other steps.

RMM for example, performs an E-R design in the first step using therefore the E-R notation. For the following steps slice, navigation and user interface design it defines its own notation. WSDM instead, does not prescribe any notation for the conceptual model, E-R as well as OMT are suggested and also uses its own defined graphical notation for the navigational model objects. EORM only presents a graphical representation for the class structure diagram using OMT notation for this purpose.

In early papers (Schwabe, Rossi and Barbosa, 1996) OMT is proposed by OOHDM; however, in a latter one they used UML, but only for the conceptual model (Schwabe and Rossi, 1998). OOHDM is not UML compliant as it uses an own notation for the so called perspectives for attributes in the class diagrams and proposes other kind of diagrams (navigational context schema, configuration diagram, ADV charts) for the navigational design and abstract interface design.

SOHDM is a scenario-based methodology using its own notation for the scenario activity diagrams, the class structure diagrams and object-oriented views. The class structure diagram is a graphically representation of the information contained in Class-Responsibility-Collaboration Cards, so called CRC Cards (Wilkinson, 1995). WSDM is not restrictive for the notation selection; E-R and OMT are both suggested for the business object modeling, user object modeling and perspective object modeling steps. For the navigational model an own notation is chosen.

4 Comparison of the Hypermedia Development Methods

The comparison of software systems development methods is a difficult task. The focus of the methodologies may be different, some try to address many aspects in the development process, others try to detail in depth one or two of them. Our comment that a specific issue is not addressed, is then not be seen as a criticism, rather than the observation that the methodology will not offer help for this issue.

We present in this work three comparative studies: The first on is shown in Table 1. It compares the steps to be performed when using a method, the modelling technique used as well as the graphical representation and notation chosen for this development. Other criteria used for the comparison is the CASE tool support they offer for the development.

The steps or phases of the process are numbered. These numbers are used in the third and fourth column to indicate what kind of graphical representation and notation are proposed for each step.

It can be observed that although the number of steps, the techniques, notations as well as the graphical representation are different, the sequence how these steps are performed are similar. First the domain model of an application is analysed and/or designed, then the methods focus on the structure and navigation and at last but not least they proceed with the user interface design.

	Process	Modelling technique	Graphical representation	Notation	Tool support
HDM	1. Authoring-in-the-large 2. Authoring-in-the-small	E-R	1.- 2. E-R diagram	E-R	
RMM	1. E-R design 2. Slice design 3. Navigational design 4. Conversion protocol design 5. UI screen design 6. Run-time behaviour design 7. Construction and testing	E-R	1. E-R diagram 2. Slice diagram 3. RMDM diagram	1. E-R 2.- 3. own	RMCase
EORM	1. Class framework 2. Composition framework 3. GUI framework	OO	1. Class diagram 3. GUI design	1. OMT	ONTOS Studio
OOHDM	1. Conceptual design 2. Navigational design 3. Abstract UI design 4. Implementation	OO	1. Class diagram 2. Navigational Class + Context schema 3. ADV Configuration diagram + ADV charts	1. OMT/ UML 2. own 3. ADVs	OOHDM-Web
SOHDM	1. Domain analysis 2. OO Modeling 3. View design 4. Navigational design 5. Implementation design 6. Construction	Scenarios OO Views	1. Scenarios activity diagrams 2. Class structure diagram 3. OO view 4. Navigational link schema 5. Page schema	1.- 5. own	
WSDM	1. User modeling 2. Conceptual design 2.1 Object modeling 2.2 Navigational design 3. Implementation design 4. Implementation	E-R/ OO	1. E-R or class diagram 2. Navigation layers	1. E-R/ OMT 2. own	
RNA	1. Stakeholder analysis 2. Element analysis 3. Relationship and metaknowledge analysis 4. Navigation analysis 5. Relationship and metaknowledge implementation analysis				DHymE
MacWeb Approach	A1. Concept elicitation B1. Generating material A2. Navigational model B2. Organising and structuring A3. Abstract Interface A4. Implementation model B3. Reorganising and restructuring A5. Testing B4. Evaluation	OO	A2. Class structure	2. own	MacWeb

