# OOHDM-WEB: Rapid Prototyping of Hypermedia Applications in the WWW

*Daniel Schwabe*

e-mail: schwabe@inf.puc-rio.br

*Rita de Almeida Pontes*

e-mail: rita@inf.puc-rio.br

PUC-RioInf.MCC-08/98, March, 1998

## Abstract

This paper presents OOHDM-Web, an environment allowing rapid prototyping of hypermedia applications designed using OOHDM in the WWW. This environment allows direct mapping of navigation and interface constructs of OOHDM into a library of functions in the CGI scripting environment CGI-LUA, extended with the DB-LUA package. This environment allows implementation of hypermedia applications as CGI scripts that produce dynamically generated pages, whose contents are fed from a database and integrated with pre-defined templates.

**Keywords:** Hypermedia Authoring, WWW, Authoring Methodologies

## Resumo

Este artigo apresenta o ambiente OOHDM-Web, que permite a prototipagem rápida na WWW de aplicações hipermidia projetadas com o OOHDM. Este ambiente fornece um mapeamento direto de construções de navegação e de interface para uma biblioteca de funções no ambiente de programação CGI CGI-Lua, estendido com o pacote DB-Lua para acesso a bancos de dados. Este ambiente permite a implementação de aplicações hipermídia na forma de "scripts" CGI que produzem páginas geradas dinamicamente, cujo conteúdo é alimentado a partir de uma base de dados, integrados com gabaritos pré-definidos.

**Palavras-chave:** Autoria hipermidia, WWW, metodologias de autoria, modelagem.

---

Este trabalho foi parcialmente patrocinado pelo CNPq.

This work was partially sponsored by CNPq.

# OOHDM-WEB: Rapid Prototyping of Hypermedia Applications in the WWW

*Daniel Schwabe*[*] *and Rita de Almeida Pontes*

Dept. of Informatics, PUC-Rio
R. M. de S. Vicente, 225
Rio de Janeiro, RJ 22453-900, Brazil
E-mail: [schwabe,rita]@inf.puc-rio.br

## Abstract

This paper presents OOHDM-Web, an environment allowing rapid prototyping of hypermedia applications designed using OOHDM in the WWW. This environment allows direct mapping of navigation and interface constructs of OOHDM into a library of functions in the CGI scripting environment CGI-LUA, extended with the DB-LUA package. This environment allows implementation of hypermedia applications as CGI scripts that produce dynamically generated pages, whose contents are fed from a database and integrated with pre-defined templates.

### 1. INTRODUCTION

One of the prime implementation platforms for hypermedia applications nowadays is the WWW. In this case, a collection of documents and CGI scripts is made available to users, allowing both access and in many cases processing of information contained in these documents. This collection of pages usually shares a common coherent interface appearance and behavior, and may be physically located in one or more web servers. One example of such applications is the Amazon Books website (www.amazon.com).

However, much in the same way as in other platforms, there is no environment that facilitates the structured development of such applications. In spite of a number of design methodologies having been proposed and adopted [OOHDM [Schwabe 96, Schwabe 95], HDM [Garzotto 93], RMM [Isakowitz 95], Matilda [Lowe 95], designers are still forced to map the primitives in such methodologies to any given implementation platform.

The tools available in the market today may be classified in three main categories: page editors, web site editors, and web site building environments.

Page editors – Hot Metal (www.softquad.com), Hot Dog (www.sausage.com), Claris Homepage (www.claris.com, PageMill (www.adobe.com) – allow the construction of HTML pages, be it in WYSIWYG mode or not, with little or no notion of a website as a coherent collection of pages.

Web site editors – FrontPage (www.microsoft.com) CyberStudio (www.golive.com), in addition to editing HTML pages, allow the management of sets of pages, but not considered as being part of a consistent, homogeneous unifying design; pages are treated in the same way as files in a file system. A more structured view of a website is allowed by the tool NetObject's Fusion (www.netobjects.com), where a site is in fact regarded as a set of pages sharing a common visual style. Is also possible to have automatically inserted navigation bars that reflect the site's structure, but unfortunately this structure is take to be the hierarchical file structure storing the pages and not a logical one defined by the designer.

Web site building environments – StoryServer (www.vignette.com), Cold Fusion (www.allaire.com), CGILua [Hester 97], (http://www.tecgraf.puc-rio.br/scripts/cgilua/manuais/cgilua/cgilua.html) and DBLua (www.tecgraf.puc-rio.br/~cgilua) allow the construction of sites where documents are defined using templates which extend standard HTML with some proprietary set of tags, typically allowing including of statements in some custom defined or standard programming/scripting language. In such environments, documents are dynamically assembled at run-time by instantiating a template with data pulled off from a database. Documents can be created and maintained using the same mechanisms, employing another set of appropriately defined templates.

In Figure 1 we make a rough comparison between these tools.

