

# Evaluation of Hypermedia Application Development and Management Systems

*S. P. Christodoulou, G. D. Styliaras, T. S. Papatheodorou*

High Performance Computing Architectures Laboratory  
Computer Engineering & Informatics Dept., University of Patras  
26500 Rion, Patras, GREECE  
E-mail: {spc,gds,tsp}@hpclab.ceid.upatras.gr

## ABSTRACT

In this paper we propose and study a framework for evaluating Hypermedia Application Development and Management Systems (HADMS) in relation to specific application requirements. We address the need for HADMS capable to efficiently support the main users involved in the life cycle of hypermedia applications, namely designers, programmers/implementers, authors/administrators and end-users. A HADMS consists of a hypermedia application development and management methodology and the respective environment. In this work, we propose and classify a set of evaluation criteria. These are mainly imposed by real life development and the need to support forthcoming, or next generation, features for hypermedia applications. We also introduce a simple framework for a comparative evaluation of HADMS. Furthermore, we demonstrate the use of the criteria and the framework proposed, for the case of three real-life applications. A representative set of seven HADMS is selected and the evaluation of these systems is carried out, leading to some useful conclusions and suggestions for future work.

**KEYWORDS:** Hypermedia application development systems, evaluation framework, criteria, methodology, Hypermedia design, Hypermedia systems, WWW.

## INTRODUCTION

The general problem addressed in this work is the need for better HADMS and for an evaluation framework for selecting the most suitable one for a certain application. This need is identified in previous work of several researchers and developers and is examined here with due consideration to additional requirements arising from practical experience. The examination of the problem also takes into account that the widest hypermedia system, WWW, will constitute the major application delivery and interface platform for the years to come [2]. Underlined issues are the ever

increasing needs of the designers to reuse design experience [20], of the authors to easily modify structure and content and of the end-users to be able to define their own ways to view and navigate through information.

More specifically, the work in this paper is motivated by the need for HADMS that exhibit sufficient support for structuring large amounts of information, designing, implementing and maintaining an advanced hypermedia application. Such a system should at least:

- provide an abstract hypermedia model, that supports next generation hypermedia features [3], capable to organize and semantically structure information material in an efficient way. This model should be extensible in order to be able to integrate upcoming hypermedia features that hypermedia systems will support.
- be able to convert (map) content of such a model into an either static or dynamic hypermedia application.
- support integration or importing of other information bases and reuse of existing applications.
- be able to easily reuse the same content to produce various kinds of hypermedia applications for different hypermedia systems (like WWW, Hyperwave, StorySpace, Microcosm, Windows Help etc.).

We provide a brief overview of existing HADMS, their approaches and describe seven representative HADMS which we later use for evaluation. We determine and present a set of evaluation criteria, grouped in three categories: Methodology evaluation criteria, Environment evaluation criteria and System Properties. These criteria try to cover the most important needs of the people involved in the life cycle of an application. They are mainly derived from our practical experience with extensive development of hypermedia applications for the Hellenic Ministry of Culture, educational hypermedia and several other applications. We introduce a simple evaluation framework in order to demonstrate how the criteria can be used in the evaluation of HADMS with respect to specific application requirements. We also demonstrate the use of the evaluation framework for the case of three real-life applications. Finally, we summarize our conclusions and propositions.

## OVERVIEW OF HADMS

In this section we provide an overview of existing HADMS and briefly describe their different approaches. We classify

them into five main categories: object-oriented approaches, entity-relationship approaches, component-based approaches, hybrid approaches and open hypermedia systems. Also, some other systems are summarized at the end of this section, under "other approaches". A set of seven representative systems was chosen for more elaborate evaluation. The six categories and the seven selected HADMS are briefly described below.

### **Object-oriented (OO) approaches**

Many features from the area of OO analysis, design and programming [21] can be extended to the hypermedia model. HADMS based on OO model allow the description of a hypermedia application using high-level constructs in an implementation independent way. The OO model focuses on the definition and inheritance of behavioral capabilities in the form of operations embedded within objects. It also provides capabilities for structuring complex objects. OO approaches support concepts such as objects, classes, methods, data abstraction, encapsulation, inheritance, reusability, concurrency control, composite objects, active objects, etc. OO approaches are capable of developing applications that require complex structures and relationships among the information items. Such approaches include the OOHDM [22,23,24] (Object-oriented Hypermedia Design Model), HSDL [14] (Hypertext Structure Description Language), EORM [15] (Enhanced Object-Relationship Model), JOE [16] (Java Ontology Editor), W3Objects [11]. OOHDM and HSDL were selected as representatives for this approach (HSDL was selected additionally because OOHDM does not provide an environment).

OOHDM [22,23,24]. The Object-Oriented Hypermedia Design Method (OOHDM) uses abstraction and composition mechanisms in an OO framework in order to allow a concise description of complex information items, and the specification of complex navigation patterns and interface transformations. In OOHDM, a hypermedia application is built in a four-step process (conceptual design, navigational design, abstract interface design and implementation) supporting an incremental or prototype process model. Each step focuses on a particular design concern, and an OO model is built.

HSDL [14]. The HSDL (Hypertext Structure Description Language) is a visual hypermedia-authoring environment, which uses the Structure Description Language as the base for the persistent data model storage mechanism. The core of the environment is the *schema* that consists of an intentional description, the *class schema*, and an extensional description, the *instance schema*. The class schema consists of classes and link-classes between them. The instance schema consists of instances and link instances. Large hypertexts are often organized as *hierarchies*. The HSDL implementation enforces strict referential integrity on the class and on the instance schemas. In HSDL several *schema evolution* operations are supported, such as reposition of a component in a class, split a class into two new classes, etc. The *translation* of a schema into HTML is controlled by a series of programs, called *expanders*.

