OOWS: A Method to Develop Web Applications from Web-Oriented Conceptual Models

Oscar Pastor, Joan Fons and Vicente Pelechano

Department of Information Systems and Computation
Technical University of Valencia
Cami de Vera s/n, E-46022, Spain
{opastor,jjfons,pele}@dsic.upv.es

ABSTRACT
This work presents an OO software production method that provides conceptual modelling extensions (in terms of models and abstraction primitives) to facilitate the web application specification and development process. It also discusses a strategy to automatically go from the problem space (conceptual models) to the solution space (final software product) in an automatic way, by implementing a set of mappings between conceptual primitives and their corresponding software representations. To include a practical view of the approach, the work describes a real case-study of a University Department Web System that has been developed with this method.

1. INTRODUCTION
Nowadays, the advance of the Internet and the emerging technologies associated to the web are universalizing information systems, allowing access to any connected potential user. The term “Web Application” [6] refers to the new family of software applications specially designed to be executed on the web. Due to the high growth of the commercial activities on the Internet, this systems are being implemented in very short periods of time, without support of appropriate tools. For this reason, Web applications has low quality and very difficult maintenance.

In the last years many methodological approaches (OOHDM [1], WebML[2], UWE [14], WSDM [5], AutoWeb[4], etc.) have been proposed. These approaches try to provide support for applications of this kind by means of abstract mechanisms that make easy the conceptualization and development of this systems. The term “Web Engineering” [3] emerges to represent the frameworks, methods and tools to develop web applications in all the web applications life cycle.

Our intention is to define a development method that enables us to specify a web application in an integrated framework. This method must capture the web applications requirements by means of conceptual models in order to automatically obtain operational prototypes from these specifications. To achieve this purpose, we have extended an existent object oriented method, called OO-Method [7]. This “classic” method allows us to capture functional system requirements to obtain an OO formal specification. We have improved this method by introducing new models to represent navigational and presentational characteristics of web applications. These specifications (with hypermedia characteristics) are a high level information repository of the system that are used as input for a conceptual model compiler. This compiler automatically generates a full system prototype for a specific target platform following a pattern oriented strategy. This extension of OO-Method with navigational and presentational capabilities is called OOWS (Object Oriented Web Solution).

In this paper we briefly present OOWS [8,9]. We describe the web application development process with this method by specifying all steps to build the conceptual model and the final web application. In order to better understand, we apply this method to the development of a University Department Web System. This system is oriented to show different kind of information about a university department, such as teaching information or research works. In addition, this Web application must provide tools for the management of the information depending on the kind of user (teachers, department staff, technicians, etc.)

The main contribution of this paper is the use of a method for the automatic production of Web applications applied to build our university department web system (www.dsic.upv.es). We follow the steps proposed by the OOWS approach to face this real web system. It allowed us to detect new navigational characteristics and refine our approach.

This document is divided in five sections. Section 2 presents the case study of a department web application. Section 3 introduces the OOWS approach applied to this case study. The new primitives detected to extend the method are presented in this section. Section 4 presents the generation strategy for automatically going from these conceptual models to a classic three (n) tier architecture web application (presentation, application and persistence). In particular, this paper focuses on the generation of the presentation tier from the specifications built in section 3. Finally, in section 5 we present some conclusions and further works.

2. UNIVERSITY DEPARTMENT’S WEB SITE
Currently, a Web application has been developed to give support for the operative/functional needs of the Department of Information Systems and Computation of the Technical University of Valencia in Spain. This application provides institutional, teaching and research information about the department. This information is put available to any department member and to (anonymous) Internet users. This application allows department members to explore and manage their personal information (teaching information, research results, resources management, etc.). Furthermore, it also allows students to see the information about their subjects, courses and teachers. This system also shows information about department groups of research and projects. Depending on the kind of member (teachers, grant holders, technicians, etc.) connected to the application, the application must provide him only the management tools that he is allowed to use (personalization).
3. WEB APPLICATION CONCEPTUAL MODELLING

This section presents the steps that we should follow to specify the functional, navigational and presentational requirements of a web application. According to the OOWS [8,9] approach, there exist two main steps: conceptual modelling and solution development. In the conceptual modelling step we obtain the system specification using conceptual models. It is divided into three sub-steps:

1. functional requirements elicitation. Techniques based on use cases and scenarios [10] are applied to build a conceptual schema
2. classic conceptual modelling. Using Structural, Functional and Dynamic models, the system structure and behavior is captured [7]
3. navigation and presentation modelling. A navigational model is built in order to model navigational requirements from the class diagram. Once the navigational model is built, presentation requirements are specified using a presentation model which is strongly based on the navigational model.

