Présentation des travaux en vue de l'obtention de

Doctorat de l'Université Toulouse 1

Spécialité : ***INFORMATIQUE***
par

Marco Antonio ALBA WINCKLER

StateWebCharts: a Formal Notation for Navigation Modelling of Web Applications

Soutenue le … devant la commission d'examen composée de :

J.	**VANDERDONCKT**	**Rapporteur**	*Professeur à l'Université Catholique de Louvain, Louvain-la-Neuve, BE*
D.	**SCAPIN**	**Rapporteur**	*Directeur de Recherche à l'INRIA Rocquencourt*
O.	**PASTOR**	**Rapporteur**	*Professeur à l' Universidad Politécnica de Valencia, Valencia, ES*
P.	**PALANQUE**	**Co-Directeur**	*Professeur à l'Université Paul Sabatier, Toulouse*
C.	**FARENC**	**Co-Directeur**	*Maître de Conférence à l'Université Paul Sabatier, Toulouse*
J.	**COUTAZ**	**Examinateur**	*Professeur à l'Université de Joseph Fourrier, Grenoble*
R.	**BASTIDE**	**Examinateur**	*Maître de Conférence à l'Université de Toulouse 1*

Laboratoire d'accueil: Institut de Recherche en Informatique de Toulouse

Acknowledgments

Contents

1. Overview

When Tim Berners-Lee presented in 1989 at the CERN laboratory his proposal of what it would be called World Wide Web (WWW or just Web), probably nobody could have anticipated the huge impact it would cause some few years later [1, 2]. The growth of the World Wide Web has had a significant impact on business, banking and finance, commerce and industry, education, government and entertainment sectors, as well as our working lives. Actually, many legacy systems are being migrated to the Web environment and commerce through the Internet is rapidly growing. A wide range of new, complex applications is emerging because of the popularity and the ubiquity of the Web itself, and the intrinsic nature of the Web. Also because its intrinsic nature and features that provide an information representation with interlinking support of any kind of content, easy access for end-users, and easy content creation [3].

The World Wide Web is the most successful distributed hypertext system ever built however it suffers from many deficiencies and problems concerning navigation. Cockburn and Jones [4, 5] identify three main origins of these navigation problems: the Web browser design, page description languages, and page design. Most Web browsers were not conceived following a user-centred design approach and as a consequence browsers are poorly understood by users who can become lost using navigation functions such as history lists and backtracking [6]. In addition, the need of standardisation for languages employed to create Web pages constrains the design. For example, the current standard HyperText Markup Language (HTML) does not support facilities such as typed links or bi-directional links that could be employed to improve navigation and usability. The third origin of navigation problems is due to inappropriate navigation design of applications. The hypertext interconnections in Web applications can be extremely complex and designers could benefit from tools and guidelines to support and assist them.

In spite of the apparent facility to build Web pages given by current visual environments, the development within the Web is complex. Web applications require regular maintenance in order to update page content, to follow a particular business workflow, to include new features for supporting new tasks and/or users, and so on. Besides, Web development is often performed by several people (from different backgrounds such as graphics designers, marketing, and editorial) at a time and thus in a parallel manner. Another important feature of a Web project is the time to delivery or to update that can be as short as few hours. Designers can change their design almost immediately, which makes Web development projects highly evolutionary in nature [7].

To deal with the complexity of Web development, modelling support is essential as it provides an abstract view of the application thus leaving details to later phases in the development process. By means of a formal description technique, models can help designers by decomposing complex applications in smaller manageable parts, increasing communication in the development team and reducing ambiguity. In addition, some models can be employed to predict problems prior to implementation saving time, money and effort.

Traditional methods and models successfully employed in development and hypertext systems and traditional software have been demonstrated insufficient to deal with Web design due to special Web features [8] [9]. Even if Web applications remain a piece of software, quite often, the development process involved has more in common with magazine publishing than with conventional software development. However, as more and more Web projects require some kind of integration with legacy systems, it seams clear that appropriate methods for Web development must combine traditional software design and the idiosyncrasies of Web design.

In recent years, some models for the development of Web applications have been proposed such as the OO-H method [10], the Object-Oriented Hypermedia Design Method

(OOHDM) [11] and the WebML approach [12]. Such models are, in some extension, adaptations from previous work on Hypermedia Systems, Object Oriented methodology and Formal Methods. Those models which have been influenced by Hypertext Systems and Object Oriented methodology, propose solutions based on the concept authoring-in-the-large; which means they provide abstract models describing the overall classes of information elements and navigation structures without much concern for implementation details [13]. Models based on such as an approach include high level constructs, which may increase productivity, but they often lead to ambiguous descriptions of requirements or impose a particular kind of implementation. These problems with informal approaches are quite well known and the most significant ones are [14]:

- To detect and to cope with ambiguous requirements;
- To separate abstract modelling from technological details of implementation;
- To go from design to implementation phases in a structured and reliable way;
- To cope with the complexity of the design, which increase fast with the size of the Web applications;
- To cope with modifiability, which is a critical point for web applications.

In contrast to informal approaches, those models originated from Formal Methods fields using notations based on Petri Nets [15] and StateCharts [16] [17] are able to provide precise and detailed information about the interface and the user interaction with the application. In addition, some aspects of the application are more difficult to grasp in an empirical analysis. For example, formal representation of navigation allows the simulation of user interaction with the system for verifying system properties with respect to the navigation. When such analysis is made at design time they inform designers of possible usability problems and act as predictive models.

Even though Formal Methods are promising for the development of Web applications, their potential to represent navigation has not been fully exploited. The current available modelling methods based on formal methods such as Hypercharts [18], Navigation Modelling by Statecharts [19], Automate-based navigation [20], Trellis [21], and Dialog Flow Notation [22], fail to provide scalable modelling for Web applications.

Formal notations are often argued to be hard to use by non expert users but appropriate tool support may ease the modelling activity. In addition, they can also be enriched with appropriate semantic elements thus increasing the level of abstraction, improving the expressiveness power of models and readability of large models.

In order to overcome limitations of informal models, this work proposes the StateWebCharts (SWC) notation for describing navigation over Web application and a tool supporting the construction of models based on that notation. SWC is a formal notation that extends Harel's StateCharts [23] by adding more necessary concepts to describe navigation in a Web context, such as appropriate semantics for states and transitions, including notions like dialog initiative control and client and transient activities. The character formal of StateCharts ensure the description non-ambiguous of requirements and the possibility to apply formal verifications and model checking of system properties.

The background to the work is presented in chapters 2 and 3. **Chapter 2** presents the current state-of-the-art in modelling methods for Web applications. Section 2.1 in chapter 2 starts with a discussion about the importance of model-based approach for the Web, followed by the section 2.1 which proposes a categorisation that is used to demonstrate the requirements for models. The most representative proposal for modelling Web applications is presented in section 2.3.

Chapter 3 provides the necessary background about hypertext and conceptual elements that compose navigation models. Our goal at this point is to discuss advanced link features that can be incorporated into navigation models to improve usability and/or enrich the semantics.

This chapter also shows at a glance some technologies for implementing advanced link features. Finally, this chapter discusses the support given by those modelling methods described in chapter 2 for specifying advanced link features through navigation models.

The major contribution of this work is presented in **chapter 4** which describes the StateWebCharts (SWC) notation, a formal notation based on StateCharts for modelling the navigation of Web applications. Section 4.1 discusses why we are using a formal notation to model navigation. Section 4.2 provides an informal description of SWC which is followed in section 4.3 by the review of the StateCharts and by the formal definition of SWC in section 4.4. The rest of the chapter present small examples showing how to use the SWC notation.

Chapter 5 proposes the set of steps, largely inspired from the *authoring-in-the-large* approach, to build successful navigation models with SWC notation.

In order to demonstrate the potential of our approach for navigation modelling, **chapter 6** provides a study case for a real Web application.

Chapter 7 presents a tool, namely SWCEditor, which has been developed as part of the present work, to support the construction of models based on SWC. We describe how SWCEditor features the edition, visualisation, simulation and evaluation of navigation models.

The last chapter describes the conclusions and perspectives of this work.

2. Modelling Web Applications

Summary

Besides simple Web Sites, many current Web Applications are complex systems supporting critical business processes and organisational workflow, integrating legacy information and database systems, and providing access to large collection of multimedia content. To deal with such complex applications a model-based approach is required.

Recently several models have been proposed to deal with design of Web applications. These models, which are issued from hypermedia and software engineering communities, have been influenced by several models such as the Entity-Relationship (ER) [24], the hypertext Dexter Reference Model [25] [26] and Unified Modelling Language (UML) [27].

This chapter presents the state-of-the-art in modelling methods for Web applications. Since navigation is one of the most critical aspects of Web Applications, most of the discussion presented hereafter focuses on how navigation modelling is supported by currently available models. The role of navigation modelling is presented as part of the development process and we discuss how it is related/to other aspects of Web design. This chapter is organised as follows:

2.1 What is Modelling in Web Applications?

2.2 Modelling Dimensions of Web Applications

2.3 On Existing Modelling Methods

2.4 Summary of Modelling Methods for Web Applications

2.5 Additional Related Work

2.6 Conclusion

2.1. WHAT IS MODELLING IN WEB APPLICATIONS?

"Web applications" is a widely-used and fuzzy term employed to assign many different kind of applications over the Web ranging from informational-centric *Web sites*, *service-oriented Web sites*, *data-intensive Web applications* and *Web information Systems (WIS)*. Such applications can be positioned according to two main axes shown in Figure 1[1]: a) the importance given to the underlying data structure; and b) the complexity of function supported by the application. Typically, *data-intensive Web applications* (e.g. digital libraries) will require more dynamic content generation based on an underlying database scheme than *Web sites* based on static documents. Similarly, *service-oriented Web sites* (e.g. chats, Web portals) support much more dynamic functions than informational *Web Sites*. Complex *Web Information Systems* (e.g. intranets) will equally require complex functions and database connectivity. In practice, this distinction is much more difficult and complex since many applications have to support static pages without any business logic, modules based on underlying data models and dynamic content generation. So modelling methods must take into account both aspects.

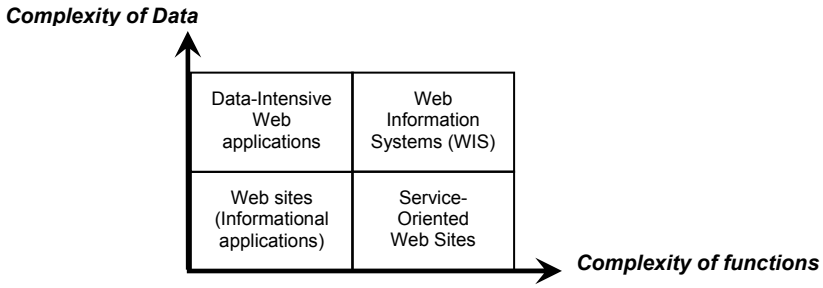


Figure 1 – Dimensions for Web applications.

1 This work does not focus on a particular category. The distinction among categories is presented here to illustrate the diversity of Web applications. So, except when explicitly mentioned the term Web application is employed here to assign indistinctly any kind of application developed over the Web.

Many current Web applications follow predefined business logic and complex transactional operations requiring integration with distributed databases and legacy systems. Notwithstanding, the early Web was born as a hypertext/hypermedia system and it still preserves many of its hypermedia influence. As a result of this hybrid heritage, the development of many Web applications needs to consider aspects typically found in document management systems (i.e. information architecture) and software engineering principles (i.e. functional architecture) [28].

To deal with complex Web applications several model-based approaches have been proposed in recent years, for example the Hypermedia Design Method (HDM) [13], the Object-Oriented Hypermedia Design Method (OOHDM) [11] and the WebML approach [12]. These methods systematise the development process and provide techniques for modelling Web applications. Since Web applications may have different styles, such as those previously mentioned data-intensive applications, services-oriented or informational Web sites (see Figure 1), many modelling methods focus on a particular kind of application in order to automate steps of the development process, increase system maintainability or reuse components. According to the modelling method employed, a particular emphasis either on information management or functional architecture aspects can also be observed. But, regardless of the emphasis given, the role of models in Web design is the same, that means to provide advanced functions to facilitate the understanding, specification, documentation, visualisation, communication, implementation, maintenance and evolution of Web applications.

Human factors aspects also have an important role in the development process of Web applications. People who develop these applications have widely varying backgrounds and approaches to development. Some of them come from non-technical areas such as graphic designers, marketing, and editorial. Thus, appropriate modelling methods are essential tools to increase the communication among such as a multidisciplinary team.

The aim of this chapter is to survey existing modelling methods to support the development of Web applications. In the section 2.2 we present a categorisation of modelling methods that will be used in the rest of this chapter to discuss existing modelling proposals for Web applications. In section 2.3 we provide a discussion about each dimension and a review the weakness of current modelling methods. Section 2.4 presents a brief summary of the most representative modelling methods for Web applications. Lastly, in section 2.5 we present final comments about this chapter.

2.2. MODELLING DIMENSIONS OF WEB APPLICATIONS

Several models are required to specify features of complex applications thus providing appropriate abstraction views for several designer roles (e.g., authors, graphical designers, programmers). Typically, modelling methods should cover the following three main dimensions (see Figure 2):

- *Modelling level:* covers the application domain, hypertext and presentation levels;
- *Modelling aspect:* is concerned with two commonly recognised aspects of models: structure and behaviour;
- *Development process:* reflects the fact that a modelling method is suitable to support a particular phase of development (i.e., requirement analysis, conceptual modelling, design, and implementation).

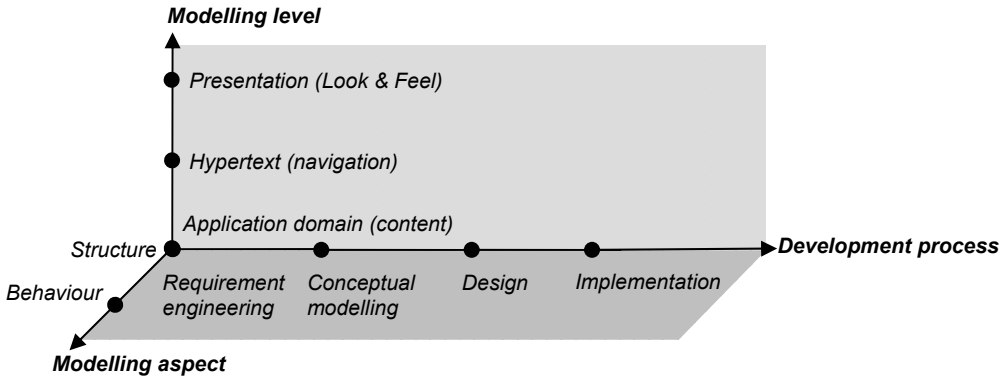


Figure 2 – Modelling dimensions for Web applications.

2.2.1. Modelling level

Current system models are better understood as a set of specific models, which are based on different views of an application. The most important views for Web application modelling are: application domain, presentation, and hypertext levels [29]. These three levels and corresponding mappings in-between are presented in Figure 3.

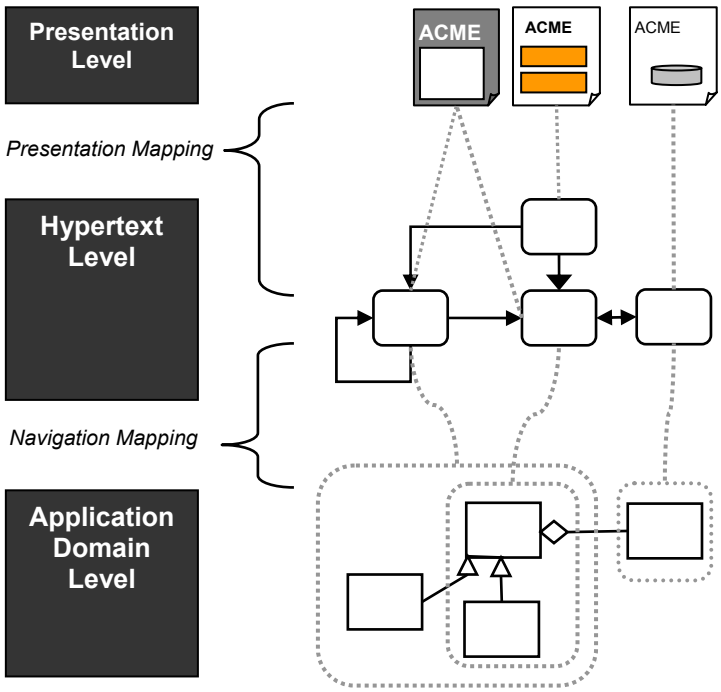


Figure 3 – Modelling levels of Web applications.

The *application domain*, *presentation* and *hypertext* levels can be compared to the Model-View-Controller pattern in an Object Oriented paradigm [30]. Such a three-tier has also been argued for hypermedia application, notably, by the Dexter Model Dexter Model [25] [26], which present three equivalent layers named storage, within-component and run-time layer, respectively.

The *application domain* (also know as content) level represents the underlying (or domain-dependent) data-model, which is often managed by a database system. In hypermedia applications, the application domain model can also serve as a model for indexing content. The content is considered as an instance (or a set of instances) of a particular concept in the application domain. Tasks models and scenarios, which may be used as the basis for modelling the navigation and the interaction between a user and the system, are typically defined at the application domain level.

The *hypertext* level (also called navigation level) denotes the logical composition of Web pages and the navigation structure linking the information. A navigation model should create appropriate views of information by grouping entities on the underlying data model and provide navigable relationships in-between according to the needs.

Navigation can be viewed from two different points of view: browsing and querying [31]. Technically they are equivalent. The difference consists only in who specifies the requests over the hyperbase: browsing is specified by a designer while querying is specified by a user.

The *presentation* level (often called Look & Feel) is concerned with the graphical presentation. It corresponds to the definition of graphical aspects of information on the screen such as page layout, graphical attributes for objects and dynamic behaviours affording user interaction.

2.2.2. Modelling aspects

The aspect *structure* refers to static properties provided by the model while the *behaviour* defines how the state of an application can change over time. The structural model describes the structure of a system (e.g., subsystems, packages) and structural relationships between them (e.g., inheritance, dependency and associations). Behavioural models describe the reactions of a system to external or internal events such as collaborations or messages passing in the system or the system's reactions to user interactions.

Structure and behavioural modelling aspects are orthogonal to the dimension modelling level. At the application domain level, the structural aspect represents content/concepts and their relationships by the means of abstraction mechanisms such as classification, generalisation and aggregation. The behavioural aspect concerns the underlining business logic associated to domain-dependent application. Behavioural models of an application domain mostly express activities, often by means of activity diagrams or flow charts.

At the hypertext level, the structural aspect represents the composition of Web pages into contexts and navigation in-between (static relationships). Contexts are structured views of the information according to a specific navigation purpose [13]. Structural models for navigation can be (partially) derived from application domain models by creating appropriate contexts over application domain concepts and transforming domain-dependent relationships into links. The behavioural aspect at the hypertext level concerns the operational semantic (i.e., when a particular event raises a transition creating dynamic links at run time). Behavioural specifications of navigation have been mainly based on models such as StateCharts as in [17] or Petri nets as in [15].

At the presentation level, structure refers to the graphical representation of the information (e.g., page layout). Behavioural models at the presentation level cover dynamic characteristics such as synchronisation of layout configuration after navigation and reactions for user interface inputs.

2.2.3. Development process

Modelling methods may be useful at a specific phase of development as well as throughout the application life cycle. The development process dimension is considered to be orthogonal to the two previous ones since it intends to highlight which aspects (structure and behaviour) and which levels (content, hypertext and presentation) are taken into account by a modelling method at each phase of development.

The development process dimension presented in Figure 2 consists of four phases (requirement analysis, conceptual modelling, design, and implementation) which are the most commonly found phases in traditional software development. However, this dimension does not take into account any particular life cycle for the development of Web applications.

Due to the diversity of Web applications, many life cycles are suitable to attend project constraints. Currently, there is no consensus on which phases of development are required

neither which life cycle describes better the development process of Web applications. Despite this, the life cycle for Web development can be generalised as an iterative process. For example, Scapin et al. [32] propose an iterative design process for the Web which is made up of six phases as follow (see Figure 4):

1) Requirements engineering phase (i.e.; *requirement analysis*) identify the main goals of the stakeholders, context of use and requirements. Activities in this phase include the collection of content for future use;

2) Specification phase (i.e.; *conceptual modelling*) produces specifications from the context of use and requirements gathered in the previous phase. Detailed models are produced to formalise requirements such as user tasks with the application and site architecture, for instance.

3) Design phase (i.e. *design*) consists of the refinement of the specifications according to their content. At the end of this phase a navigation map and page templates are prepared. This phase produces detailed specification to guide the implementation of the application Web;

4) Development phase (i.e.; *implementation*) corresponds to the physical construction of the Web application, production of the HTML pages and the eventual integration of tools for visualising media (e.g., sound, video). At the end of this phase, all the pages have content, links and graphic elements incorporated, the application is delivered;

5) Site usage and Evaluation phase intended to evaluate advanced prototypes with end-users. The product of previous phases is checked with respect to the requirements and the context identified in the first phase. Usability evaluation such as user testing or inspection of guidelines is the main activities in this phase;

6) Maintenance phase may have a long duration as it is in charge of gathering new information and planning modifications that have been requested from the use and evaluation phase.

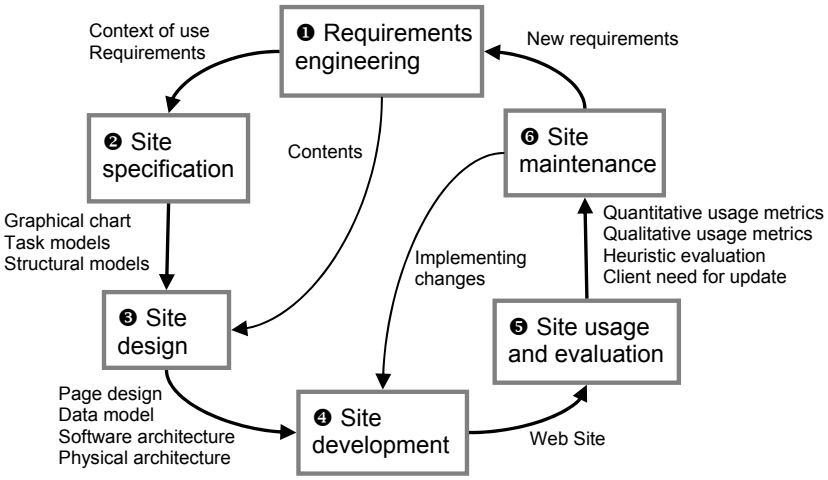


Figure 4 – Life Cycle for Web applications development.

This cyclic process does not take into account prototyping activities that are usually the core of the design process of web applications. For this reason two arrows have been added in the middle of the loop. In Figure 4, the arrow on left-hand side (in diagrams, its *contents*) represents the possible shortcut of the specification phase. Indeed, at the beginning of the design of the web application the designer may start immediately the design of the site in order to have precise information to exchange and discuss with the stakeholders. The arrow on the right-hand side (*implementation changes*) represents a possible shortcut for increasing development rapidity and taking into account in a more central manner the usage and the evaluations. Indeed, the web application developer can directly modify the application after an evaluation without going through the various top-level phases of the design process.

2.3. ON EXISTING MODELLING METHODS

The modelling methods for the Web have been influenced by different communities including database systems (notably by the Entity-Relationship (ER) model [24]), hypertext/hypermedia systems (by the Dexter Reference Model [26] [33] [9]), Object-Oriented models (by both Object Modelling Technique (OMT) [34] and Unified Modelling Language (UML) [27]), and formal methods (such as Petri Nets [35] and StateCharts [23]).

Many modelling methods have more than one influence; for example, the HDM [13] was influenced by the Dexter and ER models while OOHDM [36] was influenced by OMT and HDM. Quite often, models employed for Web development are just extended versions of previous models employed in hypermedia systems, for example the HDM-Lite [37] is the Web-specific evolution of HDM model [13]. As consequence of such diverse influences these models deal differently with several aspects of the design process and modelling, especially navigation modelling. Figure 5[2] below presents the origins of the influences received by the modelling methods surveyed here.

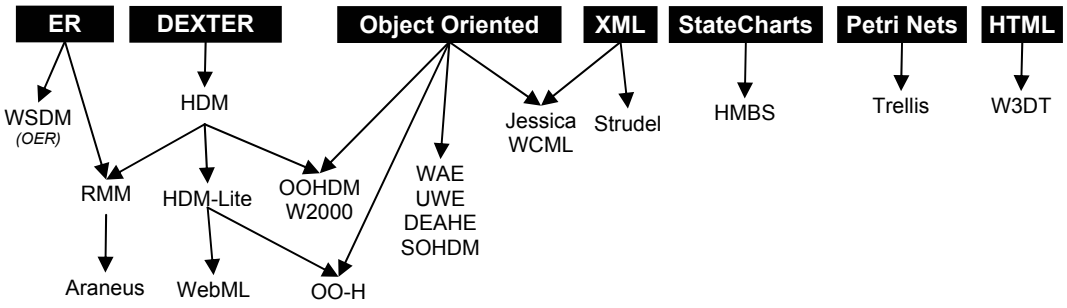


Figure 5 – Origins and influences of modelling methods for Web Applications.

In this section we discuss existing modelling methods for Web applications according to modelling requirements described hereafter. A list of methods surveyed in this work as well as individual descriptions of each method is presented in section 2.4.

2.3.1. Separation between levels

The separation between the levels is a clear requirement for models because it ensures easier maintenance, reuse and evolution of Web applications [38] [39]. For example, let us consider the implementation of different graphical presentations for a same application according to user preferences or browser specificities; without the independency between hypertext and interface levels one should be forced to duplicate the hypertext level for each different presentation.

The separation between levels can also help to reduce redundancy of information over Web pages. Very often, the same information can be found by several paths over a Web application and it also can be graphical presented in several ways. If separation between levels exists, normalisation techniques can be applied at the content level to avoid redundancy. By mapping information from the content level to the hypertext level, it is possible to have a unique reference for information elements thus facilitating updating and maintenance. Despite of this, some methods do not clearly separate these three levels.

Table 1 summarises all models surveyed here. It shows the corresponding modelling support for structural and behavioural aspects at different levels. The symbol C* in Table 1 is

[2] The graph presented in Figure 5 does not take into account the nuances of such as influences may have over modelling methods. For instance, W3DT constructs are abstractions of elements or collections of elements present in the HTML language while HMBS and Trellis just give a particular semantic for StateCharts and PetriNets constructs to describe behavioural aspects of hypertexts.

employed to show that the modelling method does not provide separation between levels. For example, HDM and W3DT do not provide clear separation between structural aspects of the application domain and the navigation. HDM does not even provide clear separation between presentations and hypertext levels. Except HMBS and Trellis models, all approaches provide solutions for representing application domain.

Table 1 also shows that the hypertext level is covered by all models surveyed while the presentation level is largely neglected by six of nineteen models surveyed.

2.3.2. Modelling behavioural and structural aspects of navigation

Regarding the aspects covered by models, we can observe that behaviour is largely neglected, especially at hypertext and presentation level. Table 1 details the support given by modelling methods to behavioural and structural aspects of navigation.

Table 1 – Modelling support for the structure and behaviour aspects at different levels.

Modelling Method	Level					
	Application domain		Hypertext		Presentation	
	STRUCTURE	BEHAVIOUR	STRUCTURE	BEHAVIOUR	STRUCTURE	BEHAVIOUR
HDM	C*	N	C*	N	C*	C
RMM	F	N	F	N	F	N
HDM-Lite	F	N	F	N	F	N
Araneus	F	N	F	N	F	N
OOHDM	F	F	F	F	F	F
Strudel	F	N	C	N	F	N
W3DT	C*	C	C*	N	F	N
WebML	F	F	F	F	F	F
WAE	F	F	F	N	N	N
UWE	F	F	F	F	F	F
OO-H	F	F	F	F	F	F
W2000	F	F	F	F	N	N
DEAHE	F	F	F	F	N	N
WSDM	F	F	F	F	N	N
Jessica	F	N	C	N	F	N
WCML	F	N	F	N	F	N
SOHDM	F	F	C	N	F	N
XHMBS	N	N	F	C	F	N
Trellis	N	N	C	F	C	F

Legend:
N No support
C Cumbersome
F Full support
* No clear separation between levels

While some models are simply not able to support a given aspect, others do partially or in a cumbersome way (e.g., Strudel, Jessica only supports navigation modelling as a restrictive view of relationships in the underlying data model). SOHDM provide a cumbersome representation of navigation modelling since it requires two models for representing navigation issues: one to describe the composition of navigation units and another for the interrelationships, which is made by means of a matrix of links. Trellis does not clearly describe structural aspects such as the content of a node in both presentation and hypertext levels.

2.3.3. Notations employed by Web modelling methods

Many existing modelling methods employ different notations to represent similar concepts. For example, while RMM [40] and Araneus [41] rely on the Entity-Relationship [24] for modelling the static structure of the application domain, OO-H employs UML class diagrams. Other methods have chosen to extend traditional notations for describing the application domain such as the HDM [13] and WebML [12], both extending the ER model.

While most models rely on traditional conceptual models like ER and class diagrams (excepting Strudel and SOHDM) to deal with the application domain level, there is huge

diversity of notations employed to deal with the hypertext and presentation levels. For instance, WAE and UWE employ class diagrams, while W2000 employs object diagrams, DEHAE employs StateCharts, and SOHDM employs a so-called navigational link matrix.

Such diversity is not limited to graphical representation but is representative of the several approaches to modelling navigation and presentation issues. Table 2 summarises modelling notations employed by several methods at different levels.

Table 2 – Summary of notation employed by modelling methods at different levels.

Modelling Method	Level		
	Application domain	Hypertext	Presentation
HDM	custom ER	graph	HTML
RMM	ER	M-slices	-
HDM-Lite	custom ER	access schema	HTML templates
Araneus	ER	N-ER/ ADM-2	Style sheet templates
OOHDM**	class diagrams	navigation class schema	ADV-charts
Strudel	eXtensible Markup Language (XML)[3]	StrudQL	Style sheet templates
W3DT	custom ER	custom ER	Simplified HTML data model
WebML	ER	WebML navigation schema	Graphical style sheets
WAE**	class diagram, technology-based stereotypes	class diagrams, sequence and activities diagrams	-
UWE**	class diagram, UWE stereotypes	class diagram, UWE stereotypes	StateCharts, UWE stereotypes
OO-H**	class diagram	Navigational Access Diagram (NAD), OCL	Abstract Presentation Diagram (APD)
W2000**	class diagram, HDM stereotypes, StateCharts	custom object diagram, use cases, and custom sequence diagram	-
DEAHE*	UML class diagrams	StateCharts	-
WSDM	ORM notation	tracks diagram	-
Jessica**	class diagrams, Jessica language	class diagrams	Style sheet templates
WCML**	XML	XML	XML
SOHDM**	SACs, CRCs, CRDs, OO views and ASN	navigational link matrix	page schema, user interface components
HMBS	-	StateCharts diagrams	schematic layout
Trellis	-	Petri nets	-

Legend: ** methods influenced by Object-Oriented methodology.

Modelling methods have slight differences considering what should be modelled at the presentation level. While HDM, Jessica and W3DT propose templates based on HTML to give a graphical representation for the content, OOHDM, OO-H and HMBS provide a notation that enable designers to specify visual layout properties such as the position of an object in a page. The methods RMM, WAE, W2000, DEAHE, WSDM and Trellis do not provide any support to deal with presentation concerns.

Models which were influenced by object-oriented approach and UML notation (identifiable by the ** marks in Table 2) may employ UML constructs to express different concepts. For example, WAE employs stereotypes to represent implementations and architectural issues while W2000 employs stereotypes to represent abstract concepts such as link types.

2.3.4. Design patterns support for navigation

In recent years many design patterns have been collected to deal with hypermedia and Web application design [42] [43] [44] [45]. These patterns have been organised around several dimensions whether they support human factors, hypermedia design and development,

[3] http://www.w3.org/XML/

application modelling and system architecture issues [46]. Most of them concerned with navigation issues of Web applications.

Both research and practical experience suggest that it is desirable to use patterns whenever it is possible [47]. In the case of the Web, design patterns concern solutions for common problems of Web applications; e.g. the shopping basket pattern [43]: "*keep track of user selections during navigation, making these selections persistent to process them when the user decides to. Decouple product selection from product consumption and/or processing*".

Despite of its guidance for the design, only a few design patterns are ready for use by means of modelling constructs. Exceptions of this are the pattern constructs such as: *indexes*, *showall, guided tours*, *indexed-guided tours, navigation chains*, and *menus*. However, these pattern constructs are only available on a few notations such as WebML and OO-H. Design patterns for the application domain or for presentation issues are not discussed by any of the methods surveyed in this chapter.

2.3.5. Support for bottom-up and top-down design

As discussed by Fraternali and Paolini [37], the development of Web applications may require both bottom-up and top-down approaches for the design. The bottom-up approach is a typical strategy employed in database systems from which the application domain level are defined first and them it is employed to derive the other levels accordingly. In opposition to the bottom-up approach, top-down design means that application domain level is derived from the other levels.

Bottom-up design is needed when an existing database should be brought to the Web, which is typical of integration with legacy systems. Top-down design is useful in case the content of already existing Web pages should be stored within a database; for example, integrating semi-structured sources of a document into a single Web application.

However, top-down and bottom-up approaches are seldom distinguished by most modelling methods. With a few exception (RMM, WAE, DEAHE), only bottom-up design is discussed by existing modelling methods. HBMS and Trellis as considered to support top-down design even though they only cover the hypertext level.

2.3.6. Supporting the development process

As mentioned before, there is no consensus about the development life cycle required for different Web applications. Many modelling methods require a specific development process while others do not even mention any specific phase in which the model is suitable to be employed.

Most of models for Web applications do not mention how to describe user requirements. The requirement analysis phase is often neglected or considered as completed when starting to model Web applications. OOHDM, WebML, OO-H, W2000, DEAHE, WSDM and SOHDM integrate requirement analysis in their development process. However, only WSDM considers both functional and navigational requirements. Other modelling methods discuss how to produce several concrete implementations from a single abstract model but they seldom describe which rules and criteria should be applied to represent user requirements.

Table 3 summarises the approaches for the design and the support according to several phases of the development process.

Table 3 – Support for development phases and design issues.

Modelling Methods	Development Process				Approach for design	
	Requirement analyses	Conceptual Modelling	Design	Implementation	Bottom-up	Top-down
HDM	-	X	X	X	X	-
RMM	-	X	X	X	X	X
HDM-Lite	-	X	X	X	X	-
Araneus	-	X	X	X	X	-
OOHDM	X	X	X	X	X	-
Strudel	-	X	X	X	X	-
W3DT	-	-	X	X	X	-
WebML	X	X	X	X	X	-
WAE	-	X	X	X	X	X
UWE	-	X	X	-	X	-
OO-H	X	X	X	X	X	-
W2000	X	X	X	X	X	-
DEAHE	X	X	X	X	X	X
WSDM	X	X	X	X	X	-
Jessica	-	-	X	X	X	-
WCML	-	-	X	X	X	-
SOHDM	X	X	X	X	X	-
HMBS	-	X	X	-	X	-
Trellis	-	X	X	-	X	-

Legend: X supported, **-** not supported.