### **Entity-Relationship (ER) approaches**

Another information modeling approach derived from the database and data manipulation fields is the ER approach. Weaker than OO, in terms of supporting advanced hypermedia features and functionalities, ER systems are better suited for the development and maintenance of simple and small-scale hypermedia applications, not including complex information items. The most representative ER approach is RMM [12,13].

RMM [12,13]. The Relationship Management Design Methodology (RMM) models application items using the ER approach and produces an application with links among the related entities and among the attributes within each entity. It has an associated development package called RMCASE [5], which can generate a stand-alone hypermedia application. It splits content design from navigational design and covers the application development process from early analysis stages to construction and testing. The user can easily switch among different phases of the development. The basic RMM methodology has been extended in [13] by the definition of *m-slices* which permit the grouping of information for various entities. This feature, along with some enhancements (specification of the anchors' content) and auxiliary tools (graphical notation, programming language), help in defining complex structures of content.

### **Component-based approaches**

Other HADMS adopt component-based approaches, in which the content of the application is hierarchically decomposed into components. A component is defined by its state (which is specified by a list of properties) and its behavior (operations on component states). Components can be derived from other components based on a prototype-instance model [26], which is simpler than the OO model. Component-based approaches are followed by Vignette's Story Server and WebComposition [7].

StoryServer. StoryServer (<http://www.vignette.com>) is a commercial product, developed by Vignette. It has a component-based architecture, which treats the functional elements of the Web site (content, format, application logic) as individual components, which can be built, managed and reused independently. StoryServer integrates a Web content management system with a content application server. The content management system creates a collaborative workflow environment for controlling the production and delivery content, while the content application server dynamically combines content with business processes and customer interaction to deliver personalized site viewing. It supports Adaptive Navigation, which personalizes the site navigation structure based on users preferences and their context depending on the way they navigate the site. Another element of StoryServer is the Production Center, a Java-based client application that facilitates the collaboration among all members of the development team - managers, designers, content creators, authors and administrators.

### **Hybrid approaches**

Some other HADMS follow hybrid approaches. Such ap-

proaches integrate data from various heterogeneous sources. Thus, they can accommodate data in various models (e.g. RDBMS, OODBMS, text DBs, filesystem etc.) without requiring storage of all data in one given repository. Such approaches include STRUDEL [6] and the system described in [9]. Hybrid approaches are appropriate for the development of mainly static applications, the content of which lays in diverse storage models.

STRUDEL [6]. STRUDEL provides a single graph model in which different data sources (e.g. existing Web sites and various types of external databases) are uniformly modeled. It allows users to manipulate the underlying data independently of where it is stored or how it is presented and to customize the web site by creating different views of the underlying data. Defining sites as views over the integrated data requires a single query language whose results are Web sites in the form of graphs of HTML pages. STRUDEL provides a query language for both data integration and view definition. However, STRUDEL cannot be used for managing the underlying data (it is an effective solution mostly for read-only sources equipped with their own management interface).

### Open Hypermedia Systems

Open Hypermedia Systems (OHS) can be identified as "originating from either Link Server Systems (LSS) or the Hyperbase Management Systems (HBMS) thread of research" [29]. The OHS could be considered as HADMS, but they actually produce hypermedia systems, i.e. applications that are tied together with the system. Thus, they are not able to produce applications for various hypermedia systems, from the same content. We evaluated two OHSs: HyperStorM [1] and HyperDisco [28]. The latter is an example of an OHS that is based on an HBMS and is the successor prototype of the Hyperform [27] hypermedia system development environment.

HyperStorM [1]. HyperStorM proposes an extensible OO hypermedia system, which supports the specification of application semantics as application classes within the hypermedia system, thereby supporting complex operations maintaining application-specific as well as application-independent constraints. In the HyperStorM hypermedia system, the storage layer and the application layer of a hypermedia system are implemented within the OO database management system VODAK. This approach facilitates both the reuse of database functionality and the support of the development of different kinds of hypermedia applications, but only for this hypermedia system.

### Other approaches

Most of the currently available commercial products for Web site management, such as Microsoft's FrontPage, NetObjects' Fusion, Netscape's Live Wire Pro and Bluestone's SapphireWeb, are essentially WYSIWYG HTML editors, including tools for managing the site structure, maintaining the consistency of links and providing a gateway to DBs. They are focused on attractive and helpful GUIs, but they are not providing efficient structuring model

and capabilities for the application content. Another approach includes systems (like ASML [19]) for developing web sites dynamically. They use the filesystem for storing the application content and they are focused on site-level development of WWW applications. They provide basic capabilities, like sharing of common page elements, searching and indexing, easy production of access structures (table of contents, indexes, guided tours, etc.), but they don't include high-level hypermedia features (i.e. typed nodes and links, conceptual data model design, navigational and interface design etc.). Some other tools that were examined include LivePage, a tool called "Navigation View Builder" described in [17] and LINCKS [25].

FrontPage. FrontPage (<http://www.microsoft.com/>) is a commercial product, developed by Microsoft. It is a graphical tool that helps design and produce simply-structured content without a strict underlying model. It incorporates a built-in HTML editor and wizards that help in embedding technologies like Java and JavaScript in WWW pages and producing database driven pages. The Hyperlink View provides various hierarchical and graphical representations of a WWW site's structure. It supports automatic update of hyperlinks. Moreover, simple navigation buttons can be automatically added inside pages. With Cascading Style Sheet (CSS) complex styles can be created and applied on a set of pages. Remote and collaborative authoring and editing is enabled. Windows NT security can be used for assigning permissions and restrictions. Supporting tools, like image editing tools and broken link detectors, are also provided. The environment is accompanied by adequate on-line help.