	Process	Modelling technique	Graphical representation	Notation	Tool support
HFPM	1. Requirement modeling 2. Project planning 3. Conceptual modeling 4. Navigational modeling 5. Abstract interface modeling 6. Design patterns employment 7. Multimedia data capturing/editing 8. Physical modeling/ integration 9. Validation/verification 10. Cognitive criteria employment 11. Quality assurance 12. Project co-ordination and management 13. Documentation	OO	3.– 5. OOHDM	= OOHDM	
OO/ Patterns Approach	1. Use case analysis 2. Conceptual design 3. Collaboration design 4. Class definition 5. Pattern-based navigational design 6. Implementation	OO	2. Class diagram 3. Collaboration diagram		
Lowe-Hall Approach	1. Domain Analysis 2. Product modelling 3. Process modelling 4. Project planning 5. Development 5.1. Analysis 5.2. Design 5.3. Production 5.4. Verification and testing 5.5. Delivery and maintenance 6. Documentation	RMM (suggested)			

Table 1: Methods for hypermedia development: Process, technique, graphical representation, notation and tools.

The second comparative study is a concept-based comparison. It is similar to the comparative table of OOSE, OOA, OOD, HOOD, and OMT elaborated by Jacobson (1992).

The results are shown in Table 2. It relates the design concepts of some of the methods outlined to the three typical levels of hypermedia design: conceptual, structural and presentational level. Most of these methods make, as mentioned above, a clear separation of the domain problem analysis from the specification of the navigation space structure as well as from the design of the user interface. The most important concepts used by six different design methods are listed in the table. Similar concepts are placed on the same row.

	HDM	RMM	EORM	OOHDM	SOHDM	WSDM
Conceptual level	entity collection perspective relationship	entity relationship	class oo-relationship - generalised - user defined	class perspective oo-relationship	scenarios: - event - activity activity flow	object perspective relationship
Structural level	link: - structural - application - perspective node component link: -collection - index - guided tour	link: - unidirectional - bidirectional slices acc. primitives: - grouping (menu) - index - guided tour - indexed guided tour	link: - simple - navigational - nodeToNode - spanToNode - structural: - set - list	link nav. class nav. context acc.structures: - index - guided tour	nav. link oo-view: - base - association - collaboration ASN (acc. structure nodes): - grouping - index - guided tour	link component - navigation - information - external nav. track
Visible level	slot frame			ADV (Abstract Data View) inContext	UI component: - button - image - list - choice - slide bar - search input text - HTML-anchor - others	

Table 2: Concepts used hypermedia development methods
Abbr.: nav: navigational, acc: access, oo: object.-oriented, UI: user interface.

The third comparison is given in Figure 1. It compares the phases covered by these methods. Note that comparing phases like this, hides other important aspects. The depth with which a method describes a phase and the guidelines given for that phase can vary enormously. Some method descriptions only have a couple of textual paragraphs while others provide tools to support the phase.