2.4. SUMMARY OF MODELLING METHODS FOR WEB APPLICATIONS

Hereafter, we describe individually existing modelling methods suitable to model Web applications. These models are discussed briefly according to the modelling dimensions presented in the previous section.

2.4.1. Hypermedia Design Method (HDM)

HDM [13] is a former model-based approach that has introduced some new concepts to hypermedia modelling such as the notion of perspective (several presentations for the same content), and the identification of several categories of links (structural links, applications links and perspective links) that have inspired other modelling methods. It has been influenced by the ER [24] and the Dexter model [25].

The primitives of the modelling language include: entities (a conceptual object of the domain which is composed of a set of components) that are grouped in entity types (correspond a classes of objects, similar to ER entities); components (an abstraction for a hierarchy of units, which are the actual container of information); units (the smallest chunks of information that can be visualized, similar to the notion of hypertext nodes); perspectives (a syntactical device organize information according to a specific presentation); structural links (links connecting components belonging to the same entity); perspective links (links connecting together different units that correspond to the same component); application links (arbitrary, domain-dependent relationships connecting together components and entities in arbitrary patterns set by a author).

These primitives are used to create a schema definition and a set of instance definitions. A schema definition specifies a set of entity types and link types of the underlying data model. Instances are allowed to be inserted in the application only if they obey the constraints specified by the schema. This notion of schema and instances has been borrowed from ER model. Many

links can be derived automatically from a schema definition to the corresponding instance definition.

Even though a clear separation between hypertext level and presentation level is pursued by the authors, there is no evident separation between domain application and hypertext levels since all relationships are built over the schema definition. However, HDM further distinguishes a hyperbase layer modelling the application domain and a so-called access layer defining a set of collections that provide users with the patterns to access the hyperbase such as index and guided tours. Such as an access layer allows to implement some navigation patterns but the concepts for flexible mapping are not described.

Structural aspects are considered in both levels by means of the specific conceptual primitives mentioned before. Some behavioural aspects can be modelled at the presentation level by means of an embodied browsing semantics. By applying different browsing semantics it would be possible to deliver the same HDM specification under different versions, each one characterised by different visual features and dynamic behaviour [48].

Concerning the dimension phase, HDM focus on the two phases named authoring-in-the-large and authoring-in-the-small. The term authoring-in-the-large refers to the specification of the overall classes of information elements and navigation structures without much concern with implementation issues. The authoring-in-the-small refers to the development of the content of the nodes.

Tool support is provided by the JWeb toolkit [49] [50].

2.4.2. Relationship Management Methodology (RMM)

The RMM methodology [40] has been influenced by HDM an ER models. It provides a modelling language, called Relationship Management Data Model (RMDM), which is based upon ER models.

The RMDM is composed of E-R domain primitives (entities, attributes and relationships); six access primitives used to model navigation; and an original concept called *slices* that is used to describe how information should be presented. The so-called *slices* are conceptual elements that enable the grouping of attributes of one or more entities in order to create appropriate views for the information. For example, a person entity with attributes name, age, address and photograph may have a general slice element containing name and address, and another slice of personal information containing name, age and photo.

The design process with RMDM starts with an ER-like model where all many-many relationships are factored into two one-to-many relationships. Over this model, navigation and the so-called *slices* views are defined following iterative steps refining both bottom-up and top-down designs [51]. Navigation is supported in RMDM by the access primitives: unidirectional link, bidirectional link, grouping, conditional index, conditional guided tour, and conditional indexed guided tour. Uni and bi-directional links are used to specify access between *slices* of an entity. Indices, guided tours and groupings complete the set of navigation patterns supported by the model. An application diagram provides a global view of the presentation level by capturing all pages and hyperlinks in-between.

The original proposal was not able to model content of complex pages which requires combining information from different entities. This problem has been solved with an extension described in [52] by the definition of m-*slices* which permit the grouping of information for several entities.

RMM only provides support for structural views for the modelling. No behaviour or dynamics is represented by the model. Relationships between entities can be used to capture contextual information during navigation. As before mentioned, navigation patterns are supported in terms of index, menus (grouping access primitive) and guided-tours. Presentation modelling is described at conceptual phases of the design by means of *slices* and m-slices views. No effective presentation modelling is supported.

Considering the development process, the design starts over ER diagrams covering the early phase of development. The RMM design process is oriented to a highly structured application that does not evolve too much. Appropriate language and tools [53] allow the code generation of application from RMDM models.

A tool support is given by the RMCase [53] that supports hypermedia design, development activities and by the means of a specific language it is able to generate HTML code.

2.4.3. HDM-Lite / AutoWeb System

HDM-Lite [37] is an evolution of HDM model tailored to the development of Web Applications which was developed as part of the AutoWeb System project [29].

An application is specified in HDM-Lite by defining its structure, navigation and presentation, which are formalised by the hyperbase schema, access schema, and presentation schema, respectively. A HDM-Lite hyperbase schema is an extended ER model featuring entities structured into a tree of components, typed attributes of components, semantic links between entities and components and cardinality constraints.

The hypertext level is defined by the access schema which defines semantic relationships of the hyperbase schema. Navigation patterns are supported by means of modes: index, guided tour, indexed guided tour and showall (all objects are presented together). HDM also include simple rules to let the designer enable/disable the access selected portions of the application such as: global to the whole application, local to an entity, local to a component or local to a set of objects.

The presentation schema defines the presentation by means of style sheets which are textual descriptions (written in a SGML-like syntax) of the appearance of pages. More than one presentation schema can be attached to the same (hyperschema, access schema) pair but no mapping constraint is supplied.

Despite the fact HDM-Lite conceptually identifies the three levels (content, hypertext and presentation), the content level and hypertext level are modelled together in the structure schema. Behaviour aspects are neglected in all three levels.

Concerning the development phases, authors describe a process to generate application pages from conceptual specification. A set of tools are made available to guide the edition and specification of structure schema, access schema and presentation schema. These conceptual models are then internally transformed into several data schemas that are used to produce the pages by the AutoWeb system.

2.4.4. Araneus

Araneus is a design methodology for data-intensive Web applications composed of a set of steps and design transformations that lead from a conceptual specification to the actual implementation of the site [54] [41].

Araneus methodology relies on the Entity-Relationship model to describe the application domain level [55]. Based on the ER scheme, the designer shapes the target hypertext in terms of nodes and navigation paths, which are defined as views over the original ER scheme. The result is a new conceptual scheme, called N-ER scheme, which is a variant of ER scheme. Based on the N-ER scheme, an ADM-d scheme is specified giving details of the organisation of the Web pages, which attributes they will contain, how these will be nested and how pages should be linked together. The next steps consist of defining a layout by means of HTML templates, one template for each ADM-d scheme.

The whole process is supported by a tool called Homer. Once a designer has defined the format for publishing the application (HTML or XML/XSL), the script language (Java Server Pages or ASP), and which portions of the modelling should be implemented, the tool support

Homer can automatically generate the source code as needed. This tool also assists the whole design process from the ER definition to the implementation.

Even though Araneus supports clear separation between the application domain, hypertext and presentation, it only emphasises the application domain and hypertext levels. Only structural aspects are considered for content and hypertext design. The presentation level relies on the HTML-templates created by an authoring tool. Patterns are not supported for any of the three levels. Even if the hypertext level is specified independently from the application domain and presentation levels, Araneus does not provide primitives to optimise the modelling of complex navigation.

2.4.5. Object-Oriented Hypermedia Design Method (OOHDM)

OOHDM[4] was originally proposed as a method for the design of hypermedia applications. Many of the concepts presented by OOHDM [11] [36] are borrowed from of HDM model. The structure and behaviour aspects of an application are formalised by using an object-oriented approach; Object-Oriented Technique (OMT) [34] is used in early papers [36] and UML [27] in more recent publications [56].

OOHDM hypermedia applications are built in a four-step process supporting an incremental or prototype process model of hypermedia applications. It comprises the activities named conceptual design, navigation design, abstract interface design and implementation. During the conceptual design phase, a conceptual schema models the application domain in terms of OMT classes and relationships (aggregation and generalisation/specialisation hierarchies).

The hypertext level is covered during the navigation design phase, which is expressed in two schemas: the navigation class schema and the navigation context schema. The navigation class schema consists of a set of classes built as conceptual views of a conceptual schema. OOHDM recognises that objects the user navigates with are not the conceptual objects, but other kind of objects that are built from one or more conceptual objects.

The navigation is structured by using the navigation context schema. Each context definition includes: the elements it contains, the specification of its internal navigation structure, an entry point, access restrictions in terms of user classes and operations and associated access structures. There is no specific notation employed but it is referred to employ StateCharts [23].

OOHDM employ the Abstract Data View (ADV) [57] design approach for describing the user interface of the hypermedia application. AVD describes the layout of navigational objects and other interface objects such as menu bars and buttons. The behavioural aspect comprising the reactions to external events is also described by ADV-charts which are derived from StateCharts.

The object-oriented approach employed by OOHDM provides some facilities for reusing components and applying navigational patterns [38]. It also provides a clear separation between all levels, even though some mapping concerning, notably, dynamic aspects are not fully described.

Recently an environment, called OOHDM-Web, has been implemented to support the generation of database definition and templates used to transform OODHM specifications into Web applications. Such tools work on form/text-based interface. An authoring environment to support the graphical edition of OODHM models under development [58].

[4] http://www.telemidia.puc-rio.br/oohdm/oohdm.html

2.4.6. Strudel

Strudel[5] is a site-implementation tool, not an environment for Web-site design [59] [60]. It does not provide a graphical notation for modelling Web applications. However, it introduces some interesting concepts concerning the architecture and the approach employed for designing of Web applications.

Strudel provides a declarative (textual) query language (StruQL) for specifying a site's content and structure, and a simple template language for specifying a HTML representation. The fundamental principle of Strudel is the use of semistructured data models providing seamless access to several data sources such as relational database, text files or XML document repositories. The different data sources are transformed by appropriate tools in a data graph. By using the declarative language StruQL, designers can construct complex queries over Strudel's data graph and explicitly define rules for the dynamic construction of Web pages. A template can be associated to a collection of pages, thus providing a defined look and feel for the pages produced. Web page composition and navigation in-between are defined according to the rules described by StrudQL queries.

Strudel provides a clear separation of management of Web application's data and its specification (concerning content, structure and visual presentation of pages). However, it is only suitable for non-transactional Web sites where the content updating is not a critical operation of web sites. Despite the fact there is no graphical notation; Strudel provides a clear mapping between all levels. Behavioural specification is neglected in all levels and no design process is discussed.

2.4.7. World Wide Web Design Technique (W3DT)

The W3DT is a former approach for designing Web applications centred on the technological aspect of Web applications [61] [62]. The most remarkable characteristic is that it makes a direct association between modelling primitives and HTML elements. An extended version of W3DT, the eW3DT [63], includes some new objects but preserves the HTML-centred basis.

W3DT provides a modelling technique, which allows drawing a graphical representation of the Web site structure and computer-based environment (called Web Designer). This gives the possibility to generate a running prototype of the system including HTML pages and CGI scripts. W3DT diagrams represent both structure and navigation. The application domain primitives include page, index, form and menu. These primitives can be represented as a singleton or a template having a similar semantic and graphical presentation of Entity-Relationship models [24]. Links can be static (a standard navigation link) or dynamic (representing the execution of a CGI script).

Some behaviour can be represented at the conceptual level by means of the tool Web Designer, which enables to set some parameters used during the automatic generation of pages and scripts. Except by the representation of dynamic links, the behaviour modelling is considered at the hypertext level.

The presentation level is covered by means of layouts comparable to textual style sheets that can be defined and assigned to pages. No design process is discussed except for the steps required by the tool Web Designer used to produce diagrams and generate the implementation.

2.4.8. The Web Modelling Language (WebML)

WebML[6] is a visual notation for specifying the composition and navigation features of hypertext applications [12] [64]. WebML has been developed as part if the EU Esprit project W3I3 (Web-based Intelligent Information Insfrastructure) [65] whose goal it is to raise the level

[5] http://www.research.att.com/~mff/strudel/

[6] http://www.webml.org

of abstraction of the specification of a DataWeb application by enriching and refocusing the classical methods of database and hypertext design.

WebML is a descendent of HDM-Lite. It distinguishes five different models called Structural Model, Derivation Model, Composition Model, Navigation Model and Presentation Model. The structural model and the derivation model describe the content level by simply using a customised ER model and derivation rules respectively. The composition model describes, by means of site views, how the concepts of the structural level are mapped to web pages for certain groups of users and provide a default mapping for the case where there is only one simple view needed.

The navigation model describes the way in which associations within the structural model should be used for navigation thus capturing contextual navigation. Navigation patterns are supported by elements called navigation chains. A denotational semantic for navigation element using StateCharts is presented in [64].

The presentation model corresponding to the presentation level uses style sheets in order to define the layout of pages, whereby a default style sheet is provided for each page. Behavioural aspects are represented and considered at all levels. Finally, W3I3 does not propose a particular design process.

2.4.9. Web Applications Extension for UML (WAE)

The approach proposed by Conallen [8] [66] for modelling Web applications is slightly different from other modelling methods since it is driven by technology. WAE extends UML constructs such as stereotypes[7], tagged values[8] and constraints[9] to represent Web pages and other architectural elements. WAE stereotypes provide special icons that graphically identify them as special elements in model diagrams.

Web applications are modelled according to a particular architecture. For example each element represents whether a page is static on the client, a script on the client or a script on the server. Data entry forms which can be part of client server pages are modelled by another class and association stereotype respectively. There are also class stereotypes for Java Applets, Java Scripts, ActiveX controls and frames. These stereotypes proposed are appropriated to model layout and implementation aspects but not to define the navigation structure of the Web application.

There is no clear separation between content, hypertext and presentation level. Web pages are represented by a UML class and its relationships to other pages (associations) represent hyperlinks. Some stereotypes for links are also proposed such as *redirect* and *submit* links.

Behaviour modelling is discussed in terms of operations which can be defined together with stereotype classes at the application domain level. UML Activity Diagrams and Sequence Diagrams are employed to describe complex operations. Even though it argues that an iterative design process is needed, no specific phases are discussed for the modelling activity.

Tool support is given by templates that can be used with ArgoUWE [67], which is build as a plug-in over the Case ArgoUML[10]. Such tool provides support for the design of Web applications using the UWE notation and the partial code generation for the Web site.

2.4.10. UML-based Web Engineering methodology (UWE)

The UWE describes a systematic design methodology using UML notation and extension mechanism [68] [69] [70]. The approach employed is quite similar of OOHDM but instead of

[7] Stereotypes allow attaching a new specific semantic to an element in the model.

[8] *Tagged values* represent new properties that can be associated to a model element.

[9] *Constraints* specify conditions under which a model can be considered "well-formed".

[10] http://argouml.tigris.org/

using a mix of different notations throughout the levels, UML is used as the basic modelling technique.

UWE separates all three levels comprising a Conceptual model in terms of pure UML diagrams, a navigation model and a presentation model. At the hypertext level the navigation class model (cf. navigation class schema in OOHDM) specifies which classes and associations of the context level are available for navigation. It is represented by means of a UML class diagram denoting the navigation classes by means of stereotypes and navigable associations by means of constraints. The navigation structure model (cf. navigation context schema in OOHDM) is based on the navigation class model and defines how each navigational class is accessed during navigation. Stereotypes are again used to represent navigational contexts and thus provide navigational patterns as index and guided tours. The stereotypes include <<navigational context>>, <<grouped context>>, <<filtered context>>, <<index>>, <<guided tour>>, <<menu>> and <<external node>>. Behaviour modelling is only mentioned with respect to defining the sequence of navigation by means of constraints.

At the presentation level, a static presentation model using the possibility of UML to represent compositions by means of graphical nesting, describes the layout of the user interface. A dynamic presentation model employs UML StateCharts for describing the activation of navigation and user interface transformations. Stereotypes representing the most frequently used interface objects such as text, images, audio and button are provided. The mapping between hypertext level and presentation level is not discussed by the approach. Concerning the dimension of phases, the same holds as for OOHDM.

2.4.11. Object-Oriented Hypermedia (OO-H)

Similarly to UWE, the OO-H method [10] [71] [72] relies upon UML notation to model the content level of Web applications by means of class diagrams. However, navigation and presentation levels are covered by two new diagrams namely: the Navigational Access Diagram (NAD) and the Abstract Presentation Diagram (APD) respectively.

NAD diagrams, which cover the hypertext level, are built by means of four types of constructs: navigation classes (modified domain classes whose attributes and methods visibility have been restricted according to navigation requirements), navigation targets (elements grouping pages and navigation classes), navigation links and collections. Links can be typed as internal links, traversal links, requirements links, exit links and service links, with a specific functional semantic in the modelling [72]. Collections are similar to HDM collections [13] which are used to describe a set of elements in a model, possibly organised hierarchically.

Behaviour can be defined in all levels using the object-constraint language (OCL) in a similar way it can be done with UML. At the hypertext level, OCL expressions are used to define navigation filters which can both inhibit navigation and/or restrict elements being visualised in the target. In addition, metamodel attributes (visualisation, user interaction, application scope) can be associated to links to express: static/dynamic link generation, manual/automatic link activation mode, and the number (simple, multiple, or universal) of objects involved when the link is traversed, respectively. OO-H also introduces the concept of *activation links*, which means links that enables (or disable) others according to contextual information from previous navigation.

Navigation patterns as those presented by HDM-Lite [29] are supported by means of special attributes which are associated to links and/or collections. The patterns supported are: index, guided tour, indexed guided tour and showall [71].

At the presentation level, OO-H employs five templates built over a XML document to specify visual appearance. The default template structure can be generated dynamically from the information provided by NAD diagrams by means of a Case tool that supports the whole design process [72]. However, a flexible mapping between navigation modelling and presentation is not fully described. OO-H does not support any kind of visual layout edition.

The OO-H method defines a design process covering the entire life cycle of development. Only the bottom-up approach is described. UML Use Cases are employed at early phases of the design to deal with requirements analysis. The design process described in [72] suggests that Use Cases can be used as a starting point to define NAD diagrams. UML activity diagrams complete the domain model by defining the inner flow of control for non- trivial use cases [73].

2.4.12. W2000 Framework

The W2000 framework is another modelling technique that integrates concepts borrowed from HDM [13] and UML notation for Web design [74]. Such an approach for integration consists in defining stereotypes and customising diagrams to render HDM and UML; specifying a set of guidelines to use UML properly for the Web design; and refining use case diagrams to describe high-level requirements related to both functional and navigational aspects.

HDM elements such as entity types, components, entities, associations, collections, link types and links are assigned by means of UML stereotypes. Use cases are employed to model both functional and navigational requirements. The content level is covered by UML class diagrams which are employed to represent the hyperbase information design. The information design is composed of the structural design and the access layer; both concepts are also borrowed from HDM.

At the hypertext level, nodes and links are derived from the hyperbase information design. Some guidelines are provided to design navigation according to the hyperbase, once no dynamic derivation is involved. Links, represented with UML arrowed associations, are organised in two categories: structural links, which are induced by part-of associations and, semantic links, which are induced by semantic associations. Navigation patterns are included in the modelling by the means of special symbols attached to associations between classes in the navigation design.

The behavioural aspect is covered at the hypertext level by extended UML interaction diagrams. Notably, sequence diagrams are extended to distinguish among different categories of objects (i.e. entities, associations, and collections) and specific primitives used for passing messages (such as constrained or free navigation, user versus system operations).

The presentation level is completely neglected. W2000 does not provide any model to structure or behaviour at the presentation level and it neither does it give any guideline to match navigation and content models during presentation design.

Similarly to HDM, W2000 identifies the design-in-the-large and design-in-the-small during the development process. Rather than to present a specific design process, the W2000 organises the design activity in a number of interdependent tasks where, to each activity corresponds a specific model or diagram. A UML StateChart diagram is employed to describe the evolution of a Web application in terms of a predefined workflow process. Even though a design process is not clearly defined, the activities proposed cover all phases of design from requirements analysis to functional design.

2.4.13. Design Engineering Approach for Hypermedia Engineering (DEAHE)

The DEAHE approach [75] employs the UML notation without any special extension for modelling Web applications. The main significant contribution of this work is to use UML StateCharts to model navigation and presentation aspects, which enables to represent behavioural aspects at the hypertext and presentation levels, respectively. Similar to previous object-oriented approaches, the information domain is modelled by traditional class diagrams. The approach also includes a user modelling view which is represented by means of a class diagram. This kind of modelling is used to generate adaptive navigation sequences in the context of e-learning applications [76].

Even though this modelling approach [75] provides clear conceptual separation between the application domain level, hypertext and presentation levels, it lacks explicit mapping rules for modelling Web applications. The authors do not describe design-in-the-large and design-in-

the-small [13]. No design process is discussed. The effort made to provide a clear UML notation without extensions, causes a lack of information about the modelling process which is left implicit by the fact that UML is being used as modelling language. Navigation patterns, for instance are not discussed. No particular stereotype is employed to represent semantic concepts of Web applications.

2.4.14. Web Site Design Method (WSDM)[11]

The WSDM method employs a user-centred approach to design kiosk Web sites that, in opposition to data-intensive Web application, mainly provide information and allows users to navigate through that information [77]. Thus, rather than taking an organisation's data or database as starting point for the modelling, WSDM takes as a starting point the needs and requirements of the intended audience of the web site by modelling user's profile and tasks.

The WSDM method consists of four phases named user modelling, conceptual design, implementation design and actual implementation. The design process includes an extensive modelling of users' requirements in terms of functional, informational and navigation requirements.

The WSDM clearly separates the conceptual design from the design of actual presentation. The conceptual design covers the information modelling, functional modelling and navigation design [78] which correspond to abstract modelling. The WSDM defines chunks of information which represents views over a larger data model in a similar way of RMM's slices [40] and HDM's perspective [13]. Information modelling and functional modelling are built by defining chunks and their relationships. An extended version of ORM notation [79], a kind of Entity-Relationship model, is employed to produce the so-called chunks of information [78].

The navigation design is made by a so-called track diagram which is composed of tracks constructs, components and links. A track element in a track diagram points to the initial element in the set. Components represent units of information or functionality. Components are connected by means of links which can be simple, conditional or a combination of typed links [80].

Even if WSDM describes a clear separation between content, hypertext and presentation levels, only the two former are covered by the method. The hypertext level depends upon content modelling which is performed by means of the information chunks. Behaviour modelling is supported at the content level and in some extension at the hypertext level by the means of conditional links. Typed links described in [80] can extend the semantic of relationships and support some navigation patterns such as index and menu. No specific tool support is discussed.

2.4.15. Jessica

The Jessica system [81] allows the description of hypermedia applications in terms of HTML elements in an object-oriented language. The components are defined with a makup language based on XML standard. The language's primitives consist of objects, templates, packages, so-called *pragmas* and elements to describe the interrelationship between them [81]. A software compiler is proposed to map abstract system definitions to dynamic Web services.

The UML class diagrams are employed in [81] to visualise Jessica packages, objects and their relationships. A set of guidelines and tool support is provided to map class diagrams into Jessica objects [82].

Jessica only provides separation between application domain and presentation which can be considered an improvement when compared to HTML language only. The presentation level is covered by templates that enable to specify the page layout and set visual properties to

[11] http://wsdm.vub.ac.be/

objects. The hypertext level is defined together with the application domain level. UML relationships are transformed into Jessica references. The behaviour aspect is not discussed.

A design process is mentioned to cover the phases, the design and implementation. Both phases are supported by a tool called JAZZ.

2.4.16. Web Composition Markup Language (WCML)

Similar to Jessica, WCML gives developers a way to describe Web applications in terms of an object-oriented approach by using XML constructs [83]. The WCML is just a language to implement a WebComposition model.

According to the WebComposition model [84], a WCML document consists of a set of components defined by a set of components declarations, with their properties and their relationships defined by inheritance and aggregation. The conceptual model behind WebComposition is a prototype-instance-model for modelling inheritance [85] which enables every component to be used as a prototype by other components. Multiple inheritances are possible. Relationships between components can be defined using a special variety of references, called link properties, which are mapped to corresponding hypertext links.

WCML uses the decoration design pattern [47] to integrate hypertext and presentation components to content thus providing separation between the three levels. The use of decoration patterns makes it possible to describe several presentations for the same application. The behaviour aspect is not discussed.

The design process is discussed in terms of how components can be reused and extended to several applications. Conceptual modelling and integration with graphical design modelling such as OOHDM is mentioned as future work [83].

No graphical notation is associated to WCML descriptions. A compiler tool performs the mapping of WCML descriptions to the target languages such as HTML, CSS and XSL.

2.4.17. Scenario-based Object-oriented Hypermedia Design Methodology (SOHDM)

SOHDM approach [86] relies upon scenarios to guide all the design activities concerning the development of Web applications. Scenarios are described by means of Scenario Activity Charts (SACs), which intend to capture (functional) user requirements. SAC diagrams are quite similar to flow diagrams in the sense they capture the business logic employed by actors. The design process proposed includes six phases: domain analysis, object modelling, view design, navigation design, implementation design and construction.

The methodology provides a set of guidelines to derive SACs scenarios into an object model, which is specified by means of class responsibilities collaboration (CRC) cards diagram. A kind of class diagram called class structure diagram (CSD) is then employed for representing relationships (aggregation, specialisation/generalisation and associations) among CRCs descriptions. A set of guidelines is provided to map navigation from SACs scenarios. The navigation modelling starts with the creation of navigational units for CSD elements and the so-called Access Structures Nodes (ASN) which are elements (such as pages) giving access to other parts of the applications. Both navigational units and access structure nodes are considered as conceptual nodes for the application. Links are presented in the form of a navigation link matrix. The presentation is made by creating a page schema which describes the elements each node contains. Page schema is enhanced to the user interface (UI) specification by the use of UI components. A graphical notation allows the description of components such as image, choice (combo box), slide bar, text and HTML anchors.

SOHDM clearly identifies the three levels and guidelines are supplied for the mapping between levels. Structural aspects are covered at all levels but behaviour description is only available at domain application level. Even if the authors use scenarios to represent user

requirements, the design process does not take into account non-functional aspects that are relevant for a user-centred design process.

2.4.18. Hypermedia Model Based on StateCharts (HMBS)

Hypermedia Model is based on StateCharts (HMBS) [87] which relies upon StateCharts notation [23] to specify both the structural organisation and the browsing semantics of hypertext applications. HBMS is a navigation-oriented formal notation that proposes a mapping of StateCharts objects such as states, transitions and events into hypertext objects such as pages links and anchors. A similar StateCharts-based approach has been proposed by Zheng and Pong [16] to model hypermedia applications; even though some remarkable differences exist concerning the browsing semantics.

An extended version of HMBS, the XHMBS [17], defines a formal model called hypercharts [18] that includes additional mechanisms for describing time sequencing and information synchronisation requirements for multimedia applications.

HMBS formally distinguishes domain application (content), hypertext and presentation levels but only provides full support for modelling navigation. HMBS allows specifying structure and behaviour at hypertext level but it is restricted to the modelling of applications with static data and no synchronisation requirements. Modelling at presentation level is suggested by means of a schematic layout for windows displaying but the authors argue that HMBS is not a model for representing user interfaces states.

Rather than propose a particular design process, the HMBS suggest that the model may be integrated into other higher-level design methods covering the whole design process. Reuse of components and navigation patterns are not discussed either. A tool support allows interactive modelling and it provides some facilities such as model checking of navigation specifications.

2.4.19. Trellis

The model Trellis [15] is a formal technique based on Petri nets for describing relationships between pieces of information into a hyperdocument and specifying the corresponding browsing semantics.

Trellis is typically a behaviour modelling. Several navigation behaviours are possible in Trellis, including special navigation patterns [21] and time-based adaptation of document presentation [88]. There is no clear separation between domain application, hypertext and presentation levels. The domain application modelling is discussed only in terms of a set of nodes to which content can be mapped. The model relies on a document database which is mapped, a set of windows display, and buttons to Petri nets elements. Only static content is supported by the Trellis. The adaptive navigation described in [88] does not include dynamic content generated but rather it enables or disables links users can navigate according to timed conditions.

A tool support is provided and a set of formal modelling analysis can be performed over Trellis graphs [89]. However, no design process is discussed.

2.5. ADDITIONAL RELATED WORK

Following different perspectives, many reviews in the literature have pointed out shortcomings with existing modelling methods for the Web.

Retschitzegger et al. [90] use a categorisation framework (similar to that presented in section 2.2) to discuss eight existing modelling methods for the Web, pointing out the lack of explicit and flexible mapping between levels and the poor support to describe behavioural aspects of applications.

Another interesting survey presented by Gu, Henderson-Seller and Lowe [28] evaluate the support for informational and functional requirements given by six models. They [28] observe the lack of support for modelling both informational and functional requirements and unclear connections between business models and the system designs. Escalona and Koch also discuss several types of requirements (e.g. navigation, functional, user interface, etc.) and techniques employed to capture project needs during the requirement engineering phase [91].

The survey presented by Barry and Lang [92] discuss the impact of Web modelling methods on the industry. They observed that influent models in the academic field such as HDM have little impact in the industry. They conclude that practitioners need more support to manage the development process and to capture user's navigational requirements.

Gaedke and Gräf [93] describe a process model putting in advance the importance of evolving concerns of Web applications and the value of object-oriented models to deal with reuse of specifications. They point out the poor support given by five existing models to support evolving Web applications but they do not compare explicitly modelling methods based on OO methodology.

2.6. CONCLUSION

In this chapter we draw a big picture of existing modelling methods for the Web covering several requirements for modelling identified in previous work. Not all, but the most representative existing modelling methods for the Web are described in section 2.4. These models are representative of the current trends for modelling Web applications. Our conclusions for the present chapter can be summarised as follows:

- Most methods do not support the presentation level with appropriate conceptual and logical modelling concepts. Rather, authoring tools are often suggested for capturing the presentation level, thus loosing the benefit of technology-independency;
- Except to a few methods ((RMM, WAE, DEAHE), top-down and bottom-up design is not clearly distinguished;
- Patterns are poorly discussed and only supported at the hypertext level;
- Most methods do not guide the design activity. They do not even support all development phases of design;
- Reuse and evolution concerns are poorly supported by most of existing modelling methods. Reuse is only supported by some of the OO-based methods described in this chapter;
- No uniform modelling notation to describe navigation level even among those relying in UML notation. The non-standard use of UML diagrams and approaches for modelling may cause cognitive overload on designers used to work with UML;
- Except for OOHDM, W2000, DEAHE, OO-H, WSDM and SOHDM, modelling methods do not clearly represent navigational and functional requirements;
- Even though the Web is considered to be a kind of hypermedia systems, multimedia concerns such as delivery of multimedia content (e.g. streaming video), media synchronisation, and user interaction are not explicitly described by any modelling method;
- The behavioural aspect is widely neglected by modelling methods especially at hypertext and presentation levels.

3. Navigation Issues When Modelling Web Applications

Summary

In the previous chapter, navigation was presented as an application level that should be taken into account by modelling methods during the development process of Web applications. This chapter proposes an overview of the conceptual elements such as link and node that compose navigation models. The analysis we propose here is justified since hypertext systems have strongly influenced the World Wide Web and most of users' tasks over Web applications (such as browsing and searching) are directly related to the navigation.

Our goal at this point is to discuss advanced link features that can be incorporated into navigation models to improve usability and/or to enrich the semantics of navigation models. A large description of link and node features is provided, even though the implementation of such features over Web applications is cumbersome or not currently available due technological constraints. This chapter is organised as follows:

3.1 What is Navigation in Web Applications?

3.2 Link Features

3.3 Languages for Link Specification

3.4 Modelling Link Features

3.5 Conclusion

3.1. WHAT IS NAVIGATION IN WEB APPLICATIONS?

The term hypertext, coined by Ted Nelson, denotes a text-based system giving to readers several pathways to explore information[12]. The first hypertext system, called Memex, was imagined by Vannevar Bush in 1945 [94] as a machine for browsing and making notes over a very large library (eventually storing all human knowledge). Bush's Memex was not conceived under a computer but it used a sort of microfilms to store information. In fact, Memex was never built but it has inspired a plethora of systems such as Engelbarts' NLS/Augment, the first hypertext system supported by a computer.

Hypertext systems[13] have strongly influenced the World Wide Web. As other hypertext/hypermedia systems, Web application features the possibility of non-sequential navigation between pieces of information or documents (i.e., there is no single order that determines a sequence in which the text is read). The central idea behind hypertext is the link and the non-linear way to explore the information space. Links not only connect chunks of information but they also provide a means to explore (by navigating) such as an information space.

The basic premise for Bush's paper "As we May Think" (1945) [94] was that links create associations between information in as similar way as human beings think and learn[14]. The network formed by the links creates a sort of direct graph[15] that is frequently employed to represent topology of hypertexts as depicted in Figure 6. Nodes (representing information or content), anchors and links complete the basic elements of a navigation model.

[12] The non-linear structure for information does not imply in an electronic format of information storage. Many books and paper-based document can also provide several paths to explore information. However, most authors insist that navigation thought a hypertext must be computer supported in order to qualify as true hypertext.

[13] Although most hypertext systems include a graphical component, a hypertext-like system including graphic or non-text media is often properly termed hypermedia. Because hypertext and hypermedia are often used together, many authors used the term hypertext to refer both. As this work is focused on navigation aspects that are out of such consideration we frequently employ the term *hypertext*.

[14] Even if that premise has not yet proven to be productive in practice, links provide an efficient way to create relationships and allowing the non-sequential access to information.

[15] If a graph is direct then the presence of an edge between any two nodes (e.g., A and B) does not imply that there must be an edge going other way too (from B to A).

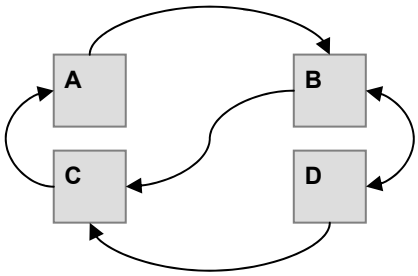


Figure 6 – A hypertext as a direct graph.

Technically, navigation is a dialogue between the users and the system following a specific protocol: a user identifies a hotspot as a source anchor for a link and s/he selects the hotspot to activate the link, then a browser checks the link connection and replaces the current document by the destination document referred by the link. This operational semantic is an important difference between windows-based applications and Web applications. While in windows-based applications any window can spawn child windows creating nested dialogues to reinforce the conceptual model, in Web application users can navigate everywhere between pages without any control by the system or the browser [95].

Browsers are often seen as just a piece of software employed to access and visualise documents. In practice, browsers play an important role ensuring cross-platform application (since only the client browser should be platform-specific) [96] and they have a huge influence on how users perceive navigation over Web applications.

Current browsers not only present documents or intermediate the communication between users and a remote systems but they also provide a variety of revisiting tools allowing users to access directly application parts without having to follow strictly navigation paths proposed by the system. These revisiting tools include history mechanisms (to keep the most recent visited pages), the famous back button (to go back to previous visited page) and bookmarks (to store favourite URI addresses). Theses functions are intended to help users during navigation over the Web but different implementations may cause the inverse effect [6] [4]. Designers should be aware that revisiting tools may interfere on user navigation and, on other hand design cannot rely on revisiting tools to ensure navigation. Since such navigation functions supplied by browsers are out of the control of designers they are not discussed in this work.