### EVALUATION CRITERIA FOR HADMS

In this section we determine and present a set of evaluation criteria for HADMS, derived from the needs of the main users involved in the life cycle of a hypermedia application. More specifically, we classify the users as end-users and application developers (designers, implementers and authors / administrators). The specific needs of these users combined with our own practical experience on developing hypermedia applications, "The Dexter hypertext reference model [10]" and the work of Garzotto F. et al [8], Bieber M. et al [3] and Bapat A. et al [1] led to the selection of the evaluation criteria.

These criteria are grouped in three categories: *Methodology Evaluation Criteria*, *Environment Evaluation Criteria* and *System Properties*. The first category includes criteria for the methodology of the HADMS, while the second includes criteria for the environment. Criteria in these two categories can be considered as quantifiable - their coverage degree by a HADMS can be approximately indicated by an integer. The System Properties category includes criteria, the coverage of which cannot be quantified. An example of such a criterion is the platform(s) that the environment is implemented for. A unique label combined by a letter and a number indicates each criterion. This label is used whenever we want to refer to a criterion.

We define for each of the quantified criteria and for each evaluated HADMS, a number that indicates the degree that the criterion is covered by the HADMS. We represent this number as  $CD_{\#crit}(system)$ , where CD stands for *Coverage Degree*,  $\#crit$  = the criterion label (e.g. D2) and *system* is one of the HADMS under evaluation. We assume that the  $CD_{\#crit}(system)$  is an integer in the range 0 to  $max_{\#crit}-1$ , where  $max_{\#crit}$  is the number of all the possible values for the  $CD_{\#crit}(system)$ . The value of  $max_{\#crit}$  depends on the criterion  $\#crit$  and could be 2 (the criterion is either supported or not) or greater (the  $CD_{\#crit}(system)$  is indicated by a fraction).

The coverage degree of most of the criteria becomes more difficult to estimate as  $max_{\#crit}$  increases, without adding much to the extracted conclusions. Also, if  $max_{\#crit}$  is too small, it wouldn't be enough for some criteria to represent the differences among HADMS. Thus, for simplicity, we assume that  $max_{\#crit}$  is the same for each criterion  $\#crit$  and equals to 4. However, one could make different assumptions concerning the value of  $max_{\#crit}$  for each criterion, considering various options e.g. including fuzzy logic. Thus,  $CD_{\#crit}(system)$  is an integer in range 0-3, with four discrete values: 0 = not supported, 1 = poor support, 2 = acceptable support and 3 = full support. We estimated  $CD_{\#crit}(system)$  for all the quantifiable evaluation criteria and for the seven representative HADMS. Straight after the description of each group of criteria, a table presents the results of this estimation, which constitutes the first part of the proposed evaluation framework.  $CD_{\#crit}(system)$  is displayed in these tables with three different tones of gray, which are mapped to a range of 1 to 3. Lack of a gray tone indicates the value 0. In some tables notes were necessary for explaining our decisions. All these tables are jointly presented in Figure 1, where one can observe the cumulative coverage degree of each criterion by all evaluated systems. Finally, for each HADMS, its system properties are specified.

### Methodology Evaluation Criteria

This category includes criteria met for the evaluation of the methodologies of HADMS. In order to present them in a well-structured and meaningful way, we examined several methodologies. Although different methodologies follow various kinds and numbers of steps, a more careful examination shows that most of them are based on four basic steps, which are explicitly observed in the OOHDM [22,23,24]. These steps are Conceptual Data Model Design, Abstract Navigational Model Design, User-Interface & Run-time Behavior Design and Implementation.

*Conceptual Data Model Design:* In this step a conceptual data model is built, capturing the characteristics of the application domain, using either an OO, component-based, ER approach, etc.

*Abstract Navigational Model Design:* In this step the navigational structure of a hypermedia application is described

in terms of navigational contexts, which are induced from navigation objects such as nodes, links and access structures, i.e. indices and guided tours. Navigational design should take into account the types of intended users and their tasks. Nodes and links represent logical views on conceptual data model defined during domain analysis. By defining the navigational semantics in terms of nodes and links, we can model movement in the navigation space independently of the conceptual model, while modifications into the conceptual model can have an impact in the navigational one, which the environment should propagate automatically.

*User-Interface & Run-time Behavior Design:* The user-interface model is built by defining perceptible objects in terms of interface classes. Interface classes are defined as aggregations of primitives classes (such as text fields and buttons) and recursively of other interface classes. Interface objects map to navigational objects, providing a perceptible appearance. Run-time behavior is declared by specifying how to handle external and user-generated events and how communication takes place between interface and navigational objects.

*Implementation issues:* In this step, the interface objects are implemented in such a way as to provide additional application features, like dynamic content or integration of external storage systems.

In the following subsections, we define the methodology evaluation criteria and present them according to these four basic steps. In addition, there are some criteria that are common to the three first steps and are classified into a separate subcategory.

#### A. Common Criteria among the Three First Steps

- A1. **Structural constraints.** The methodology must support the definition and maintenance of structural constraints (e.g. composites that may not contain themselves recursively) and guarantee their enforcement among hypermedia objects participating in any of the three design steps.
- A2. **Design evolution.** In order to provide extensibility and tailorability it is required that design evolution is supported, in each of the three first basic steps, with automatic propagation of the modifications from the step (design model) where they occur, to the other steps. The design evolution should cover the fundamental requirements for a hypermedia development environment that Nanard and Nanard have identified in [18].
- A3. **Reusing and tailoring.** This refers to the ability to allow reusing and tailoring of application-specific types of objects among different applications. Recently, design reuse was also addressed in [20].

