

Hypertext Semantics for Web Applications

Marco Brambilla, Sara Comai, Piero Fraternali

Politecnico di Milano - Dipartimento di Elettronica e Informazione
P.zza L. da Vinci, 32 – I20133 Milano, Italy
e-mail: mbrambil,comai,fraterna@elet.polimi.it

Abstract. Web applications integrate dynamic pages for publishing data, functions for content management, and generic business services. To support the model-driven design and automatic generation of Web applications, an extended notion of hypertext is required, whose semantics is scarcely investigated. In this paper, we analyse and classify the semantic problems encountered in the computation of Web pages, characterised by multiple interrelated pieces of content, inter-page and intra-page navigation, and content update operations. The identified semantic framework may help in better understanding the existing Web conceptual models, and in reasoning at a higher level about the implementation problems daily faced by Web developers.

1 Introduction

A Web application is a complex software artefact that integrates back-end data and business functions, and serves them to users by means of one or more front-end Web sites, accessible with a standard browser [Con00]. To manage the complexity of developing and maintaining a Web application, a software development methodology must be established, which blends the traditional principles and techniques of software engineering and the specific aspects of the Web. An important step towards a comprehensive Web application development methodology is the definition of proper conceptual models supporting the analysis and specification of the front-end Web sites, which is an activity not covered by state-of-practice software development methodologies, like OMT [RB+91] and UML [RJB98]. In the past few years, several authors have proposed conceptual frameworks for capturing the requirements of Web sites [AMM98, BGP01, GPS93, GCP01, SR98, KKH01], modeling different perspectives, such as data structure, page composition, navigation, and presentation. Some contributions [AMM98, CFB00, FP98, GCP01, WebR, CodeC] have also demonstrated the feasibility of building a new generation of Web-enabled CASE tools, which are able to semi-automatically implement Web sites for dynamically publishing information from conceptual specifications. However, Web applications typically comprise not only data publishing, but also content management, workflow, messaging, and more. These features pose new challenges to the Web conceptual models: first, an extended notion of hypertext must be studied, which harmoniously

integrates primitives for content publishing, navigation, content update, and generic operation invocation. This paper addresses the semantic problems of designing such an extended hypertext model for supporting the development of Web applications. In particular, we investigate the following questions:

- What are the alternative approaches for computing the dynamic pages of a Web application, when such pages contain several interrelated pieces of content?
- How is page computation affected by adding content-update operations to the hypertext model?
- What are the modeling errors or anomalous situations (e.g., non-determinism, missing input) that may arise in the definition of an extended hypertext?

The above questions are daily faced by implementors, who work on their Web applications at the source-code level, and devise ad-hoc solutions. We claim that raising the conceptual level at which such questions are investigated is mandatory, if we want to construct better CASE tools capable of semi-automatically generating Web applications from high-level specifications.

2 Basics of the WebML Conceptual Model

Throughout the paper we will present the semantic problems by means of several examples, by adopting the notations and the terminology of the Web Modelling Language (WebML) [CFB00], a language which incorporates features for expressing both read-only Web sites and Web applications with content updates and generic operations. However, the presented examples and results are independent of the specific modeling notation and could be applied to other specifications languages; moreover, they are relevant also to developers not using any conceptual modeling tool, because the problems and design choices exposed in this paper occur also at the implementation level.

In WebML, as well in most frameworks proposing conceptual models for Web applications [AMM98, BGP01, CFB00, GCP01], a conceptual Web specification consists of two major components: the *data model*, describing the conceptual organization of the application data, typically described by means of the Entity-Relationship model or an object class diagram; and one or more *hypertexts* defined on top of the data model, which express the dynamic content and linking of pages used to publish the application data. We present their main characteristics by means of an example.

Figure 1 shows an example of Entity-Relationship schema, describing the data for the publication of a hypertext about the tracks of a conference and the submitted papers. Entity Track represents all the tracks of a conference; entity Paper describes the papers submitted to the conference, and entity User represents either an Author or a Reviewer as indicated by the generalization hierarchy. Authors submit papers (relationship between Author and Paper), while reviewers review papers (relationship between Reviewer and Paper) and are associated to a track (relationship between Reviewer and Track). Finally, each paper is assigned to a unique track (relationship between Paper and Track).