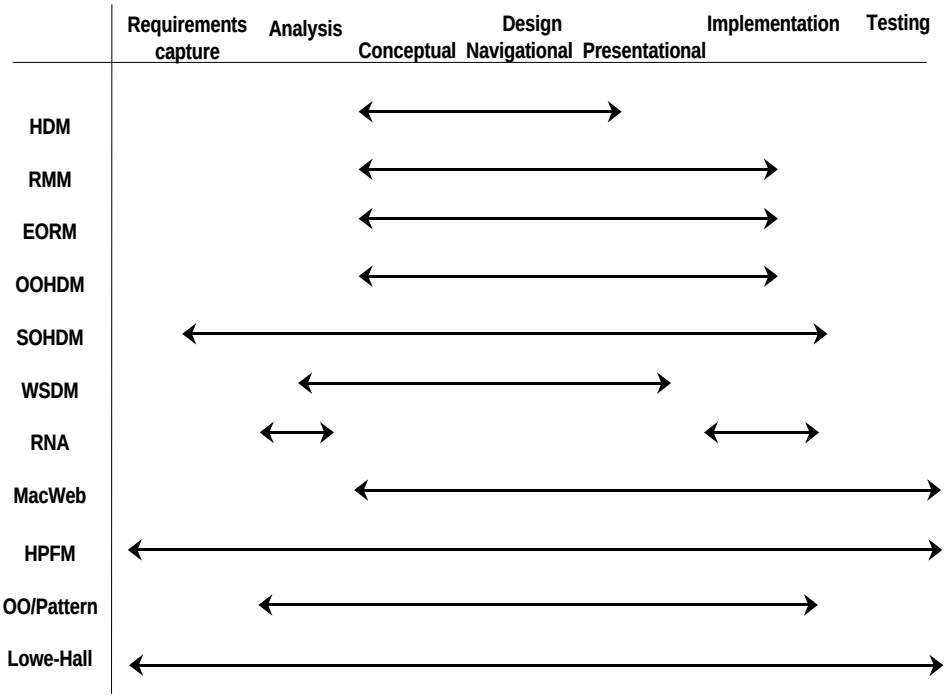

Figure 1: Phases covered by hypermedia development methods

We use here as base of the comparison the workflows proposed by the Unified Process: requirements capture, analysis, design, implementation and testing.

Other possibilities of comparison are to analyse how the model enforces production of robust applications or the degree of formalisation of the concepts. But there are more difficult to be performed. For example, to compare development results applying different methods is to expensive as it requires more than one production under same environment constraints.

5 Hypermedia Development Methods and the Unified Process

The Unified Software Development Process (in short form Unified Process) similar to the UML are becoming standards in software engineering. The Unified Process (Jacobson, Booch & Rumbaugh, 1999) proposes a development process for general software systems using models and notations of UML (Booch, Rumbaugh & Jacobson, 1999). The Unified Process is the result from the development and practical experience with other methodologies, such as the Objectory Process (Jacobson, 1992) and the Rational Unified Process (Kruchten, 1998). Although in covering the whole software life-cycle the Rational Unified Process extends the Unified Process providing workflows for project management, configuration and change management.

The methods described and compared in the previous sections are not based on the Unified Process.

A list of the most relevant characteristics of the Unified Process is given in the following. For each of these characteristics a brief analysis is done to determine which aspects are covered by the methodologies for hypermedia systems that are outlined in the previous sections.

- **Grouping of the iterations in inception, elaboration, construction and transition phases.**

The life-cycle of the development process of a software product is divided in the Unified Process in inception, elaboration, construction and transition phases. Note that maintenance is not a separate

phase. It can be considered as another development project with its own inception, elaboration, construction and transition phases. The Unified Process suggests one or more iterations for each of the four phases.

Some of the hypermedia design methods, such as OOHDM and RMM and the engineering approach of Lowe and Hall describe an iterative process but they do neither make a distinction between these iterations nor do they specify the number of iterations to be performed.

- **Covering requirements capture, analysis, design, implementation and testing.**

The Unified Process focuses on the entire development process that begins with the user's requirements to end with the software system. Most of the above described methods focuses on the design of hypermedia applications. Figure 1 depicts the phases each method covers (core workflows in the Unified Process terminology). Requirements capture and testing is only taken into account by HFPM and the Lowe-Hall approach.

- **A Use-Case-driven process**

There are two different ways to capture the requirements of the system to be built. First, the traditional functional specification that finds out what the system is supposed to do. Second, the use case strategy that consists in finding out the system functionality from the user's perspective, i.e. what the system is supposed to do for each user. Use-case driven means that the development process proceeds through a series of models derived from the use cases. Use cases are found, are specified and are the source for the analysis model and test cases. Use cases are not sufficient to define the analysis and design model, these models are also based on the architecture definition.

The hypermedia design methods start with the conceptual analysis or design with the exception of SOHDM, WSDM and HPFM. SOHDM is a scenario-based approach that uses these scenarios as source for the domain analysis. WSDM describes the domain from the user perspective; it is a user-centric approach. HFPM proposes use-case modeling for the software requirements modeling task.

- **A specification based on the architecture of the system**

The Unified Process sees the architecture as a complement to the use-cases. They define the functionality of a software system. The architecture specifies the organisation of the software system, the structural elements and their interfaces. The architecture is described by a set of five models: use-case model, analysis model, design model, deployment model and implementation model.

