Chapter 2

TASK-BASED WEB MODELING: THE WEB OBJECT LIFE CYCLE MODELING CONCEPT

Birgit Bomsdorf\(^1\) and Gerd Szwillus\(^2\)
\(^1\)FernUniversität Hagen, Praktische Informatik I, Universitätsstraße 1 – D-58097 Hagen (Germany)
E-Mail: birgit.bomsdorf@fernuni-hagen.de
URL: http://pi1.fernuni-hagen.de/mitarbeiter/bomsdorf.html
Tel: +49 2331 987-2962 – Fax: +49 2331 987-314
\(^2\)Universität Paderborn, Fakultät EIM, Institut für Informatik, Fürstenallee 11 – D-33102 Paderborn (Germany)
E-Mail: szwillus@upb.de
URL: http://www.uni-paderborn.de/cs/ag-szwillus
Tel: +49 5251 60-6624 – Fax: +49 5251 60-6620

Abstract A web modelling approach is proposed, which uses a task model of the business as underlying basis and exploits it for adequate modeling of web sites including user interaction. We argue that a variant of classical task models, which emphasizes task objects and task views, is specially well-suited to support a model-based development of web sites. Its concepts and its practical use are discussed.

Keywords: Interactive web sites, Task based design, User-centered design, Web modeling.

1. INTRODUCTION

In the early days of the WWW, web sites were created for fast and easy dissemination of information to web site visitors. Gradually, though, web sites have developed from this world of static HTML pages towards interactive sites. In addition to receiving general information, the visitor inputs data and is provided individually computed feedback from the web site. The visitor might order something, deliver or request information specific to the business, or request a specific service. In short, the dialogue going on is governed by the tasks of the user in relation to the business.
If we look at current web modeling approaches, however, we see that they do not take the interactivity of web sites appropriately into account. Dialogue modeling – as done in HCI – is not a significant part of the process. Instead, the content-centered view on the domain objects is still the dominant issue of these approaches. We claim, however, that interaction design and feedback specification from the point of view of the users should be integral parts of a modeling technique for today’s web sites. As the information presented and the user dialogues provided in the web are both dependent on events “occurring” in and with the business, while the users perform their tasks, we propose to include a task model of the business as underlying basis for a web modeling approach and exploit it for adequate modeling of interaction and for supporting the web site update process.

2. MODEL-BASED APPROACHES

The model-based idea as such implies that the creation of some complicated artifact can be divided into the handling of several distinct levels dealing with separate aspects of the overall design. Following the motto “divide and conquer” the development is structured into separate design phases producing explicit, dedicated design documents – the models. Using explicit models results in typical benefits, such as verifiable and re-usable documents. Before dealing with our concrete web modeling approach, we first sketch and compare the existing HCI and web modeling approaches.

2.1 Model-based Approaches in HCI

In HCI there exist several model-based approaches for the development of user interfaces (e.g., [1,2,7,8,10,11,12,14,17]). Although a great number of variants of model-based approaches is discussed in HCI, there is a certain consensus about a collection of models and their importance for user interface (UI) development.

– Typically a model-based HCI approach starts with task modeling, describing the tasks from the user’s point of view – the user’s goals, activities, and basic actions are described. The model contains information about the task objects operated upon by the tasks, and includes information about the different types of users, referred to as role model.

– The task model is partitioned into different views, thus defining which tasks need to be dealt with together. This implies the visibility of means to trigger task execution and visibility of task objects to the appropriate user roles. These views can be seen as corresponding to dialogue elements showing subparts of the final user interface. This grouping is
guided by the necessity or usefulness for the user’s ability to operate on a set of tasks at a time [3].

The behavior of the user interface is described in two steps: first, the navigation between views is defined; second, the processing of user input events within single views is specified. Both parts together are referred to as the dialogue model of the user interface.

The navigation model defines the visibility of the different views. It specifies which views are initially visible when a system is started, and which events in the user interface turn views visible or invisible. Hence, it defines the overall navigation structure of the user interface. As the views are derived from collections of tasks, the switch between different views is strongly dependent of the underlying task model structure.

The processing model defines the details of user interaction within the single views. Typically, this is a state-based model, specifying the different states the user interface component can be in and the effects of events on the state. Effects can be state transitions, modifications of the output presented, application function calls, or arbitrary combinations of these.