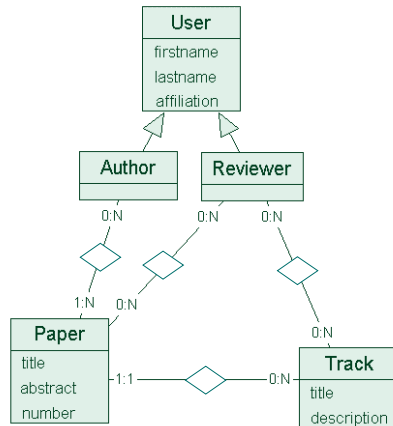


Figure 1. Example of data model

This schema represents the organization of the data of the application which will be stored in a data source. Once the data model is defined, one or more hypertexts can be defined for presenting on the Web the information extracted from the data source. Several notations have been proposed for the specification of hypertexts. In WebML an hypertext consists of a set of *pages* connected by *links*, describing the navigation and information flow in the hypertext; the content of the pages is represented by atomic content *units* representing at a conceptual level homogeneous dynamic information to be published, as shown in the following example.

Figure 2 shows an instance of a simple hypertext fragment, composed of three pages. The Home page contains only static content and provides a link (labelled with “See tracks”) to reach the Track page. The Track page shows an index of all the tracks of the conference. The user may select one of such tracks (presented to the user as anchors) to display its details in the Paper page, which shows the data of the selected track and lists the papers submitted to it.

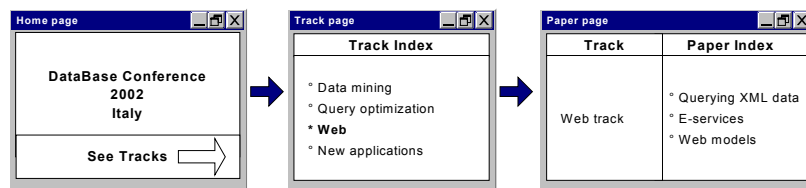


Figure 2. An instance of an hypertext defined on the conference data model

In WebML these pages are modelled like in Figure 3. From the home page a link, depicted with an arc, allows the user to reach the Track page, which shows an index of tracks, graphically represented by means of a “unit” labelled with Track index. When the user selects one track from the index the Paper page is displayed, showing the details of the selected track (represented by the unit labelled with Track data) and all the papers submitted to the selected track (represented by the unit labelled with

Paper index). Units representing indexes of objects are denoted with an index icon, while units representing a single instance are denoted with a data icon. Note that the Track index, the Track data and the Paper index units collect homogenous information: the formers collect tracks, while the latter collects papers.

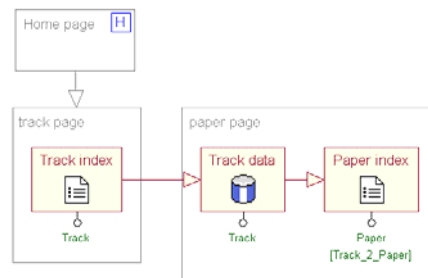


Figure 3. Extended hypertext instance and its representation in WebML

The pieces of content represented by the units can be related one to each other: for example, the data of the track shown in the Paper page depends on the choice performed by the user in the Track page; the list of papers shown in the Paper page depends on the currently displayed track.

WebML represents this coupling *between units* explicitly by means of *contextual links*: contextual links are links connecting two units and transporting some information, called *context*, used by the destination unit to display the correct information. In the above example, the Track index unit is linked to the Track data unit to provide the identifier of the selected track to be displayed in the Paper page; moreover, the Track data unit is linked to the Paper index unit to provide the data of the currently displayed track so that only the papers of such track be displayed. Note that each unit receiving an input context computes its content on the basis of such context: the Track index unit shows *all* the conference's tracks; the Track data unit shows only the track whose data are received in input; the Paper index unit shows all the papers connected by the input track (this condition is represented by the relationship [Track_2_Paper] below the icon). The context transported in output by each unit depends on the kind of unit: index units provide the data of the item selected from the index, data units provide the data of the instance they represent.

Links defined *between pages*, like for example the link exiting the Home page, do not carry any context and are used just to change page.

The basic ingredients of WebML are therefore pages, units defined inside a page to describe its content, and links between pages and units. We did not explain all the details of the WebML language, that the reader may find in [CFB00]; for the sake of simplicity, we only presented the basic concepts which will be used in the following sections.