[*] Work partially supported by CNPq.

| | flexibility | simplicity | portability (programs) | portability (data) | maintenance | efficiency | programming language |
|---|---|---|---|---|---|---|---|
| CGILUA | *** | *** | *** | *** | ** | ** | *** |
| Cold Fusion | ** | ** | * | *** | * | * | * |
| StoryServer | *** | ** | * | * | *** | *** | *** |

**Figure 1 - Comparison between web site building environments. A larger number of "*" indicates that the corresponding aspect is better developed.**

The major drawback of these environments is the fact that they lack higher level abstractions that allow the user to design applications without resorting to a node-and-link level of description of the whole system. On the other hand, they can be used as implementation environments for designs carried out using one of the previously mentioned methodologies. This paper presents how this has been achieved for applications designed employing the Object Oriented Hypermedia Design Method (OOHDM) [Schwabe96, 95], using the CGI Lua web site building environment. The reasons for this choice are discussed briefly in section 4.

## 2. THE OOHDM-WEB ENVIRONMENT

One of the foundations of the OOHDM approach is the separation of concerns in the process of hypermedia application design, embodied in four deferent phases: Conceptual Design, Navigation Design, Abstract Interface Design and Implementation. During the Conceptual Design phase the application domain is described, without concern for user profiles and tasks. In the Navigation Design phase, different user types and their corresponding tasks are used to derive the navigation objects (nodes, links and access structures) and how they are structured. In the Abstract Interface Design phase the navigation objects are mapped onto interface constructs, which will be implemented in the Implementation phase.

In many situations, especially when the interface will be implemented in the WWW, the Abstract Interface Design phase may not be cost effective, since it requires the definition of the interface behavior at a level of detail which is in most cases unwarranted. This is presently true for the interface semantics for most browsers when displaying documents in the current HTML standard and some of its dialects. In such cases, a rapid prototyping tool may be an effective substitute.

The OOHDM-Web environment addresses the Navigation, Abstract Interface and Implementation phases. It is assumed that the designer has already defined the navigation objects that will make up the application, and will define the interface appearance using templates written in an extended version of HTML. The elements that must be defined are:

1. Navigation Classes and their instances;
2. Navigation Contexts
3. Access structures
4. Interface Templates

Once these elements have been defined, the website may be deployed, using OOHDM-Web functions together with the CGI Lua scripting environment, to serve dynamically generated pages that follow the specified OOHDM design. Figure 2 shows this setup

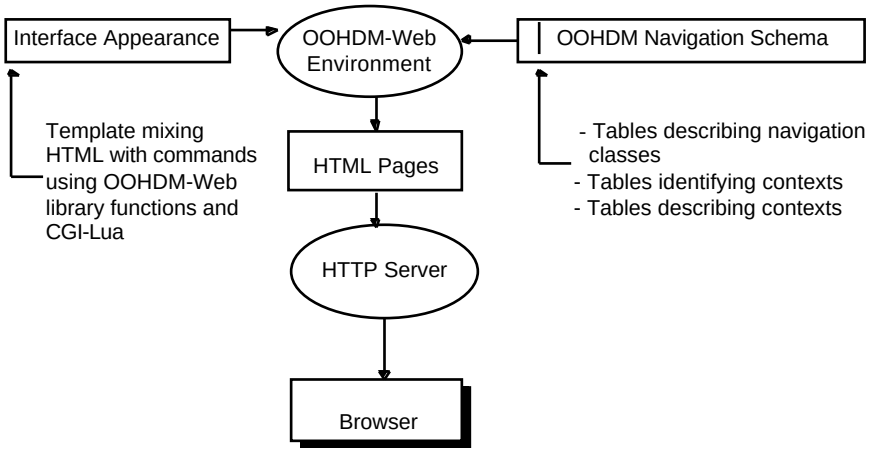


Figure 2 **- The structure of the OOHDM-Web Environment**

OOHDM-Web uses the CGI Lua scripting environment, which is written in the scripting language LUA [Ierusalimshy 96]. CGI Lua maintains a number of global variables that may be accessed by scripts, and provides several functions to ease page generation; for example, there are functions that take care of passing state information between scripts, easing the burden on the designer.

Pages in CGI Lua may be of two kinds: LUA (documents with .lua extension) and mixed HTML (documents with .html or .htm extensions). The first kind simply executes the statements in LUA, which must generate a valid HTML document, just as with other CGI scripting languages. The second kind takes HTML statements interspersed with CGI Lua statements, which are surrounded by "$|" marks or by "<!--$$" and "$$--!>".Whenever the CGI Lua interpreter runs into these statements, they are executed, and must generate valid HTML code, besides possibly setting values of global variables.

In addition to CGI Lua, the Lua environment includes another library, called DBLua. This library allows access to any ODBC compliant database, and is used extensively to allow OOHDM-Web functions to access the tables describing the hypermedia application.

Even though OOHDM is object oriented, it does not assume the implementation will be carried out in an OO environment. In particular, OOHDM-Web assumes a relational database system will be used to store the information represented in the various schemas. In the following sections, it is shown how each of the element types enumerated above is represented in the database.

### 2.1 A Simple Example Application

To illustrate the concepts involved in OOHDM-Web, we give a small example, describing part of a website for a research laboratory. The Navigation Classes are described in Figure 3.

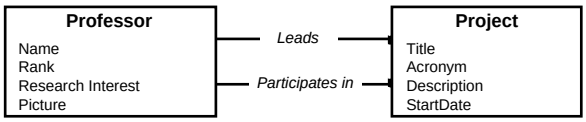


**Figure 3 - Navigational Schema for part of a website of a research lab.**

The Context Diagram for this application is given in Figure 4.