3.1.1. Nodes

Hypertext nodes may represent any kind of multimedia content. Quite often, the concept of nodes designates atomic elements (e.g., an image). However, most hypermedia systems also provide constructs to group a set of atomic elements in compound nodes (e.g., document containing text and images) with arbitrary granularity.

The content of a node can be static (defined by an author) or dynamically generated by the system (through a queries in a database). Nodes can be enriched with semantics by including the description of node types describing content, for instance. The implementation of these advanced features for nodes (semantic types and content generation) is similar to the implementation of link features discussed in sections 3.2.6 and 3.2.8 respectively.

3.1.2. Anchors

Anchors are identified as hotspots holding user interaction. Formally, an anchor is just a pointer to an interactor enabling a user to select a link over on the interface. Typically, an anchor has two parts: an anchor Id that uniquely identifies an anchor within the scope of a node; and an anchor value that specifies some location, region, item or structure within the node. All media types (text, images, audio, and video chunks) are suitable to sustain anchors [97]. However, anchors become optional if the navigation is driven by the system (e.g., links

associated to timestamps conditions) or if there is another selection mechanism to capture user inputs (e.g., gesture recognition [98]).

3.1.3. Links

Formally, a link was just a relationship of two anchors, called source and destination, which represents connection points of two nodes [26]. The idea of link has evolved in a variety of ways. Links had become complex structures with rich semantic and functionality which has been progressively incorporated into Web applications. Actually, they may contain attributes, keywords describing semantic types, or actions that are executed when the link is activated. These advanced link features are fully described in next section.

3.2. LINK FEATURES

Since Memex [94], the first hypertext system, many architectures, models and link typing strategies have been proposed, adopted, and eventually superseded [99]. Links have been extended to support advanced features that may improve usability or enriching the semantic of navigation models in hypermedia systems [100].

The Web architecture imposes constraints avoiding the implementation of some advanced link features. For example, the absence of control allowing adding and removing links may easily lead to have an application with inconsistent content and broken links. Furthermore, navigation (links), content and presentation are typically included in a single file which does not improve the reuse of components. Once the Web architecture does not propose solutions to avoid these problems they usually must be solved by the Web application itself using add-on technologies.

Hereafter, we review many different link features. More than just propose a link taxonomy/classification we present briefly link features in terms of conceptual elements for hypertext design and we then discuss their particular implementation over Web applications.

3.2.1. Direction

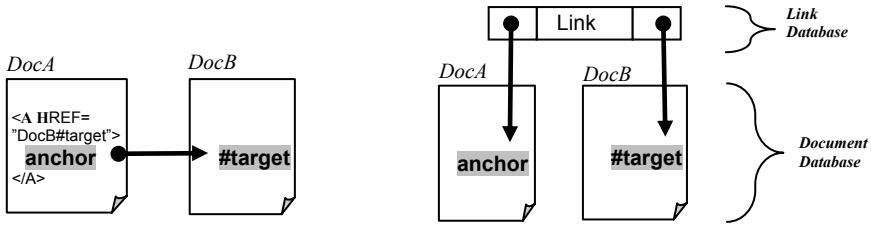
The direction is the link feature that encodes whether an anchor is to be considered a source of a link, a destination of a link, both destination and source or neither a source nor destination. In opposition to unidirectional links that can be traversed only in one direction, Bi-directional links can be traversed in both directions.

Bi-directional links do not imply revisiting tools provided by Web browsers since they are much more than just “go back links”. They enable to navigate towards all nodes referring the current node. From a technical point of view, bi-directional links help to keep links consistent and avoid broken links. In addition, users can also get benefits of bi-directional links by navigating backwards. For a concrete example, bi-directional links could help a user to find all pages referring to an old and no longer updated (but still relevant) document.

Besides ethical and legal aspects involving non-authorised use of linking, the implementation of bi-directional links throughout the Web poses serious technical problems due to the absence of a central control, as discussed in [101]. The problem is that source documents may not be notified about changes of destination documents which may cause broken links if the destination document is moved or deleted.

3.2.2. Internal or external link specification

Links are often embedded as part of documents (see Figure 7.a) as it is done by HTML. In contrast to fixed embedded links, some systems such as Microcosm [102] have been designed to keep links in a link database, separately from the documents to which they refer (see Figure 7.b).



a) *Embedded link* **b)** *External link service*

Figure 7 - Internal (embedded) x External (link service) link representation.

The main strategies to separate (or not) link [103] from content are:

- Embed all node and link information within the document;
- Embed tags to define source and destination nodes for "persistent selections" but store the links themselves externally; and,
- Store both link, and pointers to potential anchors, externally.

Embedded link and document make a self-contained object that may be moved and edited without damaging any of the links the document contains. However, if the document destination of a link is moved or deleted, all links must be checked to ensure link integrity and avoid broken links.

External link services enable several authors to add links to a *linkbase*[16] even if they are not allowed to write on the documents a link refers. These additional links can be used to annotate and supplement existing information with other information of personal importance. Such a mechanism has been the origin of annotation systems such as Annotea [104], PageSeeder[17] and ThirdVoice[18]. These annotation tools allow users to choose which linkbase s/he wants to browse. However, multiple linkbases may result in an unexpected number of links. Another major inconvenience for separating structure from the data in a linking service occurs when a file is updated causing links to point out to the wrong positions within the node.

The separation between link structure and content has been strongly advocated by the Dexter Reference Model [26] and largely preferable over embedded links in the hypertext community. However, the management of separated link structure and content is much more complex in open hypermedia systems such as the Web due to the lack of a centralised control for updating. The resulting problems from all these link strategies and the corresponding philosophies for management of link integrity are discussed in [105].

3.2.3. Link attributes

Link attributes can be used for several purposes as they can set a unique identifier for a link, help to customise the link's look and feel or to improve user feedback by giving supplementary information about the link destination. For example, the attribute colour has been extensively employed to indicate whether a link has been previously visited or not, thus providing visual feedback to user about documents s/he has seen.

Links attributes can be defined according:

- Destination characteristics (e.g., destination media type, document size, etc);

[16] The term *linkbase* is often employed to refer a collection of links managed by a database.

[17] Available at http://ps.pageseeder.com/ps/topic/ps (Jan. 4th 2004).

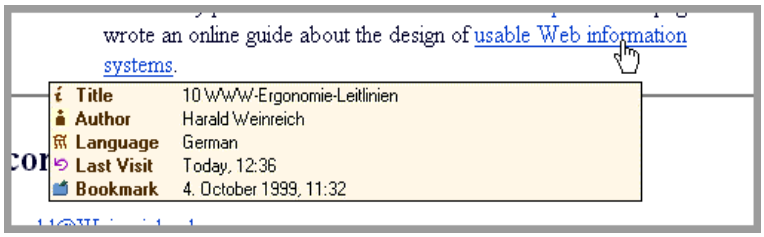
[18] Third Voice was annotation Web service that provided a way to post sticky notes with comments on any site on the Web. ThirdVoice was unpopular with many Web site owners, who were disturbed by the idea of people posting critical or obscene material that would be presented on top of their site. The project stopped in 2001 and it is no longer available at the original address at http://www.thirdvoice.com/.

- Dynamic properties at run time (e.g., link availability);
- Anchor's visibility (e.g., colour).

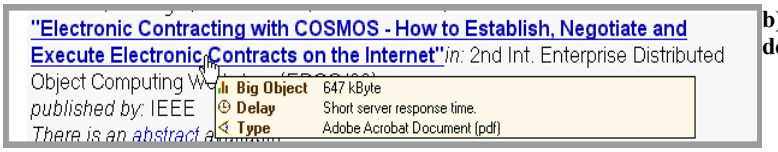
Here, we identify two categories of link attributes:

- ***Static link attributes*** that can be specified manually by the author; and,
- ***Dynamic link attributes*** that can be recovered automatically by the system (e.g., information such as whether a link dangles or it has been visited before).

The HyperScout system[19] [106] is an example of application that can obtain additional information about the link from several sources (e.g., link anchor tags, the user's history and web servers) and display them in a small box when a user passes the cursor over the link (see Figure 8). HyperScout navigate a link to discover attributes from the document the link refers. In Figure 8.a, for example, the attributes title, author and language refer to the destination document not the link itself. Even so, such additional information may improve navigation and reduce the cognitive overhead by helping users to decide if the link target worth a visit or not.



a) <title>, <author> and <language> refer to the destination document. <last visit> and <bookmark> tell that destination document has been visited and bookmarked by the user before.



b) <big object> and <type> refers to destination document while <delay> informs the connection speed.

Figure 8 - Link and document attributes by HyperScout.

3.2.4. Additional actions

Links can be empowered with functions to transfer values from a document to another, or to send instructions to be processed at the target node. Such functions are made possible by the means of additional actions that are coded as part of the link definition.

The transfer of data embedded into links became popular over the Web with the advent of Common Gateway Interface (CGI) and HTML forms. Attributes and values can be directly coded as part of links or interactively captured by forms and incorporate to a URI (Uniform Resources Identifier – URI[20]) when a user clicks on the send button.

Additional actions can also be used to modify the default links' operational semantic for replacing source documents by target document in a same window. By using additional actions one can customize the presentation of target document in multiple windows, for example.

Different actions can easily modify interaction over a same navigation scheme. Thus, additional actions must be taken into account during dialog/navigation design. [107] [108] propose a graphical notation, called Windows Transitions, which provides a means to model common actions used on windows dialogues such as setting windows size, opening and closing windows, and so on. Figure 9 presents an example for a phone order task that depicted how the

[19] Available at http://vsys-www.informatik.uni-hamburg.de/projects/hyperscout/ (Jan. 4th 2004).

[20] URI provides a seamless way to means to address objects the Internet. The term URI replace the term URL (*Uniform Resource Locator*) used by Berners-Lee in previous publications but it is no longer used in technical specifications). Currently, URL is associated with popular URI schemes: http, ftp, mailto, etc. (Available at http://www.w3.org/Addressing/. Last visit: July 2003).

Windows Transition notation looks like. Presentation units are depicted as windows-like symbols while arcs between windows describe transitions (navigation) between windows. Operations performed on source or target windows are depicted by special symbols placed, respectively, on source and target sides of arcs according to the windows they refer. For example, the symbol ☐ on the target side of an arc means target window will be opened when the transition is activated while the symbol ☒ on the source side means the source window will be closed.

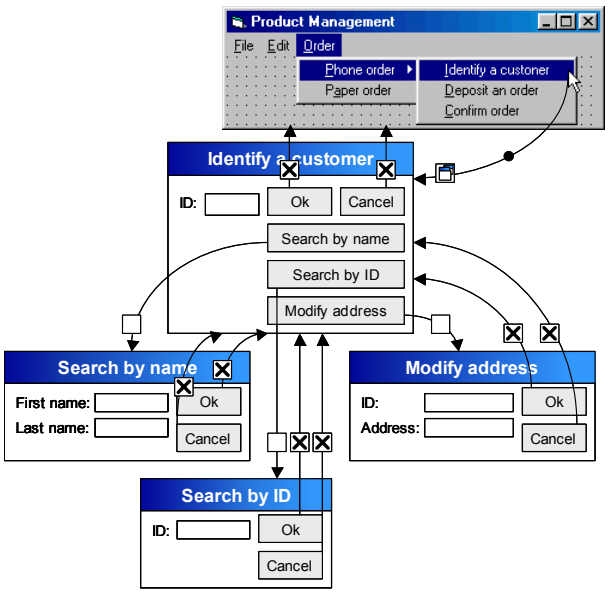


Figure 9 –Window Transition modelling for a task phone order [108].

The way in which documents are presented can change dramatically how users read or interact with the application. This concern regarding the problem of mapping dialogue and tasks modelling is discussed in [109].

3.2.5. Link cardinality (one-to-Many end points)

Often, keywords can be employed as anchors for multiple relevant destinations. For example, in Figure 10 the keyword "The Common Gateway Interface" is anchor for two alternative destinations (depicted in bold face) that could equally satisfy the user request for more information. In such a case, it is possible to use one-to-many links to connect not two nodes but a set of related nodes.

> "The Common Gateway Interface (CGI) **{CGI: Common Gateway Interface at Cern**[21], **Server Side Scripting ¦ Common Gateway Interface (CGI) at Yahoo Directory**[22]**}** is a standard for interfacing external applications with information servers, such as Web servers. It is typically used in conjunction with HTML forms to build database applications…"

Figure 10 - One-to-Many links in-line in the text.

One-to-many relationships have been extensively explored in Information Systems and Database disciplines as a means to express complex relationships. In the Hypertext field they are less frequently employed, even though many hypertext systems support one-to-many relationships (e.g., DLS [110], Microcosm [102], Hyper-G [111], and Sepia [112]). In most database applications the presentation of one-to-many relationships is solved by displaying a list

21 http://www.w3.org/CGI/

22 http://dir.yahoo.com/Computers_and_Internet/Software/Internet/World_Wide_Web/Servers/Server_Side_Scripting/Common_Gateway_Interface__CGI_/

of destinations in a relationship. However, this approach is not enough for hypertext because users must be able to move from a single document to another.

There are three strategies to define the final destination for a hypertext link. The first is to formally request users to choose between available destinations when s/he is traversing the link. In this case, after a user has selected a keyword, an intermediary page presents a list of generated link destinations. This solution has been adopted by hypertext systems such as DLS [110] and Microcosm [102].

The second approach is to show all destinations (in a pop-up menu, for instance) before traverse the link. This strategy has been employed by XLink [113]. The inconvenience with this strategy is that users have to make an additional selection from the pop-up menu each time they follow a link. Even though intermediary pages are probably the most straightforward approach and the best choice to display large lists of destination links, the pop-up menu approach may reduce navigation time since there is no intermediary page.

The third strategy is to employ filters to automatically select the best destination. Filters have been suggested by Vannevar Bush in Memex [94]. They can be used, for example, to hide already visited documents in a guided tour [114]. In Hyper-G, filters are employed to show links according to user rights [111].

3.2.6. Semantic link types

Links and nodes can be typed expressing structure, semantic, or other relevant feature that can better describe them for a specific use into the application. Link types such as <explanation>, <further details>, <argument>, may improve the user understanding about the nature of the relationships between the link's destination and the current document [114].

Systems using richer link types are quite often domain-dependent or application-dependent. For example, Benyon et al. [115] suggest that the use of <glossary>, <annotation>, <structural>, and <association> link types can improve user understanding about their multimedia courseware materials over the Web. The gIbis system, nodes and link typing are used to support argumentative dialogues among a team of software designers, including transcription of design decisions [116]. Each issue can have many positions, where a position is a statement or assertion that resolves it. Each position may have one or more arguments that either supports or rejects it. The corresponding hypertext model consists of 3 node types: <issues>, <positions>, and <arguments>, and 8 link types: <generalizes>, <specializes>, <replaces>, <questions>, <is_suggested_by>, <respond_to>, <objects_to>, and <supports> (see Table 4).

Table 4 – gIbis' enforced link and node.

Node type (source)	Possible link type(s)	Node type (destination)
Issue	generalises, specialises, replaces, questions, is–suggested–by	Issue
Issue	is–suggested–by, questions	Position
Issue	is–suggested–by, questions	Argument
Position	responds–to	Issue
Argument	objects–to, supports	Position

Similarly to gIbis, MacWeb system supports both node and link typing [117] but MacWeb does not have predefined node or link types; so, authors can specify as many types as required by the application. By defining semantic types for the hypertext, authors are explicitly describing some aspects of their mental model (and knowledge) about the application. For example, the node type <Portrait> of the node <Mona Lisa> having a link of type <Painted by> to the node of type <Italian Painter> named <Leonardo da Vinci> and another link of type <Location> to the <Public Collection> <Musée du Louvre> node represent some knowledge. These nodes types can be used latter at a more conceptual level, for instance for retrieving <Location> where are <Portrait> <Painted by> <Italian Painters> [117].

3.2.7. Structural link types

Links can also be typed to express structural relations in models such as referential and Links can also be typed to express structural relations in models such as referential and organisational links [118]. Organisational links are used to organise nodes hierarchically in a strict structure. For example, organisation links might be used to link a page to the next/previous pages in the set or to link pages to the initial page. Referential links are non-hierarchical and are the kind of links that most clearly distinguish hypertext from other forms of information storage as they might be used to link specific content such as footnotes, pictures or any other piece of information, to any page.

Similarly, Garzotto et al. [13] distinguish three types of links: structural, application and perspective links. Structural and application links are equivalent to Conklin's organisational and referential links, respectively. Perspective links interconnect nodes corresponding to the same content but having alternative presentations (i.e., a node having a text and another node having a graphic image for a same content should be connected by perspective links).

According to Garzotto et al. [13], links have a twofold role: a representational role (i.e., capturing domain relations) and a navigational role (i.e., capturing navigation patterns). These two different purposes can be consistent with others or not. It may happen that domain relationships are not relevant for the application; for example when they induce navigation patterns that are not suitable for an application or, in the opposite case useful navigational links may have vague semantic meaning.

3.2.8. Link creation

Links can be differentiated according to the way their source and target are specified:

- Static links: whose source and target are explicitly created by the document's author;
- Dynamic links: whose source is created by the document's author but whose target is determined by conditions or some kind of automated processing at run time;
- Fully extensible links: when source and target documents are determined dynamically by the system.

Static links may be complex if we consider that, occasionally, it can be useful for users to add their own links to documents in order to assist them in their own navigation or to aid other users. Such concept of users adding links to documents that other users also have access to blurs the distinction between author and user.

Dynamic links can be used, for example, to incorporate security procedures (links are available if a user has access rights), to allow users to create personal links between documents that are not usually linked, or to automatically include new node destinations in existing nodes as result of a searching process;

Another strategy for dynamically creating links is to employ search engines or some kind of information retrieval technique [31]. A typical example of dynamic link creation on the Web is search engines such as Google™ [23] and AltaVista™ [24]. These search engines compare keywords given by a user to an index of a document database and give back to the user a new document containing links created dynamically for each document corresponding to a keyword.

Actually, more and more techniques for efficient document retrieval have been proposed. Some authors even support the idea that users could take more benefits from document retrieval than navigating through information spaces. This discussion is polemical because either document retrieval or navigation is efficient for some tasks; the best approach, probably, includes both strategies.

[23] http://www.google.fr

[24] http://www.altavista.com/

Fully extensible links appear in very large hypertext system when it is almost impossible to maintain static links to every document accessible via the system. Some authors such as Andrews et al. [119] consider that the access to documents on remote databases is only possible using these kinds of links because the information in the remote database may change at any moment. This position can cause polemic but it seams clear that these kinds of links are helpful in design by reducing the number of links that are explicitly authored. For example, if an electronic book has over 100 chapters then linking each one individually to a content list could be very time-consuming. One could save time by using an algorithm that examines the text selected by a user (according to a keywords list) and then automatically links to the chapter with a title corresponding to the selected text.

Systems supporting dynamic links also support explicitly defined links since static links are often needed to define the underlying structure of a hypertext. The implementation of fully extensible links requires an independent system capable of identifying anchors at source document and to establish dynamic relationships to the corresponding nodes.

3.2.9. Timed links

Temporal constraints have been, for a long time, neglected over Web applications [120]. Such aspects have addressed by The Amsterdam model [121] which extends the conceptual model Dexter reference Model [25] to integrate temporal constraints over Web presentation.

Timed links allow authors to have control over temporal aspects such as sequence and concurrency of document presentation. Some typical examples of applications that require such a feature are the coordination of digital photos and textual comments; and training courses integrating voice and images.

Some primary timing mechanisms such as time stamps enable system-driven interaction by replacing a node by another according to timed conditions associated to links. However, full timed link support will require the coordination of timing conditions, anchors addressing any portion of streaming audio and video and actions (e.g., play, stop, replace, rotate object while translating it) specifying transformation of objects during a presentation.

3.3. LANGUAGES FOR LINK SPECIFICATION

Notwithstanding the modelling method employed to support the design of Web application, the rendering of information should be materialised over a client browser. Thus, any Web project must produce an output implementation in a language (e.g., HTML) supported by the browser.

In order to ensure the interoperability of Web applications throughout a heterogeneous network such as the Internet, most browser vendors agree[25] with use of standard technologies under the auspices of the Word Wide Web Consortium (W3C[26]). The W3C proposes several standard languages that support most of the advanced link features described in previous section, even though some advanced link features look cumbersome or not fully supported by some languages. Add-on technologies such as plug-ins, Active-X controls, or Java, can be used together with standard languages to overcome their limitation. The drawback of this approach is that programming is required for coding specific link features which may result in proprietary solutions.

Table 5 summarises some of languages promoted by the W3C and add-on technologies allowing the implementation of advanced link features over the Web. The column *add-on* refers to some technologies that are required to implement or complement the implementation of a given link feature.

[25] In fact, many browser vendors do not strictly follow the W3C recommendations for interoperability. However, proprietary languages or dialects of W3C recommendations are not covered by this work.

[26] http://www.w3c.org/

Table 5 – Web technologies supporting advanced link features.

Link Features	W3C Standards					Add-On		
	HTML2.0[27]	HTML4.0[28]	XHTML[29]	XLink[30]	SMIL	Java	Plug-ins	CGI
Unidirectional links	F	F	F	F	F	-	-	-
Bi-directional links	N	N	N	F	N	C	C	C
Embedded links	F	F	F	F	F	-	-	-
External link service	N	N	F	F	F	F	F	F
Static link attributes	N	F	F	F	F	-	-	-
Dynamic link attributes	N	C	C	N	N	F	F	C
Additional actions	C	F	F	F	F	-	-	-
One-to-many links	N	N	N	C	N	-	-	-
Semantic link types	N	F	F	F	N	-	-	-
Structural link types	N	N	N	F	N	-	-	-
Static link creation	F	F	F	F	F	-	-	-
Dynamic link creation	N	N	N	N	N	F	F	F
Fully extensible links	N	N	N	N	N	F	F	F
Timed links	N	C	C	F	F	F	F	N

Legend:
- N No support
- C Cumbersome
- F Full support
- - Non applicable

At present, there are only partial solutions for implementing bi-directional links over the Web. Chakrabarti et al. [101] present a prototype which can create a sort of go-back-link lists by asking search engines to discover referred pages. However, all automatic alternatives for implementing bi-directional links over the Web can only provide backwards navigation not avoiding broken links. In fact, the retrieval of all referring sources of a document is only part of the problem. Users may be confronted with large lists of backwards destinations with several degrees of relevance for readers. Thus, the automatic generation of bi-directional link lists may come with potential usability problems. Manual edition of bi-directional links by the document's author is a more effective solution that is only supported by XLink.

Embedded links consist of the main link strategy for the Web but several initiatives have been made to provide languages to specify links externally from documents such as XHML [122], XPointer [123] and XLink [113] [124]. However, it is important to note that the separation between structure and content through a link service is not enough to ensure the link integrity throughout the World Wide Web, it may require appropriate protocols and Web server support as discussed in [125].

Since HTML4.0[31], some attributes can be attached to links for supplying extra information about the link (e.g., media type of target document) or for specifying a particular cursor display when a user moves the mouse over a link. When employed properly, link attributes can increase usability. However, the presentation of dynamic link attributes such as those presented by Figure 8 (section 3.2.4) will require add-on tools.

Empowering link with additional actions for calling executable CGI programs on the server side was one the most significant improvement towards advanced Web applications. That was the case since HTML2.0 which innovates with forms and communication with CGI programs. More recent improvements concerning actions allow controlling browser functions such as set a link in bookmarks or specifying a window or framing to a target document.

One-to-many links are only supported in very recent proposals such as XLink. However, as mentioned in section 3.2.5, XLink does not support all strategies for presenting many destinations for a link. The XLink solution demands users to make an additional selection from the pop-up menu each time they follow a link. Other one-to-link strategies such as display

[27] (RFC1866). http://www.w3.org/MarkUp/html-spec/html-spec_toc.html (Sept. 1995).

[28] W3C Standard, http://www.w3.org/TR/REC-html40 (Dec. 1999).

[29] XHTML 1.0 is a reformulation of HTML 4.01 in XML. Available at http://www.w3.org/TR/xhtml1/ (Jan. 2000).

[30] http://www.w3c.org/TR/xlink/ (June 2001)

[31] Specification available at http://www.w3.org/TR/html401/.

destination in pop-up menu can be implemented with add-on technologies such as JavaScripts embedded in XHTML.

Concerning structural and semantic link types both can be implemented in a similar way as user defined attributes over links. However structural link types are rarely included in the implementation since they describe abstract concepts over the applications that are likely to be hidden from users. There is no special construct for specifying these link types. Both link types can be implemented by the means of user[32] defined attributes such as those provided by HTML4.0, XTML and XLink.

The W3C standard Synchronised Multimedia Integration Language (SMIL)[33] is an example of language enabling authoring of interactive audiovisual presentations integrating streaming audio and video with images, text or any other media type with temporal constraints. Advanced timed link support can also be built by using proprietary technologies and plug-ins such as Macromedia™ Flash[34]. All standard languages support manual link creation but dynamic links and fully extensible links will require programming of add-on tools.

3.4. MODELLING LINK FEATURES

Many, but not all, advanced link features presented in section 3.2 are suitable to be modelled at design time. Embedded or external linking services are features that rely on an underlying architecture that is not suitable to be described at the same level of navigation. Dynamic link attributes and full extensible links will require information that is only available at run time, so they can be hardly represented by high level navigation models. Static link attributes are most likely to be represented at presentation level. Except to these, other link features can be quite easily included into navigation models. However, a few existing modelling methods described in chapter 2 can afford their representation. The Table 6 bellow summarises the link features supported by existing modelling methods.

Table 6 – Modelling support for link features.

Link Features	Modelling Methods																		
	HDM	RMM	HDM-Lite	Araneus	OOHDM	Strudel	W3DT	WebML	WAE	UWE	OO-H	W2000	DEAHE	WSDM	Jessica	WCML	SOHDM	HMBS	Trellis
Unidirectional links	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
Bi-directional links	N	F	N	N	N	N	N	N	N	C	N	N	C	N	N	N	N	C	N
Embedded links	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
External link service	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Link attributes	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Additional actions	N	N	N	N	N	N	N	N	C	N	N	N	F	N	N	N	N	F	F
One-to-many links	N	N	N	N	N	N	N	N	N	N	N	N	N	F	N	N	N	N	N
Semantic link types	N	N	N	N	F	N	N	N	N	F	F	F	N	F	N	N	N	N	N
Structural link types	F	F	N	F	F	N	N	F	C	F	F	F	N	F	N	N	N	N	N
Static link creation	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
Dynamic link creation	N	N	F	N	F	N	F	F	F	F	N	N	N	N	N	N	N	N	N
Fully extensible links	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Timed links	N	N	N	N	N	N	N	N	C	N	F	F	F	C	C	N	N	F	F

Legend:
- N No support
- C Cumbersome
- F Full support
- - Non applicable

The graphical representation of bidirectional links over models can reduce sensibly the amount of links between nodes and thus simplify the graphical representation. RMM [40] is one of the few methods to represent bidirectional links. The inconvenience of using bidirectional links in that way is that it is not possible to set different events or conditions enabling the link forward and backward navigation. This is probably a reason that most models do not support

[32] In this case "user" means author or designer.

[33] http://www.w3.org/TR/smil20/ (SMIL 2.0 Aug. 2001; SMIL 1.0 June 1998)

[34] http://www.macromedia.com/software/flash/ (Jan. 2004)

bidirectional links. However, with the adoption in the near future of XLink standard, these modelling methods will be required to revise their approach for modelling bidirectional links or to provided mapping functions to transform conceptual models into implementations including bidirectional links.

Only a few modelling methods such as that proposed by Stotts and Furuta [88] allow the description of dynamic adaptation of document by using timed conditions. They consider in [88] a hypertext to be treated as having two levels: an underlying information structure that is created by an author and invariant, and a mutable structure that is superimposed on the former and is tuned to each user's requirement. This mutable structure can be generated dynamically changing the hypertext behaviour. Each link is associated to two time attributes, its minimum and its maximum duration, that are used to determine if a link is available or not at a time of the system execution. So, links can be included, removed and activated by the system according to time attributes. Such detailed description of timed links is not present in other models. UWE [68] support timed links by the means of conditions associated to UML StateCharts that are used to model navigation. OO-H [10] employs object-constraint language (OCL) to define several kinds of filters including timed constraints associated to links. However, most models presented in chapter 2 do not support timed links.

Some link features may remain implicit in the model and not be graphical or formally described. There are different motivations for employing link types. The first one is to improve the design by of application. By extending link types that represent a data structure (structural link types), some links can be left implicit (being induced from structural properties of the model) or can be defined intentionally or algorithmically derived. In addition, at the conceptual level, an author needs to specify a smaller number of links that will actually be present in the application, once some links can be generated automatically using the scheme definition. The second motivation is to improve usability. By extending link types that express semantic relationships referent to a specific application domain, users can better understand how the information space of the hypertext is organised.

Several link types have been subject of research but even if the authors agree on the importance of having link types they hardly agree on which types are required. Most semantic types are too domain specific to be reused with other application domain but modelling method could provide constructs to designers specify semantic over relationships. Link types supporting design activity are dependent of a methodology or require specific tools to be shown productive. With some exceptions (Strudel [59], W3DT [61], WAE [8], Jessica [81], HMBS [87] and Trellis [15]) most modelling methods for the Web distinguishes structural link type, at least as conceptual elements for the model. Such as information remains implicit in the model but is employed by tools to derivate links in instances of content.

No model surveyed in chapter 2 supports explicitly semantic link types. The reason for that is they consider semantic link types as strictly dependent of content. However, as discussed in section 3.2.6 with the gIbis system, semantic link types can be generic to the design activity. In this respect, semantic node types are better supported by models. For example, UWE [68] distinguishes stereotypes naming node constructs such as navigation context, grouped context, filtered, contexts, index, guided tour, menu and external node. We can only suppose that some UML-like notations such as DEAHE [75] that do not define any stereotype may allow authors to use stereotyped constructs to detail semantic when modelling Web applications.

3.5. CONCLUSION

This chapter presented a review of navigation constructs such as node, anchors and links. We also have discussed the advanced functionalities that can be associated to links such as actions, attributes and semantic types. Such as advanced link functionalities enrich the interaction over hypertexts and, when employed properly, may increase the usability of the applications.

Even though the implementation of some advanced link functionalities may be problematic due the underlying architecture of the Web, a significant progress has also been made in last years to provide technologies for implement advanced link functionalities over Web applications. New standards for linking such as XLink language and add-ons technologies (e.g., CGI, Java Servlets and JavaScript) enable designers to overcome the barriers imposed by the Web architecture. Notwithstanding, advance link features are not focused on a particular technology and should be considered as independent concepts for the design.

The more links include advanced features; the more modelling methods are needed to represent their functionalities and behaviour. In section 3.4, we have presented a review of modelling methods for Web in terms of their ability to deal with advanced link features. Actually, none of the modelling methods surveyed is able to represent all advanced link features. We consider this problematic since there is no mean to support their specification whenever such as features are implemented over Web applications. In this case, navigation models become imprecise or incomplete. In that sense, we suggest that more work is necessary to extends currently available navigation models for representing all linking functionalities.

4. StateWebCharts: A Formal Notation For Navigation Modelling

Summary

In the previous chapters we discussed the support given by current available models to specify advanced navigation features (chapter 3) and the lack of information those models provide concerning the behavioural aspect at the hypertext level (chapter 2). The need of more appropriate models to describe the navigation of Web applications has been the main motivation for the present work. This chapter we propose our approach for navigation modelling of Web applications.

We present here the StateWebCharts (SWC), a formal description technique based on StateCharts for describing navigation on web applications. This notation extends the classical StateCharts notation by adding more necessary concepts for describing navigation of Web applications such as appropriate semantics for states and transitions in a Web context, including notions like dialogue initiative control and client and transient activities. As well as StateCharts this formal description technique features a graphical representation thus making it easier to use for web designers and formal enough to allow rigorous reasoning about properties of navigation models.

In order to show the applicability of the SWC notation, this chapter is organised as follows:

4.1 Why a Formal Notation is Necessary to Describe Web Navigation?

4.2 Informal Presentation of SWC

4.3 StateCharts as an Underlying Notation

4.4 Formal Definition of the SWC Notation

4.5 Web Navigation Modelling with SWC

4.6 Mapping Navigation to Application Level

4.7 Discussion

4.1. WHY A FORMAL NOTATION IS NECESSARY TO DESCRIBE WEB NAVIGATION?

Model-based design is a relatively recent field over the Web but it is growing fast due the demanding need of modelling support to build more and more complex Web applications. Hypermedia and Software Engineering research have paved the way for modelling Web applications. However as discussed in previous chapters, currently available modelling methods for the Web lack appropriate notations to describe navigation. This problem is discussed in chapter 2 in terms of the lack of support to specify behavioural aspects of navigation. In chapter 3 it is discussed in terms of the difficulties to describe complex navigation constructs such as links' attributes and actions. Even though some modelling methods include high level constructs such as navigation patterns which may increase productivity, they often lead to ambiguous descriptions of requirements or impose a particular kind of implementation. These problems with informal approaches are quite well known and the most significant are [14]:

- To detect and cope with ambiguous requirements;
- To separate abstract modelling from technological details of implementation;
- To go from design to implementation phases in a structured and reliable way;
- To cope with the complexity of the design that increase exponentially with the size of the web applications;
- To cope with modifiability that is a critical point for web applications.

In order to overcome these limitations of informal models, this chapter proposes the StateWebCharts (SWC) notation for describing navigation over Web application. SWC is a formal notation that extends Harel's StateCharts [23] by adding more necessary concepts for describing navigation in a Web context. SWC adds appropriate semantics for states and

transitions, includes notions like dialogue initiative control and client and transient activities. Since SWC is a formal notation based on StateCharts, it eases the description of non-ambiguous requirements. In addition, it would be possible to apply formal verifications and check system properties (e.g., is it always possible to reach pages in one click, is it always possible to come back to the home with less than three clicks, etc.) as it is done with StateCharts models.

To gradually explain the SWC, section 4.2 introduces the concept by an informal description composed of a small case study. Section 4.3 presents an informal review of StatCharts which is the basis SWC notation. The section 4.4 formally describes the SWC. Section 4.5 presents a set of small examples demonstrating the expressive power of SWC. Section 4.6 discuss the mapping of SWC models to the application domain and presentation levels. Finally, section 4.7 presents a discussion about the limits of the SWC.

4.2. INFORMAL PRESENTATION OF SWC

The StateWebCharts is a formal notation whose main aim is to cover the hypertext level of Web applications. Similarly to other navigation models, SWC supports two main functions:

- To create views of information by grouping entities on the underling data model;
- To provide navigable relationships in-between according user needs.

The target scope of the SWC notation is presented in Figure 11. SWC models the hypertext level and it provides mapping functions to the application domain and to the presentation level which are discussed in section 4.6 and section 4.6.2 respectively.