In addition to the specification of read-only Web sites, where the user interaction is limited to information browsing, some proposed models also support applications requiring write access to the information used in a site (e.g., the filling of a shopping trolley or the update of the users' personal information). Common basic write

operations are the creation, modification and deletion of instances of an entity, or the creation and deletion of relationship instances.

Let us introduce a simple example, showing the creation of a new instance in the Track entity. Consider the hypertext represented in the left part of Figure 4. The Insert Track page shows the list of the currently available tracks and includes an entry form to insert the name of a new track. When the user clicks on the submit button the input data are used to create a new track instance into the data repository. If the creation succeeds, the New Track page is reached, otherwise an error page is displayed. In WebML this hypertext is modeled as shown in the right part of Figure 4: the Insert Track page contains two independent units, one showing the list of the available tracks, and one showing the entry form; the entry form provides the input to a create operation represented outside of the page, having two output links: one is followed when the operation succeeds the other one is followed in case of failure.

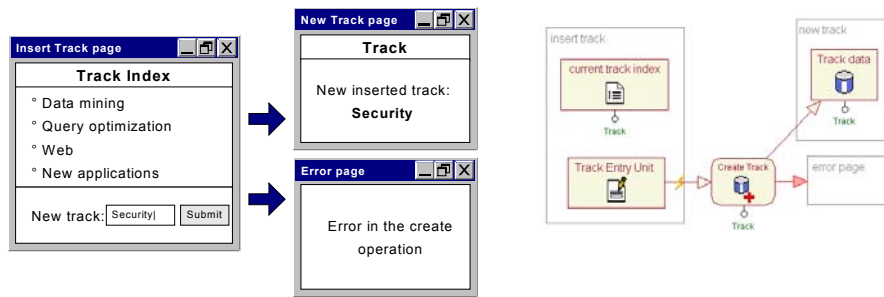


Figure 4. Example of data creation and its WebML representation

3 Semantics of Web Applications

Conceptual models like WebML allow people to better investigate the possible behaviours of hypertexts. On the basis of the fundamental concepts of WebML this section explains which are the alternative approaches for computing the dynamic pages of a Web application by individuating a set of *orthogonal semantic dimensions*. Indeed, the semantics of a page containing a set of linked units depends on several factors, such as the characteristics of the link, the policies to be adopted when more contexts are available for the same unit, and so on. On the basis of the different semantic dimensions a brief description of an algorithm for computing the content of a generic Web page is then provided.

3.1 When to Compute a Web Page

The content of a page must be computed when the following *computation events* arise:

1. *Inter-page link navigation*: the page is entered through a link originating in another page: in this case the whole content of the page must be properly initialized. For example, in the hypertext of Figure 2 the link departing from the Home page is an inter-page link, since after its navigation a new page is displayed, whose content must be computed.

2. *Intra-page link navigation*: the user clicks on an anchor inside a page and the target of the corresponding link is a unit of the same page: in this case the content of the page must be refreshed, because a user's click may produce new context information for a unit and affect the content of other units linked to it. Intra-page link navigation includes also the case of *content update*, where the user clicks on an anchor inside a page and triggers a sequence of operations and after performing the operation the same page is re-displayed. Firing the operation may have side effects on the content of the currently visualized page (e.g., it may modify content actually shown in the page) and may invalidate (partially or totally) the information used to compute the page from which the operation has been invoked. Examples of intra-page link navigation will be shown in the following sections.

3.2 Context Transportation

Links can be *contextual* or *non-contextual*, depending on the fact that they transport context or not: this dimension affects the page execution semantics because contextual link navigation may provide new context to units and therefore may affect the data content of a page. For example, in the page represented in Figure 5, a list of papers is presented and the user may select one paper. Every time a new paper is selected, the paper data must be refreshed to show the selected paper. This is an example of intra-page link: by clicking on an anchor inside the page, the same page is still displayed, but some data of the current page change. This link is also contextual since a new context is provided to the piece of page displaying the selected paper. In WebML this page is represented by two units (an index unit and a data unit) connected by an intra-page contextual link.