In a final step within model-based approaches the details of what views and their contents look like is defined in a presentation model.

Using this task-based model suite results in a user interface design which is strictly based upon the user task specification and concentrates on the functionality of the application as seen from the user’s perspective.

2.2 Model-Based Development of Web Sites

Similar as in the field of HCI the model-based idea is applied to the development of web applications with the same general goals. Typically, a web site is modeled at the conceptual level to describe the information space, its structure and possible modifications independent from implementation issues. Model-based approaches, e.g., WebML [19], OOHDM [12], Strudel [14], RMM [8], and WSDOM [5], have their origin in the development of hypermedia systems and information systems, increasingly incorporating methods and techniques known from Software Engineering. SWCEditor is a recent approach concentrating on modeling web navigation [20]. Although there are significant differences between the approaches, there is consensus on the core of the activities to be performed and aspects to be modeled.

Requirements are mostly captured by means of narrative scenarios and use cases, i.e. the objectives of the web site, the prospective user groups and their tasks are written down. The task specification is typically a high-level task enumeration, and no refined description of task internals.

The domain model conceptually describes the objects of the business domain, their properties in terms of attributes, sub-object structures, and
semantic relationships. In some approaches also the intended application functionality is specified. Techniques adopted for defining this model correspond to well-known structures from Software Engineering (such as the class model in UML) or database engineering (ER diagrams).

– Additionally, in some approaches user-related domain models are introduced (e.g. navigational class schema in OOHDM [18], audience object model in WSDM [5]), which are defined as views on the domain model, similar to external schemas as known from database development. Each user-related domain model represents a substructure of the complete model by identifying those parts which are relevant to a supposed type of user or role. Hence, these views describe properties, and semantic links to be shown to the users, while they are performing their role-specific tasks.

– The navigation model shows possible ways for the user to access the information space in terms of navigation elements, i.e. nodes and links. Hence, it defines the content and logical structure of the pages, as well as accessing criteria (such as filtering or indexing) and types of navigation (such as guided tours or object lists).

– The presentation model captures how content and navigation commands should be visualized to the user. Although this model varies in the degree of abstraction within different approaches, it is often called an abstract interface design. It shows the perceptual structure of single pages in terms of hierarchical grouping and intra-page links. It defines where additional files (images, audio, video) are to be inserted, and in which way links are to be presented (e.g. textually or graphically). The main meaning of abstraction in the context of this model concerns the independence from any particular language and from devices used to deliver the pages.

– The need for personalization of web sites to individual users or user groups is generally increasing. It concerns user interface adaptation as well as the customization of the underlying functionality, i.e. it is related to content, presentation, and navigation, thus affecting all the models described above. At the model stage personalization aspects are summarized by the term personalization model, although they are often defined as special extensions of the existing models.

The main observation here is that there is no explicit model for user interaction and for the modification of content. It is not captured how the business functions work, i.e. which tasks are performed, and how they modify data, resulting in web site changes. Emphasis is on data modeling expressed in the domain model and on navigation within this information space.

2.3 Comparing HCI Models with Web Models

To strengthen the use and the role of task modeling within web modeling
approaches, one could ask whether it would make sense to apply HCI modeling approaches directly to web modeling. The problem with this idea is that the objectives in both design fields were different from the very beginning:

- The primary goal of a user interface is to provide access to some application functionality, the functional space, to the user. It provides means to trigger functions and input data or parameter values to feed a given semantic functionality.
- The primary goal of a web site is to provide access to an information space to the unknown user. This space is described in terms of nodes carrying content connected by links allowing the user to navigate through the space, which is structured as a collection of web pages. Following a link means “going to another place”, even if this may technically not be the case.

This is a shift in paradigm from modeling flow of action in user interfaces to modeling movement in information structures in the web. Let us look closer now into the single HCI models to find out more about the differences between the two fields.

As mentioned above, task modeling is used in both fields, although the degree of its use differs a lot. Both use task modeling for requirements gathering. Within the HCI approaches, however, task models are refined much further for describing the interaction model – they are richer in structure and content, e.g., defining conditions and sequencing of task execution. In consequence, task models are used as formal input to the subsequent constructive phases.