	OOHDM	RMM	HSDL	HSM	STRUDEL	STORY	FP
A1	Note1	Note1	Note1				
A2							
A3		Note2					

Notes: 1. Referential integrity

2. Using m-slices

## B. Conceptual Data Model Design

- B1. **Basic hypermedia support.** The data model approach used to map application objects to hypermedia objects should provide at least a set of basic types for hypermedia objects such as nodes, links, anchors, etc.
- B2. **Multimedia support** of existing multimedia types and ability to integrate future ones.

	OOHDM	RMM	HSDL	HSM	STRUDEL	STORY	FP
B1							
B2		Note1					Note1

Notes: 1. Stored in filesystem

## C. Abstract Navigational Model Design

The criteria in this category are further categorized into Node, Link and Navigational Features.

### Node Features

- C1. **Semantically typed nodes** categorize a node's contents, help authors organize information more effectively and lend context to readers.
- C2. **Composites.** Dealing with groups of links and nodes.

### Link Features

- C3. **Semantically typed links.** Information contained by a link type can be a user-interface level action triggered by a user following the link or the resource the link points to or the way two, or more, nodes are related, such as "explanation".
- C4. **Link attributes.** In addition to types, other semantics can be attached to links, such as labels, names, keywords or timestamps. Smarter links could contain other information like abstracts and sizes of the linked-to files that would aid a user in his selection.
- C5. **Bi-directional links.** This refers to the ability of the system to support links that can be accessed from either endpoint.
- C6. **Anchors embedded in node content:** Ability to include either the start or end-point of a link inside the content of a node.
- C7. **Internal link update mechanisms,** guaranteeing consistency (avoiding broken links). Whenever a node is removed, every internal link of the application pointing to it should be removed.
- C8. **Automatic creation of links** among nodes, according to their contents and types. While most links are handcrafted, the larger the information base, the less feasible authors find it to compose all links manually.

### Navigational Features

- C9. **Global and local views,** improve spatial context and reduce disorientation in a hypermedia network [3, 14]. *Global views* provide an over-all picture and can also provide entry points for local views. *Local views* provide a detailed view of the local neighborhood of a node.
- C10. **Node and link participation in different views.** Support of different behavior for hypermedia objects, since many hypermedia applications support multiple uses of hypermedia objects in different contexts.

C11. **Access structures.** This refers to the ability to create structural parts of a hypermedia application (summary objects), e.g. menus, indexes, guided tours, indexed guided tours.

C12. **Personalization features (user profiling).** Such features are the *Path-tracking* and *Adaptive Navigation Support*. A reader's path (documents a reader's earlier visited) could be saved and restored on reader's demand. A path can be used to collect all interesting documents to form a linear document that can be preserved in printed form or to help developers find the more frequently visited paths and to avoid readers to revisit the same information units. *Adaptive Navigation Support* controls which links and their corresponding properties should be activated based on the user's previous interactions. Hypermedia applications should include filtering mechanisms to present only the most relevant links, based on the users' current goals.

	OOHDM	RMM	HSDL	HSM	STRUDEL	STORY	FP
C1							
C2		Note1	Note2			Note3	
C3							
C4							
C5							
C6							
C7	Note4	Note4		Note4	Note4		
C8				Note5			
C9							
C10							
C11	Note6					Note7	
C12							

Notes: 1. m-slices of attributes  
 2. Hierarchy of objects  
 3. Hierarchy of components  
 4. Referential integrity  
 5. When constraints are violated  
 6. Except indexed guided tours  
 7. Templates

## D. User-Interface & Run-time Behavior Design

- D1. **How interface objects map to navigational objects.** The design approach for describing the user interface of an application. Such approaches are ADV-charts[4] and StateCharts[30].
- D2. **Specification of run-time behavior to interface objects.** Specification of the reaction of interface objects to external events.
- D3. **Synchronization of interface objects,** particularly when dynamic media such as audio and video are involved.
- D4. **User-defined instantiation.** The environment should allow a reader to create different views from the same semantic hypertext and give the readers the chance to choose the way they prefer to see the information, including jumping from view to view. The readers should be able to issue commands to the hypermedia application that will allow the traversal of information in their own personal manner.

	OOHDM	RMM	HSDL	HSM	STRUDEL	STORY	FP
D1	Note1	Note2	Note3	Note4	Note2		Note5
D2							Note5
D3							Note5
D4							

Notes: 1. Using ADV charts  
 2. Templates  
 3. Using Expanders  
 4. Using VQL  
 5. Using FP embedded editor

E. Implementation issues

- E1. **Dynamic content.** Ability to generate and/or display content at run time, dynamically generated from the storage model of the system.
- E2. **External storage systems.** Ability to integrate or co-operate with external storage systems, from RDBMSs and OODBMSs to video servers, document management systems, knowledge bases or GISs.
- E3. **Simplicity.** Ease of using the methodology for designing and implementing an application.

	OOHDM	RMM	HSDL	HSM	STRUDEL	STORY	FP
E1							
E2							
E3							

**Environment Evaluation Criteria**

This category includes criteria used for the evaluation of the implementation environments of HADMS. These criteria concern the support for the implementation, testing and maintenance stages of the application life cycle.

F. General

This category introduces various kinds of evaluation criteria requiring from the environment to provide capabilities like distribution, extensibility and information retrieval.