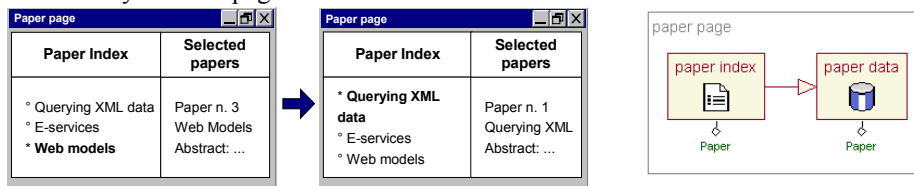


Figure 5. Contextual link: example of two subsequent choices from the paper index

Conversely, non contextual links do not transport input context, and thus cannot alter the input context of units.

3.3 Context Propagation

A Web page typically contains several related information; in WebML this corresponds to having several units linked in a network topology. When the page is accessed, some units are automatically instantiated, while some other units are “left blank” (e.g., the result of a search) and are filled only when the user clicks on the anchor of the unit’s input link. To capture this distinction, links may be classified as *automatic* or *manual*. The former are “automatically clicked” by the system at page entry, to propagate context from the source to the destination unit of the link even in absence of user’s action. The latter do not exhibit such behaviour, but the user must explicitly activate the link for context propagation to occur. We call *default context* the context automatically transported by an automatic link.

Consider the example of Figure 5 and suppose to enter Paper page for the first time: if the link exiting the index of papers is manual, no paper data are shown until the user selects an item from the index. If the link is automatic, the default paper (e.g., the first paper in the index) is chosen and the paper data are automatically shown without user’s intervention. Figure 6 shows what happens when the Paper page is accessed for the first time, in the case the intra-page link is automatic (Figure 6.a), and manual (Figure 6.b).

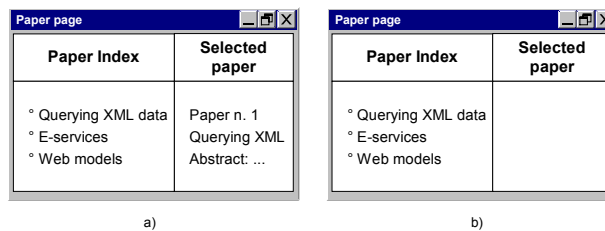


Figure 6. Possible rendition of Paper page, when the intra-page link is automatic (a) and manual (b)

3.4 Interaction History

When the user navigates a link, the content of the destination page is refreshed, in a way that may depend on the past history of the user interaction. For example, if the page includes several indexes for the user to choose among various items, it may be possible to record past choices. The alternatives for re-computing the page are captured by a further semantic dimension, called interaction history, which dictates the “degree of memory” of page computation:

1. *Without history*: the content of the units is computed as if the page were accessed for the first time.
2. *With context history*: the content of the units is calculated based on the units’ input context existing prior to the last navigation event. We call such context also *past context*.

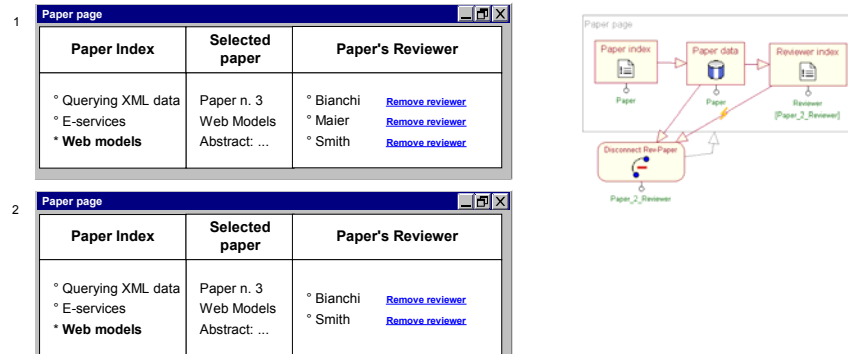


Figure 7. Links and history: the initial page (1), the page after computation with context history (2)

Consider the example 1) of Figure 7: Paper page lists all the papers and allows one to select one paper. The reviewers associated with the selected paper are also displayed. The user may disconnect a reviewer from the paper: an operation is invoked (by clicking on the anchor “Remove reviewer”), which deletes the relationship between the selected paper and the selected reviewer (i.e., the pair <paper-X, reviewer-Y>). Suppose that after this operation the page is redisplayed with the **computation with context history** semantics : the data content of the units is recomputed using the past context existing prior to firing the operation. The past context contains the paper selected before firing the operation which is needed to compute both the currently selected paper and its reviewers. The index of the papers is calculated afresh, while the currently selected paper and its reviewers are recomputed using the past context: if the pair <paper-X, reviewer-Y> has been removed, reviewer-Y no longer appears in the reviewers list of paper-X. See example 2) in Figure 7, where reviewer Maier has been removed.