In web modeling, task models are used primarily as a means to explore requirements and are limited to high-level task descriptions (i.e. use cases or scenarios). They are used as starting point to describe either the global domain model or the user-related domain models (if supported). In WSDM [5], for example, task models are used to guide the design of the navigation, i.e. the content units and navigation tracks for different user groups, as well as the respective presentations. In OOHDM [18], task descriptions are analyzed to identify the data items, which are to be exchanged between the user and the web application. Hence, the task model is not exploited as a formal basis for the design of the site structure and interaction.

View modeling is used in both fields too, but with slightly different meanings. In the field of HCI, views are derived from the formal task model based on task grouping, leading to the definition of interaction elements for triggering tasks and information elements for displaying relevant task objects. Hence, the view definition originates essentially from task grouping.

In web modeling, the term view is used for two different aspects: On one hand, user-related domain models are defined as views on the domain model, which are similar to the models of task objects in UI modeling. On the other
hand, single nodes are considered as views on the underlying domain model. Defining such a view means to design the structure and content of a web page, typically derived from a data structure model of the web site. Hence, in web modeling the derivation of both kinds of views is strongly based on data modeling.

**Navigation Modeling** means the transition between different views (logical “windows” or pages) in both fields. In a user interface, navigation corresponds to a transition between groups of tasks, hence UI navigation between views corresponds to movement within a function space.

Navigation in web modeling, however, means movement within an information space. The view model provides the necessary starting and end points for the movement, which may additionally transport data provided by the user for selection and filtering content from the underlying information base. Hence it is strongly data oriented. As interaction with the web site is mapped onto navigation too, there is no explicit orientation of the navigation model towards the functions provided by the web site to the visitor.

The **processing model** as introduced by the field of HCI is the most detailed part, specifying what is happening within single views.

In web modeling, there is no concept for specifying operations on a single view. As mentioned before, interaction is described and implemented by means of links. As links are used for different purposes, the definition of interaction is mixed with other modeling aspects. In existing web modeling approaches it is “visible” to the designer, e.g., in the form of extensions (new symbols, e.g., in [3]). From the viewpoint of the user, the effects of submitting information to the web site or invoking an application operation, for instance by clicking on a button, is mapped onto navigation. As described in [4] invoking operations is modeled as side effects of activating links. Hence, there is no explicit concept in web modeling for dealing with the user’s action steps to execute a task – the user’s movement possibilities are modeled along the information structures underlying the web site.

Altogether, a sequence of web interactions while performing a task is modeled as a sequence of pages the user has to navigate through. Hence, each dialogue state is represented by a single view (page) and a dialogue sequence is defined by means of links, without distinguishing between the navigation and processing dialogue. Furthermore, there is no explicit concept of defining groups of related task, i.e. views as introduced by HCI, which is important to match the mental model of the user and thus for supporting usability.

Based on the approaches as known from HCI and web modeling, we started to develop a web modeling approach with the objective to integrate benefits from HCI methods with the needs of web modeling. The overall goal is the strict user/task-orientation of the process throughout all develop-
ment phases and the switch to data-centric issues only later in the process. Hence, the emphasis is on processes first and on information structures second, which is typically the other way round in web modeling approaches.

Within our approach we are developing a system, referred to as Task-Object-Based Web Site Management system [2], [15], [16]. In the following, we will use WOLM (Web Object Life Cycle Model) as an abbreviation for this concept, as this model is the kernel model of our approach.

3. TASK-OBJECT-BASED WEB SITES

Our approach is based strongly on an underlying task model, as is the case in HCI modeling. As mentioned above, a classical task model specifies primarily the task hierarchy, pre- and postconditions, and conditions for task execution. Most existing models take user roles and task objects into account too. Altogether, emphasis is on modeling of users’ “action” while they are performing their tasks; modeling of “things”, i.e. task objects, which are affected by those actions comes second.

![Diagram](image)

*Figure 1. Overview of the concept*

When applying the task model approach to web modeling, it is important to look at task modeling the other way round: The task objects modified during task execution are to be displayed in the web, and thus are of primary importance within the design process. Therefore, in our approach the core part is a task object model, referred to as Web Object Life cycle Model (WOLM), describing the task objects which will ultimately be represented in the web as well as their modifications resulting from task execution. The second basic part of the approach is the Abstract Website Structure (AWS), which structures the information described by the WOLM into web
In the following we discuss the two main parts of the concept and how they contribute to task-based modeling of web sites.