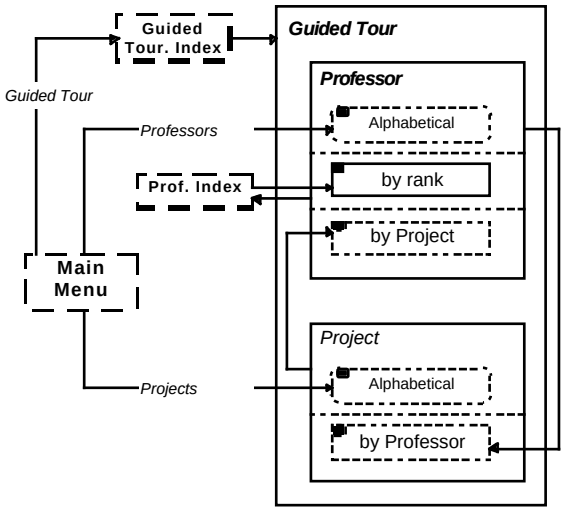


**Figure 4 - Context Diagram for the example application.**

### 2.2 Navigation Classes

Each navigation class and its instances is implemented as a table with the following structure

Class CL

| # objectID | Unique key identifying the object instance |
|---|---|
| attribute 1 | Value of attribute 1 |
| attribute 2 | Value of attribute 2 |
| ... | ... |
| attribute n | Value of attribute n |

For tthe example, the corresponding tables are

ClassProfessor

| #ObjectId | ProfName | ProfImg | Rank | ResearchInterest |
|---|---|---|---|---|
| lfgs | Luis F. G. Soares | <IMG src= "lfgs.gif"> | associate | High speed networks... |
| schwabe | Daniel Schwabe | <IMG src= "daniel.gif"> | associate | Advanced Information Systems ... |
| noemi | Noemi Rodriguez | <IMG src= "noemi.gif"> | assistant | Distributed systems ... |

ClassProject

| #ObjectID | Acronym | ProjTitle | ProjDescr | ProjDate |
|---|---|---|---|---|
| redpuc | REDPUC | Project REDPUC | This project developed the first Brazilian LAN... | 1980 |
| oohdm | OOHDM | Projeto OOHDM | OOHDM is an authoring methodology for hypermedia applications... | 1994 |
| hyperprop | HYPERPROP | Hyperprop | The Hyperprop architecture allows the representation of hypermedia documents... | 1992 |

The name of the table, as well as the names of the attribute fields are given by the designer. The only condition is that objectID be a key field of the table.

### 2.3 Navigation Contexts

To recap, navigational contexts are structured sets of navigational objects, formed according to several possible criteria; a navigation order may be also specified. A. There is a master table specifying the navigation contexts that are present in the application:

Context

| # ContextName | The name that identifies the context |
|---|---|
| Type | The type of context |

Contexts may be of five different types:

1. Class derived - Contexts of this type are formed by objects of some class, selected according to some criterion. For example, "all full professors" would include objects of class "Person" who are "full professors";

2.. Class derived group of contexts - Instead of specifying a single context, it is often more concise and convenient to specify groups of related contexts. In this case, the contexts in the group are formed according to some property of objects in the same class. For example, "professors by rank" would form a context group, in which each member of the group is a context formed by objects of class "Person" who are of a given "rank". There is one context in the group for each possible value of "rank".

3. Enumerated context - is formed by individually enumerating the objects that make it up.

4. Link derived - is a context formed by objects that participate in a relation. For example, "Students enrolled in "Hypermedia Authoring Seminar" is formed by objects of class "Person" who are "students" and are related to the object "Hypermedia Class Seminar" of class "Course".

5. Link derived group of contexts - Analogously to the class derived group, this is a set of contexts formed by objects connected by some type of link. For example, "Students by course" is a group

such that each of its members is a context formed by objects of class "Person" who is a "Student" and is linked to an object of class "Course" by a link of type "is enrolled in". There is one such context for each possible object in class "Course".

Since each context type requires different types of information, there are five different tables to represent them:

Type1 : Class derived context

| | |
|---|---|
| **#ContextName** | Name identifying the context |
| **ClassName** | Class to which the elements of the context belong |
| **Selection** | SQL query selecting the elements of the context |
| **Order** | Name of the attribute used to order the objects in the context |
| **TargetPage** | Name of the template page used to show the elements of the context |
| **NavigationType** | Type of navigation allowed in the context |

The valid navigation types are free, sequential, circular and index. In this last case, navigation within the context is only allowed from its index to an object and back; no paths are provided for navigating from object to object without first going through the index.