The **computation without context history** option can be used to “reset” and forget the choices done by the user in a page.

3.5 Context Invalidation

Due to the side effects produced by operations and to the possibility of accessing the page in several different ways, it may happen that the past context existing prior to the last page computation event is no longer meaningful when the page is re-accessed after invoking an operation or following a contextual link. In such occurrence, the page context is said to be *invalid*. Context invalidation occurs in three cases:

1. Object deletion: when an instance of an entity is deleted all the units showing such instance are invalid.
2. Relationship deletion: when a relationship instance <object1, object2> of a binary relationship is deleted, all the units showing object1 related to object2 or, vice versa, object2 related to object1 are invalid.

- Contextual access: if a page is accessed through a contextual link (inter-page or intra-page) pointing to a unit U, the context of all units that depend on U is no longer valid.

As an example of point 1, consider the page in Figure 8, showing a list of papers, one selected paper and the index of the authors of such paper. The selected paper can be deleted, using the anchor labelled “Remove paper”. If after invoking the operation the same page is re-displayed, the previously selected paper is not *valid*.

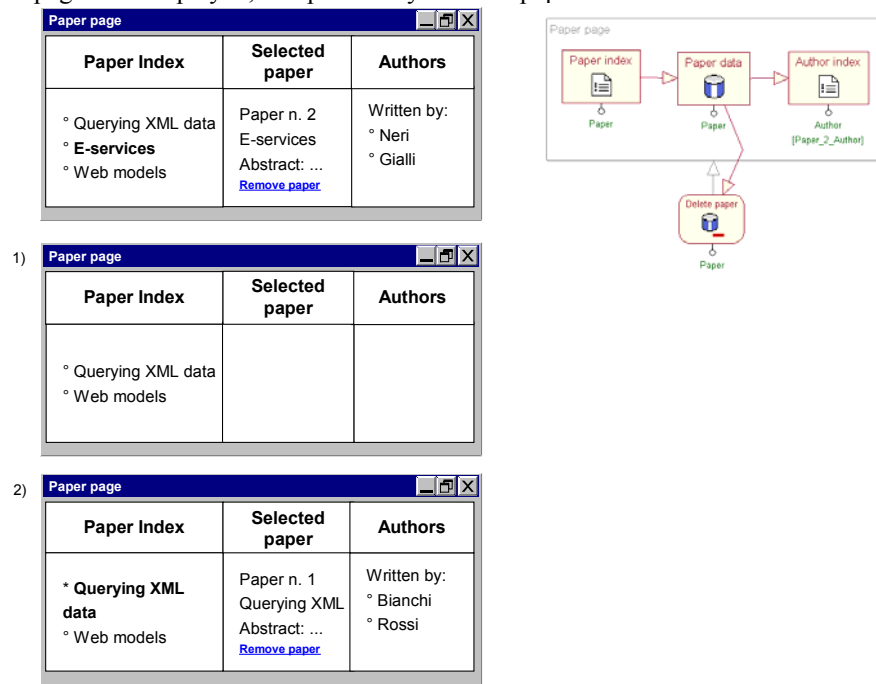


Figure 8. Example of context invalidation: the initial page, the page after deletion with cascaded content deletion (1) and with cascaded default replacement (2)

In response to context invalidation, the following behaviours are possible:

- Cascaded content deletion: the invalidated unit and all its dependent units are not computed (e.g., they are left blank). In the example of Figure 8, the data about the paper and its authors are not shown. See the resulting page 1).
- Cascaded default replacement: The invalidated unit is recomputed, using a default context, provided that such default context can be obtained from some automatic incoming link. All units that depend on the invalid unit are also recalculated in the same way. Supposing that all the contextual links in the example of Figure 8 are automatic, the default paper and its authors are shown. See the resulting page 2).