This paper focuses on navigational and presentational models. However, class diagram class is presented due to its importance in the definition of navigational aspects.

3.1. Case Study Class Diagram

This section presents a piece of the class diagram for the case study. It shows the classes used to model the structured information about the department, its members and its research groups.

![Class Diagram](image1.png)

Figure 1. Partial view of the class diagram for this case study

3.2. Navigational Modelling

In this section we present the navigational model. It is used to capture the navigational requirements of web applications by defining a "navigational view" (navigational map) for each kind of relevant users of the system. This view provides an accessibility structure (defining all possible navigational paths) that personalizes the system depending on the user type. To build this model, the method introduces two tasks:

1. User identification and categorization. In order to model navigation, first this approach provides a user diagram where we specify the system kind of users and how they are inter-related. This diagram also specifies the accessibility of each kind of user to the system information and functionality. Detailed information about this diagram can be found in [11]. In this case study, we have defined the following users: teacher, technician, grant hold, management and internet user.
2. Navigational diagram specification. For each kind of user, we should specify its navigational view (navigational map) of the system. This navigational diagram defines the basic interaction units with the user, describing which information is available user and how this information can be "navigated". Navigation personalisation capabilities can be captured within this model as discussed in [12].

3.2.1. Navigational Diagram Specification

In this step, a navigational map for each system user is defined. This map is represented using a directed graph whose nodes (navigational contexts) represent the basic interaction units between system and user (represented as UML [13] packages) and its arcs represent valid predefined navigational paths. Navigational contexts (stereotyped with the «context» reserved word) allow us to define a view over a set of class attributes and operations from the class diagram. Depending on the context reachability, there exist contexts of two kinds:

- "Always reachable nodes" or exploration navigational contexts (depicted with the "E" label) represent nodes that are reachable from any node. These contexts define implicit navigational links from any node and explicitly from the root of the map that is represented by the user (see Links and DSIC Info contexts in Figure 2). One exploration context can be set as default or home node (depicted with the "H" label). This primitive allows the user to automatically reach this context when it connects to the system (DSIC Info context).
- Sequence navigational contexts (depicted with the "S" label) can only be accessed following a predefined navigational path (see Tutorships context in Figure 3). When we define complex Web applications we need to organize and structure the navigational contexts. To deal with this requirement, this paper introduces a new primitive to the OOWS navigational model: the navigational subsystem. Navigational subsystems (stereotyped with «subsystem» reserved word) allow us to represent a logical group of navigational contexts and other navigational subsystems which share navigational properties. It is also represented as a navigational map that gets active when the user navigates to the subsystem.

![Navigational Map](image2.png)

Figure 2. Navigational Map for the Internet user

![Navigational Map](image3.png)

Figure 3. Navigational Map for the Directory Subsystem
Figure 2 shows the navigational map for the internet user. This map defines the (personalized) view of the system for this user, providing access to a set of navigational nodes (contexts and subsystems). Each node allows the user to obtain information about the Department, Teaching, Directory, Research, Intranet (subsystems), interesting Links and contact information about the department DSIC Info (contexts). Figure 3 presents the navigational map related to the Directory subsystem. It has a set of exploration contexts to provide information about a map place for the department, its collaborators and its staff. From this staff, it is possible to navigate to a sequential context to get information about the tutorships of the teachers.

Navigational links (graph links) represent reachability between navigational nodes, specifying valid navigational paths of the system for this user. New navigational requirements lead us to refine this primitive introducing the notion of user type change. This property allows us to model a change of interactive user type (after identification) to access the target node. Graphically, it is represented by a UML actor within the navigational link. As shown in Figure 2, the user must be identified as a Technician, Management user, Grant Hold or Teacher, to access the Intranet subsystem.