In the example, we have

| | |
|---|---|
| **#ContextName** | Prof_Alphabetical |
| **ClassName** | Professor |
| **Selection** | * |
| **Order** | ProfName |
| **TargetPage** | prof.html |
| **NavigationType** | Circular |

Type 2 : Class derived context group

| | |
|---|---|
| **#ContextName** | Name identifying the context |
| **ClassName** | Class to which the elements of the context belong |
| **ClassAtribute** | Name of attribute that will be used to group contexts |
| **NavigationType** | Type of navigation allowed in the context |
| **TargetPage** | Name of the template page used to show the elements of the context |
| **Order** | Name of the attribute used to order the objects in the context |

The elements of a group are contexts that are defined according to the possible values of an attribute of the class. For example, to group "Professors" by "Rank", the following table is defined:

Table Professor_Rank

| | |
|---|---|
| **#ContextName** | Prof_Rank |
| **ClassName** | Professor |
| **ClassAtribute** | ProfRank |
| **NavigationType** | Circular |
| **TargetPage** | prof.html |
| **Order** | ProfName |

Type 3 : Arbitrary Context

| | |
|---|---|
| **#ContextName** | Name identifying the context |
| **Def_Table** | Name of the table enumerating the elements of the context |
| **NavigationType** | Type of navigation allowed in the context |

Since the elements of an arbitrary context do not possess any rule of formation, they must be individually named. This is achieved by having an entry in the "Def_Table" for each enumerated element. The format of this table is as follows

| | |
|---|---|
| **# ClassName** | Class to which the elements of the context belong |
| **# ElementId** | A key identifying the element |
| **Anchor** | Name of anchor to be used in the index to the context |
| **TargetPage** | Name of the template page used to show the elements of the |

| | |
|---|---|
| | context |

For example, a simple guided tour of the laboratory, listing the main projects and their respective leaders, would be defined as follows

Table Guided_Tour

| | |
|---|---|
| **#ContextName** | GuidedTour |
| **Def_Table** | Guided_Tour_Def |
| **NavigationType** | Sequential |

Table Guided_Tour_Def

| #ClassName | # ElementId | Anchor | TargetPage |
|---|---|---|---|
| Professor | lfgs | ProfName | prof.html |
| Project | HYPERPROP | ProjTitle | proj.html |
| Professor | schwabe | ProfName | prof.html |
| Project | OOHDM | ProjTitle | proj.html |

Type 4 : Link derived context

| | |
|---|---|
| **#ContextName** | Name identifying the context |
| **RelationTable** | Name of table representing the links |
| **SourceClass** | Class to which the source element of the link belong |
| **DestinationClass** | Class to which the destination element of the link belong |
| **NavigationType** | Type of navigation allowed in the context |
| **TargetPage** | Name of the template page used to show the elements of the context |

For example, the context "Professors by Project" is represented by

| | |
|---|---|
| **#ContextName** | Prof_Proj |
| **RelationTable** | Professor_Project |
| **SourceClass** | Project |
| **DestinationClass** | Professor |
| **NavigationType** | Sequential |
| **TargetPage** | prof.html |

The table representing the link is defined as follows

| | |
|---|---|
| **#Source Element** | Key identifying the source element of the link |
| **#Destination Element** | Key identifying the destination element of the link |

For example, the relation between "Professors" and "Projects" is represented by

Table Professor_Project

| #Source Element | #Destination Element |
|---|---|
| lfgs | Hyperprop |
| schwabe | OOHDM |
| noemi | Hyperprop |

Type 5 - Link derived context group

| | |
|---|---|
| **#Context Name** | Name identifying the context |
| **Relation Table** | Name of table representing the links |
| **Source Class** | Class to which the source elements of the link belong |
| **Destination Class** | Class to which the destination element of the link belong |
| **DestTarget Page** | Name of the template page used to show the elements of the context |
| **Navigation Type** | Type of navigation allowed in the context |
| **SourceTarget Page** | Name of the template page used to show the elements that are the source of the link |
| **Source Context** | Name of the context that is formed by the sources of the links |

In the example, we have

| | |
|---|---|
| **#Context Name** | Proj_Prof |
| **Relation Table** | Professor_Project |
| **Source Class** | Professor |
| **Destination Class** | Project |
| **DestTarget Page** | proj.html |
| **Navigation Type** | circular |
| **SourceTarget Page** | prof.html |
| **Source Context** | Prof_Alphabetical |

In OOHDM it is possible to extend a class when its objects are accessed within a certain context. This is achieved via a mechanism called "InContext Class", which functions as a decorator to the original class, adding new attributes to the decorated class. This is implemented by having two tables, one that indicates which navigation classes have extensions via InContext classes, and the second one to specify the particular extensions for each

Identification of InContext Classes

| | |
|---|---|
| **#Class** | Name of class to be extended |
| **ContextTable** | Name of table containing the additional attributes for a given context |

Attribute specification for InContext Classes

| | |
|---|---|
| **# objectID** | Unique key identifying the object instance |
| **#ContextName** | Name identifying the context |
| **attribute 1** | Value of attribute 1 for element "objectID" in the context "#ContextName" |
| **attribute 2** | Value of attribute 2 for element "objectID" in the context "#ContextName" |
| **...** | ... |
| **attribute n** | Value of attribute n for element "objectID" in the context "#ContextName" |

### 2.4 Functions in the OOHDM-Web Library

The designer builds page templates that mix HTML tags with special commands that are interpreted by the CGI Lua scripting environment. These commands can be, in general, valid statements in the Lua programming language. In addition, a set of functions is provided to allow the designer to make reference to OOHDM navigation primitives as defined in the tables above, in such a way that (s)he does not have to know the actual internal representation used to store these elements. There are four categories of functions: index generation; page layout; context navigation; and miscellaneous, which are described next.