3.6 Conflict Resolution Strategy

A *conflict* arises when a unit receives more than one input context to be computed: the *conflict resolution strategy* (CRS) specifies which input context is selected to compute the data content of the unit. Three strategies are possible:

1. *Non-deterministic choice*: one input context is chosen non-deterministically at run-time among the set of available input contexts.
2. *With priorities*: priorities are assigned at compile-time to the input links of the unit and in case of run-time conflict the context transported by the link with highest priority is chosen.
3. *Mixed*: a partial order is defined at compile-time over the input links of a unit and in case of run-time conflict the context transported by the link with highest priority (if unique) is chosen. If the page is accessed at run-time in such a way that multiple links with highest priority are in conflict, a non-deterministic choice is taken.

Consider the example of Figure 9: an index listing the regular papers and an index containing the PC member papers are shown, and one paper can be selected from one of the two indexes. Suppose that the links exiting the indexes are automatic and therefore carry a default context to the unit showing the details of the selected paper. This unit could be instantiated either with the default entry of the Regular papers index unit or with the default entry of the PC papers index unit. In the non-deterministic conflict resolution strategy, the input context of the data unit is non-deterministically chosen at run-time between the two possible alternatives. In the priority-based conflict resolution strategy, priorities can be specified at compile-time on the links entering the Paper details data unit, to deterministically select one of the two input contexts when the page is accessed at run-time. The mixed solutions cannot be applied to this simple example, because at least three different input context are necessary.

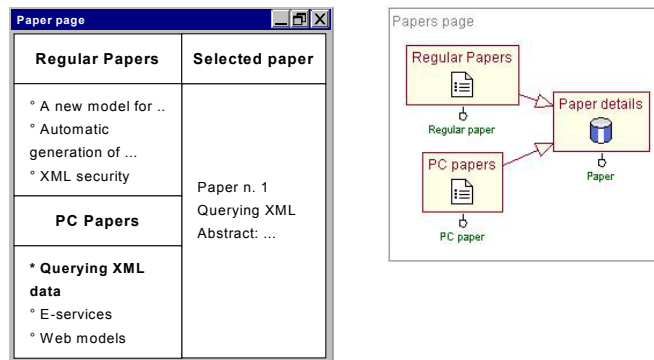


Figure 9. Non-deterministic or priority-based conflict resolution strategy

3.7 Page Computation Process

In this section we provide a brief description of an algorithm for computing the content of a generic Web page, summarising in a concise manner all the computation semantics issues illustrated in the preceding sections.

The page computation process is performed every time a computation event arises. The process tries to determine the data content of all the units of the page, taking into account the semantic dimensions presented in the previous subsections. Intuitively, the process determines at each step the set of “computable” units, i.e., of the units that receive their input contexts and therefore can be calculated, and instantiate such units. A unit is computable if it has no incoming links or if it has incoming links and the following conditions are satisfied: 1) the unit has not been already computed (unit cannot be computed more than once); 2) all the units from which the unit may receive a context have already been computed; 3) all the input contexts needed to compute the unit have a value. If the page computation semantics is without history default contexts are considered in point 3); if the page computation semantics is with context history units may draw their input values either from default contexts or from the past context existing prior to the last link navigation.

The algorithm computes a page starting from the following input parameters: it must receive the page to compute, the set of units to be considered in the computation (initially all the units of the page), the conflict resolution strategy, the interaction history policy, the past contexts of all the units prior to the last link navigation, the invalidation policy, the destination unit of the link whose navigation has produced the computation event together with the past context transported by the link. The basic steps of the algorithm are then computed:

1. Unit invalidation: If the destination of the navigated link is a unit, all its dependent units are invalidated. (We say that unit u_1 *depends* from unit u_2 if u_1 can be reached through contextual links from u_2 .)
2. Non-invalidated unit computation: one unit at a time is computed, until all possible units are considered: at each step, if there is at least one computable unit, one of them is selected and its content is computed, based on the conflict resolution strategy, the interaction history policy, and the values in the past context. In particular:
 - 2.1 If a unit does not depend from any other unit, i.e. it does not receive any input context, it can always be computed.
 - 2.2 If a unit is the destination unit of the link whose navigation has produced the computation event, then the past context and the new values of the link parameters are used for instantiating the unit.
 - 2.3 In all the other cases, the interaction history policy determines which context must be used: if the interaction history policy is “without history”, one of the possible input default contexts is chosen, according to the conflict resolution strategy; if the interaction history policy is “with context history” the past context is considered. If the past context is available and valid, it is used to instantiate the unit; if it is available but invalid, the unit cannot be computed and all its dependent units are invalidated; if no past context for the unit is

available, one of the possible default contexts is chosen according to the conflict resolution strategy.