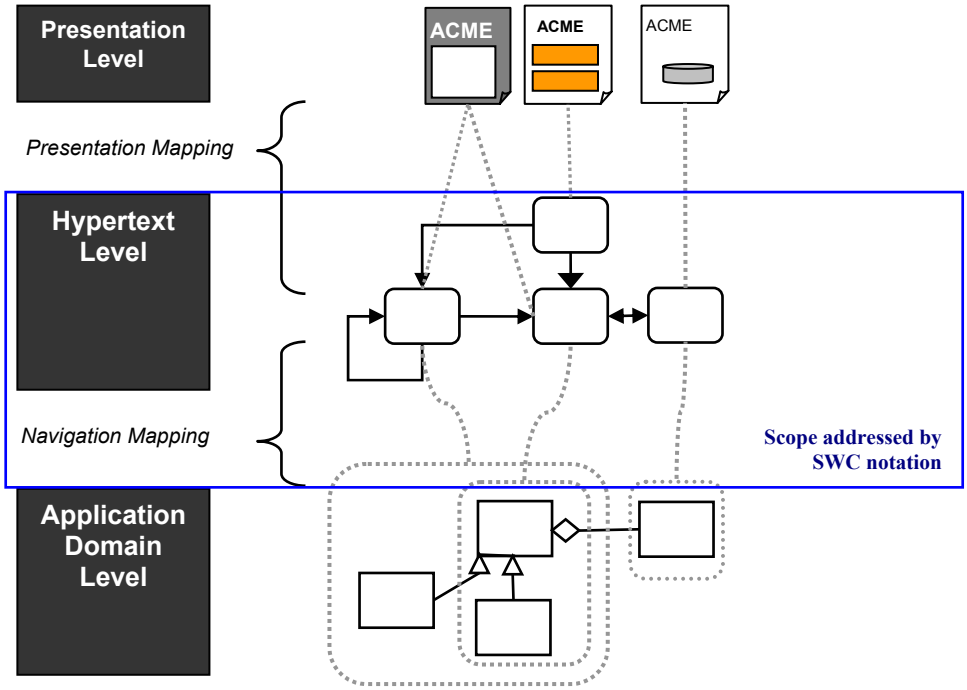


Figure 11 – Scope addressed by SWC notation.

The basis of SWC is the StateCharts notation [23] which has been extensively used to model complex/reactive systems. StateCharts can be defined as a set of the states, transitions, events, conditions and variables and their inter-relations [126]. Typically, these elements are used to describe the behaviour of reactive systems. SWC extends these elements to give appropriate semantics for states and transitions in a Web context. So, states in SWC are represented according to their function in the modelling. In a similar way, a SWC transition explicitly represents the agent activating it.

More generally, SWC states are used to specify navigation between views for information at the application domain. Views are stored or dynamically generated by the systems providing a container of objects (graphic or executable objects) in the application domain. For example, each individual Web HTML page is considered a container for objects and each container is associated to a state. Links are represented by transitions between states. Thus, each state describes which objects users can see when navigating over the applications.

A state may be associated to at most a single container. So, to attend more complex contexts of Web applications (e.g. when a container can provide different navigation views for content) SWC employs compound states describing the behaviour of multiple views of objects.

The operational semantic for a SWC state is: current states and their containers are visible for users while non-current states are hidden. Users can only navigate outgoing relationships (represented by the means of transitions in the model) from current states. When a user selects a transition the system leaves the source state which becomes inactive letting the target state to be the next active state in the configuration.

4.2.1. Case study for the Spider Web Project's Web site

Here we present a small case study for the SpiderWeb project's Web site. This Web application consists of a few pages whose goal it is to publish information relating to an international project allowing the cooperation between French and Brazilian researchers[35]. The site hosts information about members, publications, ongoing activities as well pointers to partners. It is publicly available to anyone thus no specific user need have been addressed. All pages on this Web site are updated manually having static content.

The Figure 12 presents an informal storyboard for the application. The nodes named *"Capes", "Cofecub", II-UFRGS", 'LIIHS-IRIT"* and *"IHC'2002"* represent foreign Web applications on which the site has connections.

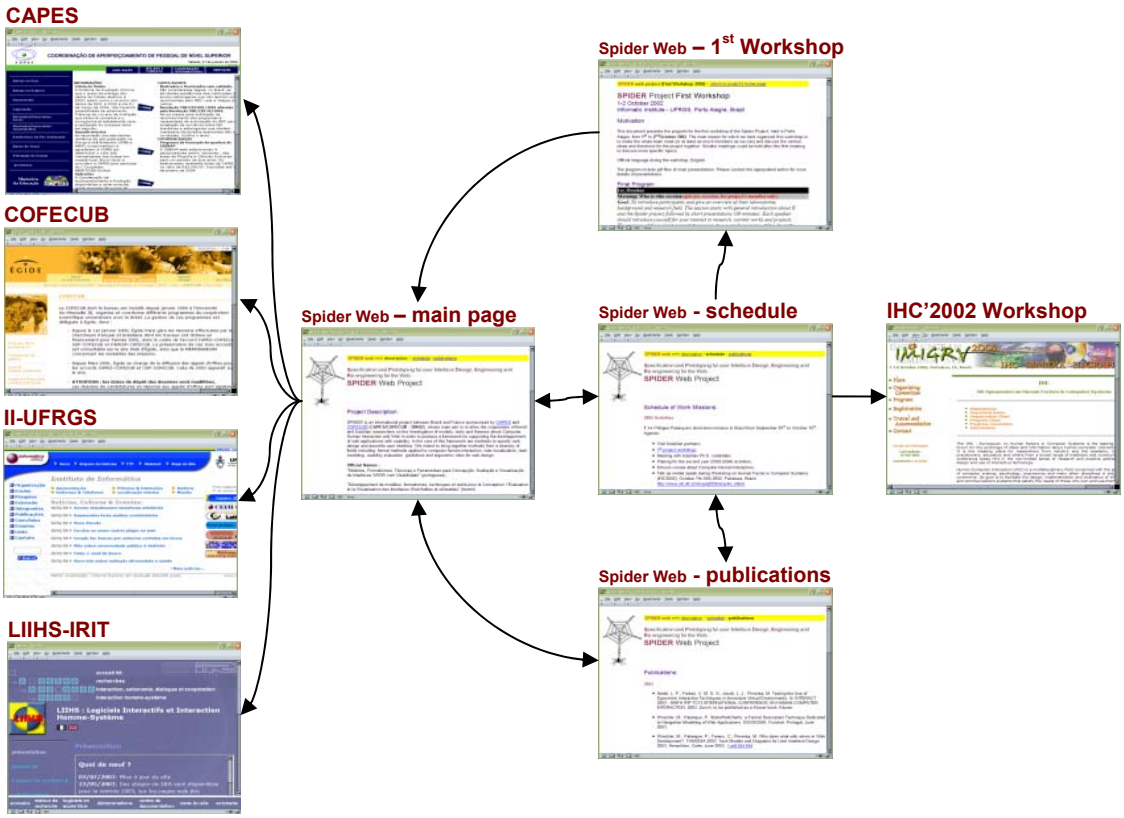


Figure 12 – Navigation Schema for the Spider Web Project's Web site.

[35] The site is available at: http://liihs.irit.fr/winckler/projects/spider/spider_intro.html (Jan 21st 2004).

4.2.2. Case study modelling

To provide a glance of SWC constructs, Figure 13 shows the equivalent navigation modelling with SWC notation for the Spider Web project's Web site. SWC is a typical exemplar of a finite state machine. At the top level a compound state named *"SpiderWeb project Web Site"* group all states taking part of the Web site. Rounded-corner rectangles (i.e., "*main intro", "first workshop", "schedule"* and *"publications"*) are *static states* in the SWC notation which means they refer to pages with static content. External or foreign Web sites (i.e., *"Capes", "Cofecub", II-UFRGS", 'LIIHS-IRIT"* and *"IHC'2002"*) are depicted by cloud-like symbols (i.e. *external states* in the SWC notation). The labelled transitions from *"t1"* to "*t13"* correspond to the links between pages.

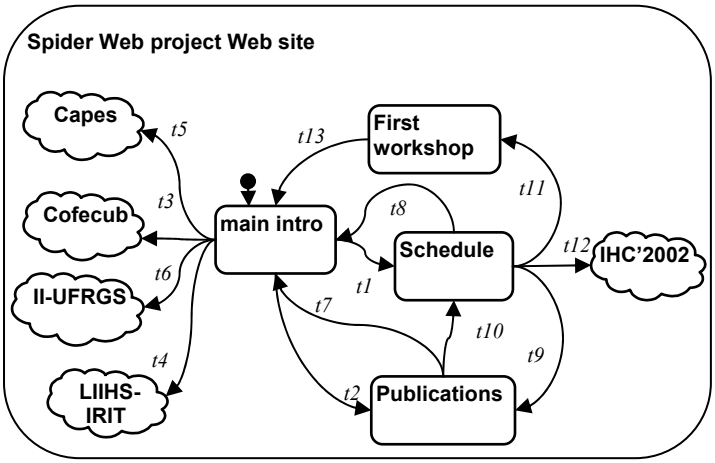


Figure 13 – Navigation Modelling with SWC for the Web site of Spider Web project.

In Figure 13, the state *main intro* is set to be the initial or default state for that model. Initial states are depicted as a decoration over a state by an arriving arrow starting with a black dot. This concept of initial state as well as many others constructs is a heritage of StateCharts.

External states are included in the notation to represent relationships to states that are out of the control of the designers since they typically do not have access right to modify their content or ensure that content will be always available. External states are drawn as part of the notation because it would not be suitable to present broken links over models and we consider that links towards other Web applications should be considered as part of the design. Similarly, SWC provides other types of states to represent dynamic content generation and executable state (transient states).

4.3. STATECHARTS AS AN UNDERLYING NOTATION

Since SWC is strongly based on Harel's StateCharts [23], this section introduces briefly the StateCharts notation (see section 4.3.1) and the extensions we have made (see section 4.3.2) to describe navigation of Web applications with SWC.

4.3.1. The original StateCharts notation

A StateCharts [23] [127] [126] model is defined by a statemachine structure composed of states, transitions, events, conditions, variables, actions and their inter-relations. States are graphically represented by rounded rectangles and transitions are represented by unidirectional arrows going from a source state to a target state (see Figure 14). Transitions are usually represented by arrows that are labelled by the expression *event/[condition]:action* (see Figure 14.a). Optionally, the label could be just a generic identification (*t1*, as in Figure 14.b). Guard conditions and actions are optional. If the guard condition is not given it is assumed to be *true*. When actions are not given, control is given to the arrival state. Parameters can be transferred from one state to another. Only events are explicitly required on transitions.

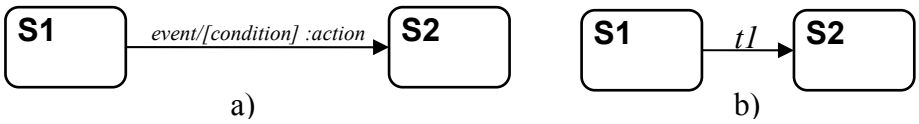


Figure 14 - Graphical representation of states and transitions.

In opposition to state diagrams which are "flat" and inherently sequential in nature [127], StateCharts propose three basic structuring mechanisms: hierarchy of states, orthogonally and broadcasting communication. The first two which are critical for modelling the navigation of Web applications are presented in more detail hereafter.

The hierarchy is represented by composition of nested states (XOR-states) thus allowing an efficient use of transition arrows. XOR-states can have exclusively one active sub-state at a time. Figure 15.a, Figure 15.b and Figure 15.c are equivalent representations concerning the transitions k and j.

- In Figure 15.b, states *A1* and *A2* are nested into the XOR-state *A*. All transitions going from a composite state are propagated to sub-states;
- Figure 15.c hides details for the XOR-state *A*. This is a useful abstraction mechanism in StateCharts to represent large hierarchies.

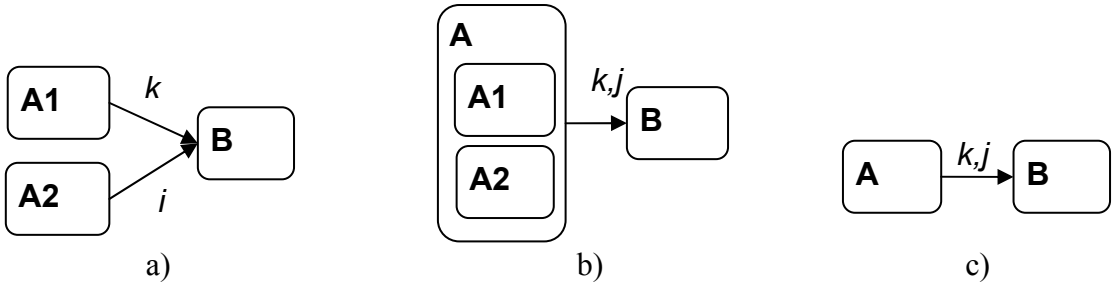


Figure 15 - Hierarchy of states in StateCharts with XOR-states.

Orthogonally is the decomposition of composite states into concurrent regions, each one representing an independent modules of a system. Each concurrent region in an AND-state is illustrated using a dashed row. Figure 16 shows three concurrent states: D, C and A. Like the XOR-state, each concurrent region can have at most one active state at a time.

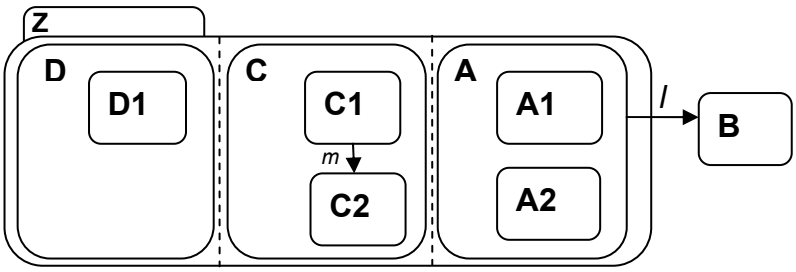


Figure 16 - Concurrent states in StateCharts with AND-states.

As with a state diagram, a StateCharts model starts in an initial state represented by an arrow with a black circle at its starting end (see Figure 17). Each XOR-state may contain at most one initial state as shown in Figure 17.a (state A1).

In both figures 17.b and 17.a, the execution starts by state *B* which is marked by an initial state at that level. If transition *p* is activated the system enters state A1 (the initial state in the composite state *A*). Transition *q* in both figures 17.b and 17.a is going from state *B* to a *final state* which is represented by a black dot inside a circle. *Final states* mean that the execution is

complete. When the enclosing state is on the top state it means that the entire state machine has completed.

There is a remarkable difference between models presented in Figure 17 concerning the use of a history state which is only presented in Figure 17.b by means of an "**H**" inside a circle. The role of a history state is to keep the information about the most recent state inside a compound state. Consider the example in Figure 17: let us to assume the state B is the initial configuration with the activation of the following sequence of transitions: $p \rightarrow k \rightarrow l \rightarrow p$. Such as a sequence will produce the configurations: (A, A1) → (A, A2) → (B) → (A, **A1**) in Figure 17.a; and (A, A1) → (A, A2) → (B) → (A, **A2**) in Figure 17.b. The first time the state A is entered there is no previous active state in the set, so the history state in Figure 17.b transfers the control to the initial state A1. The transition k occurs and then the state A is left when in state A2 by the transition l. When the transition p is reactivated, the history state tracks the last current state in the set (i.e. A2) and transfers the control to A2. Whenever the previous configuration inside the state A in Figure 17.a, the transition p will always transfer the control to the state A1.

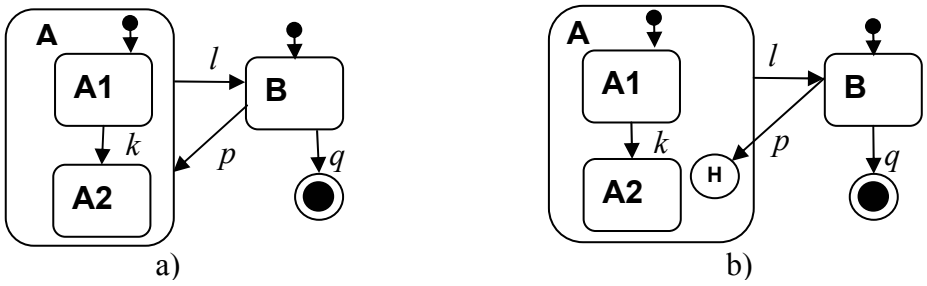


Figure 17 - Initial states, history states and final states in StateCharts.

The operational semantics for StateCharts is given by a sequence of steps. At each step the state machine evaluates a single transition and may assume a new state configuration (the set of currently active states). When an event happens the system transmits it to the transition associated to the triggered event. The corresponding guard condition is then evaluated and if it is true the statemachine sets the target state as active.

An optional activity can be added to the transition label indicating which activity will take place when the transition happens. The triggered activity can in turn be received by the system as another event to trigger other transitions creating compounding transitions. The broadcasting mechanism in StateCharts is represented by events that are associated to more than one transition. In that case, when an event happens, all transitions associated to the triggered event are evaluated and executed if the guarding conditions are true. In classical StateCharts, activities and events are considered to be instantaneous (they take no time to perform).

4.3.2. SWC extensions to StateCharts

There are numerous extensions of StateCharts to support different modelling needs and have different semantics [126]. Purposely, the basic elements and the operational semantics presented in the previous section is the same employed by SWC. However SWC also introduces the following functional semantics for StateCharts elements to describe the navigation of Web applications:

- A mapping function allowing association between content and state;
- The graphical representation of the states according to their function in the modelling (e.g. states representing static or dynamic content generation);
- A mapping function allowing the determination of transitions as complex links supporting conditional activation and customised actions;

- A mapping function identifying transitions suitable to be activated by users or by the system;
- A mapping function allowing the mapping of user defined stereotypes to states and/or transitions;

The operation semantic of SWC defines that the content of active states in a configuration and their corresponding outgoing transitions are visible to the users. When a user selects a link, the corresponding transition is validated and the system changes the configuration to the target state returning the appropriate content to the user.

A basic state can be associated to one or no container which is assumed in any target language for the implementation of the Web application (e.g. HTML, XML, PHP[36], etc.). Empty dynamic states are possible when unpredictable content is generated on-the-fly. The SWC notation employs state types for describing special functions performed by a state in a SWC state machine. Figure 18 shows a graphical representation of all state sub-types.

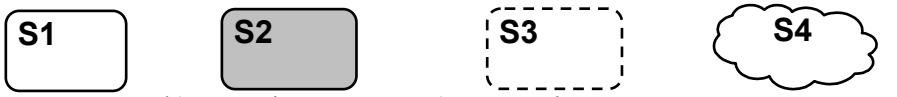


Figure 18 - Basic state sub-types in SWC notation.

Static states (Figure 18.a) are the most basic structures to represent information in SWC. A static state refers to a container with a static set of objects; once in a static state the same set of objects is always present. However, the objects it contains are not necessarily static by themselves; they could have dynamic behaviour as we usually find, for example, in applets, JavaScript or animated images. Static is the default type.

Transient states (Figure 18.b) describe a non-deterministic behaviour in the state machine. Transient states are needed when a single transition cannot determine the next state of the state machine (see Figure 22 for an example). The formal definition for a transient state says that only completion or system events are accepted as outgoing transitions. They can be used to refer to server-side parts of web applications, such as CGI and Java Servelts programs. Transient states only process instructions and they do not have a visual representation towards users.

Dynamic states (Figure 18.c) represent content that is dynamically generated at runtime. Usually they are the result of a transient state process. The associated container of a dynamic state is empty. The semantics for this state is that in the modelling phase designers are not able to determine which content (transitions and objects) will be made available at run time. However, designers can include static objects and transitions inside dynamic states; in such case transitions are represented, but designer must keep in mind that missing transitions may appear at run time and change the navigation behaviour.

External states (Figure 18.d) represent information that is accessible through relationships (transitions) but are not part of the current design. For example, consider two states A and B. While creating a transition from A to B, the content of B is not accessible and cannot be modified. Thus B is considered external to the current design. Usually they represent connections to external sites. External states avoid representing transitions with no target state, however all activities (whatever it is entry, do, exit) in external states are null.

Events are classified in the SWC notation according to the agent triggering them: user (e.g. a mouse click), system (e.g. a method invocation that affects the activity in a state) or completion (e.g. execute the next activity). A completion event is a fictional event that is associated to transitions without triggers, e.g. change the system state after a timestamp.

[36] PHP (which stands PHP: Hypertext Preprocessor) is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. Available at: http://www.php.net/ (Feb. 2004).

Fictional completion events allow providing the same representation for all transitions in SWC machines. This classification of event sources is propagated to the representation of transitions. Transitions whose event is triggered by a user are graphically drawn as continuous arrows (Figure 19.a.) while transitions triggered by system or completion events are drawn as dashed arrows (figure Figure 19.b).

Figure 19 - Graphical representation of SWC transitions.

In Figure 19 one can promptly identify which owns control on the activation of a transition: the system (transition *t2*) or a user (transition *t1*). In order to be able to use the SWC models to perform usability evaluation, the fact that a transition is related to a user event or not is critical. Thus, SWC puts in a single graphical representation those transitions (*completion* and *system* transitions) that are not triggered by the users. If explicit representation is required for distinguishing between completion and system events a full label for transition can be included (e.g., *t=completion/[true]:action*).

SWC employs pseudo-states (i.e. *shallow history, deep history, end state* and *initial state*) as defined by StateCharts. Figure 20 below presents all elements of the SWC notation.

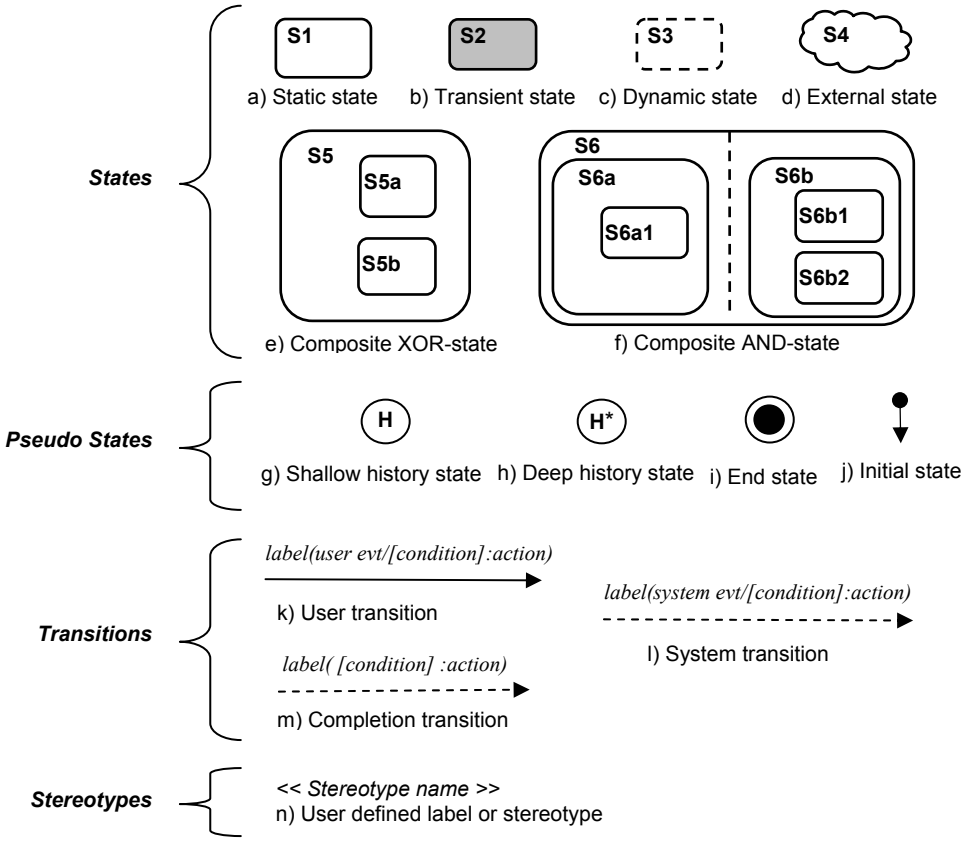


Figure 20 - *Graphical representation of StateWebCharts elements.*

In addition to *states, pseudostates* and *transitions*, Figure 20 also presents a category namely *stereotypes* (see Figure 20.n). SWC *stereotypes* are a kind of user defined label that allows designers to enrich the semantic of a state or a transition. By means of *stereotypes* it is possible to define typed links according to their *semantic* as discussed in sections 3.2.6 (see chapter 3). The concept of *stereotypes* was borrowed from UML [27] since there is no equivalent structure in the proposal of Harel [23] for the StateCharts.

4.4. FORMAL DEFINITION OF THE SWC NOTATION

In order to make it easier to understand the formal definition of SWC section 4.4.1 presents the formal definition of StateCharts. We then formalise in section 4.4.2 the extensions proposed for SWC.

4.4.1. Formal definition of StateCharts

A formal definition of StateCharts (or the statemachine) is:

S is defined as the set of states;

$\rho : S \rightarrow 2^S$, is the map function that associates each state to its sub-states, where $\rho(s)=0$ means *s* is a *basic state* with no children inside;

$\psi : S \rightarrow$ **{AND / XOR}** is the function that defines whether $s \in S$ is a *composed AND / XOR* state or not

H is the set of *history symbols*

$\gamma : H \rightarrow S$ is the function that match *history symbols* to *states*

$\delta : S \rightarrow 2^{S \cup H}$ is the *default function* that defines the initial states in *S*

Φ is the set of final state symbols

V is the set of variables

C is the set of *conditions*

E is the set of *events*

A is the set of *actions*, where each action is a term of a language $£_a$, which defines the allowed operations in a machine. Actions can also be associated to a state where $A=A_{entry} \cup A_{do} \cup A_{exit}$, where $\forall a \in A$, A_{entry} is the action executed when entering a state, A_{do} is the main action executed by it, A_{exit} is the action executed before the state is left.

$\Sigma : S \rightarrow A$ is the mapping function that associates a state to its activities

$L = E \times C \times A$ is the set of *labels* on transitions

$T \subset 2^S \times L \times 2^{S \cup H \cup \Phi}$, is the set of transitions represented by a source state (2^S), a label (L) and a target state ($2^{S \cup H \cup \Phi}$)

StateCharts may have two types[37] of *History states*: (*shallow* and *deep* history). The difference between them concerns how many levels are taken into account to determine the next stable statemachine configuration. A *shallow history states* the statemachine assumes the next stable configuration as the most recent active state at the same level of the history state. A *deep history state* will look for the most recent configuration inside the most recent active state at the same level. *Shallow history* states are represented by an "*H"* inside a circle (see Figure 20.g) while *deep history states* are represented by and "*H*"*inside a circle (see Figure 20.h).

4.4.2. Formal definition of StateWebCharts

States and transitions in SWC are represented according to their function in the model. The basis of SWC modelling is a state machine, as described above, in addition to the following elements:

P is the set of containers storing information content that will be exhibited as a whole in a web application (generally web pages). **P** is defined by the set (**o**, **k**) where **o** is the set of objects

[37] Some StateCharts dialects only take into account *shallow history states* calling them simply *history states.*

(text, sound, image, etc) it contains and **k** is the set of anchors in the set. The set **P** include an empty container

$\Omega : S \rightarrow P$, is the map function that associates each state to a container

$M : k \rightarrow E$, is map function that associates a anchor to an event in the state machine

Y = {static/transient/dynamic/external} is the set of sub-types for a basic state

$\varpi : S \rightarrow Y$ is the function that maps a sub-type to a basic state in the state machine $\forall s \in S \,.\, \rho(s) = 0, \ \varpi \neq \varnothing, \exists y \in Y, \varpi(s) = y$

W = {user/system/completion} is the set of events sub-types where each event type indicates an agent triggering the event in the system

$E = W_{user} \cup W_{system} \cup W_{completion}$ is the redefined set of event in a system

Z is the set of user defined labels (or stereotypes)

$\lambda t : T \rightarrow Z$, is the map function that associates a user label to a transition

$\lambda s : S \rightarrow Z$, is the map function that associates a user label to a state

A container *P* is considered as a compound document according to W3C DOM[38] definition, which may contain objects (text, images, etc) as well as other documents.

By the function $\varpi : S \rightarrow Y$ we make each basic state $s \in S$ assume an appropriate sub-type *static, transient, dynamic* or *external*.

4.5. WEB NAVIGATION MODELLING WITH SWC

This section describes the most important features related to navigation design for web applications and their corresponding representation with SWC notation, when it is applicable. Sections 4.5.1 to 4.5.6 describes the use of the SWC notation. The final sub-sections correspond to link features discussed in chapter 3.

4.5.1. Dynamic content generation

A particular feature of web application is the dynamic generation of pages on an application by server-side applications. Dynamic pages do not exist on the Web server and they represent unpredictable content for the page but it does not exclude the possibility to represent required transitions for the design.

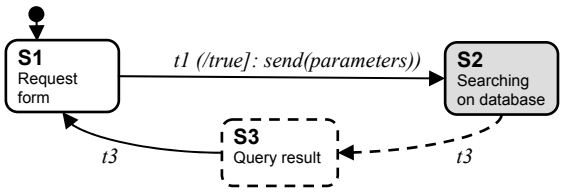


Figure 21 - Query search.

Figure 21 shows a classical example of dynamic content generation as a result of a query in a database. Notice that the *dynamic state S3* has a *user transition* that allows the user to return to the request form (state *S1*). It is also important to note that, at run time, the page resulting by the database searching can include links that are not represented in the model and may alter the navigation on the web application.

[38] http://www.w3.org/DOM/

4.5.2. System-driven navigation (the use of transient states)

In many case the combination of events plus conditions determines the next state. However it not true for all cases. Consider the case of user authentication in Figure 22. In this example, the event *press button* (to send user name and password) in transition *t1* does not count to determine whether the user will get access to the system or not. But it is the result of processing the transient state *S2*. Notice transitions *t2* and *t3* going out from *S2* presenting system events.

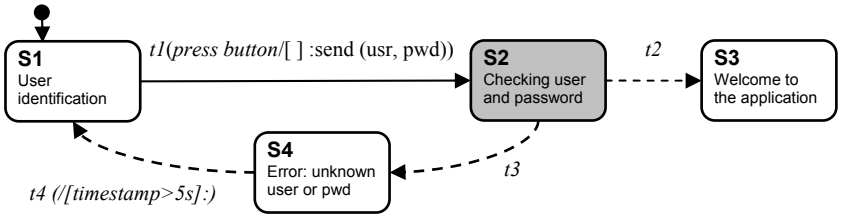


Figure 22 - Example of a simple user authentication.

In most cases, the user will send additional information filling in forms or following parameterised links to a server-side application (represented by a transient state) that will execute some processing and then send the appropriate answer back to the user.

4.5.3. Frames

Frames are elements that split the browser's windows into two or more concurrent visual areas were each region can load and display a different document. *Frames* were introduced as a standard in HTML 4.0. Links in a frame region can alter the exhibition of documents in another frame region. *Frames* are modelled in SWC with AND-states where each orthogonal region represents an individual frame as shown in Figure 23. When entering state *A*, two concurrent regions are activated *A'* and *A''* which pass the control to their initial states *B* and *C*, respectively. When the transition *t2* is fired the configuration in region *A''* changes arbitrarily to states *D*, region *A'* maintains its configuration.

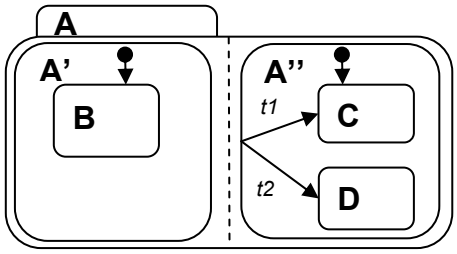
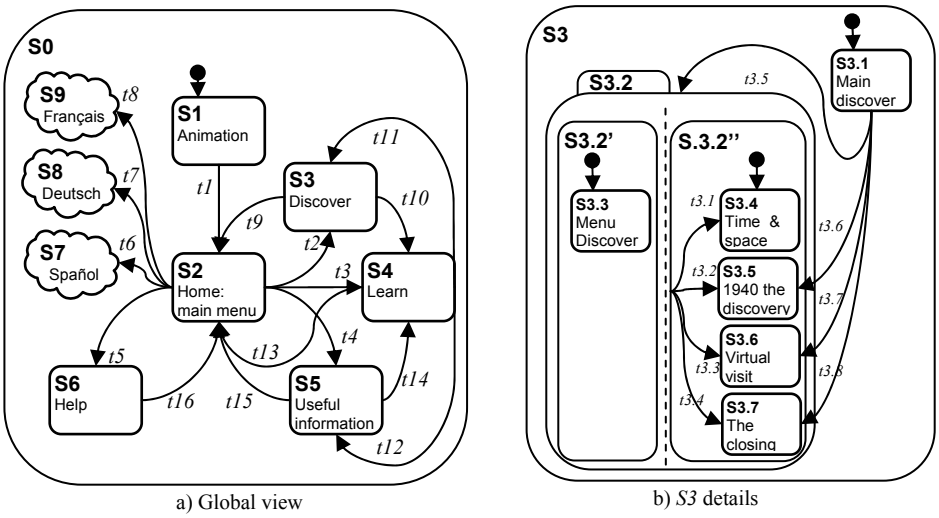


Figure 23 - Concurrent visual area representations (frames).

4.5.4. Modularisation

The number of pages on most web applications increases very quickly and the representation of documents and links became a problem in flat-notation such as automata. In addition, large projects must be cut in small part and distributed among the member of development team. The modularisation is also required to deal with the complexity during the development. SWC takes benefits from the multi-level hierarchy from classical StateCharts to better manage large web applications. Figure 24 presents a partial modelling for the web site *The Cave of Lascaux*[39].

[39] http://www.culture.fr/culture/arcnat/lascaux/en/



a) Global view b) *S3* details

Figure 24 - Hierarchical view for the Web Site '*The Cave of Lascaux*'

Figure 24.a presents a global view for the application which contains nine states. Some of those states are of compound states and whose internal content is not represented in higher level. For example *S3* is a composite state whose details are shown in Figure 24.b. In such as approach, composite states represent classes of pages which share the same structure. Sub-states inherit relationships from their parents. For example in Figure 24.a, the transitions (i.e., *t9 a*nd *t10*) going from state *S3-Discover* to states *S2-Home-main menu* and *S4-Learn*, respectively are shared by all *S3* sub-states (i.e., *S3.1, S3.2, S3.2', S3.2'', S3.3, S3.4, S3.5, S3.6* and *S3.7*). The states at the left are instances of classes of pages that have their own navigation. Due to space constraints only state *S3* is detailed in this model although S3 and some sub-states (*S3.4, S3.5, S3.6* and *S3.7*) are at their turn suitable to be decomposed.

4.5.5. Dialogue control modelling

Modelling dialogue control means identifying who (system or user) causes events changing the interface. SWC explicitly represents system interaction (by *system* and *completion* events) and user control (by *user events*). A typical example of a system event is a timed transition used to redirect Web pages. In the Figure 25 users start at state *S1* which contains two associated transitions: *t1* and *t2*. The transition *t2* represents a system event that once activated, will change system state to *S2* after five seconds [5s] in *S1*. Users can also cause a transition by selecting a link associated to user event *t1*.

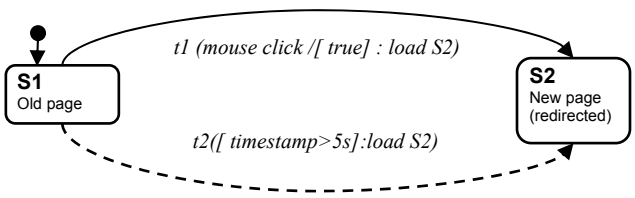


Figure 25 - User x System dialog control.