3.1 The Web Object Life Cycle Model

The WOLM contains web objects corresponding to the task objects of the business to be represented in the web site, as they have been identified during the task analysis phase. WOLM not only captures the objects’ data structures as done in a domain model, but specifies the modifications they undergo during task execution as well. Technically, objects are described by means of a class specification, which specifies the properties of a given type of objects. The class specification contains attribute declarations, initialization information, and methods to be performed “on” the objects of the specified type. Object modifications are described as state transitions corresponding to the task description they are derived from. The different web object states represent “situations” or “configurations” a web object can be in. They are used to describe the changes a web object undergoes during its life time through execution of tasks. The following example shows a specification derived from the task model describing the user task “prolongate a book” while using a library system:

```java
class Book { string signature;...

  prolongate() {...}

  field {prolongable, not_prolongable};

  prolongable -prolongate() -
  [numberProlongation < 3]: prolongable

  prolongable -prolongate() -
  [numberProlongation >= 3]: not_prolongable

...}
```

To define the dynamics, a WOLM class specification contains a definition of rules governing the transitions between states, specifying which state transitions are possible and what the respective target states are. Together with the state transition triggered by an event the effect of the transition on the object’s attribute values is described. The performance of methods, i.e. the execution of attribute modifications, may request input parameters from the user. The transition below, for example, specifies that within the login procedure the user has to fill in his “login” and “password”:

```java
not_logged_in -tryLogin(login, password) -
logged_in
```
The input request is an abstract model of the interaction process which will later on be mapped onto web site interaction elements.

WOLM specifies the modifications in the business from a strictly user centered point of view. It is a mirror of the task execution as performed in reality, described by the changes the task objects undergo. When specifying the state transition possibilities and their effects on objects, it is not defined where triggers for the state transitions originate. The model is explicitly concentrating on the effects and their interrelations between the objects. The model is independent from any web representation specification, hence the two design spaces “task model” and “web representation” are completely separated in this approach.

### 3.2 The Abstract Web Site Structure

The Abstract Web Site Structure (AWS) contains a hierarchy of elements structuring a web site, e.g., web pages, lists, graphics, text elements, links, and interaction elements such as buttons and text input fields. AWS is a pure structure model, as it defines what is shown on the web pages and does not specify how things look. While the WOLM defines the content to be displayed and the functions accessible, the AWS defines the distribution of the content onto web pages and the access to functions through the web pages, i.e. the AWS covers the specification of views, navigation and processing dialog. Before describing this let us consider the interrelations of both models.

AWS and WOLM are linked through references to single web objects or object lists, which are expressed in the AWS through specification of the web objects’ properties. At runtime, the attribute values of linked WOLM objects are available as information source to the AWS, and therefore to the representation in the web. This concept is similar as in content management systems, which link abstract objects (typically database records) to web pages and derive the content of the pages from their contents.

Another important link between WOLM and AWS is the fact that different states of objects can lead to different AWS fragments representing an object. The AWS contains condition clauses defining which elements are shown and not. Hence, the state transition of an object related to an AWS fragment, can change its AWS representation by changing its state in the WOLM model. This mechanism allows the visualization of semantically important object state information to the web site visitor.

Furthermore, based on task grouping, the AWS is used to define views, which have a similar function as in HCI. A view in HCI specifies groups of related tasks to match the mental model of the user. A page in the AWS also groups related tasks; the modifications occurring within such a view are modeled within WOLM, while the corresponding changing representation is
described within the AWS. The following AWS excerpt exemplifies a view on the task object “book” by means of which users of a library system can prolong an already borrowed book:

```java
page BorrowBook (Book b)

... button Prolongate Activate [b inState prolongable]
  action b.prolongate() -> this {processing dialog}
  link close -> HomePage {navigation dialog}
...
```

The example also shows a part of the processing dialog: An abstract button `Prolongate` is defined. Activating it will invoke the function `prolongate` of the WOLM book object `b`. As denoted by the `this` specification the same view, i.e. the same page, will be shown afterwards, of course in an updated version if necessary.