3. Invalidated unit computation: if the cascaded default replacement policy applies the algorithm for computing the content of the page is invoked for the invalidated units with the “without history” policy, to automatically compute the invalidated units. If the cascaded deletion policy applies the process stops, since the invalidated units must not be computed.

4 Related Work

To our knowledge, very few authors have investigated the semantic issues involved in the design of conceptual models for dynamic hypertexts. Among the early works, [SF89] used Petri Nets to describe static hypertexts, where pages do not access content dynamically. Also [ZP92] addresses the navigation semantics of static hypertexts, using Statecharts instead of Petri Nets. [TFM99,FTM01] introduces HMSB - Hypermedia Model Based on Statecharts, to specify both the structural organization and the browsing semantics of static hypermedia applications. Here the focus is on synchronization of multimedia data (i.e. text, audio, animations, images and so on).

Among the Web conceptual design methodologies, *OO-HDM* [SR98] adopts an object-oriented approach, which permits a high degree of flexibility in defining the hypertext semantics. [SR98] presents several implementation techniques, but does not discuss context semantics at the level of the conceptual language.

HDM [GPS93] is another Web modeling notation, which has been implemented in two CASE tools (Autoweb [FP98] and JWEB [BPP99]). In the implementations of HDM, both contextual and non-contextual links are available, inter-page links are manual, whereas intra-page links are automatic, and a limited form of context history navigation is provided. A recent evolution of HDM, called W2000 [BGP01] introduces operations in the hypertext model, by leveraging extended UML interaction diagrams to describe how the information objects cooperate to accomplish a particular operation.

In *Araneus*, a logical model, called ADM (Araneus Data Model), allows one to represent the hypertext organization, with a notation similar to nested relations. Araneus uses manual inter-page links and does not allow intra-page links. Presently, ADM does not include operations: a second version is under development, called ADM2 [AP01], which associates actions to pages and navigation.

Finally, *OO-HMETHOD* [GCP01] is an UML-compliant approach for modelling and implementing Web application interfaces, where also operations can be considered. The dynamics of the site is represented with a diagram called Navigation Access Diagram (NAD), providing the constructs to describe the navigation behaviour, the object population selection, and the order in which objects should be navigated. The NAD model caters for several of the semantic dimensions analysed in this paper, like automatic/manual navigation and context history.

In the industry, Conallen [Con00] has proposed a popular set of UML extensions for specifying Web applications. Web pages are represented as UML components. In

particular, sequence diagrams are used to specify which operations to invoke, in which order and so on, thus completely specifying the behaviour of a page. However, Conallen does not envision a conceptual modeling phase, but all the models are very tied with the implementation aspects.

A few commercial tools have recently emerged, which leverage conceptual modeling and code generation in the Web context. *WebRatio* [<http://www.webratio.com>] adopts the WebML notation and generates dynamic Web applications coded in JSP. It supports many of the semantic options discussed in this paper: manual and automatic links, alternative interaction history options, and context invalidation. However, predefined interaction history policies are hard-wired in the code generator, and can be somehow tailored only using explicit session parameters storing the required context information.

CodeCharge [<http://www.codecharge.com>] is a visual environment for the specification of dynamic, data-intensive Web applications. Differently from WebRatio, CodeCharge includes a gallery of predefined patterns typically found in Web applications, which can be combined to obtain a dynamic Web site. However, the included patterns are not flexible enough to allow all the semantic dimensions, and hand-written code should be integrated to obtain the different behaviours described in the previous sections.

5 Conclusions

This paper investigates the semantic implications of extended hypertext models, i.e., conceptual models of Web applications including pages with multiple interconnected content units and operations. A set of semantic dimensions that affect page computation, and the investigation of the possible choices for such dimensions have been analysed. In this way, the paper proposes a much-needed terminology, which can be used by researchers and practitioners to classify and reason about the alternative ways of computing complex pages. This approach is traditional in other software fields (e.g., active databases [FT95]) and may help a deeper understanding of a new technology. Moreover, an abstract page computation algorithm is proposed, which may serve as guideline for understanding how complex pages are computed after inter-page link navigation, intra-page link navigation, and content update operations.