- F1. **Client/server distribution.** The ability of the environment to be based on a client/server architecture, allowing its users to have remote access to it through the network.
- F2. **External tools.** The ability to integrate external tools (e.g. text or HTML editors, media processing tools) into the environment.
- F3. **Extensibility (Scripting capabilities / API).** Flexibility for the developer to enhance existing or add new features to the current implementation.
- F4. **Importing.** Support of batch information importing from databases, filesystems, WWW sites (locally or over network through HTTP), etc. The system should also provide extensibility to the supported formats.
- F5. **Content-based information retrieval mechanisms.** The application developer should have the ability of content-based search on the content of the application.
- F6. **Structure-based information retrieval mechanisms.** In addition to a text string search facility for node content, structure-based queries based on node and link attributes are often useful. An example of such a query is "find the abstracts of all technical reports modified during the previous month".
- F7. **Supporting utilities,** such as find & replace expressions in the application content, spell checking, etc.
- F8. **Quality testing.** How much does the environment assist users in evaluating their application design.

	OOHDM	RMCASE	HSDL	HSM	STRUDEL	STORY	FP
F1	-						
F2	-		Note1				
F3	-		Note2				
F4	-		Note3			Note3	
F5	-					Note4	Note4
F6	-						
F7	-						
F8	-						

Notes: 1. Preview browser, HTML editor 2. LISP support 3. From filesystem and DB 4. By integrating search tools

G. Collaborative work

In order to serve an organization's working groups, hypermedia development methodology features have to coexist with collaborative features that enable people to work together during the design, development and maintenance phases. Collaborative tools maintain many relationships between a working group's members, tasks, and information.

- G1. **Simultaneous multi-user access, including robust concurrency control mechanisms.** Allow multiple users to access concurrently the development environment while preserving the consistency and integrity of data.
- G2. **Access control.** Besides standard read and write access permissions on documents, system administrators may consider analogous read and write permissions for hypermedia objects. For certain documents one could allow link authoring and update while prohibiting write access to the underlying content. Access permissions may prove most useful for allowing individuals to maintain a personal set of nodes, links and annotations, while collaborating workgroups maintain shared sets of objects.
- G3. **Member activity tracking.** Tracking actions and / or modifications made by each member of a team.
- G4. **Notification controls.** Users can be informed about modifications that have taken place on contents or structures.
- G5. **Versioning** at the level of individual entities e.g. links, nodes and also at the level of the hypermedia application schema. Version control provides one way to control the disruptive effects of change without the worse solution of preventing or obstructing it.
- G6. **Annotations (private, workgroup or public).** Creation of annotations (comments) is a basic right for hypermedia readers as well as a basic tool for collaboration and exchange of ideas. The environment should support private, workgroup or public annotations, providing them to any person who has the appropriate access permissions.
- G7. **Interactive collaboration:** Support of audio and video teleconferencing and interactive electronic whiteboards.

	OOHDM	RMCASE	HSDL	HSM	STRUDEL	STORY	FP
G1	-	Note1	Note2			Note3	
G2	-					Note4	Note5
G3	-						
G4	-					Note6	
G5	-		Note7			Note8	Note9
G6	-						
G7	-						

Notes: 1. RDBMS 2. Built-in check 3. Locking 4. Different levels of access 5. NT security system 6. via mail 7. Using different SDL files 8. Project-based 9. Using SourceSafe

H. User Interface (UI) - Friendliness and ease of use

This category includes criteria relative to the friendliness of the environment's UI and the ease of using it.

- H1. **UI of the design and implementation environment,** which is used by designers and implementers during the basic methodology steps.

- H2. **UI of the authoring environment**, used by authors to insert, update or delete information.
- H3. **Intelligent user assistance**, e.g. wizards, functional on-line help, etc.

	OOHDM	RMCASE	HSDL	HSM	STRUDEL	STORY	FP
H1	-				-		
H2	-				-		
H3	-				-		

### System Properties

This category includes criteria whose coverage by the HADMS is not quantified. For these criteria, the specification of certain value(s) rather than a degree of coverage is required.

- I1. **The model approach** used to map application objects to hypermedia objects (OO, ER, component-based, etc.).
- I2. **Storage System**, which the environment uses to store the conceptual, navigational and interface design and application contents. The most commonly used are OODBMS, RDBMS and Filesystem.
- I3. **Kinds of hypermedia applications supported**. Using the same data, the system could be able to produce various kinds of hypermedia applications (a Web site, a CD-ROM based application, a Windows Help application, etc.) for diverse hypermedia systems (e.g. WWW, Hyper-Wave, Microcosm).
- I4. **Platforms**. For which platforms the components of the environment have been implemented.

	OOHDM	RMCASE	HSDL	HSM	STRUDEL	STORY	FP
I1	OO	ER	OO	Extended OO	Component-based	Component-based	-
I2	-	RDBMS	OODBMS	OODBMS (VODAK)	-	RDBMS	Filesystem
I3	Any hypermedia system	WWW, ToolBook, Hypercard	WWW	Application and system together	WWW	WWW	WWW
I4	-	Windows	Windows	UNIX	-	Windows, UNIX	Windows, Mac

### EVALUATION FRAMEWORK AND A CASE STUDY

We specify an evaluation framework in order to demonstrate how the criteria can be used in the evaluation of HADMS with respect to specific application requirements. We also demonstrate the use of the evaluation framework for the case of three applications arising in practice. The first part of this framework has already been described in the evaluation criteria section.

Based on the application requirements, a developer may specify the degree that each quantified evaluation criterion should be supported by the desired HADMS. This degree is represented as  $RCD_{\#crit}$ , where RCD stands for *Required Coverage Degree*.  $RCD_{\#crit}$  is an integer in the range 0 to  $max_{\#crit}-1$ . We assume that  $max_{\#crit}$  is the same for each criterion  $\#crit$  and equals to 4. Thus,  $RCD_{\#crit}$  is an integer in range 0-3, with four discrete values: 0 = irrelevant, 1 = desired, 2 = highly recommended and 3 = required (must be supported). Finally, the developers should specify the desired System Properties (criteria I1, I2, I3, I4).