4.5.6. Client-side and server-side execution

Web applications were originally built over a client/server architecture where the server-side is responsible for all processing leaving to the client-side (the web browser) just the display of the content information. The advent of new technologies such as JavaScript, Java and Active-X for example, put on the client more interactive than just display functions. We define client-side execution as any processing changing the state of the application without communication with a web server. Server-side execution at its turn, is defined as any instruction processed on a web server following a client's request.

Transient states and *system/completion* transitions in SWC are suitable to describe executable states and system initiative on web applications however such elements say nothing about where (on the client- or server-sides) it occurs. SWC do not impose a particular architecture for the design and a modelling can be quite easily implemented using as thin-client architecture (no processing in the client-side) or a robust-client (full client-side functionality). However at this time we have not included a description about how to model objects in a container, thus we could consider that *transient* states are always on the server-side.

4.5.7. Bi-directional links

SWC does not support graphically bi-directional links. The reasons for that is bi-directional transitions could be a source of ambiguity in the model since a single transition would hold two events and two different actions (i.e., navigate towards and backwards).

However SWC can supply information on which (incoming) transitions arrives at a state. This kind of information is formally given by the statemachine and can be used to implement backtracking navigation on the user interface.

4.5.8. External navigation specification

SWC provides a means to separate abstract modelling navigation specifications from other models required to describe the application domain and presentation issues. The relationships between these three levels are made by optional mapping functions. The SWC notation can be employed independently of other models thus assuring flexible modelling and reuse of specifications.

Implementation details concerning the physical implementation of embedded or external linking services are out of the scope of the notation since as previously discussed in section 3.4, such features rely on an underlying architecture which is not suitable to be described by abstract navigation models.

4.5.9. Static link attributes

Link attributes refer to static attributes (e.g. link colour, destination media type) or *dynamic attributes* recovered at run time (e.g. network speed, link availability). The attributes may have an influence on users by helping them to decide to follow a link or not but, in any case they affect the navigation schema behind the application. In addition, their presence over a Web application should be considered to be more likely described at the presentation level (see section 3.2.3). Therefore, these elements are not supported by the SWC notation.

4.5.10. Additional actions

SWC provide supports for additional actions by means of constructs purposely namely *actions* that can be associated to *states* and *transitions*. All possible additional actions are defined in a language namely *£a,* which enables the construction of executable expressions in the system. SWC does not define a particular *£a* language to be used of Web applications but we used in our examples the convention *actionName(attribute_1, attribute_2, ... , attribute_n)* where *actionName* is the method invocation and *attribute_1* to *attribute_n* is the set of attributes passed as arguments. For example Figure 22 presents a transition namely *t1* having an additional action namely *send (usr, pwd)*.

4.5.11. Link cardinality (one-to-many end points)

At the present the SWC notation does not provide a construct for implementing *one-to-many* end points.

4.5.12. Semantic link types

Semantic link types are supported by SWC notation by means of *stereotypes*. Stereotypes are represented as user defined labels inside double quotations (e.g. << >>). For example Figure 26 presents an example for an academic Web site where transitions *t9* and *t10* from the states *S6 Lectures* to *S3 Professor* are assigned by the stereotypes <<*assistant*>> and <<*chair*>>, respectively. It is important to note that the use *a stereotypes* remains optional and it may be marginal for most applications.

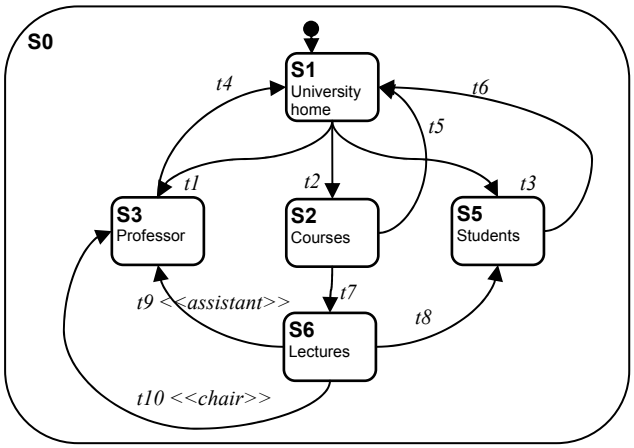


Figure 26 – Employing stereotypes in SWC models.

4.5.13. Structural link types

When analysing how pages are related to each other on web applications we can observe three different types of links: a) internal-pages link, b) inter-page links and c) external links. *Internal-page* links relate different parts of a same web page which can be very helpful for long documents. These links present the same semantic behaviour of scroll bars in windows browsers. Therefore, at first sight we can consider irrelevant to include the specification of such elements into navigation design. If required, *internal-links* can be easily represented with SWC by decomposing the page in a composite state and create links between the sub-parts of the document as presented by Figure 27.a. As we can see in Figure 27.a is a *spaghetti*-link interconnection between all sections of a same document.

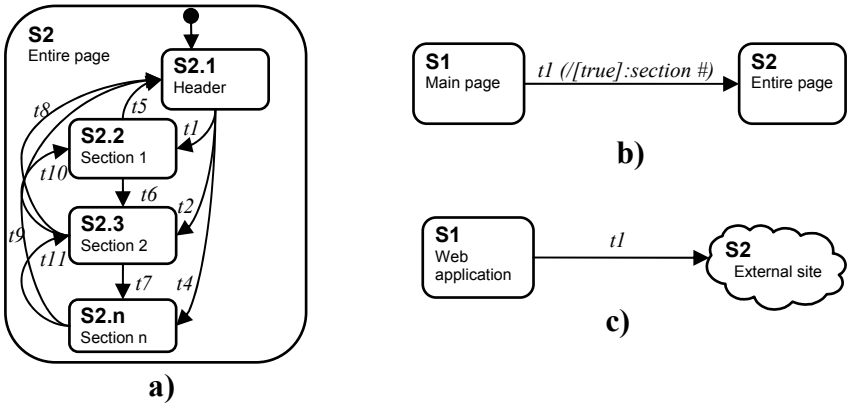


Figure 27 - Links types: a) internal-page, b) inter-page and c) external.

Inter-page links are the most classical example; it means a simple connection of two pages belonging to the same web site. We can see in Figure 27.b how two pages can be connected by an *inter-page* link with passage of a parameter that indicates the subsection in the document to be displayed. Figure 27.b is a preferable representation for the same problem described above (*internal-pages*) because it increases the legibility of diagrams.

External links are links connecting the web application with foreign web sites or non-relevant parts of the web application for the current design. Even though its name makes a reference to a link, this concept is treated as a state because it is not possible to represent transitions without a destination in SWC, even though the transition makes references to an external site.

4.5.14. Link creation

As discussed in section 3.2.8 (see chapter 3), links can be differentiated according to the way their source and target is specified. Thus we have *static links* (i.e. both source and target are defined at design time), *dynamic links* (i.e. when only the anchor is defined at run time), and *fully extensible links* (i.e. when both source and target are dynamically determined at run time).

SWC is able to represent *static* and *dynamic links*. *Static* links are represented by means of transitions. Since the creation of *dynamic links* requires information only available at run time, SWC only provides constructors to indicate where *dynamic links* may appear during the navigation. *Dynamic links* are represented by means of *dynamic states* as those represented in Figure 21.

Since *fully extensible links* are calculated dynamically by the system at run time, there is no possible representation of them with the SWC notation.

4.5.15. Timed links

Timed links can be easily specified with SWC by means of guard conditions associated to transitions as those presented in Figure 25 and other SWC constructs allowing to create sequence and concurrence by means of compound states. SWC also gives the possibility to add additional actions over transitions which allows to introduce others advanced functions for executing temporal transformation during a presentation (e.g. replacing the current page using a cross fading effect).

4.6. MAPPING NAVIGATION TO APPLICATION LEVELS

This section presents how to map navigation models to the domain application level and presentation level.

4.6.1. Mapping navigation to application domain level

We consider the following situations for mapping navigation specification to the underlying domain application level of the Web applications:

- *Document-based structure*: in this case the application domain is covered by a set of documents without a common structure (e.g. single Web pages);
- *Service-based structure:* in this case the application domain is only accessible by executable scripts or programs on the server side (e.g. CGI programs or PHP scripts);
- *Highly structured application domain:* in this case Object-Oriented techniques such as UML class diagrams are employed to describe the application domain.

If the application domain is defined by a *document-based structure,* each individual file for a Web page is mapped to a *basic* state.

Similarly when the application domain is defined by a *service-based structure* we map the script files to *transient* states. In such as a case, the *transient* state represents all the treatment performed to transform the domain application data to the Web application; for example to update a database of a legacy system. The result of a *transient* process is mapped to *dynamic states* if new content is generated.

Complex Web applications present *highly structured application domain* that are better described by means of Object-Oriented models. Our approach in that case is very similar to some modelling methods such as OOHDM (section 2.4.5) and DEAHE (section 2.4.13) that employ OO-like notations to describe the application domain level and StateCharts notation to cover the hypertext level. In this case, a class element in a domain application model is not mapped to a single state but to a set of states describing the behaviour of the class concerning the navigation. SWC is in its essence a StateCharts diagram with a richer semantic for Web applications; so that SWC models can be easily mapped to class diagrams following an approach as those suggested by OOHDM and DEAHE. Chapter 5 describes how SWC models can be employed to describe the behaviour of class diagrams.

For many Web applications, the three situations for mapping navigation to application domain would be required. The SWC can adopt any situation using the same basic constructs to attend such application needs.

4.6.2. Mapping navigation to presentation level

The SWC notation is completely independent from the presentation level. The linking between navigation and presentation is the container of objects assigned to each SWC *state*. So, the presentation aspects of objects should be defined externally of navigation models. Several techniques are possible to define the presentation level:

- By attaching templates defining the presentation to the container of objects; for example, templates *Cascading Style Sheets - CSS*[40]; or,
- By defining presentation concerns together with content a in a HTML-like style where content and presentation aspects defined by tags in a same file.

Such independency of SWC navigation models and presentation allows the reuse of navigation specification to several presentations of documents without sacrificing device-independence.

4.7. DISCUSSION AND RELATED WORK

Designers can consider modelling navigation as a cumbersome task but it provides them with many benefits. Those benefits are the same as the ones for modelling dialogue in interactive systems. Modelling can help designers during design phases by defining formally the requirements, dealing with navigation in an abstract way, providing multi-levels of details as well as providing support for testing prior to implementation. Support from modelling can also be obtained in later phases via, for instance, support for verification prior to implementation [128].

StateCharts and StateChart-like notations have already been widely used for modelling various aspects of web applications. For instance, they have been previously used for navigation modelling of hypertext/hypermedia applications [16] [17], web applications [19] and even WIMP interfaces [129]. Howevere these previous work shows some limitations in the expressive power of StateCharts for handling specific and critical aspects of web application modelling. For instance, it is not possible, using StateCharts, to represent who (the user or the system) is at the source of an event to be received by the application. Even previous work that focused on navigation modelling for web applications, such as [19], do not clearly explain how StateCharts can be effectively used to model other Web applications feature such as dynamic content generation.

For this reason, we have extended StateCharts to the StateWebCharts notation (SWC) that provides dedicated constructs for modelling idiosyncrasies of states and transitions in Web applications. Our aim is to provide a visual notation easy-to-apply for web designers and formal

[40] Available at http://www.w3.org/Style/CSS/.

enough to be subject of automatic verification, thus supporting designer's activity throughout the design process. Most elements included in the SWC notation aims to provid explicit feedback about the interaction between users and the system. The fact that the StateCharts notation is part of the UML notation pack may help designers having knowledge about UML notations to employ SWC.

SWC can only describe navigation. It is not suitable to model domain application data neither graphical aspect related to presentation. However, by associating containers to states and transitions to links we provide a mechanism to match SWC models and underlying data models. The presentation level is not directly addressed by SWC neither explicit mapping is supported by the notation.

5. Development Process

Summary

A development process is necessary to help developers to manage the complexity, minimise risks of development, monitor the progression, deal with maintenance and evolution of the Web applications. Currently, there is no consensus on which phases of development are required neither which life cycle describes better the development process of Web applications.

Rather than survey several approach for development process, this chapter is limited to discuss the steps required to produce navigation models with the StateWebCharts notation. The approach for navigation modelling presented in this chapter is largely inspired from the *authoring-in-the-large* and *authoring-in-the-small* approach introduced by the HDM method [13]. Our approach does not intend to cover all phases of development of Web applications but we show how it can be integrated into the life cycle of development.

This chapter is composed of the following sections:

5.1 Introduction

5.2 A Method for Navigation Modelling

5.3 Discussion

5.4 Conclusion

5.1. INTRODUCTION

In order to make the design and development of Web applications manageable, systematic and repeatable, we need a process that outlines the several phases of development of Web applications. A development process is necessary to help developers to manage the complexity, minimise risks of development, monitor the progression, and deal with maintenance and evolution of the Web applications. Indeed, the lack of guidance by current modelling methods during the development process is pointed out as one of the most important problem for the Web development [92] [93].

The development over the Web is complex due to many factors such as the evolving nature of applications [7], the multidisciplinary nature of development team [130], the competitive points of views for the application, complexity of user requirements [131], unrealistic and narrow schedules for the development [132] [133]. Web applications require regular maintenance to update content, follow a particular business workflow, and include new features to support tasks and/or user needs. Besides, Web development is often performed by several people (from different backgrounds such as graphic designers, marketing, and editorial) at a time and thus in a parallel manner. Another important feature of a Web project is the time to delivery or to update the application can be as short as few hours. Designers can change almost immediately their design, so this makes Web development projects highly evolutionary in nature [7].

Currently, there is no consensus on which phases of development are required neither which life cycle describes better the development process of Web applications. Many modelling methods discussed in chapter 2 focus on a particular domain of application in order to automate some steps of the development process, increase the maintainability or reuse of components. For example, the WSDM method [77] (section 2.4.14) tackles many usability concerns and user requirements which are quite often neglected by other methods. However WSDM is only suitable for the design of kiosks-based Web applications that mainly provide a means to navigate an information space. The OOHDM [11] (section 2.4.5) focus on hypermedia-based Web applications proposing a four-step design process supporting the reuse of components. Araneus [54] (section 2.4.4) and WebML [12] (section 2.4.8) are two examples of methods ensuring high productivity of development for Web applications base on an underlying database (e.g. data-intensive Web applications) but they provide little support to deal with informational-based applications.

Rather than survey proposals for the development process of Web application, this chapter intends to discuss our approach for navigation modelling. It is not our intention to provide a systematic development process covering all the aspect required for the development of Web applications. Our aim is just to demonstrate how SWC can be used in practice of Web design.

The development process we propose here is focused on the specification of the navigation modelling so, it does not intend to cover the specification of requirements engineering. We consider that, by the beginning the navigation modelling, designers are aware about informational and functional requirements as well as project constraints. We consider that information is available somewhere in a format that is understandable for designers but we do not impose a particular notation to represent them. In the case study presented in chapter 6 we suggest to employ Concurrent Task Tree model (CTT) [134, 135] to model user tasks and functional requirements. For informal requirements and project constraints we used to use the document-type for informal specification of Web Applications, which is presented in Annexe B. However, even though these documents and models are helpful during the requirement engineering phase they should not be considered as preconditions to employ the SWC notation.

This chapter is organized as follows: the section 5.2 presents our approach for modelling navigation, section 5.3 discusses how to exploit navigation models, notably SWC models, during the life cycle of Web applications; and section 5.4 presents our conclusions.

5.2. A METHOD FOR NAVIGATION MODELLING WITH SWC

The approach proposed hereafter consists of a set of steps describing how to employ the StateWebCharts (SWC) notation for specifying Web applications. Such as an approach considers that the phase of requirements engineering has been completed and the following information are available:

- Description of user roles;
- List of informational and functional requirements;
- Description of user tasks;
- Class diagram describing the underlying application domain.

We do not propose a specific notation for describe user roles, requirements or the underlying application model. For the description of users roles, informal and functional requirements we used to use the document-type template in Annexe B. To model user tasks we suggest the use of Concurrent Task Tree model (CTT) [134, 135]. However, the underlying domain application model should be provided in the form of a UML class diagram. The use of these notational or templates, or any other model, is not a precondition for building navigation models with SWC notation; even though we encourage the use of models for describing requirements and design intentions.

Our approach for navigation modelling with the SWC notation is largely inspired from the *authoring-in-the-large* and *authoring-in-the-small* approach introduced by the HDM method [13]. So, the navigation specification is made in two phases. The first phase is made up by defining a *navigation schema* to describe a generic navigation for the application. The second step is subsequent refinement of the navigation schema into a *detailed navigation* by taking into account the content.

The *navigation schema* is based on mapping the class diagrams representing the domain model into navigation models to support user tasks. The class diagrams made the connection between the navigation model and the underlying application domain. Each class diagram is considered in terms of their public attributes and public methods that can be used by an end-user to explore the information space. We use StateWebCharts notation to describe the behaviour of each class according user tasks. Additional transitions and states can be included into the model

to represent single pages, external relationships, index, guide-tours or any other navigational requirements.

The second phase, *detailed navigation*, is made up by detailing the *navigation schema.* At this step we include transitions (i.e. relationships) and states that are based on the content. The *detailed navigation* intends to provide a navigation modelling that is very close to the implementation.

A *navigation schema* is made up by following the steps:

1) Defining a class model for the navigation;

2) Creating SWC diagrams to describe class behaviour;

3) Creating SWC diagrams to describe user tasks;

4) Integration of individual SWC diagrams into a single model;

5) Including navigation paths;

These steps are not necessarily sequential. For example, designers can first develop individual SWC diagrams and then generalize it to produce class diagrams. These steps complete the specification for the navigation.

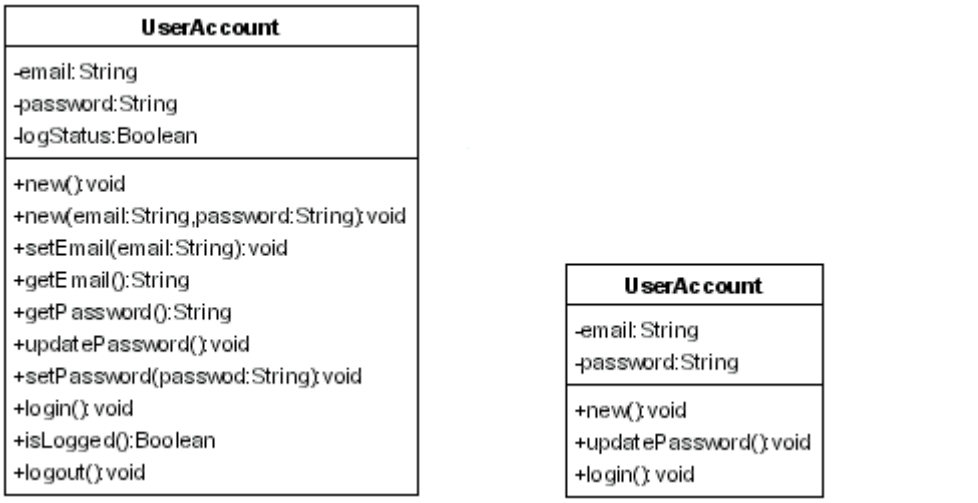
The next step covers the *detailed navigation* phase:

6) Defining content-dependent navigation

These steps are detailed hereafter.

5.2.1. Defining a class model for the navigation

To model the navigation, we consider an existing UML class diagram representing conceptual elements for the application domain. We start by restricting the visibility of attributes and methods of a class diagram that are not necessary for the navigation. This restriction intends to focus on the on elements that will be employed for specifying the navigation. For an example, Figure 28.a shows the equivalent domain class for the navigational class presented by Figure 28.b which has access restrict for attributes and methods. In the example shown by Figure 28, only the methods *new(), updatePassword(),* and *login()* are relevant for the navigation. The relationships between classes are not taken into account at this step.



a) Domain application class **b) Class showing attributes and methods used in navigation**

Figure 28 – Class diagrams and methods that are relevant for navigation.

5.2.2. Creating SWC diagrams to describe class behaviour

After we have defined the visibility for the domain class, we create SWC diagrams describing the behaviour of each class. Figure 29 shows the navigation modelling the class *UserAccount* presented by Figure 28.b.

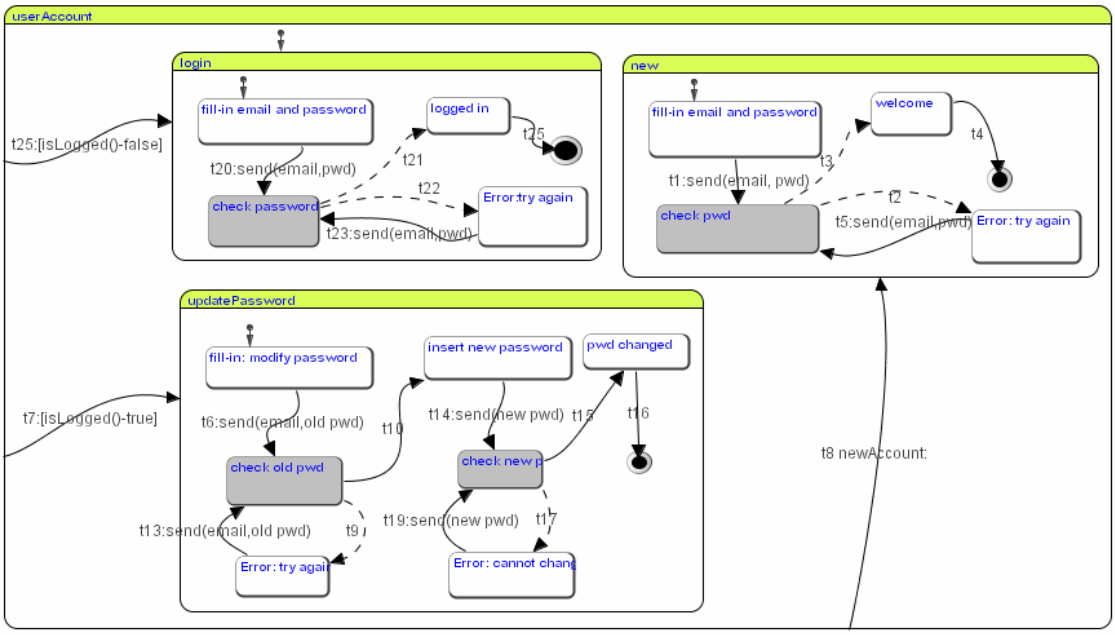


Figure 29 – SWC modelling for the class UserAccount.

Each public method concerning the navigation is represented as a composite state containing the detailed description about the dialogue. For example, the method *login* in the SWC model presented by Figure 29 describes:

a) The conditions to access the method (e.g. *isLogged()=false*);

b) The actions and attributes (e.g. at t20, action *send* and attributes *email and pwd*).

c) The set of elements that should be implemented by the means of static pages (i.e. basic states; e.g. the state fill-in email and password inside the state login);

d) The transient elements that should be implemented as scripts (i.e. transient states *check password*) and which describe the processing performed by the system;

e) The result of processing (i.e. state *logged in* or *error: try again*, inside sub-state *login*);

f) The agent activating transitions (i.e. system or the user).

5.2.3. Integration of individual SWC diagrams into a single model

This step is made up by integrating all individuals SWC models into a single SWC model describing the entire application. We start by creating a new SWC diagram where the top level state represents the whole navigation space of the Web application. Then, individual SWC diagrams describing the behaviour of each class are imported to form a single SWC model.

The relationships between classes are not transformed automatically into SWC relationships. The reason for that is that relationships in class diagrams are more likely to represent dependency of data than user requirements for navigation. In order to set relationships between SWC diagrams representing class behaviour we need to perform a task analysis to identify user requirements for navigation (see next step).

5.2.4. Creating SWC diagrams to describe user tasks

Task models are performed in early phases of design. They should be supplied before starting the navigation modelling. They help designers to specify the behaviour of the application according to user requirements. Task models present a great advantage over UML Use Cases and text-based scenarios to describe user tasks since they provide means to clearly represent hierarchy of sub-tasks, order for tasks (e.g. sequence, concurrence), the users and stakeholders of the system. Several task models such as Concurrent Task Tree model (CTT) [134, 135] or the "*Méthode Analytique de Description des tâches"* (MAD) [136] are suitable to help designers to describe user tasks with the application.

For an example, Figure 30 presents a task modelling with CTT for the *login* task. The task *login* can be described as a sequence of tasks starting by *showing the form fill* for email and password followed the tasks *check password* (i.e *check pwd*) and *close window*. The task *check pwd* is composed of 4 sub-tasks describing how the system executes the verification of a user account as follows: a user *supply email and password*, then the system that validates it (i.e. sub-task *validate pwd*) and provides the corresponding feedback (i.e. show error: try again or show welcome and login). CTT employs different symbols to express complex tasks (i.e. cloud-like symbol; e.g. *check pwd* task), tasks performed by the system (e.g. *show fill-in form*) or interactive tasks which performed by users (e.g. *close window*). In addition to these tasks, CTT also have a so-called *user-task* that expresses cognitive tasks performed by a user, which are not employed in this example. CTT also employ operators such as "[ ]" to represent the concurrency between tasks and the operator ">>"to represent the sequence.

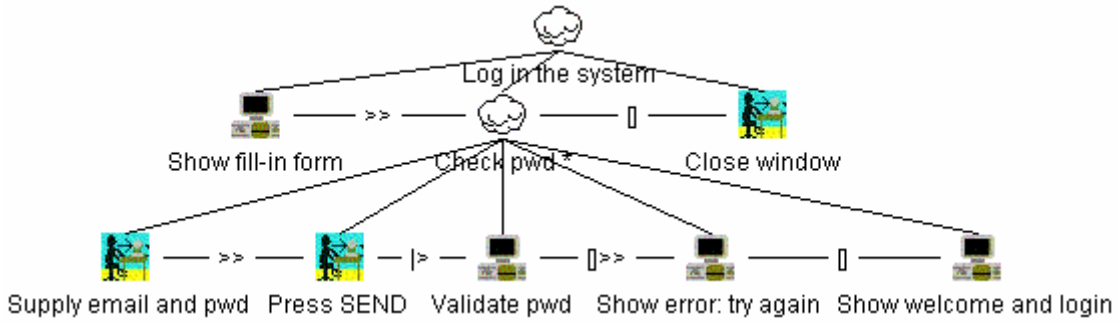


Figure 30 – Example of task modelling with CTTE for the *login* task.

Task models not only help to take into account user requirements but also can help to specify the class behaviour. For example the task model presented in Figure 30 was employed to detail the behaviour for the method *login* in Figure 29. Tasks models can be used to oversee where a given task is required in the application. For example, the task modelling for the ACME conference Web application presented by Figure 31 shows that the *login sub-task* is required to *apply a submission* and to *modify a submission*.

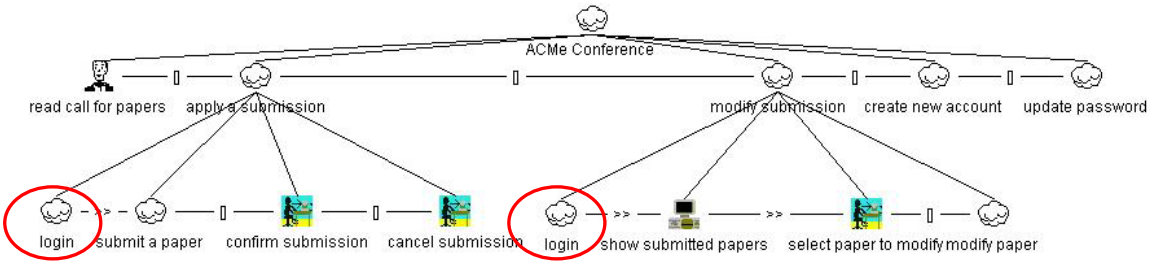


Figure 31 – Task model for the ACME conference Web application.

For some Web applications task models can be used as primary sources of information for navigation models. However, task models are only suitable to model structured tasks, that means, tasks which requires a specific set of steps to be accomplished. Since navigation of Web

applications often have non-linear relationships to explore the information space, what is usual on hypertext-based systems, tasks models lost they efficiency and meaning.

5.2.5. Including navigation paths

In this step designers must include navigational elements such as links or navigation patterns (e.g. indexes, menus, guided-tours) to easier the access to information. The exploration of information space (e.g. *browsing*) is a typical task in hypertext systems where there is no a single path to access the information. Such a browsing task is often difficult to represent without a navigation model.

At this point designers can create new states and transitions to represent menus or indexes pages, and any other navigational elements necessary for navigation. For example, Figure 32 presents the addition of menu to the ACME conference Web application. We can observe that a new static state namely *home* has been introduced. This state has set as the default (i.e. initial state) for the application and, thus, it represents the main page for that application. The state *home* has two outgoing transitions: *log the system* and *create an account*. The former links going from the state *home* to the state *userAccount* automatically transfers the control for the initial sub-state *login*. The second transition links directly the state *home* to the sub-state *new* insider the state *userAccount*. The two transitions will be translated to links in the implementation.

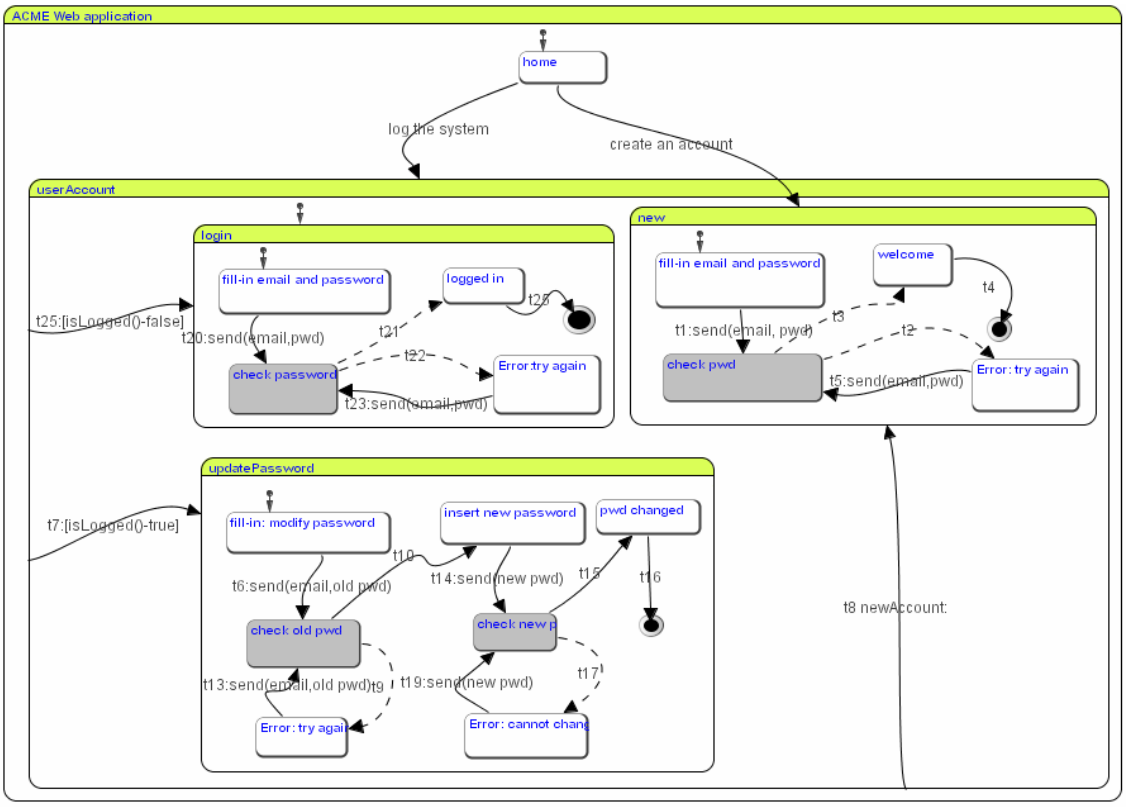
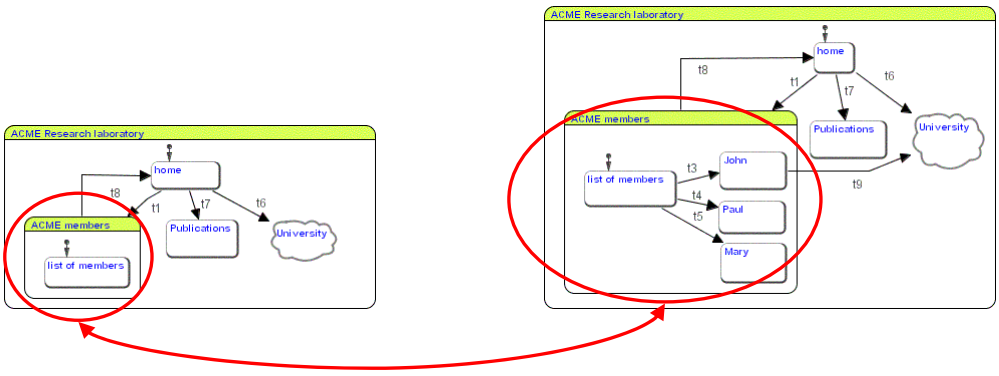


Figure 32 – Navigation paths for the ACME Web applications.

5.2.6. Defining content-dependent navigation

Once we have defined the general navigation by following the previous steps, we can start by populating the states with content in order to define navigation that is dependent of the content. At this point, we have to know exactly which content should be included in each state. Figure 33 presents a example for the ACME Research laboratory Web site: Figure 33.a shows the global view created by following the previous steps while Figure 33.b give details about how the state *ACME members* is populated with three states (namely *John, Mary,* and *Paul*) to describe specific content. Figure 33.b also present new transitions that have been added to

describe the behaviour inside the state *ACME members* as well a particular link going from the state *John* to the state *University* by following the transition *t9*.



a) Navigation schema **b) Detailed navigation**

Figure 33 – Populate states with content.

5.3. DISCUSSION

The method described above is made up in two phases: the first phase, namely *navigation schema,* builds a generic navigation for the application; the second phase, namely *detailed navigation.* Figure 34 shows how these two phases are placed inside the life cycle of development of Web applications proposed by Scapin et al. [32]. Typically, the *navigation schema* is developed during the phase (2) of *Site Specification* and *Detailed Navigation* is developed at the phase (3) of *Site Design*.

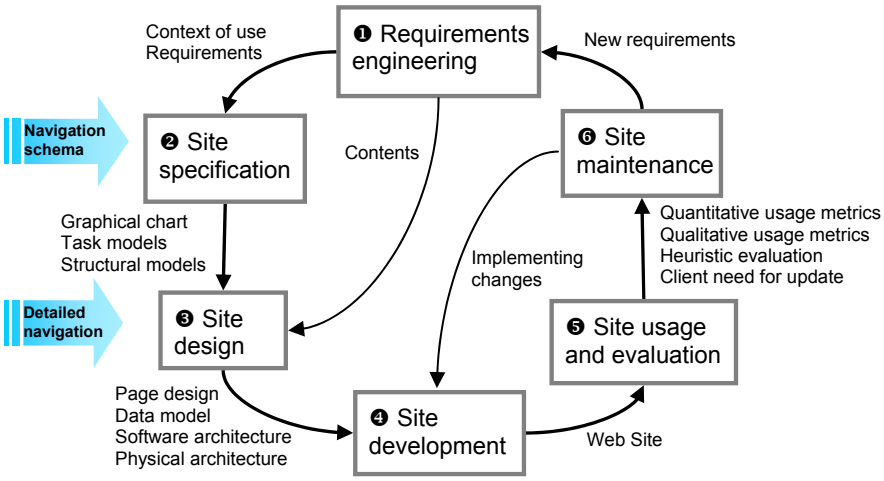


Figure 34 – Life Cycle for Web applications development.