3.2.2 Navigational Context Definition

Navigational contexts are made up of a set of navigational classes (stereotyped with the «view» reserved word) referring to classes from the class diagram. Figure 4 shows the navigational classes Department Member, Office and Teacher in the Staff context. The set of class attributes and operations that will be visible for the user must be specified. All navigational contexts have a manager class from which all navigation starts (Department Member in Staff context and Teacher in Tutorships context) and optionally, complementary classes that provide additional information (see Figure 4, Office and Professor classes in Staff context and Subject and Tutor in Tutorships context). All navigational classes are connected by unidirectional binary relationships that can be defined over structural (association/aggregation/composition or specialization/generalization) relationships. Navigational relationships of two kind can be defined depending on a navigability capability: a (1) context dependency relationship (graphically represented using dashed arrows inside a context), that represents an information recovery of instances of the complementary class by crossing the structural relationship (see Figure 4, Department Member – Office relationship in the Staff context and Teacher - Subject and Subject - Tutor in Tutorships context). A (2) context relationship (graphically represented using solid arrows), that is a context dependency relationship that also defines a navigation capability to a target navigational context and creates a (sequence) link in the navigational map. To define that kind of relationships, the target context of the navigation must be specified (through a target attribute, depicted as [targetContext]), as well as the attribute that will be used as the anchor to activate the navigation to the target context (a link attribute).

Figure 4 shows the definition of the Staff and Tutorships navigational contexts for the Intranet user. The Staff context includes information about the members of the department (name, photograph and e-mail address) and their respective offices (code and phone number) by using a context dependency relationship. A navigation capability to the Tutorships context is defined with the context relationship with the Teacher navigational class. In the Tutorship context, the internaut user can see detailed information about the tutorships of the selected member (teacher) of the department (subject, day, start time and end time).

Due to Internet characteristics, some additional properties for improving the access to the information can be specified. These properties are described in the following section.

### 3.2.3 Advanced Navigational Features

Once navigational maps are built, additional mechanisms to structure the access and to filter the population inside navigational contexts might be defined. This information is captured in the specification of the navigational context (see the bottom area in the Staff context in Figure 4). These mechanisms are the following:

1. An index provides an indexed access to the information (by a property or a related object property) with respect to the objects of the manager class. Indexes create a list of summarized information allowing the user to choose one instance from the list. This selection causes this instance to become active in the navigational context. In our approach, indexes of two kinds can be specified: (1) Attribute indexes, which are defined over a subset of attributes of the manager class. They create the index using instances from that manager class, and (2) Relationship indexes, which are defined using a subset of attributes of a class related to the manager class. In this case, one index of all attribute values of the related class. When an index is activated, a list of the attribute values of the involved instances is built. At least one of these attributes must act as the link attribute “anchor” to select instances. Otherwise, designer can indicate the impossibility of showing duplicates values (DISTINCT VALUES), for retrieve only distinct values.

Figure 4 shows the Staff context where one index (Surname_char) has been defined to retrieve the department members’ information whose surnames begin with a specified character.

2. Search mechanisms, expressed as information filters, which allow constraining the objects space of the manager class to be retrieved, in function of one predefined condition over some property of the manager class or obtained from a relationship between classes. We provide searching mechanisms of three types: (a) exact filters, that return all the instances whose attribute value matches exactly with the indicated value; (b) approximate filters, that return the set of instances whose attribute value is similar to the indicated value; and (c) range filters, that return the set of instances whose attribute value fit within the range defined by two indicated values.

In the Staff context in Figure 4, an approximate filter has been defined, to retrieve the department members by surname. Due to no population condition has been defined, the application will have to ask user to submit one value for the name of attribute in run-time.
3.3 Presentational Modelling

Once the navigational model is built, we should specify presentational requirements of web applications. For this purpose, a Presentation Model is introduced. This model uses the navigational contexts as basic entities to define the presentation properties. Presentation requirements are specified by means of simple presentation patterns, which can be associated to the elements of the navigational context. The presentation properties can be applied at navigational context level, access structure or search mechanisms level, and at relationships between classes. These presentation patterns are:

1. **Information Paging**. This pattern allows capturing the semantics of logic information scrolling. When paging is specified, the set of instances that will be presented is “broken” into “logical blocks”, so that only one block is visible at a time. Mechanisms to move forward or backward are provided. Over this pattern, next properties can be defined: (1) **Access mode**, whose value can be **sequential**, providing mechanisms to go to the next, previous, first and last logical block; or **random**, allowing the user to go directly to a desired block; (2) **Cardinality**, that can be **static**, when the block always has the same number of objects; or **dynamic**, when the number of objects can be changed by the user in run-time; and (3) **Circularity**, allowing that the set of logical blocks behave as a circular buffer. This paging information can be applied to the manager class, as well as a navigational relationship or an index or a filter.