The identified dimensions and their values can be seen as the building blocks of the semantics of a conceptual Web design method applicable to extended hypertexts. However, also Web application programmers not using a model-driven approach may benefit from the proposed semantic framework, which brings into unity a variety of problems that typically arise in the hand-coding of Web template pages, like non-determinism, missing input, and page context invalidation.

Our future work includes further extending the analysis of Web application semantics along several directions: adding more primitives to the extended hypertext model, and investigating their impact on the identified page computation dimensions, or on new, not yet discovered, dimensions; analysing applications built by real-world users (both by hand and with model-driven CASE tools), to establish a connection between application requirements and best practices in the selection of the mix of page

computation options; evaluating existing conceptual models supported by CASE tools to discover their “hidden semantics”.

References

- [AMM98] Paolo Atzeni, Giansalvatore Mecca, Paolo Merialdo: Design and Maintenance of Data-Intensive Web Sites. EDBT 1998: 436-450
- [AP01] Paolo Atzeni, Alessio Parente: Specification of Web applications with ADM-2, 1st Int. Work. on Web-Oriented Software Technology (IWWOST'01) - informal proceedings
- [BPP99] M.A. Bochicchio, R. Paiano, P. Paolini: Jweb: an HDM Environment for Fast Development of Web Applications. Proc. Of IEEE Multimedia Computing and System 1999, Vol. 2, pp. 809-813
- [BGP01] Luciano Baresi, Franca Garzotto, Paolo Paolini: Extending UML for Modeling Web Applications. HICSS 2001
- [CFB00] S. Ceri, P. Fraternali, A. Bongio. "Web Modeling Language (WebML): a Modeling Language for Designing Web Sites". Computer Networks, 33, pp. 137-157 (2000).
- [CodeC] Codecharge. <http://www.codecharge.com>
- [Con00] J. Conallen. Building Web Applications with UML. Addison Wesley, 2000.
- [F99] P. Fraternali. Tools and Approaches for Developing Data-Intensive Web Applications: A Survey. ACM Computing Surveys 31(3): 227-263 (1999)
- [FP98] P. Fraternali, P. Paolini: A Conceptual Model and a Tool Environment for Developing More Scalable, Dynamic, and Customizable Web Applications. EDBT 1998: 421-435
- [FT95] P. Fraternali, L. Tanca: A Structured Approach for the Definition of the Semantics of Active Databases. TODS 20(4): 414-471 (1995)
- [FTM01] M.C. Ferreira De Oliveira, M.A.S. Turine, P.C. Masiero: "A Statechart-based Model for Modeling Hypermedia Applications". ACM TOIS, April, 2001
- [GPS93] F. Garzotto, P. Paolini, D. Schwabe. HDM - a Model-based Approach to Hypertext Application Design. ACM Transaction on Information Systems 11(1), January, 1-26, 1993.
- [GCP01] J. Gómez, C. Cachero, O. Pastor: Conceptual Modeling of Device-Independent Web applications. IEEE Multimedia. Special Issue on Web Engineering, IEEE 04 2001.
- [RB+91] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorenson: Object-Oriented Modeling and Design. Prentice Hall International, 1991.
- [RJB98] James Rumbaugh, Ivar Jacobson, Grady Booch: The Unified Modeling Language Reference Manual. Addison-Wesley Object Technology Series, 1998.
- [SF89] P. Stotts, R. Furuta: Petri-Net-Based Hypertext: Document Structure with Browsing Semantics. TOIS 7(1): 3-29 (1989)
- [SR98] Daniel Schwabe and Gustavo Rossi, An Object Oriented Approach to Web-Based Application Design. Theory and Practice of Object Systems 4(4), 1998. Wiley and Sons, New York, ISSN 1074-3224)
- [TFM99] M. A. S. Turine, M. C. Ferreira de Oliveira, P. C. Masiero: HySCharts: A Statechart-Based Environment for Hyperdocument Authoring and Browsing. Multimedia Tools and Applications 8(3): 309-324 (1999)
- [WebR] WebRatio: <http://www.webratio.com>
- [ZP92] Y. Zheng, M. Pong: Using Statecharts to Model Hypertext. ECHT 1992: 242-250, 1992