At the Figure 34, the numbers on each phase suggest a natural order for the life cycle starting from the *requirement engineering* phase (1), going to *site specification* phase (2), and so on. This sequence reinforces the idea of a model-based approach for the development. The phase (2) and (3) are those that most clearly emphasize the idea of a model-based approach by including modelling activities. However, in the practice, some activities on each phase can be run in parallel. For example, it is possible to imagine that the *site specification* phase can be done at a time a designer defines the *physical architecture* (activity corresponding to the *site design* phase (3)) for the application.

The so-called "O"-life cycle does not even impose a particular start point for the design process. We suggest starting at the *requirement engineering* phase (1) but practical evidence has proved that many Web projects may start in as to late phases as *site design* phase (3) or *site development* phase (4). In such as a case, we are not talking about a model-based approach. On

what we are concerned, our approach for development process requires the modelling activities covered during the phase (2) and (3) at the Figure 34.

Navigation models can be employed exploited at different phase of the life cycle of Web applications. In the early phase of the development process, they help to manage the complexity of the application, to specify behaviour and requirements for navigation, and to document the application. During the development phase, navigation models guide the implementation according to the requirements that have been set implicitly inside models. During evaluation and maintenance phase, models help to choose the best way to correct a problem by simulate the changes prior to implementation. In fact, as we discuss in [137], the use combined of task models and navigation models provide valuable information for usability assessment of Web application, thus facilitating the maintenance of Web applications.

5.4. CONCLUSION

This chapter have presented the steps required to describe Web applications with the StatWebChart notation. The approach proposed is based in two phase (*navigation schema* and *detailed navigation*) that allows to detail incrementally the navigation of Web applications.

SWC models can be produced without any task model. However we suggest strongly the use of task models to take into account user requirements when describing navigation. The use of task model reinforces the user-centred approach for the design and it provides a mean for specifying the user activity. At the present, we have employed the task model Concurrent Task Tree model (CTT) [134, 135] in our case studies but other task model would be used with similar results.

We do not expect to contribute to the development process of Web application with this chapter. Our intention was just to describe the method we used to build navigation models for Web application using the SWC notation and place the activities required for model the navigation inside a life cycle of development. More case studies developed by other development teams are necessary to verify the efficiency of the approach presented here. The use of the StateWebCharts notation is not limited to this any other development process. So, designers can adopt or adapt the present approach for modelling navigation with SWC to other development processes according to the requirement and constraints of Web projects.

6. Case Study

Summary

In order to demonstrate the potential of the SWC notation for the development of Web applications this chapter presents a case study showing the scalability of our approach. The case study presented here describes a Web service for the *Association Francophone d'Interaction Homme-Machine AFIHM* allowing users to navigate, search and update a catalogue of theses. This case study intends to highlight the advantages of the use of a formal notation to describe the navigation in a real Web project.

The chapter starts with the subject of the case study and general requirements for the application. The remainder presents a detailed task analysis and task modelling, the underlying data model, the navigation modelling with SWC and implementation issues. Finally we described the lessons learned.

The chapter is composed of the following sections:

6.1 Foreword

6.2 The AFIHM's Web-Based Catalogue of Theses

- 6.2.1 General requirements
- 6.2.2 Target audience and users
- 6.2.3 Informational requirements
- 6.2.4 Functional requirements

6.3 Underlying Data Model

6.4 Task Analysis

6.5 Navigation Modelling

6.6 Implementation Issues

6.7 Discussion and Lessons Learned

6.1. FOREWORD

This chapter presents a real case study for a Web application using the SWC notation which was described in chapter 4. The main objectives of this case study are:

- To demonstrate the potential of the SWC notation to describe the navigation of Web applications;
- To show the scalability of our approach;
- To exemplify our approach for modelling as described in chapter 5.

Due to the requirements of the application employed for the present case study, the models shown in this chapter do not feature some constructs of SWC notation. In addition, since we have employed the SWCEditor to build SWC models, graphical presentations of states and transitions may have small differences when compared to the examples presented in chapter 4.

A particularity of the case study is that the target audience consists of French-speaking users. Even though the final implementation is delivered in the French language, we present in this chapter an English version of such an implementation which was conceived specially to be coherent with the language used in this document.

The chapter starts in section 6.2 by describing the subject of the case study and general requirements for the application including functional and navigational requirements. The rest of the chapter is organised as follows: section 6.3 describes the underlying data model; section 6.4 presents the task analysis for the application; section 6.5 presents the navigation modelling; section 6.6 presents the implementation issues, and, lastly, in section 6.7 we present the lessons learned within this case study.

6.2. THE AFIHM'S WEB-BASED CATALOGUE OF THESES

The AFIHM[41] is a francophone association whose main goal is to promote the research of Human-Computer Interaction (HCI) and related disciplines to improve the development, implementation and evaluation of interactive systems. Among the actions developed by AFIHM is the organisation of annual conferences in the HCI field and the publication of the "*RIHM : la Revue d'Interaction Homme-Machine*" journal.

Currently the AFIHM publishes on its Web site a list of theses made in French. To have a thesis published on the AFIHM Web site an author must send by email the information concerning his/her thesis to the AFIHM secretariat who will update a Web page (i.e. a single HTML file) containing the list of theses. In order to make this process more interactive and provide searching facilities, the AFIHM decided to create a database of the theses publicly available. This decision is the origin for the present case study, namely *AFIHM's catalogue of theses* which has been created in conjunction with the team responsible for the development of the application.

6.2.1. General requirements

The general idea behind this application is to allow users to feed the database with little effort and control by the AFIHM rather than simply providing searching and browsing and facilitates. Thus any typical user (e.g., a former PhD student) would easily add his/her thesis to the catalogue without the need of a heavy bureaucratic procedure by the AFIHM secretariat. Since some users could create unexpected records (e.g. supplying incomplete/inappropriate information, mistakenly changing a record) the Web application should support some form of minimum review process. Such a review process is made up by forming a special user namely a *system administrator* who may give the authorisation to publish a thesis in the catalogue.

Three main requirements guide the design of the AFIHM's theses catalogue:

- To provide searching and browsing facilities over a thesis catalogue;
- To ease updating the catalogue's contents by users;
- To support a fast review process for users submitting their thesis to the catalogue.

6.2.2. Target audience and users

The target audience consists of French-speakers users. The application is mainly addressed to doctorate students and researchers working in the Human-Computer Interaction field.

Three main user roles (or actors) have been identified, which are: "*everyone", "registered user"* and "*system administrator"*. Table 7 presents these three main roles according to the tasks they are allowed to do with the application.

The "*everyone"* role represents any user visiting the Web site. The browsing and searching facilities of the database are available to *everyone*. However user authentication is required to submit a thesis.

Everyone can create an account using the Web site giving access to other functions. The role "*registered user"* indicates those users who have been assigned an account. *Registered users* are allowed to modify their account information and submit their thesis to the catalogue.

The user role "*system administrator"* refers to someone who is responsible for supervising the submissions.

[41] AFIHM means : "*Association Francophone d'Interaction Homme-Machine"*. The AFIHM Web site is available at: http://www.afihm.org/ (3/5/2004).

Table 7 – Target audience for the AFIHM's theses Web site.

Role Name	User Profile	Pre-conditions	(allowed) Tasks
Everyone	- Possibly a HCI researcher	- French-speaker	- Search database - Create an account - Log into the system
Registered user	- HCI researcher or former PhD student - HCI researcher taking part in a jury thesis	- Logged in	- Update account information - Submit a thesis to the catalogue
System administrator	- A AFIHM member - Has full control over the catalogue	- Have access rights	- Review submissions

6.2.3. Informational requirements

The informational requirements for the application can be summarised as follows:

- Provide general information about the theses including author, title, keywords and abstract;
- Show the category to which a thesis belongs (e.g. master, PhD, *HDR*[42]);
- Provide the text of the thesis in electronic version (e.g. PDF format) if it is available;
- Inform about the defence including: date of defence, jury members with their corresponding titles (e.g. advisor, supervisor, president of the jury, etc.) and affiliations;
- Supply contact information of authors and jury members.
- Include a link to the AFIHM's Web site homepage.

6.2.4. Functional requirements

Functional requirement include:

- Support both simple keyword-based searching as well as advanced queries according to fields combinations (e.g. author, title, keywords, domain, institutional affiliation, etc.);
- Support browsing the database by author name and thesis title;
- Allow *everyone* to create a new account;
- Allow *registered users* to submit a new thesis to the database;
- Allow the *system administrator* to check a submission before giving authorisation to publish it in the catalogue. The system should provide the *system administrator* an easy-to-use interface to accept, modify or reject user requests;
- Inform the users about the status (i.e. in reviewing process, new entry accepted in the database, changes accepted, modified by administrator and accepted, or refused) of his/her request;
- Once registered, users should be able to create new accounts for someone else (e.g. a new entry for a jury member). In such a case, the system administrator is requested to accept, modify or refuse the new inscription. If the inscription for a new account is accepted, the owner of the new account is informed by email;
- Allow users to modify their own password, their contact details and information any time without supervision.

[42] HDR means *Habilitation à Diriger des Recherches (HDR),* it is particular kind of thesis giving some privileges to his/her owner such as to supervise PHD students alone.

6.3. UNDERLYING DATA MODEL

The catalogue contains general information about a thesis (e.g. title, author, date of defence, knowledge domain, keywords used to describe the content) as well as information about the people taking part in the jury (e.g. jury members, jury qualifications, roles in the thesis, institutional affiliation).

The underlying data model for the thesis catalogue is described by means of a UML class diagram. The model presented in Figure 35 is composed of seven classes which are arranged around the class *Thesis*. The attributes and methods describe public operations that should be taken into account during navigation.

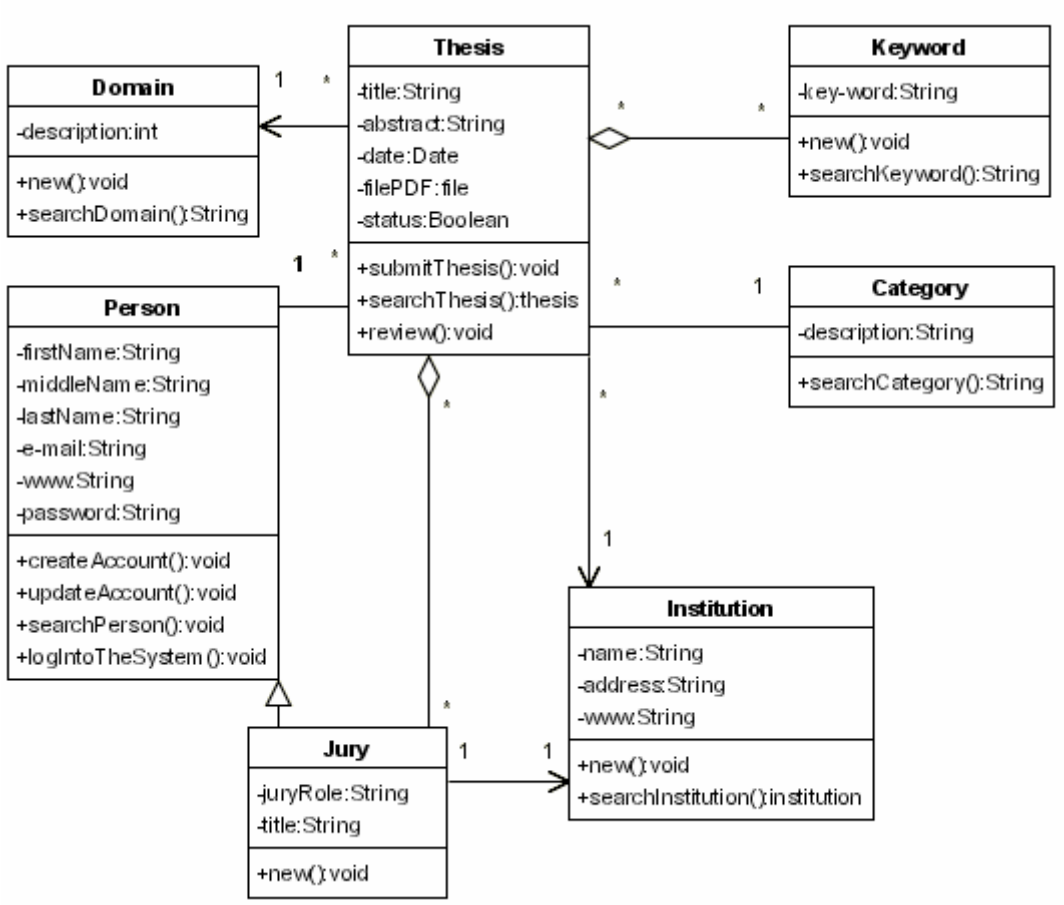


Figure 35 – Underlying class diagram "AFIHM's *theses catalogue"*.

The class *Thesis* consists of the following set of attributes: *title, abstract, date* of thesis defence and a link to the thesis in electronic format (i.e. *filePDF*). The attribute *status* is used to set whether a thesis has been reviewed by the system administrator. A *thesis* also has associations to *keywords*, *jury*, *category, domain* and *person* (i.e. thesis author). The class *category* means possible categories for a *thesis* (*e.g. Ph.D., HDR*).

The class *Jury* is a specialisation of a *Person* playing a particular role (e.g. advisor, supervisor) for the duration of a thesis. The association between the class *thesis* and *institution* expresses the affiliation (e.g. research laboratory, university).

6.4. TASK ANALYSIS

This section presents the main user tasks identified for the application. The presented task analysis is employed in conjunction with the underlying class diagram presented in previous section to build the navigation model for the application. Tasks are represented by means of

Concurrent Task Tree models (CTT) [134, 135]. We also provide an informal scenario illustrating each task described in CTT. Tasks described from section 6.4.1 to 6.4.6 correspond to the tasks presented in Table 7.

6.4.1. Task: Search Database

Searching the AFIHM theses catalogue is functionality publicly available to *everyone*. Users can search according several criteria: *thesis keyword, author name, date, institution, thesis category, keyword in the title, thesis domain.* These criteria can be supplied alone or in combination for advanced searching. Figure 36 presents the CTT model corresponding to the task *search* database.

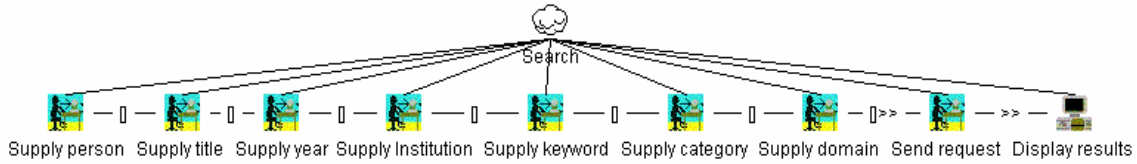


Figure 36 – Task: search database.

Scenario for *search a thesis in the database*:

> "Marie is French researcher in HCI who would like to update their personal homepage with the list of all theses she has supervised. However, Marie does not remember the members of jury of Johan, one of her the students. So, she decided to take a look at the AFIHM's thesis Web Site. From the main page of the Web site, she follows the link *search* and she fills-in the form the query to find *theses* defended in *1981*. After had pressed the *send* button, the system answers with a list of twenty theses ordered by author. She locates in the list the Johan's thesis, the student she was looking for. Afterwards, she selects the entry to do a copy and paste to updating her personal homepage."

6.4.2. Task: Create an account

Each person cited in the catalogue as an author of a thesis or jury member must have an account containing personal information containing their address and affiliation. User accounts are protected by a password. The creation of new accounts is followed by notification to the user by email. The system administrator is also notified. Figure 37 presents the CTT model for the task: *create an account*.

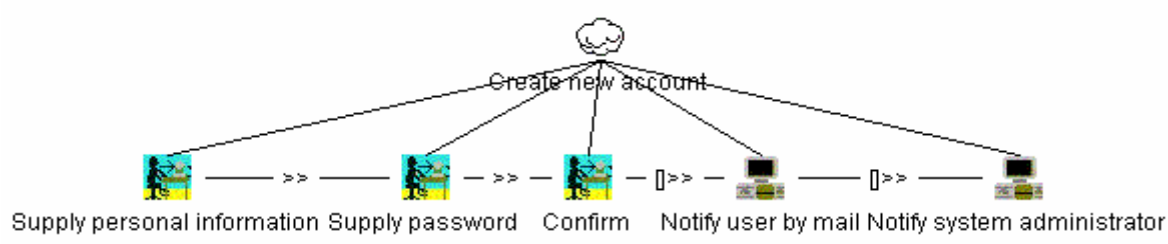


Figure 37 – Task: create an account.

Scenario for *creating a new user account*:

> "Pierre is a student in the HCI field. He has heard about the AFIHM's electronic catalogue of theses during the last IHM conference and he then decided to publish his thesis on that catalogue. On arriving on the Web site, he discovers that he needs to create an account. Pierre fills-in a form with his name, e-mail and Web page addresses. He also supplies a password to protect his account. Once Pierre press the button to submit his account information, the system create the new account and notify the both user and the system administrator that a new account has been created."

6.4.3. Task: Log into the system

Only registered users who have an account are allowed to request updating of the theses catalogue. The task *log into the system* (see Figure 38) can be described as a sequence of tasks starting by the task "*show the form fill"*, followed by the tasks "*check password"* (i.e *check pwd*) and "*close window"*. The task *check pwd* is composed of sub-tasks describing how the

system verifies the user account. A user *supplies email and password* and *presses send* to log in. The system then validates the email address and password (i.e. *pwd*) providing the corresponding feedback: *Show error: try again* or *Show welcome and login*.

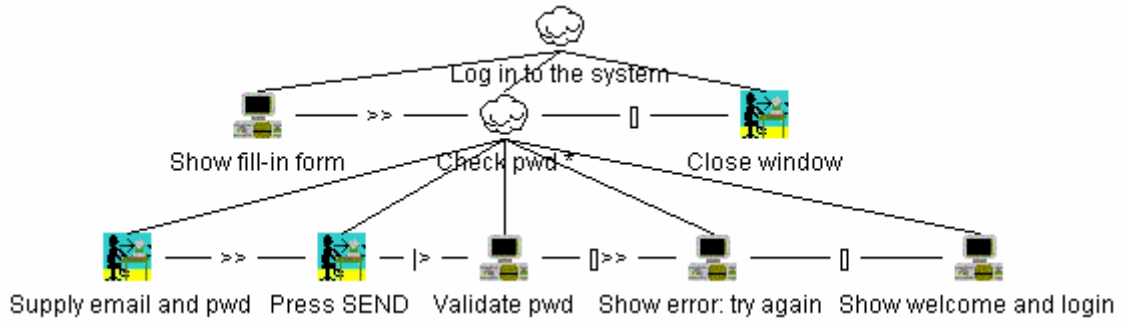


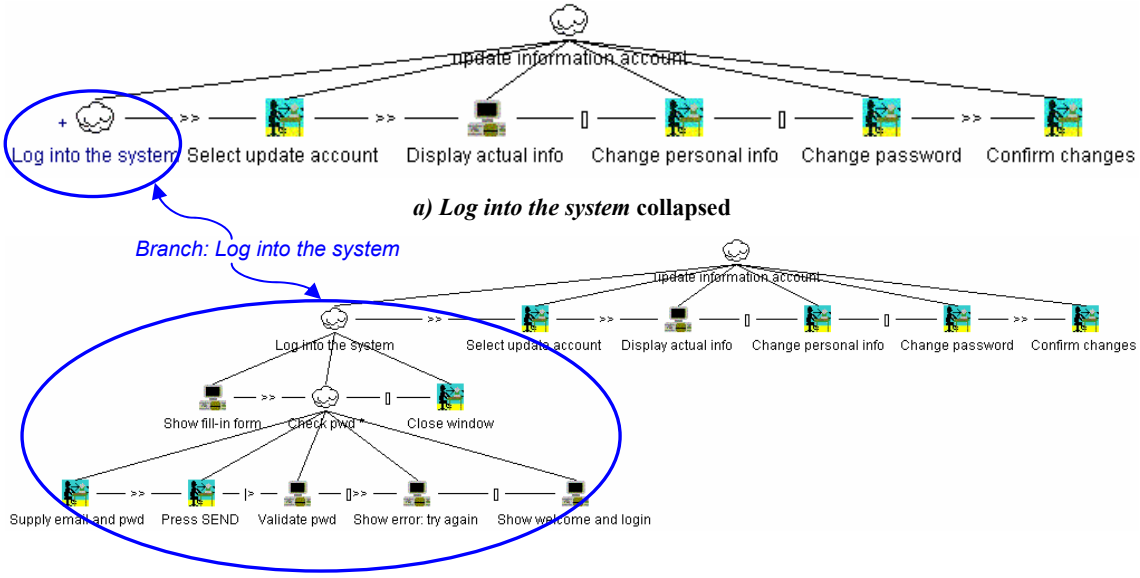
Figure 38 – Task: Log into the system.

Scenario: *logging into the system:*

"Marie has moved to a new research laboratory and she would like to update her email address and affiliation on the theses catalogue. She knows that to perform that operation she needs to log into the system. So, she opens the main page of the AFIHM's theses Web site and she fills-in a form her user name and password. The system answer that password is not correct. Marie succeeds the second try. The system informs she is logged in… "(*Continue in next section*)

6.4.4. Task: Update information account

Users can manage their own account changing personal information (e.g. email, postal addresses) and/or password. In order to update account information, a user must log into the system. We reuse the specification of the task *log into the system* presented in the previous section to build this more complex task *update information account*. Figure 39.a presents the task model collapsing the branch *log into the system* that has been reuses. Figure 39.b presents the same task model showing all of the sub-tasks involved.



***a) Log into the system* collapsed**

b) Log into the system expanded

Figure 39 – Task: Update information account.

Scenario: Updating address.

(Continuation) "…Once logged in the system, Marie pursue updating his email address and affiliation. He selects the link for managing personal account. The system, them shows him her actual information and let her to modify it. Marie confirms the changes and the system updates the account."

6.4.5. Task: Submit a thesis to the catalogue

Two steps are needed to add a new thesis in the catalogue. The first one is done by a *registered user* who submits his thesis to the catalogue. The second step is done by the *system administrator* who reviews the user request before accepting it in the database. The former step is described in this section while the second step is detailed in section 6.4.6.

Figure 40 presents the submission of a thesis by a *registered user*. This task requires the following sub-tasks: *log into the system*, *select the link to add thesis, supply thesis information, upload PDF file*, *fill-in* each one of *jury members*, *confirm submission, notify user* and *notify system administrator*. The sub-tasks *notify user* and *notify system administrator* are performed by the system.

Similarly to the task *update information account* presented in section 6.4.4, the task *submit a thesis* reuses the specification of other sub-tasks as follows: *log into the system, search person* in database, and *create a new account*. These sub-tasks (i.e. *log into the system, search people* and *create new account*) are shown in collapsed format in Figure 40.

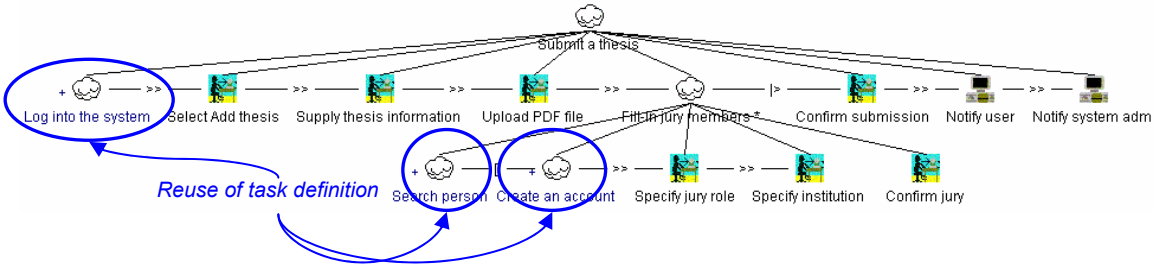


Figure 40 – Task: Submit a thesis to the catalogue.

The sub-task *fill-in jury member* is repeated until all jury members are included. For each jury member, users have to *search a person* in the database or create a *new account* when the person is not in the catalogue.

Scenario: adding a new thesis to the catalogue.

> "Pierre has presented successfully his thesis. His advisor asked him to include his thesis in the AFIMH's theses ctalogue. Once in the Web site, Pierre logs in since it has already an account. He starts by filling-in a form with information about thesis *title*, *date of defence*, *abstract*, *keywords* and corresponding *discipline*. He then uploads the PDF file of the thesis. He also informs in which category his thesis fits (in the case a *PhD*). Then he starts filling the form with information about each member in the jury which is composed of five members. Pierre searches in the system the names of people in the jury. Pierre picks from a list one by one the names but his advisor is there. So, he selects the button "create new user account" and he creates a new account for Duffy, his advisor. Pierre can now complete the list of jury members. Presently, he confirms the submission. Pierre The system sends to him an email telling that his submission has been successful received by the system but it will not be available until the system administrator check it and confirm the thesis inscription in the database".

6.4.6. Review submissions

This task describes how the *system administrator* supervises user submissions to the catalogue. Typically, the *system administrator* is notified by an email containing a direct link to a secure area of the application showing the thesis submission. The system administrator verifies the submission and replies giving his/her approval or refusal. He/she can also change the submission before approving it. When the system administrator confirms his/her review, the system updates the database and notifies the user that the review process is finished. Figure 41 describes the corresponding task model.

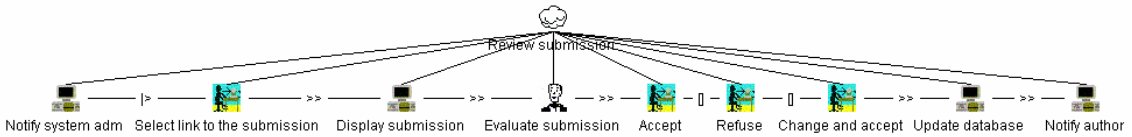


Figure 41 – Task: review submission.

Scenario: Review submission

"Thomas has been assigned as the new system administrator for the AFIHM's theses Web site. His main activity is to verify the 10 new submissions for the catalogue. The first mail in the list is from Pierre. Thomas follows a link the mail which opens the application with the Pierre's submission. Thomas verifies that title and abstract are well-spelled, the keywords and the categories supplied for the theses are coherent with the subject. Thomas would change the information supplied by Pierre but he thinks that everything looks fine and he decided to accept the submission without changes. Thomas selects the link named *accept modifications* and the system does the rest updating the database and sending an email to Pierre. Thomas does the same for the others submissions".

6.5. NAVIGATION MODELLING

In section 6.2.2 three user roles have been identified: *everyone*, *registered users* and *system administrator*. We decided to define a secure area where the *system administrator* can *review the submissions*; such a private area is independent from the rest of the application. This decision implies separately modelling the navigation concerning the review submission process and other user tasks. Hereafter, sub-section 6.5.1 presents the navigation model concerning the task *review submission,* which is performed by the role of the *system administrator.* The other sections describe progressively the application available for the user roles *everyone* and *registered users*.

6.5.1. Navigation diagram for the review submission

The navigation model for the submission reviews follows the general sequence of activities identified during the task analysis describe in section 6.4.5. The *system administrator* receives by email a notification containing a link to the new submissions in the catalogue. The model concerning the navigation starts when the *system administrator* decides to follow the link generated by the system. Figure 42 presents the corresponding SWC model. The *thesis review* (identified by a composite state with the same name) is accessible by a guarded link, namely *isSystemADM* which protects the access to non-authorised users[43].

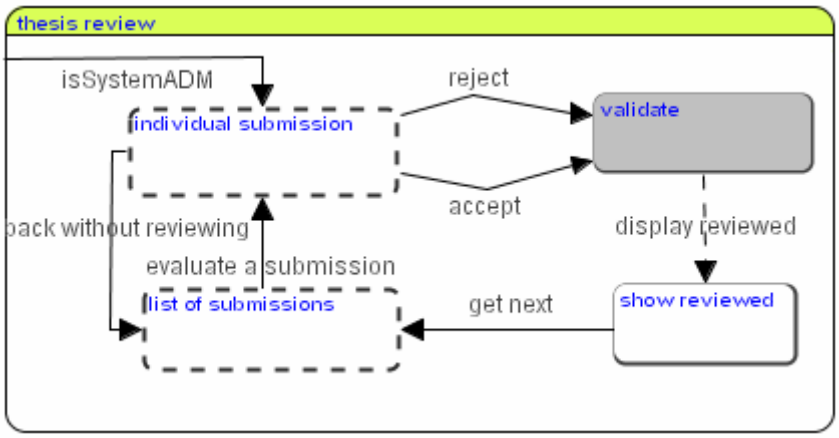


Figure 42 – SWC navigation modelling for the review submission task.

As depicted in Figure 42, the navigation starts at the state named *individualSystemADM* whose content is generated automatically by the system. This state contains the information concerning a single thesis submission. At this point, the *system administrator* can *reject* the submission, *accept it* or go *back* to list of submissions *without reviewing.* Note the activity described in the task analysis *change and accept* a thesis is not represented in Figure 42. The reason for this is that the activity is considered to be performed inside the state *individual submission*; a typical implementation of this navigation may show the thesis information by

[43] We do not describe over a navigation model how the system checks the authenticity of a link since it concerns the implementation. We suppose here that the application is able to generate a link, send it automatically to the *system administrator* and, therefore, verify whether the link is valid or not.

means of editable form fields allowing the *system administrator* to change the content before to *accepting* the submission.

6.5.2. General navigation for *everyone* and *registered users*

The general navigation concerning the roles *everyone* and *registered* users covers the publicly available part of the application. In opposition to the previous model concerning the *thesis review* which has been done in a single step, this section describes the set of steps which were required to produce the navigation model.

- CREATING SWC MODELS TO REPRESENT THE CLASS BEHAVIOUR

Once we have modelled the application domain by means of the UML class diagram presented in Figure 35 we start by modelling the behaviour of each class with SWC models. Only the behaviour related to navigation is covered by these models. Each class was first represented in a single SWC diagram. Figure 43 presents the SWC model describing the behaviour concerning the class *Person*.

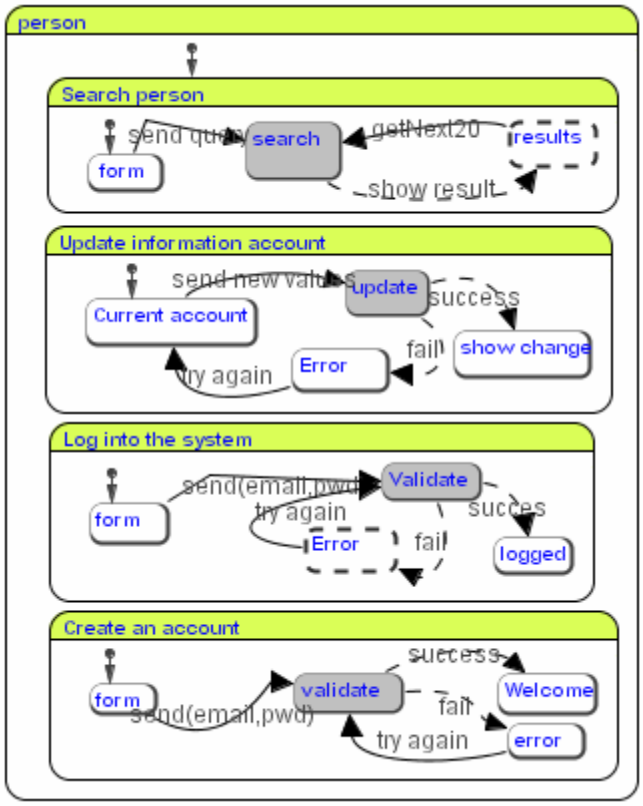


Figure 43 – Navigation model for the class Person.

The class *Person* is represented by a compound state with the same name *person*. Each one of its four methods, namely *search person, update information account, log into the system* and *create an account,* are also represented as compound states. Inside each compound state representing a method, we describe the details concerning the navigation. In Figure 43 the compound state *search person* consist of three states: a) *basic state* named *form* that typical contains the form fields for the query; b) *transient state* named *search* expressing the system activity for searching; c) *dynamic state* named *results* representing the query results. The transitions *send query* and *getNext20* are expressing the user interactions with the application.

- INTEGRATING SWC DIAGRAMS CONCERNING USER NAVIGATION

In this step, a single SWC diagram is created holding individual SWC diagrams, as shown in Figure 44. Each class in the class model is represented by means of a SWC composite state that contains the specification of public methods. At this point, the diagram does not show the

relationship between individual diagrams representing classes. One can note that the method *review* concerning the class *Thesis* is not represented in Figure 44 since it has already taken into account in section 6.5.1.

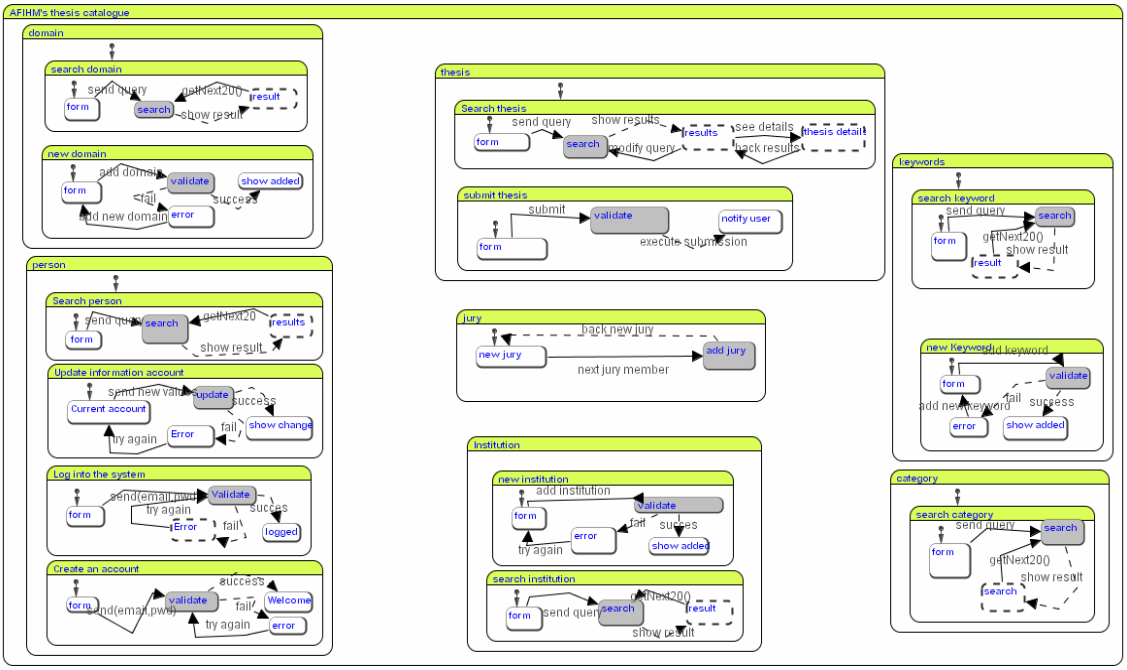


Figure 44 – SWC diagrams showing class and methods used in the navigation.