Basically, there exist two interaction elements in the web: text input fields and buttons. The user can click into text input fields and type text. If at some point this information is collected by the web site, e.g. by pressing the submit button of the corresponding form, the data is sent from the user’s web browser to the server. This event triggering behavior constitutes the second type of user interaction. When clicking on a button on a web page, as shown in the example above, the user triggers the execution of some application functionality.

The effect of a button click is twofold: first, reading user input from text input elements on the web page (if the button implements the “submit” functionality) and, second, displaying a new or redisplaying the same page (`this`). For the description of the navigation dialog we use a similar notation, e.g. `link close -> HomePage` specifies a link `close`, which will lead the user to the home page.

In the co-operation between AWS and WOLM, these two elementary functions are mapped onto two corresponding effects of a button press:

- First, the button can trigger a state transition of some WOLM object. In this case the values input by the web site visitor are transferred as parameters to this state transition. The state transition is then executed until the WOLM model has performed the corresponding modifications, including attribute modifications and inferred state transitions.
- Second, once the model is stable again, the same or some other page is displayed, using the “new” values of the WOLM model for this purpose.

Once an AWS is linked to a WOLM model, the remaining work to create the final web pages is to combine it with layout files containing templates for
defining the visual representation.

### 3.3 Cooperation between the Components

WOLM and AWS cover very different design spaces within the web modeling approach. By means of the AWS the structure of information presentation, including the navigation structure between these elements is designed. Most important is the dependency of AWS parts of WOLM objects, defined by declarative constraints, and the variants in the representation, dependent on objects’ states. Classical hyperlinks are modeled as well as the effect of interaction elements, and the two important components of a button element – the trigger of semantic changes and the display of some new page – are explicitly and separately specified. Hence, the two design spaces, modeling the functionality on one hand, and the displayed information structure and their properties on the other hand, are clearly separated, which allows to update, refine, and debug both parts independently of each other.

WOLM allows the designer to base the web site creation explicitly on a task model. The kernel model is a task object model specifying the web objects to be presented in the web, and defining their modification through task execution. The AWS contains both the view model as well as the model of the navigation and processing dialog. The views are defined by spreading the information to be displayed onto web pages and web page parts. The navigation is defined by specifying links from one node within the AWS to another one while the processing dialog is defined by abstract interaction elements whereby navigation within a view and between views is clearly separated.

Other than in HCI modeling, there is no dialogue model concentrated in a single component, which encapsulates the aspects “context” of a user interaction and its effect. These two aspects are separated in our approach, as the perceived context of an interaction is specified in the AWS, and the effect is specified in the WOLM kernel. This separation makes sense in the web modeling case, however, as designing these aspects needs developers with very different background and expertise.

In addition, exploiting the clear execution semantics of the WOLM allows the performance of a simulation of the model at a very early development stage. In addition, the “running” WOLM model can serve as underlying directing device for the live web site while the AWS deals with the resulting changes of the web presentation.

### 3.4 Status of the Work

To work successfully with a model-based approach such as WOLM, a tool environment is needed. We designed such an environment and identified
the following suite of tools:
- an **editor** to create and edit the WOLM,
- a **simulator** to execute the kernel model for to validate the dynamics of the specification,
- an **admin pages generator** to enable the web site owner to manipulate and define the contents of his web site via the simulator (see the screen shot in figure 2 below),
- an **AWS editor** to create and edit abstract web site structure documents, based upon an existing kernel WOLM model,
- an **AWS generator** to create an initial AWS automatically from the WOLM to allow fast and easy early-phase testing of the WOLM,
- a **layout Linker** tool to create and merge concrete HTML design files containing appropriate templates with the AWS, and
- a **runtime system** for WOLM, based upon the simulator, which incorporates the complete functionality for a living web site, including admin pages and a time stamp management for keeping the site up-to-date.

Most tools are currently under development and will soon be completed; emphasis up to now has been on the design and specification of the WOLM and AWS models, and their representation as XML files. A preliminary version of the runtime system has been developed and is working. In the context of language definition we currently look into applying the concept to realistically sized web projects.

![First prototype of the generated administration](image-url)
REFERENCES