#### 2.4.1 Index Generation Functions

These functions build HTML fragments that, when interpreted by the HTTP server together with the CGI Lua environment, will generate an index to a context. The index is such that, when the users selects an entry, the corresponding page is generated. To aid in this generation process the CGI Lua environment will make available to the script that generate the page three global variables, that may be accessed within the LUA code embedded in the page: *cgi.context* contains the name of the context in which the page is being accessed; *cgi.group* contains the name of the group, if applicable; and *cgi.inst* contains the objectID of the selected instance in the context.

For class derived contexts, the function is

> Index{ContextName, Anchor, Formatting Function}.

In this case, "Anchor" is a class attribute to be used as an anchor for each object in the context. "Formatting Function" is a Lua function call that should result in valid HTML code to exhibit the index. The designer may use the formatting functions provided in the OOHDM-Web environment described later on.

Using the example, for the class derived context "Professors_Alphabetical" containing all professors in alphabetical order, the call

> Index{context="Professors_Alphabetical" , anchor="ProfName", function= 'Horizontal( separator = <IMG WIDTH="4" HEIGHT="4" ALIGN="TOP" SRC="IMG/DOT.GIF">)'}

would produce a horizontal list of professor names, separated by small squares, where the "ProfName" is used as an anchor to point to a page that would show information about the professor. The function "Horizontal" is described later.

For class derived context groups, the function is

> Index{ContextName, Anchor, Group, Formatting Function}.

For example, for the group called "Prof_Rank" the call

> Index{ context="Prof_Rank", anchor="ProfRank", group= "Prof_Rank", function= 'Horizontal_Tab(col = 2; par_table = BORDER=1)' }

would produce a 2-column table with the possible ranks; for each rank, the anchor would point to the first professor of that rank.

For arbitrary contexts the function is

> Index{ContextName, Formatting Function}

In the example, the call

> Index{ context="Guided_Tour", function= 'Vertical_Tab(col = 2; par_table = BORDER=1)' }

generates the guided tour index as a vertical table with 2 rows.

For link derived contexts the call is

> Index{ContextName, SourceAnchor, Formatting Function}.

where "SourceAnchor" is the anchor of the source of the link. For example, the call

> Index {context = 'Profs_Proj', SourceAnchor = 'ProjName', function = 'Vertical()'}

would generate a vertical table whose elements are the project names, pointing to the first professor that participates in the corresponding project.

For link derived context groups, the function is

> Index{ ContextName, Group, DestAnchor, SourceAnchor, Formatting Function}. or
>
> Index{ ContextName, Group, DestAnchor, SourceAttribute, Formatting Function}.

The difference between the two calls is whether the index will also serve as an index for the source elements of the links (first option) or simply as a grouping for the destination elements (second option). For example, the call

> Index {context ='Proj_Prof', group = 'associate' , DestAnchor = 'ProjName', SourceAnchor='ProfName', function = 'Horizontal_Tab(col = 2; par_table = BORDER=1)'}

would produce an index of "Projects" lead by "Professors" of rank "Associate" organized as a 2-column horizontal table. In this index, for each "Professor". This is illustrated in Figure 5.

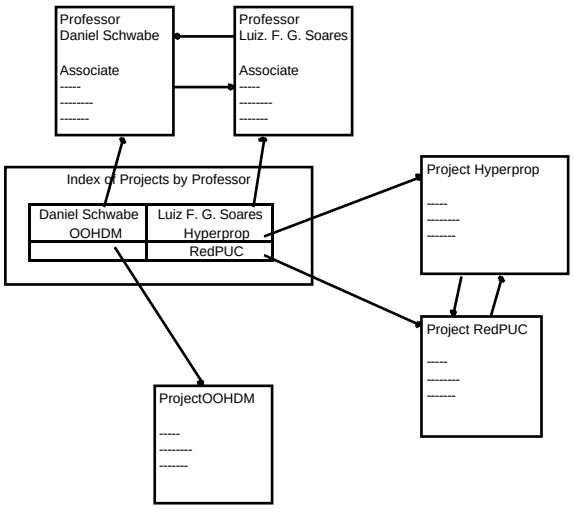


**Figure 5 - Example of index generated by the Index function for link derived context group. The other pages are part of other contexts and are included to ease comprehension.**

### 2.4.2 Formatting Functions

As already mentioned in the previous sub-section, OOHDM-Web provides functions to help position dynamically generated elements on a page. The functions are:

Horizontal(separator) -

> generates a list of elements arranged horizontally, separated by the string "separator"

Vertical() -

> generates a list of elements arranged one per line

Horizontal_Tab(cols, par_table, par_rows, par_cols, par_cell) -

> Generates a table with "cols" columns, and a variable number of rows, depending on the number of elements. "par_table" is a string passed as parameter to the <TABLE> tag; "par_rows" is the same for the <TR> tag; "par_cols" is the same for the <TD> tag; and "par_cell" is the same for the table cell.