- INCLUDING DEPENDENCY TO NAVIGATIONAL PATHS

In order to compose the navigation available to the end-user, this step defines two additional states to provide access from a main page of the application (see state *home* in Figure 45) and a link to the AFIHM Web site (see state in Figure 45) which is considered external to the current study case.

Next step creates the missing relationships between states. The transitions included in the model represent the navigation suitable for the user roles *everyone* and *registered users*. It is important to note that the model keeps track of which sequence of states is available at a time. For example, when arriving at the state *home,* users can navigate towards the states *log in to the system, create an account* and *search thesis*. Since users log into the system (which is represented by the state *logged* inside the *log into the system* and *person*) they have access to other functions such as *submit a thesis*.

Links to other states representing classes such as *domain, keywords* and *jury* and *institution* are only made available after the user have engaged a thesis submission. The same applies for the state *search person* representing methods of the class *person*.

These models were built by using the tool support SWCEditor, which is better described in chapter 7. SWCEditor provides a simulator tool called *StateMachinePlayer* that helped us to verify that transitions included in the model correspond to the intended navigation. For example, Figure 46 shows the list of enable transitions when a user reaches the state *submit thesis* shown in red.

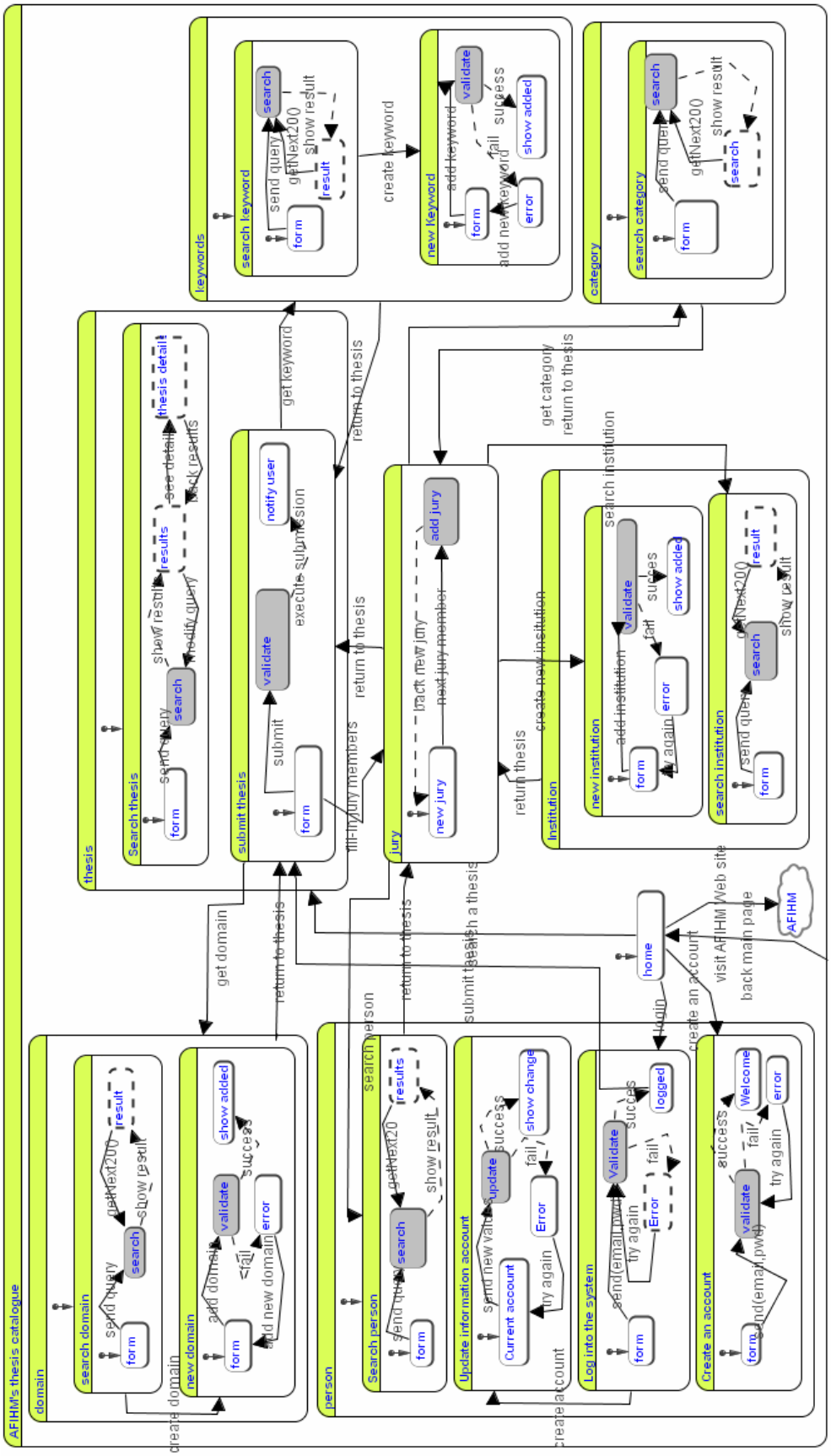


Figure 45 – Navigation SWC model for AFIHM's theses catalogue.

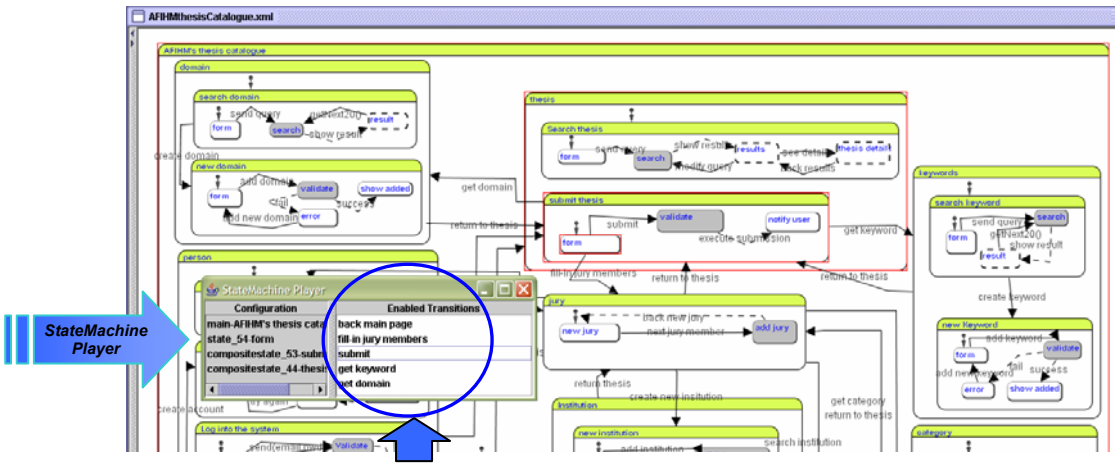


Figure 46 – Available transitions/links when navigating the state *submit thesis* in the AFIHM's theses catalogue application.

6.5.3. Prototyping the application

Once we have verified that the navigation built with SWC holds in our requirements we start by creating the Web pages and scripts that correspond to the SWC model. Figure 47 presents the main page of the AFIHM's (at right) beside the SWC model (at left) showing in red the corresponding state (i.e. *home*). The corresponding links are depicted in blue in Figure 47.

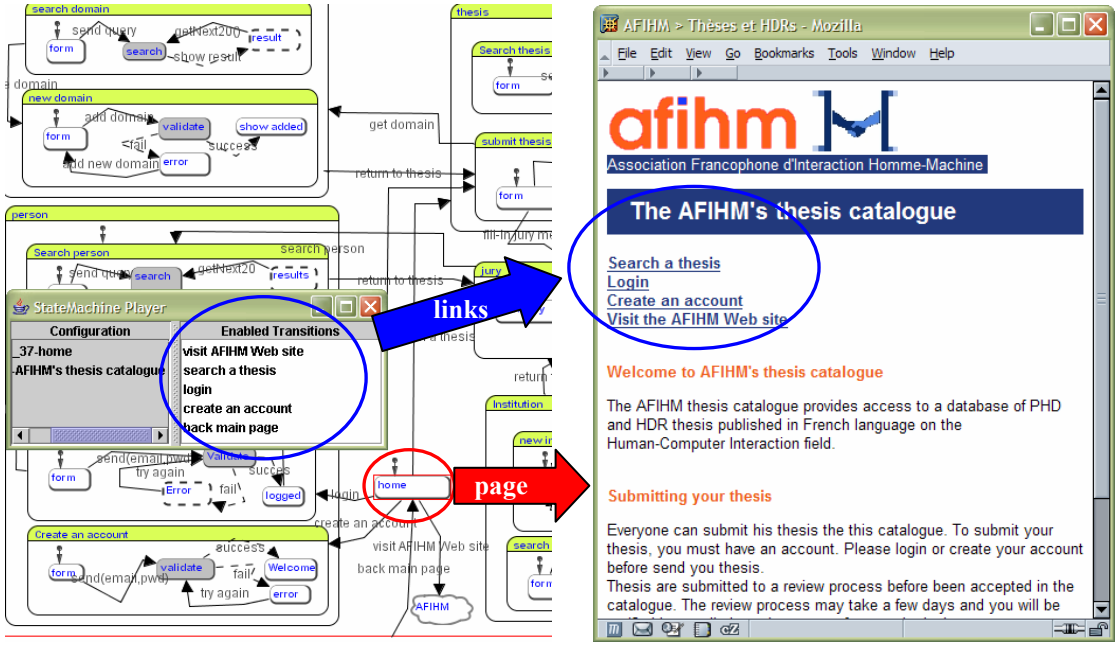


Figure 47 – Prototyping the main page for AFIHM's theses catalogue.

The Web pages were built using a visual environment independent from SWCEditor since, at the present, there is no Web page editor integrated to the SWCEditor.

Once we have prototyped all Web pages for the application we returned to the model specification and by means of the SWCEditor we have associated each state visible at the user presentation of a Web page. Each state visible at the user presentation is associated to a Web

page which includes the content and the graphical presentation for the objects. Figure 48 shows the pages corresponding to the states in the SWC model for the state *Search thesis*.

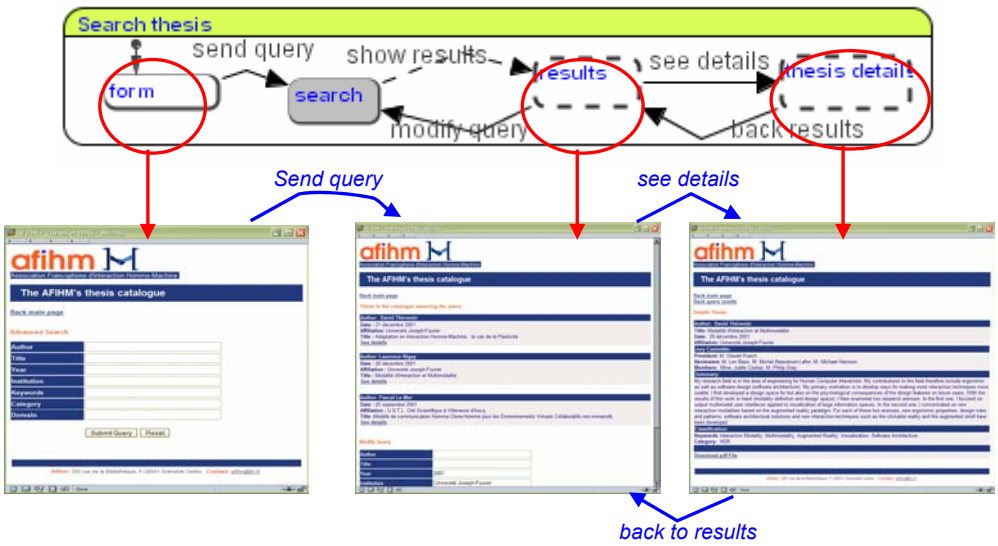


Figure 48 – Mapping SWC states to Web pages.

As we can observe in Figure 48 *user transitions* (represented by continuous lines) are visible as links in the application (represented by blue lines). *Completion* and *system transitions,* which according to the SWC notation are represented by dashed lines, are internally coded by the scripts which are represented by the transient state namely *search*.

6.6. IMPLEMENTATION ISSUES

Presentational aspects were defined by means of the template Cascade Style Sheet (CSS)[44] that has been defined by the AFIHM Web site. This ensures a uniform identity for theses catalogue.

The Web pages for the application were developed using the editor Mozilla Composer 1.4[45]. The server side application and scripts were coded using PHP: Hypertext Preprocessor [46].

6.7. DISCUSSION AND LESSONS LEARNED

The case study presented in this chapter could be considered small when compared to very large Web applications such as institutional Web sites or electronic commerce Web sites, for instance. However this can be considered sufficient to demonstrate the scalability of our approach. The model presented in Figure 45 consists of forty-nine basic states (including all sub-types: *static, dynamic, transient* and *external*) that have been translated to Web pages and/scripts at the implementation. That navigation model provides detailed information on how users can navigate the application and which are the required paths to access to the information. It is important to note that all of relationships are held in a single page and are still legible.

Since the prototype for the AFIHM's Web catalogue not yet has been published we do not have results concerning how real users perceive the navigation. Inspection testing has been performed to ensure that the navigation follows the navigation requirements defined by the task analysis. Usability testing is required to confirm the choices we have made for the design.

[44] Such as template is available at http://www.afihm.org/styles/style-afihm.

[45] Available at http://www.mozilla.org/

[46] Available at: http://www.php.net/ (Feb. 2004).

However the absence of testing with users does not affect the validity of this case study since it demonstrated how to model the navigation with SWC and how such models can be employed during the implementation. A French version of the prototype is currently in development and it will be tested with real users in the near future. We expect to evolve the modelling to integrate new requirements originated from the users of the application.

The use of task models remains essential to understand how to build navigation by taking into account user requirements. In section 6.5.1 we presented an example concerning the navigation for the task *review submission*. The task *review submission* clearly define the options the *system administrator* has for reviewing a submission (i.e. *accept, reject* or *change and accept*). In this example, the corresponding navigation model only provide relationships for *accept* and *reject* of the submission since the sub-task *change and accept* a submission is considered as an atomic activity of a state *individual submission* (see Figure 42). This illustrate that the description of tasks should be adapted to the context of the design.

This case study also allowed testing the SWCEditor. In addition to the facilities of edition and visualisation of models, the functionality allowing the simulation of SWC models was particularly useful to inspect the navigation. We have used it as a kind of automated walkthrough method for inspecting the initial model and in the sequence to verify that correct links have been included in each state.

7. Tool support

Summary

In chapter 4 we have proposed a notation called StateWebCharts (SWC) to help Web designers formally describe navigation of Web applications. However most advantages of having formalisms like SWC can only be fully exploited if appropriate software tools exist. The advantages of tool support for interactive design have been extensively discussed in the literature; they range from facilities for editing (such as providing automatic layout to avoid overlapping lines in models) to advanced model analysis (i.e.; prediction of system performance or automatic detection of conflicts in the system).

This chapter presents a tool named SWCEditor supporting the edition, visualisation and simulation of SWC models. We describe hereafter how such a tool can support the design and evaluation of navigation of Web applications modelled using the SWC notation.

This chapter consists of the following sections:

7.1 The SWCEditor

7.2 General Architecture

7.3 Support for Creation, Edition and Visualisation

7.4 Support for Simulation

7.5 Discussion and Future Work

7.1. THE SWCEDITOR

The use of formal methods implies questions concerning the complexity of development using formal notations. In order to overcome such an inherent complexity and make formal methods more usable, a tool support is required. The advantages of tool support for interactive design have been extensively discussed in the literature; they range from facilities for editing (such as provide automatic layout to avoid overlapping lines in models) to advanced model analysis (i.e.; prediction of system performance or automatic detection of conflicts in the system).

This chapter presents a tool called SWCEditor to support the construction of StateWebCharts models. Such a tool does not only support the edition, visualisation, simulation and analysis of models built with the notation but also it eases the prototyping activity. SWCEditor also features ease integration with the application domain and presentation models which allows using SWC models in conjunction with other development tools such as visual environments.

This chapter is organised as follows: section 7.2 presents the general architecture of the SWCEditor, section 7.3 presents the functions supporting the creation, edition and visualisation of SWC models, section 7.4 demonstrates the support for simulation of SWC models. Section 7.5 presents a discussion and future work.

7.2. GENERAL ARCHITECTURE

The general architecture for the application is presented in Figure 49. The grey rectangular areas named *StateMachinePlayer, GraphicEditor* and *StateMachine* correspond to the modules we have developed. The *StateMachine* module is the core of the SWCEditor as it stores the models created using this tool.

The *Graphic Editor* supports user interaction by providing functions to edit and visualise models. The *simulator* module corresponds to an add-on tool called *StateMachinePlayer* supporting the simulation of SWC models. SWCEditor allows the association of files (e.g. HTML or any other kind of file describing the content of a Web page) and states thus providing a means to simulate models while viewing the file content as a Web browser (see section 7.4 for details).

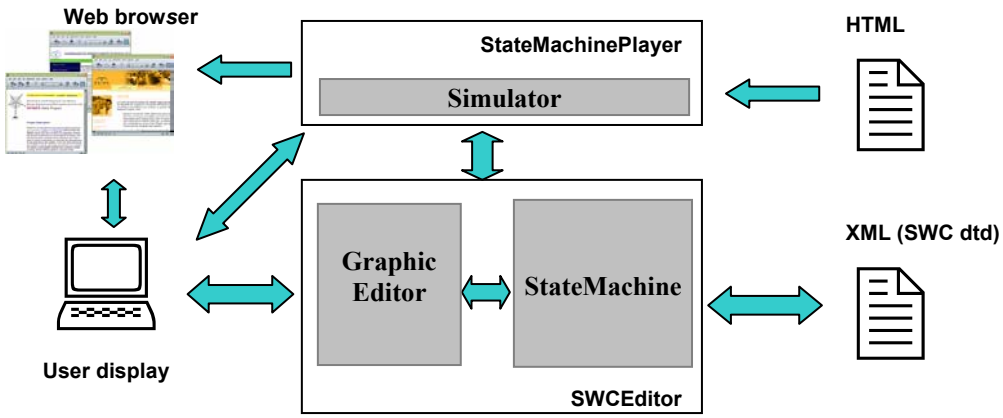


Figure 49 – General architecture of the SWCEditor.

SWC models are stored in XML format files. A Document Type Definition (DTD) named SWC DTD provides the structure for XML file. Such a DTD is presented in the annexe B. According to this DTD SWC elements are grouped in two main sections:

a) State section, which contains the hierarchy of a state and pseudo-states elements; and,

b) Transition section, which presents the list of all transitions in a model.