All methods described above construct at least a static model of the domain; with the exception of RNA they also proposes a graphic representation of these models. These models are part of the description of the architecture of the software system.

- **Component-base implementation.**

The Unified Process is a component-based methodology as the software systems being built are made up of software components interconnected via interfaces. For all the hypermedia methods to perform a component-base architecture is not a restriction posed by the methodology. It is a decision of the designer to perform a component-based design and implementation or not.

- **An incremental and iterative process.**

An incremental and iterative process provides the strategy for developing software systems in small manageable steps. The iterations in the inception and elaboration phases of the Unified Process are concerned with scoping the project, removing critical risks and baselining the architecture. In the construction and transition phases the objective is to implement and integrate the components step by step, i.e. in small increments.

All hypermedia methods can be applied in successive iterations, but only few propose it, such as OOHDM and the Lowe-Hall approach. For the Unified Process an iterative and incremental

development process is one of the cornerstones of the method.

- **Initiating the development process with a feasibility study.**

HFPM and the Lowe-Hall engineering approach suggests a first step with a feasibility study. The purpose is to evaluate the viability of the project to convert a vision into a software system. The Unified Process just mentions it by the way. But this step is of particular importance, if an expensive software system is to be developed.

- **Elaboration of a business model.**

Business modeling is used in the Unified Process to describe the business processes of an organisation in terms of business use cases and business actors corresponding to business processes and customers, respectively.

The business model represents the business from the usage perspective and outlines how it provides value to the users. Although the business aspects are relevant in most hypermedia systems, only WSDM proposes to built a business object model.

- **Planning and assessment.**

The Unified Process proposes the elaboration of a project plan as well as activities for the project assessment and so does HPFM and the engineering approach of Lowe-Hall. The project plan comprises of a list of milestones, at least one for each phase, a time schedule, evaluation criteria and a number of iterations for each phase. The objectives of the assessment are to examine what has been accomplished since the last evaluation, to review progress against the project plan and to determine if quality requirements are satisfied.

- **Risk management.**

Risk management consists of risk identification, risk description, impact analysis, assignment of priorities, risk monitoring, determination of risk responsible, risk mitigation and alternative plans in case risk materialises. The Unified Process gives only a set of brief explanations, hints and rules for risk management.

Only Lowe and Hall mention the importance of risk management. Although general risk management concepts and techniques can be used, the hypermedia development process has its own risk factors.

- **A model-based development**

Models are built, updated and consolidated in the Unified Process in all core workflows and in all iterations of the phases inception, elaboration, construction and transition. These models are built based on the techniques and notations defined by the UML. All other methods mentioned in this chapter also construct one or more static and/or dynamic models. The difference is that they do not use the Unified Modeling Language with the exception of Conallen and OOHDM for the conceptual model.

- **Use of UML techniques and notations**

The Unified Process employs only models defined in the UML and notations for the graphical elements as specified in the UML. Section 3 compares the notations used by the hypermedia methods and a summary of techniques and notations is shown in Table 1. These notations are not unified at all, most of the methods proposes their own notations for all or for part of their models.

6 Conclusions

This work outlines and compares the most relevant methods for hypermedia development and gives a comparison between the Unified Process and these methods.

The evolution of hypermedia development methods can be compared with the steps through which went the software development. We agree with Lowe and Hall (1999) that development processes for hypermedia will continue to evolve and improve. State of the art reports and comparative studies like this, allow for a better comprehension to understand what is missing and failing in the current methods. It can be helpful showing in which direction research has to continue. On the other hand, this work describes the strength of each method giving ideas to combine these strengths in an unified and improved method.

In particular, some research is needed to improve and test methods that cover the complete life cycle of hypermedia applications. Weak points are the capture of requirements, that is performed using informal techniques, such as interviewing or prototyping. Validation, verification and testing along the whole process also require to be focused on.

References

Avison D. and Fitzgerald G. (1995). *Information systems development: methodologies, techniques and tools.* Mc Graw-Hill.

Balasubramanian V., Bieber M. and Isakowitz T. (1996). Systematic hypermedia design. *Technical report, CRIS Working Paper Series. Stern School of Business, New York University.*

Baumeister H., Koch N. and Mandel L. (1999). Towards a UML extension for hypermedia design. *In Proceedings of The Unified Modeling Language: Beyond the Standard (UML´99).* France R. and Rumpe B. (Eds). LNCS 1723, 614-629.