Vertical_Tab(rows, par_table, par_rows, par_cols, par_cell) -

> Generates a table with "rows" rows, and a variable number of columns, depending on the number of elements. Elements are filled top to bottom, column by column. "par_table" is a string passed as parameter to the <TABLE> tag; "par_rows" is the same for the <TR> tag; "par_cols" is the same for the <TD> tag; and "par_cell" is the same for the table cell.

### 2.4.3 Navigation Functions

These functions generate anchor for links to be included in the page which allow navigation within contexts. The functions are:

NextLink(html_anchor, context) or NextLink(atr_anchor, context) -

> generates an anchor for a link to the next element in the context. This anchor may be an HTML snippet (first option) or an attribute of the class the elements belong to (second option)

PrevLink(html_anchor, context) or PrevLink(atr_anchor, context) -

> same as NextLink, but for the previous element in the context.

FirstLink(html_anchor, context) or FirstLink(atr_anchor, context) -

> same as NextLink, but for the first

element in the context.

LastLink(html_anchor, context) or LastLink(atr_anchor, context) -

same as NextLink, but for the last element in the context.

AllLinks(html_anchor, formatting function, context) or AllLinks(atr_anchor, formatting function, context) -

generates links to all elements of the context, using the formatting function to arrange elements on the page.

### 2.4.4 Miscellaneous Functions

These are functions to help manipulate objects. The are:

Attrib(Class, ObjectID, Attribute, Context)

returns the value of attribute "Attribute" of object "ObjectID" of class "Class" in context "Context").

Repeat(String, #times)

Generates a string made or "#times" repetitions of "String".

## 3. AN EXAMPLE

To illustrate how the actual templates are built, we show in this section portions of the Telemedia Laboratory website, which is similar to the example shown in the previous sections; the generalizations from one to the other should be obvious. For example, instead of "Professor", this site uses "Researcher".

In Figure 6 an index to the "Researcher in Alphabetical Order" context (analogous to "Professor in Alphabetical Order") is shown. The list is actually generated by a function call; in Figure 7 the uninterpreted template is shown, with the function call spelled out.

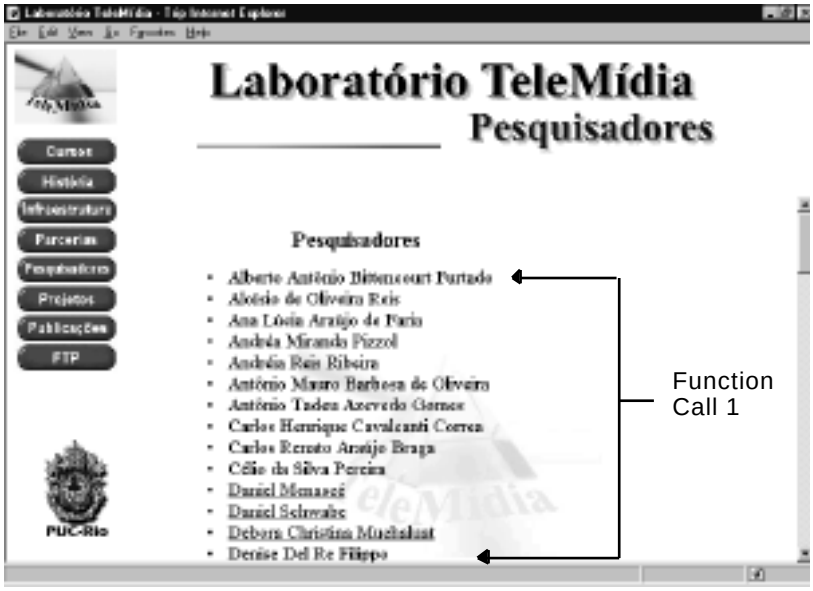


**Figure 6 - An index to "Researchers in Alphabetical Order" context**

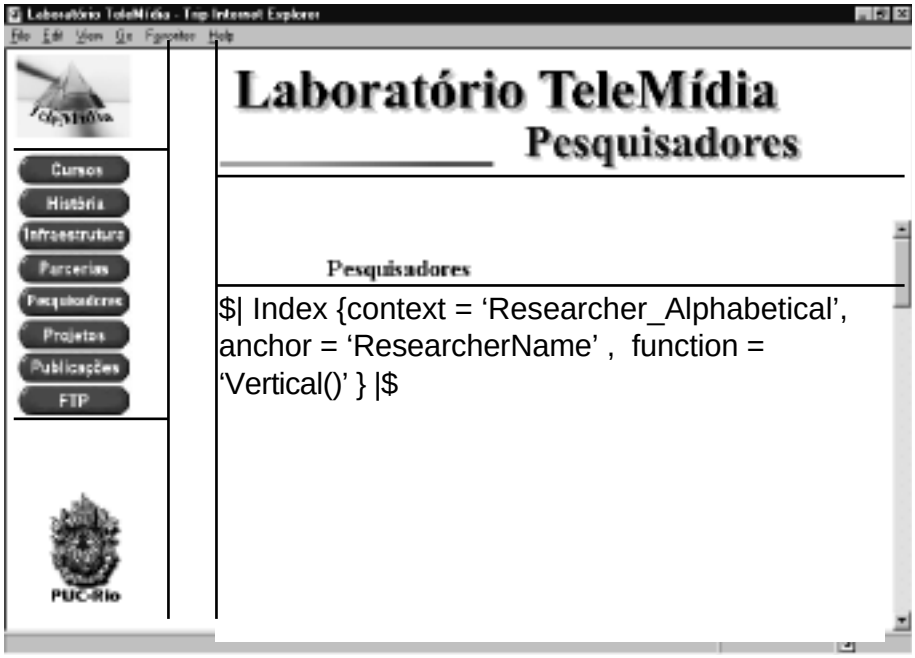


**Figure 7 - Uninterpreted template for the index shown in Figure 6. "Function Call 1" in that figure is shown in full in this template.**