Let N be the number of criteria for which  $RCD_{\#crit} > 0$ . For each of the candidate HADMS, a number  $RANK(system)$  is calculated that indicates the ability of the system to efficiently support the development of such an application:

```

RANK(system) = 1
for each #crit that RCD#crit > 0
  if RCD#crit > CD#crit(system) then
    RANK(system) = RANK(system) -
      (RCD#crit - CD#crit(system)) / N * (max#crit - 1)
  end-if
end-for

```

The  $RANK(system)$  is a number between 0 and 1. The larger this number is, the better the N criteria are covered. In addition, the developers should also consider the

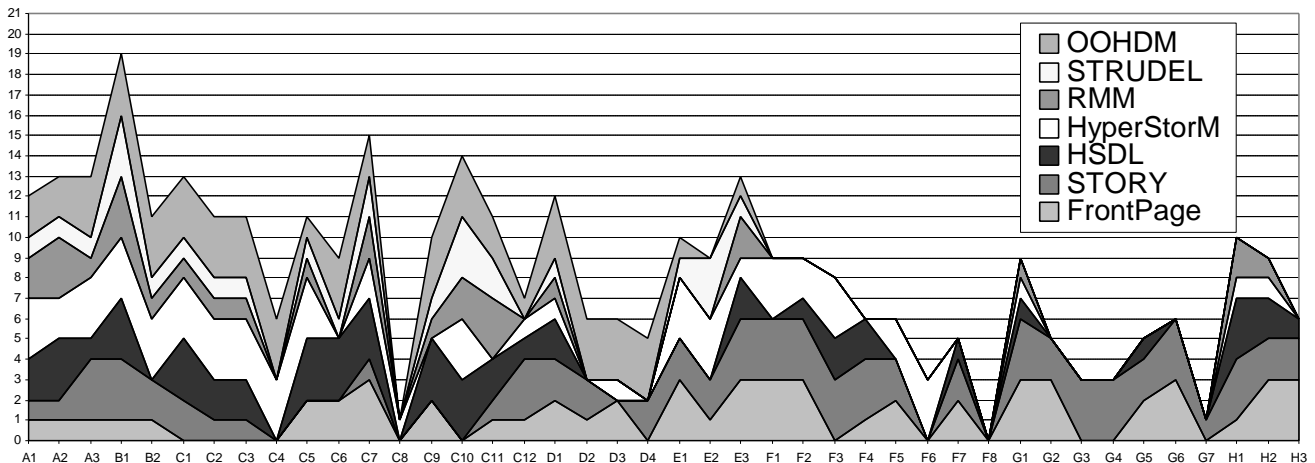


Figure 1: Coverage degrees of each criterion by the evaluated HADMS (jointly presented)

app	A1	A2	A3	B1	B2	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	D1	D2	D3	D4	E1	E2	E3
1	2	1	2	3	1	0	0	0	0	0	2	2	0	1	3	3	2	0	1	0	0	3	3	2
2	3	3	3	3	1	3	3	3	3	3	3	3	2	3	3	3	3	1	2	0	3	3	0	2
3	2	1	2	3	3	3	2	3	3	3	3	1	0	3	3	3	3	3	3	1	3	0	1	1

app	F1	F2	F3	F4	F5	F6	F7	F8	G1	G2	G3	G4	G5	G6	G7	H1	H2	H3
1	3	1	0	3	3	2	3	1	3	3	2	2	2	1	0	2	1	1
2	2	1	1	1	3	3	2	0	2	2	1	1	1	1	0	2	3	2
3	3	3	1	0	3	3	3	2	3	3	3	3	3	3	3	3	3	3

I1	I2	I3	I4
Hybrid	WWW	any	any
OO	WWW	any	any
any	WWW,CDROM	Windows	any

Figure 2: Application requirements

	OOHDM			RMM			HSDL			HyperStorM			STRUDEL			StoryServer			FrontPage		
	M	E	T	M	E	T	M	E	T	M	E	T	M	E	T	M	E	T	M	E	T
App1	0.82	-	-	0.71	0.40	0.55	0.76	0.52	0.63	0.80	0.54	0.67	0.78	-	-	0.73	0.88	0.81	0.73	0.75	0.74
App2	0.80	-	-	0.48	0.50	0.49	0.67	0.65	0.66	0.70	0.67	0.68	0.48	-	-	0.52	0.90	0.68	0.45	0.83	0.61
App3	0.91	-	-	0.52	0.14	0.35	0.70	0.27	0.51	0.71	0.33	0.55	0.53	-	-	0.59	0.75	0.66	0.50	0.61	0.55

Figure 3: Case-study results (M: Methodology, E: Environment, T: Total)

required system properties and whether they are met by the properties of the qualified system. Using the same algorithm with the corresponding subsets of criteria, a developer can separately evaluate the methodologies and the environments of the candidate HADMS.

#### Evaluation of Representative Systems – A case study

As a case study, we specify three example applications and we use our evaluation framework to evaluate the seven representative HADMS, in order to select the most appropriate HADMS for each application. The three applications are as follows:

*Application1.* The Hellenic Ministry of Culture maintains heterogeneous information repositories, including an OODBMS for all Greek artists, archaeologists and authors and their activities, an RDBMS for all the cultural activities, their organizers, the time and the location where they take place, the sponsors, etc. and some HTML pages containing information for Greek archaeological sites, monuments and museums. The Ministry plans to develop a hypermedia application, with both static and dynamically generated HTML pages. The latter would be dynamically extracted from the different sources and uniformly presented to the end-users upon their request. Some of the application contents require frequent updating from non-expert users, who need a friendly and easy to use authoring environment. Also, the presentation of the application should be easily mutable.