Figure 50 presents an extract of a SWC XML file. In Figure 50, we have highlighted in bold the attribute "*file*" enabling the mapping of a state to an external file representing the corresponding page or script.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with SWCEditor -->
<swc>
<CompositeState  id="root" label="root" file="null" initial="S1" concurrent="false">
<BasicState  id="S1" label="main intro" type="BasicState" file=" C:\swceditor\demo\spider\spider_intro.html" >
</BasicState>
<BasicState id="S2" label="schedule" type="BasicState" file="C:\swceditor\demo\spider\spider_schedule.html">
</BasicState>
...
</CompositeState>
<Transition id="t1" type="user" label="" source="S1" target="S2" trigger="mouseClick" guard="true" action="">
</Transition>
<Transition id="t2" type="user" label="" source="S1" target="S3" trigger="mouseClick" guard="true" action="">
</Transition>
...
</swc>
```

Figure 50 - *XML file describing a SWC model.*

SWCEditor was built using Java[47] technology. The management of graphical objects was developed on top of the open source graphical library JGraph[48] which is also based on Java. The prototype and the classes of SWCEditor are open source and publicly available at http://www.irit.fr/cgi-bin/viewcvs.cgi/swceditor/?cvsroot=swceditor.

[47] http://java.sun.com/

[48] Available at: http://www.jgraph.com/

7.3. SUPPORT FOR CREATION, EDITION AND VISUALISATION

One of the most basic requirements for model-based tools is support the creation of new models and the updating of existing ones. Figure 51 presents the edition of a basic state in a model. All attributes of a state are shown in a fill-in form; including actions and URL address to files containing the content of the application.

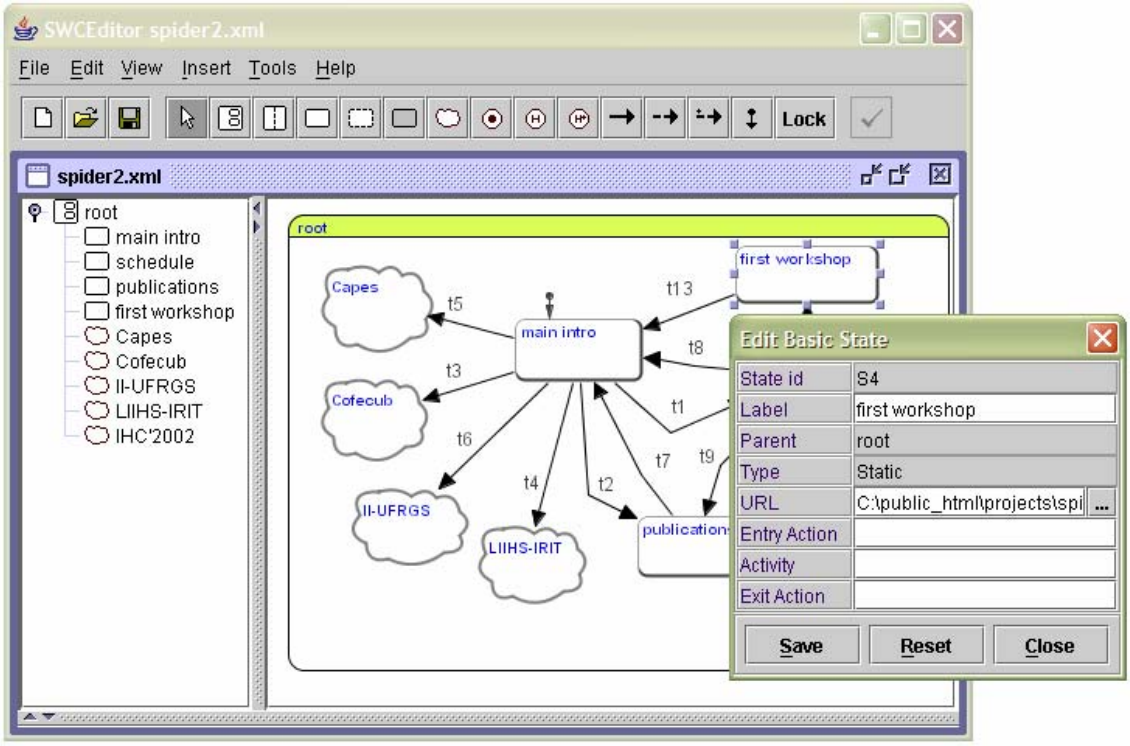


Figure 51 - *Editing function with SWCEditor.*

The field name "URL" (see dialogue "Edit Basic State" in the figure above) links an abstract representation of a state to a concrete Web page (a container of objects HTML, CSS, JavaScript, etc). As SWCEditor does not support code generation of Web pages, designers can use their preferred editor to implement the application. Once Web pages have been created, it is possible to associate each state to a corresponding Web page. This matching of states and Web pages is required for simulating the application in advanced phases of development.

Even though the SWCEditor provides many facilities for creating new SWC models, it is also possible to imagine SWCEditor being used to visualise SWC specifications resulting from reengineering process of existing Web sites.

SWCEditor includes functions of zooming and panning to explore and visualise large diagrams. In addition, a panel on the left showing the hierarchy of states helps designers to identify modules of the application (e.g. a set of states describing the behaviour of a class).

7.4. SUPPORT FOR SIMULATION

During early evaluation phases of the development process, designers have to check if abstract modelling will behave as expected. Simulations of models can be useful for exactly this purpose.

Figure 52 presents how SWCEditor allows simulation of SWC models. First of all, let us focus on the left part of Figure 52. There are two windows: the simulator window (top-left) and the visualisation window (bottom-left). The window simulator is comprised of two panels showing: the set of active states (left grey panel) and the set of enabled transitions at any time

(right white panel). The visualisation window is the main graphical editor of SWC models (i.e. the SWCEditor).

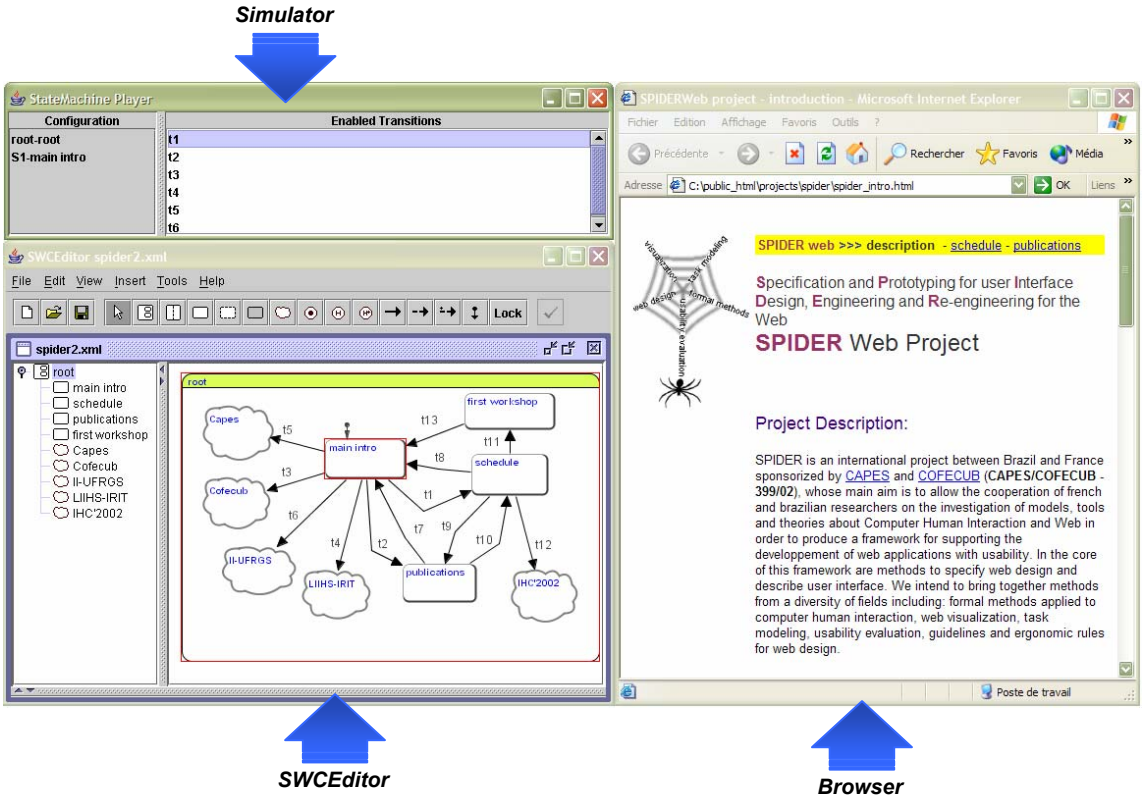


Figure 52 - *Tools for simulating SWC models.*

In the current implementation all interactions concerning simulation are performed in the simulator window. When an enabled transition is selected (by double-clicking) the system fires it immediately causing the system change displaying the next stable configuration. The interpretation of changes within the system is eased by animation reducing the user's activity required to compare the current state with the previous one. The new active states become red in the visualisation window then the set of current active states and the set of current enabled transitions are updated in the simulator window.

If a file is associated to a state it is possible to concurrently display the corresponding container (typically a Web page) in a browser during the simulation. The concurrent simulation of a model and implementation is suitable during the prototyping activity. Therefore, designers can follow the changes in the abstract specification using the SWCEditor as well as its concrete implementation at the Web browser. Figure 52 shows the corresponding Web page for the current state in simulation, in a browser window (right side).

The simulation can be used as a kind of automated walkthrough method for inspecting both navigation specifications and their corresponding Web applications. The task of testing the application for different displays is eased by repeating a simulation with different browsers. By simulating scenarios over navigation modelling, it is possible to verify the system's conformance with the user requirements specified through task models.

7.5. DISCUSSION AND FUTURE WORK

The development of Web applications can benefit from model-based approaches. Model-based tools can give abstract views of applications and guide the implementation passing the interactive phase of prototyping (which is especially useful during the development of

interactive systems such as Web applications) and supporting evaluation prior to implementation. Moreover, as discussed in [137] model-based tools can help designers to evaluate applications and support rational redesign of Web applications.

Surprisingly, only a few tools such as WebRatio[49] OOHDM-Web [58], Oliva Nova[50] and AutoWeb [29] support a model-based approach for Web applications. Table 8 presents a summary of tools supporting the development of Web application following a model-based approach.

Table 8 – Tools supporting the development of Web application following a model-based approach.

Tool Support	Modelling method	Navigation model	Analysis support	Approach for design
JWeb	HDM	graph	*not given*	bottom-up
RM Case	RMM	M-sclices	*not given*	bottom-up and top-down
AutoWeb system	HDM-Lite	acess schema	*not given*	bottom-up
Homer	Araneus	N-ER / ADM-2	model checking	bottom-up
OOHDM-Web	OOHDM	navigation class schema	model checking	bottom-up
WebRatio	WebML	WebML navigation schema	model checking	bottom-up
Rational Rose	WAE	class diagram	*not given*	bottom-up and top-down
Oliva Nova	OO-H	Navigational access Diagrams	*not given*	top-down

The tools presented in Table 8 are built upon modelling methods discussed in chapter 2. Since the models on which these tools are based do not provide detailed information about the user interaction of the interface there are no functions for simulating user interaction over the applications. Furthermore, due to little information about how users may interact with the application, these tools can barely reap the benefits of advanced analysis of system properties concerning usability of the applications.

Simulation is a very useful support for checking whether user requirements for navigation hold in models. In our practical experience notably in the case study presented in chapter 6, simulation of SWC models was particularly useful to inspect the navigation according to user tasks. In addition, SWC eases the task of testing the application for different displays by repeating a simulation with different browsers.

We have presented in this chapter the current version of a prototype of SWCEditor. Even though the current implementation does not include all tools to analyse system properties, SWCEditor provides at least the core functions enabling the development of such tools. Our ongoing work consists of developing and integrating such analysis tools for our prototype. We are also working on the development of more advanced visualisation features to make easier the interaction and edition of very large models.

We also have identified five categories of analysis techniques suitable to be automated by model checkers and employed to analyse navigation models:

- ***Static analysis:*** directly checks a model without having to provide any additional information. For example: check if all states are accessible; if there is no duplicated states or links; if all transitions have a valid target, and so on.
- ***Navigation path analysis:*** given a start point and a destination for a navigation path it is possible to check: the availability of a path to the destination; the shortest path to destination; the occurrence of path constraints (for example, to get destination ones have to visit some arbitrary states first).

[49] Available at http://www.webratio.com/

[50] Available at http://www.care-t.com/

- ***System properties analysis:*** one can choose to check if the systems obeys predefined rules such as if all states are accessible within a given number of links (more or less five for instance).
- ***Assessment of task and navigation models:*** check if scenarios obtained from task modelling can be performed over a navigation modelling. This kind of analysis requires simulation of the system.
- ***Comparing alternative models:*** if there is more than one navigation model for an application, it is possible to compare whether these models provide similar or equivalent navigation paths. By comparing alternative models, one can determine in which extension they are equivalent or not and if they support the same set of tasks.

Up to now, there is no analysis tool integrated to SWCEditor but this work is in progress. At present, we have selected a set of ergonomic rules that can be checked automatically or supported by semi-automatic tools. Our aim is to transform these ergonomic rules in a formal description that can be interpreted by a model checker over SWC models.

8. Conclusion et Perspective

8.1. CONCLUSIONS AND LIMITATIONS

The contribution of this work concerns the navigational aspect of Web applications. We have proposed a formal notation, named *StateWebCharts* (SWC), to model the navigation of Web applications and a tool, named SWCEditor, supporting the edition, visualisation, simulation and analysis of models built with that notation.

More generally, this work discussed the need of model-based approaches to build complex, reliable, scalable, and manageable Web applications. Web applications are a special kind of interactive system whose development is complex due to many factors such as the evolving nature of applications [7], the multidisciplinary nature of development team [130], the competitive points of views for the application, complexity of user requirements [131], unrealistic and narrow schedules for the development [132] [133].

In chapter 2 we have presented the state-of-the-art on modelling methods for Web applications. These methods have been strongly influenced by previous models usually employed in "traditional" software development and hypermedia system. However, despite of the efforts made to make these models suitable to deal with the development of Web applications many of them are only suitable for a particular kind of Web application. In addition, a detailed analysis of the currently available modelling methods revealed that many of them are not able to deal with the behavioural and navigational aspects of Web applications.

In chapter 3 we have discussed the idiosyncrasies of Web navigation and advanced functionalities, which can be associated to links such as actions, attributes and semantic types. We have demonstrated that such advanced link functionalities enrich the interaction over hypertexts and when employed properly may increase the usability of the applications.

Chapter 4 presented the StateWebCharts (SWC) notation, which extends Harel's StateCharts [23] to help Web designers to formally describe the navigation of Web application. StateCharts-based notations have been employed to model navigation of hypertext systems [16] [17] [18] and more recently to model Web applications [19] [22] [75]. However this previous work does not cover the idiosyncrasies of Web navigation described in chapter 3. Chapter 4 also demonstrated how navigation can be successfully described by means of the SWC notation.

One of the main contributions of the SWC notation proposed here is that it makes explicit in the models the points where users interact with the application with respect to those where the system drives and controls the navigation. Moreover, elements in SWC have a clear semantic with a corresponding visual representation, which is supposed to increase the legibility of the models. SWC supports client-side execution and server-side execution with some limitations as explained in chapter 4. However this is an intended limitation as solving this problem (for instance by including architectural information within the notation) would bind models to implementation/architectural concerns too early on the design process. In the same way, SWC is not the best solution for representing interaction on objects inside states. Here again, the focus of SWC is more on the early design phases where low level interaction modelling is premature. Besides, several notations deal very efficiently with these aspects and our goal is to integrate more the SWC with such approaches rather than making it suitable for all purposes.

Chapter 5 proposed a set of steps to build navigation models using the SWC notation. It describes the approach we have employed to build the case study in chapter 6. However the development process of Web applications with SWC has been only partially covered in this work since more case studies with other development teams are required to evaluate the effectiveness of our approach. Despite of this, the use of the StateWebCharts notation is not limited to a particular development process.

The use of formal description techniques raises issues about the increased complexity of the development of the applications. It has been shown that in order to overcome these difficulties that will be faced by the developers, adequate tool support must be provided. To this end, we have presented in chapter 7 the SWCEditor. Such as a tool does not only support the edition, visualisation, simulation and analysis of models built with that notation but it also eases the prototyping activity that are critical in the field of web applications. The functionality allowing the simulation of SWC models is particularly useful since it can be used as a kind of automated walkthrough method for inspecting both navigation specifications and their corresponding Web applications. The task of testing the application for different displays is eased by repeating a simulation with different browsers. SWCEditor also features an intended lazy integration with the application domain and presentation models, which allows using SWC models in conjunction with other development tools such as visual environments

8.2. PERSPECTIVES

The work presented here provides the foundations for further development concerning the design process and evaluation of Web applications. Concerning the evaluation aspect, a formal description technique such as SWC opens the perspective for the development of usability evaluation method that could be applied prior to implementation. More than just a set of states and transition, SWC models provide a semantic concerning the navigation (e.g. user and system activity, dynamic and static elements of the application). The semantic of SWC constructs can be exploited by model checker tools, for instance, to reason about the design. Model checking tools are already available to verify StateCharts models for analysis purposes. Since SWC can be translated into an equivalent StateCharts representation, it is intended to refine and adapt such model checking tools to handle SWC models.

As demonstrated by the tool support in chapter 7, models can be simulated allowing the kind of automated walkthrough inspection. We suggested that this kind of evaluation could be better explored if the simulation is structured around task models. In this direction, we have already started [137] the development of tools exploiting in a synergistic way both navigation models described with SWC and task models in order to identify some usability problems at various stages of the development process. However this question has only been addressed superficially and its critical impact on the usability of web application requires a much deeper and thorough treatment. This is one of the directions we really want to move towards.

Even though the evaluation is the most promising perspective for further development, much remains to do concerning the design. In this work we do not investigate the contributions of navigation model during several phases of the development process and more particularly the design phase. Indeed, the work presented here does not discuss in detail how to integrate other design issues of web applications. We have focussed our integration with user centred design methods and more particularly with the exploitation of task models and not with the work done by graphical or interaction designers. This is also considered as a very promising extension to this work.

One of the characteristics of Web applications is their continuous evolution both in terms of content and structure which requires frequent maintenance. Another perspective of the work in this thesis is to investigate how navigation models can help during the evolution of the application. We suggest that SWC models can be used to reason about changes before implementing them and to document the evolution of the navigation of the application. However, many more case studies and analyses are necessary to demonstrate the applicability of such a hypothesis.

In this thesis, we have selected several case studies to show the applicability (chapter 4) and the scalability (chapter 6) of our approach. However, we are aware that we need to apply it to more complex and more diverse applications to validate our proposal and provide concrete arguments that would encourage others to take up our results. An of particular concerns of ours is in the efficiency of the notation in terms of modelling i.e. how much time it takes to build the

navigation model of a web application. We are aware that if such issues are not adequately addressed they may jeopardise the usability of our work. We have already envisaged some extensions to the SWC formal description technique that could provide significant benefits. First the definition of reusable patterns (i.e. ready to use and ready to reuse templates) that could be exploited by designers during the early stages of the development process.

Similarly to software components, a key issue for design patterns is the need for a significant amount of them to be made readily available to designers in order to be beneficial. Even though we have identified some patterns while working on the case studies a far more intensive application of the notation must be implemented to identify a catalogue of patterns represented using the SWC notation.

References

[1] T. Berners-Lee "Information Management: A Proposal," no. August 25th, 2003, 1989.

[2] T. Berners-Lee, R. Cailliau, A. Luotonen, H.F. Nielsen and A. Secret "The World-Wide Web," Communications of the ACM vol. 37, no. 8, pp. 76-82, 1994.

[3] S. Murugesan and Y. Deshpande "Web Engineering: managing diversity and complexity of web application development," 2001.

[4] A. Cockburn and S. Jones "Which way now? Analysing and easing inadequacies in WWW navigation," International Journal of HumanComputer Studies pp. 105-129, 1996.

[5] A. Cockburn and S. Jones "Design issues for World Wide Web navigation visualisation tools," RIAO'97: The Fifth Conference on Computer-Assisted Research of Information pp. 55-74, 1997.

[6] L. Tauscher and S. Greenberg "How people revisit web pages: empirical findings and implications for the design of history systems," Int. J. Human–Computer Studies vol. 47, pp. 97-137, 1997.

[7] K. Norton "Applying Cross-Functional Evolutionary Methodologies to Web Development," Web Engineering: Managing Diversity and Complexity of Web Application Development pp. 48-57, 2001.

[8] J. Conallen "Building Web Applications with UML," -300, 1999.

[9] K. Grønbæk, N.O. Bouvin and L. Sloth "Designing Dexter-based hypermedia services for the World Wide Web," Proceedings of the eighth ACM conference on Hypertext pp. 146-156, 1997.

[10] C.C. J. Gómez, O. Pastor "Extending an Object-Oriented Conceptual Modelling Approach to Web Application Design," CAiSE'2000 pp. 79-93, 2000.

[11] D. Schwabe and G. Rossi "Building Hypermedia Applications as Navigational Views of information Models," 28th Hawaii International Conference on System Sciences pp. 231-240, 1995.

[12] S. Ceri, P. Fraternali and A. Bongio "Web Modeling Language (WebML): a Modeling Language for Designing Web Sites," WWW9 Conference 2000.

[13] F. Garzotto, P. Paolini and D. Schwabe "HDM—a model-based approach to hypertext application design," ACM Transactions on Information Systems (TOIS) vol. 11, no. 1, pp. 1-26, 1993.

[14] A. Dix "Formal Methods for Interactive Systems," 1991.

[15] P.D. Stotts and R. Furuta "Petri-net-based hypertext: document structure with browsing semantics," ACM Transactions on Information Systems (TOIS) vol. 7, no. 1, pp. 3-29, 1989.

[16] Y.i. Zheng and M.-C. Pong "Using statecharts to model hypertext," Proceedings of the ACM conference on Hypertext pp. 242-250, 1992.

[17] M.A.S. Turine, M.C.F.d. Oliveira and P.C. Masiero "A navigation-oriented hypertext model based on statecharts," Proceedings of the eighth ACM conference on Hypertext pp. 102-111, 1997.

[18] F.B. Paulo, P.C. Masiero and M.C.F.d. Oliveira "Hypercharts: Extended Statecharts to Support Hypermedia Specification," IEEE Transactions on Software Engineering vol. 25, no. 1, pp. 33-49, 1999.

[19] K.R.P.H. Leung, L.C.K. Hui, S.M. Yiu and W.M. Tang "Modelling Web Navigation by StateCharts.," 24th Annual International Computer Software and Applications Conference (COMPSAC 2000) pp. 41-47, 2000.

[20] Dimuro and A.C.R. Costa "Towards an automata-based navigation model for the specification of web sites," 5th Workshop on Formal Methods 2002.

[21] R. Furuta and P.D. Stotts "Programmable browsing semantics in Trellis," Proceedings of the second annual ACM conference on Hypertext pp. 27-42, 1989.

[22] M. Book and V. Gruhn "A Dialog Flow Notation for Web-based Applications," 7th IASTED International Conference on Software Engineering and Applications 2003.

[23] D. Harel "StateCharts: A Visual Formalism for Complex Systems," Science of Computer Programming vol. 8, no. 3, pp. 231-274, 1987.

[24] P. Chen "The Entity-Relationship Model - Towards a Unified View of Data," ACM Transactions on Database Systems vol. 1, no. 1, pp. 9-36, 1976.

[25] F. Halasz and M. Schwartz "The Dexter Hypertext Reference Model," Hypertext Standardization Workshop 1990.

[26] F. Halasz and M. Schwartz "The Dexter Hypertext Reference Model," Communications of the ACM vol. 37, no. 2, pp. 30-39, 1994.

[27] J. Rumbaugh, I. Jacobson and G. Booch "The Unified Modeling Language Reference Manual," 1998.

[28] A. Gu, B. Henderson-Sellers and D. Lowe "Web Modelling Languages: The Gap Between Requirements and Current Exemplars," 8th Australian World Wide Web Conference (AusWeb'2002) 2002.

[29] P. Fraternali and P. Paolini "Model-driven development of Web applications: the AutoWeb system," ACM Transactions on Information Systems (TOIS) vol. 18, no. 4, pp. 323-382, 2000.

[30] G.E. Krasner "A Cookbook for using the Model-View-Controller User Interface Paradigm in Smalltalk," Journal of Object-Oriented Programming vol. 1, no. 3, pp. 26-49, 1988.

[31] G. Golovchinsky "Queries? Links? Is there a difference?," Proceedings of the SIGCHI conference on Human factors in computing systems pp. 407-414, 1997.

[32] D. Scapin, J. Vanderdonckt, C. Farenc, R. Bastide, C. Bastien, C. Leulier, C. Mariage and P. Palanque "Transferring Knowledge of User Interfaces Guidelines to the Web," Tools for Working With Guidelines pp. 293-304, 2000.

[33] K. Grønbæk and R.H. Trigg "Toward a Dexter-based model for open hypermedia: unifying embedded references and link objects," Proceedings of the the seventh ACM conference on Hypertext pp. 149-160, 1996.

[34] Rumbaugh,J., M. Blaha, W. Premerlani, F. Eddy and W. Lorenson "Object-Oriented Modeling and Design," 1991.

[35] J.L. Peterson "Petri Net Theory and The Modeling of Systems," 1981.

[36] D. Schwabe, G. Rossi and S.D.J. Barbosa "Systematic hypermedia application design with OOHDM," Proceedings of the the seventh ACM conference on Hypertext pp. 116-128, 1996.

[37] P. Fraternali and P. Paolini "A Conceptual Model and a Tool Environment for Developing More Scalable, Dynamic, and Customizable Web Applications," Advances in Database Technology - EDBT'98, 6th International Conference on Extending Database Technology, Valencia, Spain, March 23-27, 1998, Proceedings vol. 1377, pp. 421-435, 1998.

[38] G. Rossi, D. Schwabe and A. Garrido "Design reuse in hypermedia applications development," Proceedings of the eighth ACM conference on Hypertext pp. 57-66, 1997.

[39] D. Florescu, A. Levy and A. Mendelzon "Database techniques for the World-Wide Web: a survey," ACM SIGMOD Record vol. 27, no. 3, pp. 59-74, 1998.

[40] T. Isakowitz, E.A. Stohr and P. Balasubramanian "RMM: a methodology for structured hypermedia design," Communications of the ACM vol. 38, no. 8, pp. 34-44, 1995.

[41] P. Atzeni, G. Mecca and P. Merialdo "Design and Implementation of Data-Intensive Web Sites," Conference on Extended Database Technology (EDBT'98) 1998.

[42] M. Gaedke, F. Lyardet, Werner-Gellersen and H. "Hypermedia Patterns and Components for Building better Web Information Systems," 2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia 1999.

[43] G. Rossi, D. Schwabe and F. Lyardet "Improving Web Information Systems with Navigational Patterns," WWW8 Conference Refereed Papers 1999.

[44] D. German and D. Cowan "Towards a unified catalog of hypermedia design patterns," 33rd Hawaii International Conference on System Sciences 2000.

[45] M. Bernstein "Structural patterns and hypertext rhetoric," ACM Computing Surveys (CSUR) vol. 31, no. 4es, pp. 191999.

[46] J. Nanard and M. Nanard "Toward an Hypermedia Design Patterns Space," 2nd Workshop in Hypermedia Development: Design Patterns in Hypermedia 1999.

[47] E. Gamma, R. Helm, R. Johnson and J. Vlissides "Design patterns: elements of reusable object-oriented software," 1995.

[48] F. Garzotto, L. Mainetti and P. Paolini "Designing modal hypermedia applications," 8th ACM conference on Hypertext pp. 38-47, 1997.

[49] F. Garzotto, P. Paolini and L. Baresi "Supporting Reusable Web Design with HDM-Edit," 34th Annual Hawaii International Conference on System Sciences (HICSS-34) vol. 7, pp. 70762001.

[50] M.A. Bochicchio, R. Paiano and P. Paolini "JWeb: An HDM Environment for Fast Development of Web Applications," IEEE International Conference on Multimedia Computing and Systems pp. 8091999.

[51] T. Isakowitz, A. Kamis and M. Koufaris "Reconciling top-down and bottom-up design approaches in RMM," ACM SIGMIS Database vol. 29, no. 4, pp. 58-67, 1998.

[52] T. Isakowitz, A. Kamis and M. Koufaris "Extending the capabilities of RMM: Russian Dolls and Hypertext," 30th Hawaii International Conference on System Sciences (HICSS) vol. Volume 6: Digital Do, pp. 1771997.

[53] A. Diaz, T. Isakowitz, V. Maiorana and G. Gilabert "RMC: A Tool to Design WWW Applications," 4th International World Wide Web Conference 1995.

[54] P. Merialdo, P. Atzeni and G. Mecca "Design and development of data-intensive web sites: The Araneus approach," ACM Transactions on Internet Technology (TOIT) vol. 3, no. 1, pp. 49-92, 2003.

[55] P. Atzeni and A. Parente "Specification of Web Applications with ADM-2," 1st International Workshop on Object Oriented Software Technology 2001.

[56] D. Schwabe and G. Rossi "Developing hypermedia applications using OOHDM," Workshop on Hypermedia Development Processes: Methods and Models, Hypertext'98 1998.

[57] D.D. Cowan and C.J.P. Lucena "Abstarct Data Views, an Interface Specification Concept to Enhance Design for reuse," IEEE Transactions on Software Engineering vol. 21, no. 3, 1995.

[58] D. Schwabe, R.A. Pontes and I. Moura "OOHDM-Web: An Environment for Implementation of Hypermedia Applications in the WWW," SigWEB Newsletter vol. 8, no. 2, 1999.

[59] M. Fernández, D. Florescu, A. Levy and D. Suciu "Declarative specification of Web sites with STRUDEL," The VLDB Journal — The International Journal on Very Large Data Bases vol. 9, no. 1, pp. 38-55, 2000.

[60] M. Fernández, D. Suciu and I. Tatarinov "Declarative specification of data-intensive Web sites," Proceedings of the 2nd conference on Domain-specific languages pp. 135-148, 1999.

[61] M. Bichler and S. Nusser "Modular design of complex Web-applications with W3DT," 5th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE'96) pp. 3281996.

[62] M. Bichler and S. Nusser "Developing Structured WWW-Sites with W3DT," WebNet96 1996.

[63] A. Scharl "A conceptual, User-Centric Approach to Modelling Web Information Systems," The Fifth Australian World Wide Web Conference 1999.

[64] S. Comai and P. Fraternali "A semantic model for specifying data-intensive Web applications using WebML," International Semantic Web Working Symposium (SWWS) 2001.

[65] S. Ceri, P. Fraternali and S. Paraboschi "Data-Driven, One-To-One Web Site Generation for Data-Intensive Applications," VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK pp. 615-626, 1999.

[66] J. Conallen "Modelling Aplication Architectures with UML," Communications of the ACM vol. 42, no. 10, pp. 63-70, 1999.

[67] A. Knapp, N. Koch, F. Moser and G. Zhang "ArgoUWE: A CASE Tool for Web Applications," First International Workshop on Engineering Methods to Support Information Systems Evolution (EMSISE03) held in conjunction with OOIS03 2003.

[68] N. Koch and A. Kraus "The expressive Power of UML-based Web Engineering," 2nd International Workshop on Web-oriented Software Technology (IWWOST02) 2002.

[69] R. Hennicker and N. Koch "Systematic Design of Web Applications with UML," Unified Modeling Language: Systems Analysis, Design and Development Issues 2001.

[70] H. Baumeister, N. Koch and L. Mandel "Towards a UML Extension for Hypermedia Design," UML'99 - The Unified Modeling Language. Beyond the Standard. vol. 1723, pp. 614-629, 1999.

[71] C.C. J. Gómez, O. Pastor "On Conceptual Modeling of Device-Independent Web Applications: Towards a Web Engineering Approach," IEEE Multimedia vol. 8, no. 2, pp. 20-32, 2001.

[72] J. Gómez and C. Cachero "OO-H Method: extending UML to model web interfaces," Information modeling for internet applications pp. 144-173, 2003.

[73] N. Koch, A. Kraus, C. Cachero and S. Meliá "Modeling Web Business Processes with OO-H and UWE," Third International Workshop on Web-oriented Software Technology (IWWOST03) 2003.

[74] L. Baresi, F. Garzotto and P. Paolini "Extending UML for Modeling Web Applications," 34th Annual Hawaii International Conference on System Sciences (HICSS-34) vol. 3, pp. 370-379, 2001.

[75] P. Dolog and M. Bieliková "Hypermedia Modelling Using UML," 5th International Conference on Information Systems Modelling pp. 79-86, 2002.

[76] P. Dolog and W. Nejdl "Using UML and XMI for Generating Adaptive Navigation Sequences in Web-Based Systems," 6th International Conference on the Unified Modeling Language: Modeling Languages and Applications (UML'2003) 2003.

[77] O. De Troyer and C.J. Leune "WSDM: A User Centered Design Method for Web Sites," 7th International World Wide Web Conference 1998.

[78] O. De Troyer and S. Casteleyn "The Conference Review System with WSDM," IWWOST 2001 2001.

[79] T. Halpin "Information Modelling and Relational Databases: From Conceptual Analysis to Logical Design," 2001.

[80] O. De Troyer and S. Casteleyn "Exploiting Link Types during the Conceptual Design of Web Sites," International Journal of Web Engineering Technology vol. 1, no. 1, 2003.

[81] R.A. Barta and M.W. Schranz "JESSICA: An Object-Oriented Hypermedia Publishing Processor," 7th International World Wide Web Conference pp. 281-290, 1998.

[82] M.W. Schranz, J. Weidl, K.M. Göschka and S. Zechmeister "Engineering Complex World Wide Web Services with JESSICA and UML," 33rd Hawaii International Conference on System Sciences vol. 6, 2000.

[83] H.-W. Gellersen and M. Gaedke "Object-Oriented Web Applications Development," IEEE Internet Computing no. Janary-February, pp. 60-68, 1999.

[84] H.-W. Gellersen, R. Wicke and M. Gaedke "WebComposition: An object-oriented support system for the Web engineering lifecycle," Computer Networks and ISDN Systems vol. 29, pp. 1429-1437, 1997.

[85] M. Gaedke, C. Segor and H.-W. Gellersen "WCML: Paving the Way for Reuse in Object-Oriented Web Engineering," ACM Symposium on Applied Computing (SAC 2000) 2000.

[86] H. Lee, C. Lee and C. Yoo "A scenario-based object-oriented hypermedia design methodology," Elsevier Information & Management no. 36, pp. 121-138, 1999.

[87] M.C.F.d. Oliveira, M.A.S. Turine and P.C. Masiero "A statechart-based model for hypermedia applications," ACM Transactions on Information Systems (TOIS) vol. 19, no. 1, pp. 28-52, 2001.

[88] P.D. Stotts and R. Furuta "Dynamic adaptation of hypertext structure," Proceedings of the third annual ACM conference on Hypertext pp. 219-231, 1991.

[89] P.D. Stotts, R. Furuta and C.R. Cabarrus "Hyperdocuments as automata: verification of trace-based browsing properties by model checking," ACM Transactions on Information Systems (TOIS) vol. 16, no. 1, pp. 1-30, 1998.

[90] W. Retschitzegger and W. Schwinger "Towards Modeling of DataWeb Applications - A Requirements' Perspective," Proceedings of the Americas Conferenc on Information Systems pp. 149-155, 2000.

[91] M.J. Escalona and N. Koch "Ingeniería de Requisitos en Aplicaciones para la Web: Un studio comparativo," Iberoamerican Workshop on Requirements Engineering and Software Environments (IDEAS'2003) 2003.

[92] C. Barry and M. Lang "A survey of Multimedia and Web Development Techniques and Methodology Usage," IEEE Multimedia vol. 8, no. 1, pp. 52-60, 2001.

[93] M. Gaedke and G. Gräf "Development and Evolution of Web-Applications Using the Web Composition Process Model," Web Engineering: Managing Diversity and Complexity of Web Application Development pp. 58-76, 2001.

[94] V. Bush "As We May Think," The Antlantic Montly vol. 176, no. 1, pp. 101-108, 1945.

[95] J. Rice, A. Farquhar, P. Piernot and T. Gruber "Using the Web instead of a window system," Proceedings of the SIGCHI conference on Human factors in computing systems pp. 103-110, 1996.

[96] H. Berghel "The client's side of the World-Wide Web," Communications of the ACM vol. 39, no. 1, pp. 30-40, 1996.

[97] P.H. Lewis, H.C. Davis, S.R. Griffiths, W. Hall and R.J. Wilkins "Media-based navigation with generic links," Proceedings of the the seventh ACM conference on Hypertext pp. 215-223, 1996.

[98] M. Moyle and A. Cockburn "The design and evaluation of a flick gesture for 'back' and 'forward' in web browsers," Proceedings of the Fourth Australian user interface conference on User interfaces 2003 pp. 39-46, 2003.

[99] P.H. Lewis, W. Hall, L.A. Carr and D.D. Roure "The significance of linking," ACM Computing Surveys (CSUR) vol. 31, no. 4es, pp. 101999.

[100] M. Noirhomme-Fraiture and V. Serpe "Visual representation of hypermedia links according to their types," Advanced Visual Interfaces (AVI'98) pp. 146-155, 1998.

[101] S. Chakrabarti, Gibson and K.S. McCurley "Surfing the Web Backwards," 8th International World Wide Web Conference pp. 601-615, 1999.

[102] A.M. Fountain, W. Hall, I. Heath and H. Davis "MICROCOSM: An Open Model for Hypermedia with Dynamic Linking," European Conference on Hypertext pp. 298-311, 1990.

[103] H. Davis "To embed or not to embed…," Communications of the ACM vol. 38, no. 8, pp. 108-109, 1995.

[104] J. Kahan and M.-R. Koivunen "Annotea: An Open RDF Infrastructure for Sgared Web Annotations," WWW10 pp. 623-632, 2001.

[105] H. Davis "Hypertext Link Integrity," ACM Computing Surveys vol. 31, no. 4, 1999.

[106] H. Weinreich, H. Obendorf and W. Lamersdorf "The look of the link - concepts for the user interface of extended hyperlinks," Proceedings of the twelfth ACM conference on Hypertext and Hypermedia pp. 19-28, 2001.

[107] E. Mbaki, F. Bodart, J.M. Leheureux and J. Vanderdonckt "Windows Transitions: A Graphical Notation for Specifying Mid-Level Dialogue Models," 7th DSVIS 2000.

[108] E. Mbaki and J. Vanderdonckt "Window Transitions: A Graphical Notation for Specifying Mid-level Window Transitions," Proceedings of 1st International Workshop on Task Models and Diagrams for user interface design TAMODIA'2002 pp. 55-63, 2002.

[109] J. Vanderdonckt, Q. Limbourg and M. Florins "Deriving the Navigational Structure of a User Interface," Interact 2003 pp. 455-462, 2003.

[110] L. Carr, D. De Roure, W. Hall and G. Hill "The Distributed Link Service: A Tool for Publishers, Authors and Readers," The 4th World Wide Conference 1995.

[111] H. Maurer "HyperWave: The Next-Generation Web Solution," 1996.

[112] N. Streitz, J. Haake, J. Hannemann, A. Lemke, W. Schuler, H. Schütt and M. Thüring "SEPIA: a cooperative hypermedia authoring environment," Proceedings of the ACM conference on Hypertext pp. 11-22, 1992.

[113] S.J. DeRose "XML linking," ACM Computing Surveys (CSUR) vol. 31, no. 4, pp. 211999.

[114] M. Bieber and F. Vitali "Fourth Generation Hypermedia: Some Missing Links for the World Wide Web," vol. 47, no. 1, pp. 31-65, 1997.

[115] D. Benyon, D. Stone and M. Wookdroffe "Experience with developing multimedia courseware for the WWW: the need for better tools," International Journal of HumanComputer Studies vol. 47, 1997.

[116] J. Conklin and M.L. Begeman "gIBIS: a hypertext tool for exploratory policy discussion," ACM Transactions on Information Systems (TOIS) vol. 6, no. 4, pp. 303-331, 1988.

[117] J. Nanard and M. Nanard "Using structured types to incorporate knowledge in hypertext," Proceedings of the third annual ACM conference on Hypertext pp. 329-343, 1991.

[118] J. Conklin "Hypertex: A Survey and Introduction," IEEE Computer vol. 20, no. 9, pp. 17-41, 1987.

[119] K. Andrews, F. Kappe and H. Maurer "Serving information to the Web with Hyper-G," 3rd International WorldWide Web Conference vol. 27, 1995.

[120] L. Hardman, J. van Ossenbruggen, L. Rutledge, K.S. Mullender and D.C.A. Bulterman "Do You Have the Time? Composition and Linking in Time-based Hypermedia.," 10th ACM conference on Hypertext and Hypermedia pp. 189-196, 1999.

[121] L. Hardman, D.C.A. Bulterman and G.v. Rossum "The Amsterdam hypermedia model: adding time and context to the Dexter model," Communications of the ACM vol. 37, no. 2, pp. 50-62, 1994.

[122] S. Pemberton "XHTML 1.0: The Extensible HyperText Markup Language. A Reformulation of HTML 4.0 in XML 1.0," 1999.

[123] S.J. DeRose, E. Maler and R. Daniel "XML Pointer Language (XPointer) Version 1.0," no. October 2003, 2001.

[124] S. DeRose, E. Maler and D. Orchard "XML Linking Language (XLink) Version 1.0," 2001.

[125] D. Brailsford "Separable hyperstructure and delayed link binding," ACM Computing Surveys vol. 31, no. 4, 1999.

[126] D. Harel and A. Naamad "The STATEMATE semantics of statecharts," ACM Trans. Software Engineering Methodology vol. 5, no. 4, pp. 293-333, 1996.

[127] D. Harel "On visual formalisms," Communications of the ACM vol. 31, no. 5, pp. 514-530, 1988.

[128] J.C. Campos, Harrison, M. D. "Formally Verifying Interactive Systems: A Review," DSVIS'97 pp. 109-124, 1997.

[129] I. Horrocks "Constructiong the user interface with statecharts," -253, 1999.

[130] M. Winckler, P. Palanque, F. C. and M.S. Pimenta "What does what with whom in Web Development?," Human-Computer Interaction Internationnal 2003.

[131] J. Fleming "Web Navigation: Designing the User Experience," 1998.

[132] W. Janssen and M. Steen "Rapid Service Development: An Integral Approach to e-Business Engineering," pp. 119-135, 2001.

[133] V. Balasubramanian and A. Bashian "Document management and Web technologies: Alice marries the Mad Hatter," Commun. ACM vol. 41, no. 7, pp. 107-115, 1998.

[134] P.F. Mori G., Santoro C. "CTTE: Support for Developing and Analyzing Task Models for Interactive System Design," IEEE Transactions on Software Engineering no. August 2002, pp. 797-813, 2002.

[135] F. Paterno, C. Mancini and S. Meniconi "ConcurTaskTree: A diagrammatic notation for specifying task models.," 13 International Conference on Human-Computer Interaction Interact'97 1997.

[136] D. Scapin and C. Pierret-Golbreich "Towards a method for task description: MAD," Work with Display Units WWU'89 pp. 27-34, 1989.

[137] M. Winckler, P. Palanque, C. Farenc and M. Pimenta "Task-Based Assessment of Web Navigation Design," TAMODIA'02 Task Models and Diagrams for User Interface Design 2002.

[138] M. Winckler, E. Barboni, C. Farenc and P. Palanque "SWCEditor: a Model-Based Tool for Interactive Modelling of Web Navigation," Computer-Aided Design of User Interfaces (CADUI'2004) 2004.

List of Figures

List of Tables

Annexe A – SWC Document Type Definition (DTD)

```
<!ELEMENT swc (CompositeState, Transition*)>
<!ATTLIST swc
    project CDATA #IMPLIED
    author CDATA #IMPLIED
    date CDATA #IMPLIED
    version CDATA #IMPLIED
>
<!ELEMENT CompositeState ((CompositeState | BasicState)*, DeepHistory?, ShallowHistory?, EndState?, Stereotype*,
GraphicBounds?)>
<!ATTLIST CompositeState
    id ID #REQUIRED
    label CDATA #IMPLIED
    file CDATA #IMPLIED
    initial IDREF #IMPLIED
    concurrent (false | true) #REQUIRED
    entryAction CDATA #IMPLIED
    doAction CDATA #IMPLIED
    exitAction CDATA #IMPLIED
>
<!ELEMENT BasicState (Stereotype*, GraphicBounds?)>
<!ATTLIST BasicState
    id ID #REQUIRED
    label CDATA #IMPLIED
    type (BasicState | TransientState | DynamicState | ExternalState) #REQUIRED
    file CDATA #IMPLIED
    entryAction CDATA #IMPLIED
    doAction CDATA #IMPLIED
    exitAction CDATA #IMPLIED
>
<!ELEMENT DeepHistory (GraphicBounds?)>
<!ATTLIST DeepHistory
    id ID #REQUIRED
>
<!ELEMENT ShallowHistory (GraphicBounds?)>
<!ATTLIST ShallowHistory
    id ID #REQUIRED
>
<!ELEMENT EndState (GraphicBounds?)>
<!ATTLIST EndState
    id ID #REQUIRED
>
<!ELEMENT Transition (Stereotype*, Point*)>
<!ATTLIST Transition
    id ID #REQUIRED
    type (user | system | completion) #REQUIRED
    label CDATA #IMPLIED
    source IDREF #REQUIRED
    target IDREF #REQUIRED
    trigger CDATA #IMPLIED
    guard CDATA #IMPLIED
    action CDATA #IMPLIED
>
<!ATTLIST GraphicBounds
    x CDATA #IMPLIED
    y CDATA #IMPLIED
    width CDATA #IMPLIED
    height CDATA #IMPLIED
>
<!ELEMENT Stereotype (#PCDATA)>
<!ELEMENT GraphicBounds EMPTY>
<!ENTITY % version "StateWebCharts 1.0">
<!ENTITY % copyright "Marco Winckler - winckler@univ-tlse1.fr">
<!ELEMENT Point EMPTY>
<!ATTLIST Point
    x CDATA #REQUIRED
    y CDATA #REQUIRED
>
```

Annexe B – Document-type for specification of Web Applications

Plan-type d'un Dossier de Spécification pour la réalisation d'un site Web (DSW) V0-R4 du 06/10/03

Groupe de travail ENAC, LIIHS-IRIT, ICAM

Ce document expose les exigences du « client » pour la réalisation du site complet (partie statique et dynamique).
L'objectif est que ce document soit approuvé par le client et le fournisseur avant la conception et l'implémentation du site.
Il est donc rédigé après prise de connaissance du cahier des charges et discussions éventuelles pour précisions, complément, reformulations. Il peut d'ailleurs avantageusement servir de trame à ces discussions.

Il expose ce que le site doit faire mais ne présente pas les solutions techniques choisies sauf si ce sont des exigences, des contraintes.

Certains paragraphes du document ne s'appliqueront pas à votre projet; indiquez dans ce cas la mention: NEANT ou NON APPLICABLE.

1. But du document

Ce document décrit ce que doit faire le site

2. Documents applicables et de référence

2.1 Documents applicables

Figureront ici les documents que vous devez impérativement respecter. Ex : charte graphique, plan documentaire, méthodes de réalisation, cahier des charges, recommandation du W3C Web Accesibility Guidelines (WCAG 2.0) ...

2.2 Documents de référence

Figureront ici les documents dont vous pouvez vous inspirer, les références des sites dont vous vous inspirez.

3. Terminologie

Explication des sigles et abréviations, non usuels, utilisés dans le document.

4. Objectifs du site

Indiquez ici la(les) raison(s) d'être du site : à quoi va-t-il servir ?
Ex : "Donner des conseils sur la gestion de projets web aux créateurs de sites".

Indiquez les objectifs qualité.

Ayez en tête que l'atteinte de ces objectifs va être vérifiée donc mentionnez des objectifs vérifiables, quantifiables. Mentionnez les critères pour l'évaluation de l'utilisabilité.

5. Présentation des visiteurs

Dressez le(s) portrait(s), le(s) plus précis possible, des internautes que vous ciblez : tranche d'âge, degré de dextérité à naviguer sur le Web, nationalité, ... Le profil des visiteurs aura une influence sur la construction du site.

6. Spécifications générales

6.1 Scénarii d'utilisation

Ex : modélisation des scenarii avec Use Cases UML, modèle de tâches…

6.2 Description des fonctions du site

*Présentez ici, de façon détaillée, toutes les fonctions du site **issues des besoins du client et des utilisateurs**. Ex :*

- *présentation de société,*
- *site monolingue, bilingue, multilingue,*
- *choix d'articles,*
- *moteur de recherche,*
- *paiement en ligne,*
- *composition d'images,*
- *affiliation,*
- *webrings,*
- *mesure d'audience*
- *espace de discussion (chat)*
- *foire aux questions*
- *services mail…*

6.3 Référencement auprès de moteurs de recherche

Indiquez ici si un référencement est demandé. Indiquez les mots clés qui caractérisent l' (les) activité(s) présentées via le site et ceux par lesquels les internautes vous rechercheront le plus probablement.

6.4 Exigences opérationnelles

6.4.1 Politique de maintenance

Exemples :

- rapports sur les problèmes d'utilisabilité
- gestion du serveur web
- alimentation des informations du site

6.4.2 Exigences de présentation du site

Exemples :

- fil conducteur
- ordre de présentation des pages…

6.4.3 Règles d'ergonomie

Donnez toutes les caractéristiques qui complètent la charte graphique qui est applicable (cf paragraphe 2.1). Exemples :

- *layout*
- *couleur*
- *polices*
- *taille en Kb pour le pages*
- *photos*
- *images animées*
- *animation interactive*
- *son*
- *vidéo…*
- *Règles générales pour le site. Ces règles comprennent de règles ergonomiques aussi bien que d'autres règles générales pour le bon fonctionnement du site. ex. inclure (ou pas) la date de la dernière actualisation en bas (ou ailleurs) de chaque page ; rajouter (ou pas) le nom du Webmaster ou de la personne à contacter en cas de problèmes avec le site. Ex. de règles ergonomiques applicables : toutes les pages du site doivent avoir un lien vers la page principale; toute les pages doivent impérativement avoir un titre significatif que désigne proprement le contenu de la page en question, etc.*

6.4.4 Composants d'interface utilisables dans l'application

Doit-on utiliser une librairie d'objets graphiques déjà établis ou bien doit-on développer la librairie ?

6.4.5 Echange de bannières et de liens

Doit-on prévoir des échanges de bannières et de liens ?

6.5 Contraintes

6.5.1 Hébergement

Volume à stocker chez l'hébergeur exprimés en Mo.
Volume du trafic sur le site exprimés en Mo.
La bande passante.
Localisation géographique des ordinateurs de l'hébergeur.
Convivialité, simplicité de l'interface d'administration de l'hébergeur
Autonomie de gestion.
Services et support demandés (assistance, scripts serveurs, bases de données...).

6.5.2 Nom du site, du domaine

Pour vérifier au plus tôt la disponibilité du nom du domaine. Le nom (monsite. com, .net, .org, .fr) sera choisi en fonction de l'activité (professionnelle commerciale ou non…), de l'image à véhiculer…

6.5.3 Matériel imposé

Mentionnez ici le matériel imposé pour le développement.

6.5.4 Méthodes de développement imposées

Mentionnez ici les méthodes de développement imposées pour le développement de la partie dynamique mais aussi les méthodes requises pour l'évaluation d'utilisabilité.

6.5.5 Logiciels imposés

Y compris langage de programmation

6.5.6 Algorithmes imposés

7. Matrice de traçabilité avec l'expression de besoin