2. **Ordering**. This pattern allows defining a class population ordering, according to the value of one or more attributes. Ordering can be **Ascendant (asc)** or **Descendant (desc)** and applied to navigational classes, specifying how the retrieved instances will be ordered (attributes the search is done over have to be specified in the context), or it can be applied to access structures and search mechanisms, ordering the obtained results by some of the attributes specified by these mechanisms.

3. **Presentation of Instances (Layout)**. Four types of these patterns have been defined: **register**, **tabular**, **master-detail** (allowing to recursively indicate how to show the detail element) and **tree**. They can be applied to the navigational class and relationships.

Figure 5 shows the presentation requirements specified for the **Staff** and **Tutorships** navigational contexts. In the Staff context, register patterns have been applied to the manager class (**Department member**) as well as to the navigational relationships and the complementary classes. On the other hand, in each “screen” one logical block, always formed by four instances (**static cardinality**), will be showed. Sequential access is provided for accessing the different logical blocks, i.e., access to the first block, previous, next and last one. In the case of Tutorships navigational context, the tabular layout pattern has been applied to the manager class. The **Teacher - Subject** relationship will be shown by means of a master-detail pattern, with a **register** pattern for the Subject’s instances.

These presentation patterns, together with the specified navigation features, capture the essential requirements for the web interface development. The specified information will be used by the model compiler to generate the web pages interface.

4. IMPLEMENTING WEB APPLICATIONS FROM CONCEPTUAL MODELS

The second main step in the OOWS approach is the **solution development**. In this step, it is determined the target platform, and it is chosen a specific architectural style. Then, a set of correspondences between abstraction primitives and the elements that implement each tier of the architectural style are applied in order to automatically obtain the final system.

For implementing web applications, we use a three-tiered architecture style. Architectures of this kind allow us to clearly structure web applications, easing their adaptability and extensibility and improving scalability. The tiers of the architecture are the following:

- **Presentation Tier**. It includes the graphical user interface (in this case, Web pages and visual components) for interacting with the user (visualizing information, allowing service access and easing navigation).
- **Application Tier**. It implements the business logic (functionality) of the web application.
- **Persistence Tier**. It implements the persistence and the access to the data in order to hide the details of the data repositories to the upper tiers.

In this work, we focus on presenting the extensions that allow for the navigation and presentation requirements modelling. These requirements direct influence on the implementation of the presentation tier. Next section presents the followed strategy to implement the presentation tier from these enhanced conceptual models.

4.1 Implementing the Presentation Tier

Starting from the navigational map, it is possible to systematically obtain a web application skeleton made up of a set of interconnected web pages. These web pages represent the user interface that provides access to the relevant information and functionality for a given user, and they provide an structured (navigable) view of the system. With the presentational requirements captured in the presentation model, we can apply “layout” formatting rules. The first step of the generation process is to obtain a skeleton of the web system structure by producing a set of interconnected web pages. This skeleton structure is directly obtained from the navigational map structure (see Figure 6), where graph nodes are converted into web pages and graphs links denote navigational links between web pages. That is, each context or subsystem is converted to a web page.

The second step of the generation process is to obtain the content of each web page using the context definitions (see Figures 4 and 5). Following this basic conversion rule, the generator must assure that each page fulfills two basic requirements: (1) each page must provide the user the (relevant) specified information to perform certain activity, and (2) web pages must avoid the user to get lost by providing him
information about where the user is, where can the user go, how the user reached there, etc [15]. To better structure and clarify web page generation, we divide them into two logical areas according to these requirements:

- **The information area** presents a specific view of the system defined by a navigational context. This area is the responsible for requesting the persistency tier for the data that are specified in the navigational context definition. As these data comes from a structured model (class diagram), the generator knows how to obtain these data. The presentation model specifies how to obtain the “layout” and some specific requirements for the final interface. To obtain this area, we apply the following strategy: first, we recover the instances of the manager class specifying the population filter condition (if defined) and pagination requirements. The specified attributes and services are shown according to the layout pattern. For each navigational relationship that starts from this manager class, it is retrieved the related instances for each instance of the manager class. Context relationships also define a link, using the value of the attribute link attribute as the anchor, to the web page that implements the specified context attribute context. Recursively, it is applied the same strategy to generate the full specification of the navigational context for the rest of the complementary classes.

- **The navigational area** provides navigation meta-information to the user: where is he located?, how do he reached there? and where can he go to?. In addition, this area allocates indexes and search mechanisms.