*Application2.* A large organization such as the Hellenic Ministry of Environment, Physical Planning and Public Works intends to develop a WWW application to describe the hierarchy of the Ministry, its several directorates and departments, its employees and their activities. The application should also be used for facilitating project monitoring and management. Complex relationships among information items are required. Moreover, most of the content needs to be frequently updated. Part of the application will be dynamic. The application should support different views of its contents for different kinds of users (e.g. external us-

ers, employees, managers, directors).

*Application3.* The Hellenic Ministry of Culture wants to collect and organize information material (rich in media but without complex relationships among information items) for the Acropolis of Athens, in order to develop a stand-alone multimedia application and a static web site. Such an application requires high collaboration support among different kinds of people (archaeologists, directors, graphic designer, authors, programmers, etc.). Also, there are needs for high quality presentation of the application.

In this case study,  $system \in \{ "OOHDM", "RMM", "HSDL", "HyperStorM", "STRUDEL", "StoryServer" \}$ . We estimated the  $RCD_{\#crit}(system)$  for all the quantifiable evaluation criteria and for each of the three applications (see Figure 2). For each application and system evaluated, three values are calculated:  $M$ (ethodology),  $E$ (nvironment) and  $T$ (otal). As OOHDM and STRUDEL don't have an environment, the respective columns for the environment and overall grade are left empty. The results of the evaluation are presented in Figure 3. Of course important parameters on extracting such conclusions are the required system properties. For instance, if a system cannot run on a UNIX platform (criterion I4), then this system may be a-priori rejected. However, if the RANK is very high (or RCDs are very high) indicating a system of a very high performance, then one may consider purchasing a non-UNIX platform.

#### Conclusions derived from the case-study

Application1 has relatively low requirements, but because of its special need to integrate heterogeneous data sources, STRUDEL seems to be the only suitable methodology, but it does not provide an implementation environment. This application indicates the influence that system properties can impose on the selection of a HADMS. Although StoryServer has a very high rank at the environment evaluation, it does not constitute the best choice. Application2 needs a strong methodology, whereas Application3 requires a powerful environment. It seems that none of these two applications can be acceptably supported by any of the evaluating

HADMS. However, it is important to notice that some of the existing methodologies (OOHDM, HyperStorM) and some other existing environments (StoryServer, FrontPage) can acceptably cover their requirements for the methodology and environment respectively, but there is lack of a complete HADMS.

More specifically, for the systems evaluated and for the case-study of the three example hypermedia applications, the following are observed:

- OOHDM seems to be the most powerful methodology for most of the application types. However, it lacks a complete implementation environment, although the methodology has been used for developing several hypermedia applications.
- RMM's main advantage is the use of a simple ER design approach and, therefore, is better suited for simple or small-scale applications. Another important feature of RMM is the ability to easily switch among different design steps, enforcing data consistency. However, it cannot support applications that require complex structures among information items and its implementation environment (RMCASE) behaves rather poorly.
- HSDL appears to be the best-balanced system regarding the methodology, the implementation environment and system properties. However HSDL cannot cover some advanced hypermedia features of the applications, like collaboration work and interface design.
- HyperStorM is the most complete HADMS, but it can only produce applications for its own hypermedia system.
- STRUDEL is based on a hybrid approach and thus it can accommodate data in various models, without requiring the storage of all data in one repository. Generally it doesn't provide an efficient methodology and moreover it doesn't provide any implementation environment.
- StoryServer covers efficiently the environment evaluation criteria and is more appropriate for the development of applications that require collaboration features, advanced user interface and scripting capabilities. FrontPage includes many wizards, a graphical user interface and more than adequate help, but in fact, it is a graphical tool for managing site structures (at page level), authoring HTML pages and providing DB gateways. The main disadvantage of both systems is their weakness to develop applications that require complex structures and relationships in data model and navigational design.

## CONCLUSIONS & PROPOSITIONS

In this paper we proposed and studied an approach for evaluating Hypermedia Application Development and Management Systems in relation to specific application requirements. Existing HADMS were classified into six categories and a set of seven representative systems was used for the evaluation. Advanced criteria were defined and classified. In order to demonstrate the use of these criteria, an evaluation framework was specified and actual evaluation was carried out for three on-going applications en-

countered in our practical work. Besides this demonstration, several conclusions were derived regarding the behavior of the seven representative HADMS. It was observed that no HADMS (methodology and the corresponding environment) is efficiently covering the evaluation criteria.

Also, there is no existing full-featured HADMS that one can use to develop different kinds of applications with different requirements. Thus, if one wants to develop three different applications, she/he might need to install, learn or probably purchase three different HADMS. In order to create a full-featured system one could propose either to extend the functionalities of any of the existing HADMS (e.g. by integrating a better methodology in an existing environment), to implement a totally new one, or to define a "meta-HADMS" that could integrate the currently most effective methodologies under a common decision and implementation environment. This meta-HADMS would take as input the application requirements and would transparently decide which methodology to use. It should provide the developers with the ability to move through the four basic methodology steps, designing the application in a methodology-independent way.

In some cases (like a large web site), a hypermedia application consists of a set of "smaller" applications that have different requirements. For instance, a part of a web site contains only static pages, while another is a dynamic application over a multimedia DB. In such cases the developer may need to use different methodologies for the implementation of the "smaller" applications. This requires the maintenance of different HADMS that possibly do not inter-operate. By providing a "meta-HADMS" that covers the underlying methodologies transparently, one may facilitate the process of developing applications with different requirements, giving the developer the impression of a full-featured HADMS.