If the user chooses the anchor "Daniel Schwabe", (s)he will navigate to the following page, shown in Figure 8. The corresponding uninterpreted template is shown in Figure 9


**Figure 8 - A page showing information about a researcher in the Lab.**

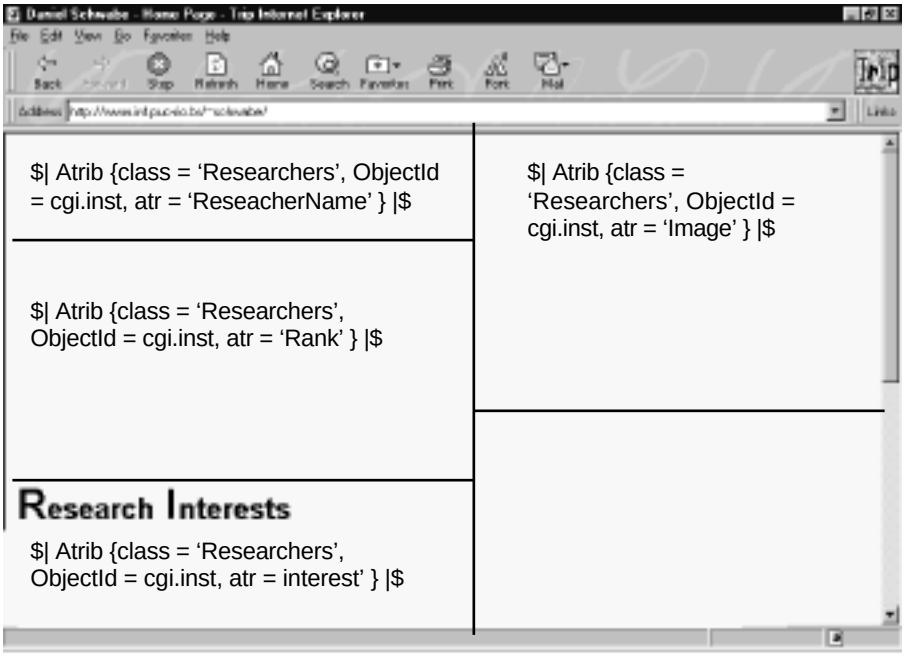


**Figure 9 - Uninterpreted template corresponding to the page in Figure 8.**

In Figure 10 , a page of a project within the "Project by Researcher" context is shown, and the respective uninterpreted template is shown in Figure 11.

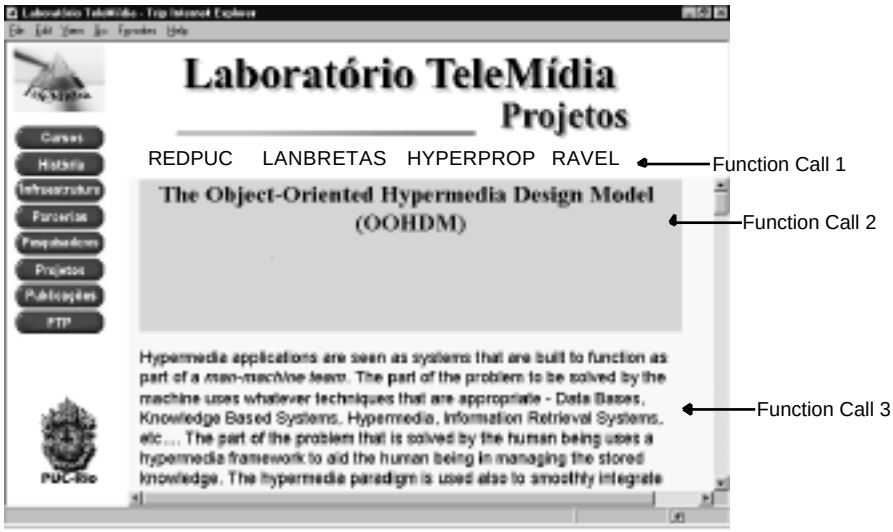


**Figure 10 - A page describing a project within the context "Projects by Researcher".**

It is interesting to notice how the "AllLinks" navigation function is used to generate an index to all projects in the context. The use of OOHDM-Web global variables, such as "cgi.context" and "cgi.inst" should also be noticed; their value is set in the link that is generated (using the OOHDM-Web Index function) in the index page (not shown) that leads to this page.