A navigational subsystem captures a logical set of contexts and subsystems. It is represented as a navigational map inside the global navigational map. When the user accesses a navigational subsystem, its navigational map gets activated and it is shown their exploration nodes as reachable. One of its graph nodes can also be marked as default. So, the generation strategy applies the navigational map translation patterns to each subsystem.

Figure 6. Web pages skeleton structure generated for the **Staff** subsystem.

Figure 6 presents the web pages skeleton structure generated for the navigational map of the subsystem **Directory**. Four web pages are generated: **Staff**, **Collaborators**, **DSIC Place Map** (exploration web pages) and **Tutorships** (sequential web page). As none of the exploration pages are marked as default, a home page is introduced. This page provides access (links) to all exploration pages.

Then, each web page is implemented. Figure 7 shows an implementation for the **Staff** context according to its specifications in Figures 4 and 5.

The **information area** (see box number 1) is implemented by querying the system to retrieve all the specified data and operations. This context requires information about a department member (full name, photo and email), its office (code and phone extension) and a “link” to its tutorships (**Tutorships context**) that is specified by the context relationship defined in Figure 4. This information is presented (see box number 2) as it has been specified in the presentation model (figure 5):

**Pagination**

- There are shown four instances of department members (cardinality is set to 4).
- This number of instances can not be modified by the user (static cardinality).
- We can only access to the next and previous block of instances and to the first and the last blocks (sequential access) See box number 7.

**Layout**

The specified layout is applied to each element of the navigational context as follows:

- We have chosen the register layout pattern for the manager class. So, the values of the instances for the members of the department are shown in a classic “register” manner by showing the name of the attribute and its value, instance by instance (see box number 2).
- We also apply the register layout pattern to the navigational relationships **department member – office**. So, the information about the office for each department member is also shown in a register manner (see box number 2).

The **navigational area** allocates the following navigational meta-data (boxes number 3, 4, 5 and 6):

- A “navigational menu” that provide access (links) to exploration contexts or subsystems (see box number 3), and if we are inside a subsystem, also its exploration nodes are shown. The navigational menu of Figure 7 provides access to **Department**, **Directory**, **Teaching**, **Research**, **Intranet** and **Links** exploration nodes from the navigational map, and to the **Staff**, **Collaborators** and **DSIC map** from the **Directory** subsystem.
- A “navigational path” (see box number 6) specifying where the user is and how the user has reached there (a set of ordered navigational contexts and subsystems that the user have navigated).
- Search mechanisms (see box number 4). In this example, the **Surname** filter allows the user to find department members by their surname.
- Indexes (see box number 5). In this example, the **Surname_char** index has been implemented to allow the user to access to department members by their surname initial character.

If an Internet user is located in this context and he selects the “Tutorships” link for a specific teacher, the system navigates to the Tutorships web page. This page (sequential navigational context) is only reachable following this navigational path. It
shows the tutorship information for a given (the selected) teacher. Figure 8 presents an implementation for this context. As can be seen, it have been applied the same implementation strategy.

![Diagram of Tutorships context]

Figure 8. Web page of the Tutorships context.

The information area (see box number 1) provides the specified information (Figure 4) about the selected teacher (full name and photo), and the tutorships (day, start and end hour) for its subjects (name).

Then, the specified presentation patterns (see Figure 5) are applied to “formatting the layout”. First the tabular layout pattern is applied to the Teacher (manager class. See box number 2). Next, a master-detail layout pattern is applied the Subjects wrt the given teacher. These subjects are presented in a register manner (see box number 3). Also a master-detail layout pattern is applied to the Tutorships wrt the subjects and they are presented in a register manner (see box number 4).

The navigation area has the “navigational menu” and the “navigational path” (see boxes number 5 and 6). In this case, there not appear search mechanisms nor indexes because of these features have not been specified.

5. CONCLUSIONS AND FURTHER WORK

In this work, we have presented OOWS, a methodological approach to develop web applications in an OO modelling oriented software development environment. This framework integrates appropriate models to capture the structure, behaviour, navigation and presentation requirements of a web application.

New primitives (subsystem, user type change) have been introduced to the previous version of the Navigational Model.

The method has been successfully applied to a real case of study, starting from the conceptual modelling stages and ending in the final software system solution.

A CASE tool is being developed to support the full development process of web applications using our integrated model-driven approach.

Acknowledgments. This work has been partially supported by the VII. 18 – WEST CYTED project, the MCYT project with ref. TIC2001-3530-C02-01 and by the Technical University of Valencia, Spain.

References