## ACKNOWLEDGMENTS

This work was supported in part by the General Secretariat of Research and Technology of Greece, Grant 1656 ED 95 and by the "POLINET" project of the Hellenic Ministry of Culture.

## REFERENCES

1. Bapat, A. Wasch, J. Aberer, K. and Haake, J. HyperStorM: An Extensible Object-Oriented Hypermedia Engine. Hypertext '96.
2. Bieber, M. and Vitali, F. Toward Support for Hypermedia on the World Wide Web. *COMPUTER* Vol. 30, No. 1. (January 1997), pp. 62-70.
3. Bieber, M. Vitali, F. Ashman, H. Balasubramanian, V. and Oinas-Kukkonen, H. Fourth Generation Hypermedia: Some Missing Links for the World Wide Web. *International Journal on Human Computer Systems*, 1997.

4. Cowan, D.D. and Lucena, C.J.P. Abstract Data Views, an interface specification concept to enhance design reuse. *IEEE Transactions on Software Engineering*, Vol.21, No.3, March 1995.
5. Diaz, A. and Isakowitz, T. RMCASE: Computer-Aided Support for Hypermedia Design and Development. *International Workshop on Hypermedia Design 1995*.
6. Fernandez, M. Florescu, D. Kang, J. Levy, A. and Suciu, D. STRUDEL: A Web-site Management System. *ACM SIGMOD 1997*.
7. Gallersen, H. Wicke, R. and Gaedke, M. WebComposition: An Object-Oriented Support System for the Web Engineering Lifecycle. *Sixth International World Wide Conference, 1997*.
8. Garzotto, F. Mainetti, L. and Paolini, P. Hypertext Design, Analysis, and Evaluation Issues. *Comm. ACM*, Aug. 1995, pp. 74-86.
9. German, D. and Cowan, D. A federated database for hypermedia development for the WWW. *Codas 96*. Kyoto, Japan. Dec. 1996.
10. Halasz, F. and Schwartz, M. The Dexter hypertext reference model. *Comm. ACM*, Feb. 1994, pp. 30-39.
11. Ingham, S. Caughey, J. and Little M. C. Supporting Highly Manageable Web Services. *Proceedings of the 6<sup>th</sup> WWW Conference*.
12. Isakowitz, T. Stohr, E. and Balasubramanian, P. RMM: A Methodology for structuring hypermedia design. *Comm. ACM*, Aug. 1995, pp. 34-44.
13. Isakowitz T., Kamis A. and Koufaris M., Extending the Capabilities of RMM: Russian Dolls and Hypertext, *Proceedings of the 30<sup>th</sup> Annual Hawaii International Conference on System Sciences*, IEEE Press, Washington, D.C., Vol. VI, January 1997, pp. 177-186.
14. Kessler, M. A Schema-Based Approach to HTML Authoring. *W3 Journal*.
15. Lange, D. An Object-Oriented Design Approach for Developing Hypermedia Information Systems. *Journal of Organizational Computing*, October 1996.
16. Mahalingam, K. and Huhns, M. N. A Tool for Organizing Web Information, *Computer*, Vol. 30, No. 6, June 1997, pp. 80-83.
17. Mukherjea, S. Foley, J. D. Hudson, S. Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views. *Computer Human Interaction 95 Proceedings*.
18. Nanard, J. and Nanard, M. Hypertext Design Environments and the Hypertext Design Process. *Comm. ACM*, Aug. 1995, pp. 49-56.
19. Owen, C. B. Makedon, F. Frank, G. and Kenyon, M. ASML: Automatic site markup language. *Proceedings of Webnet'97 World Conference on the WWW, Internet, and Intranet*. Toronto, Canada, 1997.
20. Rossi, G. Schwabe, D. and Garrido, A. Design Reuse in Hypermedia Application Development. *Proceedings of Hypertext'97*, Southampton, UK, 1997.
21. Rumbaugh, J. Blaha, M. Premerlani, W. Eddy, F. and Lorensen, W. *Object Oriented Modeling and Design*. Prentice Hall Inc. 1991.
22. Schwabe, D. Rossi, G. The Object-Oriented Hypermedia Design Model. *Communications of the ACM*, Aug. 1995, pp. 45-46.
23. Schwabe, D. and Rossi, G. Abstraction, Composition and Lay-out Definition Mechanisms in OOHDM. *ACM Workshop on Effective Abstractions in Multimedia*, Nov. 4 1995.
24. Schwabe, D. Rossi, G. and Barbosa, S.D.J. Systematic Hypermedia Application Design with OOHDM, in *Proc. Hypertext 96*, ACM Press, New York, pp. 116-128.
25. Sjolín, M. Introduction to LINCKS, *Linux Journal*, vol. 1 (Mar. 1995.), pp. 26—30.
26. Ungar, D. and Smith, R.B. *Self: The Power of Simplicity*, OOPSLA '87 Proceedings, pp. 227-242, 1987.
27. Wiil, U.K. and Leggett, J.J. Hyperform: Using Extensibility to Develop Dynamic, Open and Distributed Hypertext Systems. *Proceedings of the ACM Conference on Hypertext (ECHT '92)*, Milano, Italy, 1992, pp. 251-261.
28. Wiil, U. K. and Leggett, J. The HyperDisco Approach to Open Hypermedia Systems. *Proceedings of the 7th ACM Conference on Hypertext*, pp. 140-148.
29. Wiil, U. K. and Leggett, J. Workspaces: The HyperDisco Approach to Internet Distribution. In *Hypertext '97 Proceedings*, (Southampton, England, April 1997), ACM Press.
30. Zheng, Y. and Pong, M-C. Using Statecharts to Model Hypertext. *Proceedings of the ACM European Conference on Hypertext*, Milano, December 1992.