Bieber M., Galnares R. and Lu Q. (1998). Web engineering and flexible hypermedia. *In Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia, Hypertext´98.*

Booch G., Rumbaugh J. and Jacobson I. (1999). *The Unified Modeling Language: A User Guide.* Addison Wesley.

Carneiro L., Cowan D. and Lucena C. (1993). Introducing ADVcharts: A graphical specification of abstract data views. *In Proceedings of CASCON´93.*

Conallen J. (1998). Modeling Web application design with UML. Available at http://www.conallen.com/ModelingWebApplications.html

De Troyer O. and Leune C. (1997). WSDM: A user-centered design method for Web sites. In *Proceedings of the 7th International World Wide Web Conference.*

Diaz A., Isakowitz T., Maiorana V. and Gilabert G. (1995). RMC: A tool to design WWW applications. In *Proceedings of the 5th International World Wide Web Conference.*

Garzotto F., Mainetti L. and Paolini P. (1995). Hypermedia design analysis. *Communications of the ACM,* 8(38), 74-86.

Garzotto F., Paolini P and Schwabe D. (1993). HDM: A model-based approach to Hypertext application design. *ACM Transactions of Information Systems,* 11(1), 1-26.

Harel D. (1987). Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8(3), 1987.

Henderson-Sellers B. (1995). Who needs an object-oriented methodology anyway? *Journal of Object Oriented Programming*, Vol 8(6), 6-8.

Iivari J. and Maansaari J. (1998). The usage of systems development methods: Are we stuck to old practices*? Information and software Technology, Elsevier*, 40, 501-510.

Isakowitz T., Stohr E. and Balasubramanian P. (1995). A methodology for the design of structured hypermedia applications. *Communications of the ACM*, 8(38), 34-44.

Jacobson I. (1992). Object-oriented software engineering: A use case driven approach. *Addison Wesley*.

Jacobson I., Booch G. and Rumbaugh J. (1999). *The Unified Software Development Process.* Addison Wesley.

Kruchten P. (1998). *The Rational Unified Process: An Introduction*. Addison Wesley.

Lange D. (1996). An object-oriented design approach for developing hypermedia information systems. *Journal of Organizational Computing and Electronic Commerce*, 6(3),269-293.

Lee H., Lee C. and Yoo C. (1998). A scenario-based object-oriented methodology for developing hypermedia information systems. In *Proceedings of 31st Annual Conference on Systems Science*, Eds. Sprague R.

Lowe D. and Hall W. (1999). *Hypermedia & the Web: An engineering approach*. John Wiley & Sons.

Nanard J. and Nanard M. (1995). Hypertext design environments and the hypertext design process. *Communication of the ACM*, August 1995, Vol 38 (8), 49-56.

Olsina L. (1998). Building a Web-based information system applying the hypermedia flexible process modeling strategy. *1st International Workshop on Hypermedia Development, Hypertext´98.*

Palvia P. and Nosek J. (1993). A field examination of system life cycle techniques and methodologies. *Information and Management*, Vol 25(2), 73-84.

Rossi G. (1996). OOHDM: Object-Oriented Hypermedia Design Method (in portuguese). PhD thesis, PUC-Rio, Brazil.

Rumbaugh J. (1995). What is a method? *Journal of Object Oriented Programming*, Vol 8(6), 10-16.

Schwabe D. and Rossi G. (1998). Developing hypermedia applications using OOHDM. In Proceedings of *Workshop on Hypermedia development Process, Methods and Models, Hypertext´98.*

Schwabe D., Rossi G. and Barbosa S. (1996). Systematic hypermedia design with OOHDM. In *Proceedings of the ACM International Conference on Hypertext, Hypertext´96.*

Thomson J., Greer J. and Cooke J. (1998). Algorithmically detectable design patterns for hypermedia collections. In Proceedings of *Workshop on Hypermedia development Process, Methods and Models, Hypertext´98.*

Wilkinson N. (1995). Using CRC cards: an informal approach to object-oriented development. *SIGS Books*.