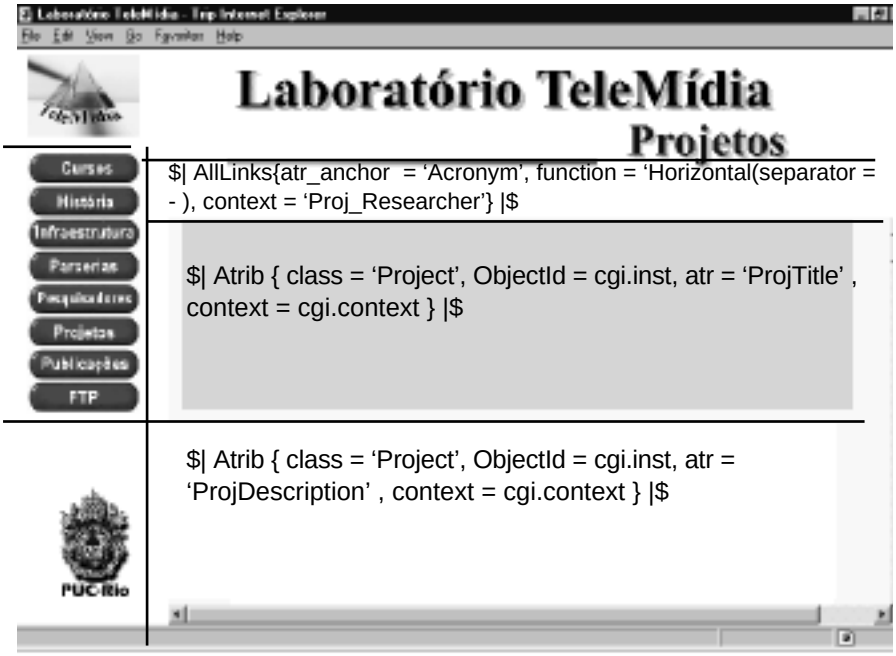


**Figure 11 - The uninterpreted template corresponding to the page shown in Figure 10. Notice the use of "AllLinks" OOHDM-Web function, and the OOHDM-Web variables "cgi.context" and "cgi.inst".**

## 4. CONCLUSIONS

We have described the OOHDM-Web environment for implementing web-based hypermedia applications. This environment is currently being used in a number of real-world complex websites, some on the Internet and some on private intranets.

As discussed in section 1, there are a number of commercial tools comparable to the CGI Lua+DB Lua environment, notably Cold Fusion and StoryServer, and OOHDM-Web environment could be implemented in any one of them. There are several reasons why have chosen CGI Lua, the foremost being portability - CGI Lua is available for the largest number of operating systems. A second important reason is cost; CGI Lua is freely available. A third reason is the fact that CGI Lua, Lua and DB Lua have been developed in another lab in our department, and it was very convenient to have the developers close at hand during implementation.

In terms of runtime efficiency, we have noticed that the slower part of the system tends to be the database interface. This is due to the way CGI scripts access databases, since a new transaction must be opened for each access. We are investing ways to optimize this interface.

In terms of continuing research, our current efforts are in the direction of providing a richer set of formatting functions, following the idea of design patterns [Rossi 97]. Another direction being pursued is the integration of processing functions (as opposed to pure browsing) in classes, to allow operations such as dynamic creation of instances, computations over dynamically defined contexts, etc...In addition, we are developing a web-based environment in which OOHDM designs can be developed, with a standard graphical interface.

## 5. REFERENCES

[Garzotto93] F. Garzotto, D. Schwabe and P. Paolini: "HDM - A Model Based Approach to Hypermedia Application Design". ACM Transactions on information Systems, 11 (1), Jan. 1993, pp. 1-26.

[Hester 97] A.M. Hester; R.C.Borges; R. Ierusalimshy; "CGILua: A Multi-Paradigmatic Tool for Creating Dynamic WWW Pages", Proceedings of the XI Brazilian Software Engineering Symposium (SBES'97) pp.347-360, Fortaleza, Brasil, 1997 (available at http://www.tecgraf.puc-rio.br/~anna/cgilua/ cgilua.ps.gz)

[Ierusalimshy 96] R. Ierusalimschy, L. H. de Figueiredo and W. Celes, "Lua - an extensible extension language",

Software: Practice & Experience 26 #6 (1996) 635-652. (see also http://www.tecgraf.puc-rio.br/lua/).

[Isakowitz95] T. Isakowitz, E. Stohr and P. Balasubramanian. “RMM: A Methodology for Structured Hypermedia Design”. Communications of the ACM 38 (8), 1995, pp. 34-44.

[Rossi97] G. Rossi, D. Schwabe and A. Garrido: “Design Reuse in Hypermedia Applications Development” Proceedings of ACM International Conference on Hypertext (Hypertext’97), Southampton, April 7-11, 1997, ACM Press.

[Schwabe95] D. Schwabe and G. Rossi: "The Object-Oriented Hypermedia Design Model (OOHDM)", Comm ACM, August 1995.

[Schwabe96] Schwabe, G. Rossi and S. Barbosa: "Systematic Hypermedia Design with OOHDM". Proceedings of the ACM International Conference on Hypertext (Hypertext'96), Washington, March 1996.