Chapter 2

Methodologies for Web Information Systems Design

Modern Web Information Systems (WIS) are characterized as data-intensive systems in which data integration and personalization aspects play important roles. Designing such WIS is far from trivial: the good old software engineering principles need to be adapted to the peculiarities of the Web. With respect to this researchers have proposed several WIS design methodologies among which we mention the model-driven methodologies due to the advantages they offer. Several WIS design methodologies and their accompanying tools are presented in this chapter. The emerging Semantic Web offers numerous benefits/opportunities for the WIS designers. WIS that use Semantic Web technologies are called Semantic Web Information Systems (SWIS). In this chapter, we also present some of the pioneering work in developing SWIS design methodologies. A brief comparison of the presented (S)WIS methodologies is given emphasizing for each methodology its strong and weak points.

2.1 Introduction

The Web of today has more than ten trillion pages and around one billion users. Its success lies in its very characteristics: it is unbound in space and time (it is available everyday, around the clock, and around the world), it uses the hypermedia paradigm (it provides flexible access to information according to the associative nature of the human mind [Bush, 1945]), it is distributed (it uses the popular client/server architecture with multiple clients and servers), and it is for free (there is no organization that owns the Web). If the nineteenth century was dominated by the “industrial revolution”, the beginning of this century is marked by the “information revolution” having the Web as its main engine.

A Web Information System (WIS) [Isakowitz et al., 1998] is an information system that uses the Web to present data to its users. The first generation of WIS presented the data in terms of carefully authored hypermedia documents. Typically, this involved the handcrafting of a static collection of pages and links between these pages in order to convey information to the users.

Due to its popularity there was an increasing need to make available on the Web more data sources to present up-to-date information. As such the second generation of WIS was characterized as data-intensive applications, that usually produced on-the-fly Web presentations from data stored in databases. The databases connected to the Web form the so-called "deep Web" as opposed to the "surface Web" composed of static pages. It is the "deep Web" that is the most interesting one for WIS developers as it is 500 times larger and offers much better quality than the "surface Web".

The next generation Web, the Semantic Web, is an "extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation" [Berners-Lee et al., 2001]. One can view the Semantic Web as a large decentralized global knowledge representation system. The third generation WIS are the Semantic Web Information Systems (SWIS) , i.e., WIS that make use of the Semantic Web technologies.

Some typical examples of Web Information Systems are: commerce sites, online newspapers, educational sites, Web order tracking systems, etc. All these systems share a number of characteristics. First of all, as indicated above, they are data-intensive applications. This data can come from a single source or from different sources that need to be integrated. Based on the (integrated) data a WIS automatically generates a hypermedia presentation. The one-size-fits-all approach for traditional hypermedia is not suitable for delivering information dynamically to different users with different platforms (e.g., PC, PDA, WAP phone, WebTV) and different network connections (e.g., dial-up modem, network copper cable, network fiber optic cable). The personalization component in a WIS will take care exactly of these user/platform features so that the user has a pleasant browsing experience.

The lack of rigor in developing WIS leads to serious problems (with respect to maintenance, evolution) when the complexity of these applications grows. Web Engineering, a new discipline, is responsible for proposing a systematic approach to the successful development of Web applications [Murugesan et al., 2001]. As in software engineering, Web engineering emphasizes the need to carefully design your application before implementing it. Also the existence of reusable components simplifies a lot the development of new Web applications. Differently than the classical software engineering approach, Web engineering needs to consider the peculiarities of Web applications, e.g., the navigational aspects of these application.

The design of WIS is a highly complex task that needs to consider all WIS features (e.g., data-intensive, data integration, automatic generation of the presentation, personalization). It is the consideration of these WIS characteristics at an early stage in the WIS development life-cycle, i.e., at design time, that, in our opinion, ensures the Web application success. We also believe that a methodology that clearly identifies different steps for coping with different WIS aspects will greatly reduce the WIS development effort.

Several design methodologies have been proposed to help the designer to specify WIS. A distinguished group of methodologies are the model-driven methodologies, i.e., methodologies that use models to specify the different aspects of a WIS. A model-based approach for WIS design has numerous benefits: better communication and understanding of the system among stakeholders, model reuse, improved system maintainability and evolution, possibility for checking validity and consistency between models, etc.

Figure 2.1 shows on a timeline some of the most popular (S)WIS design methodologies.

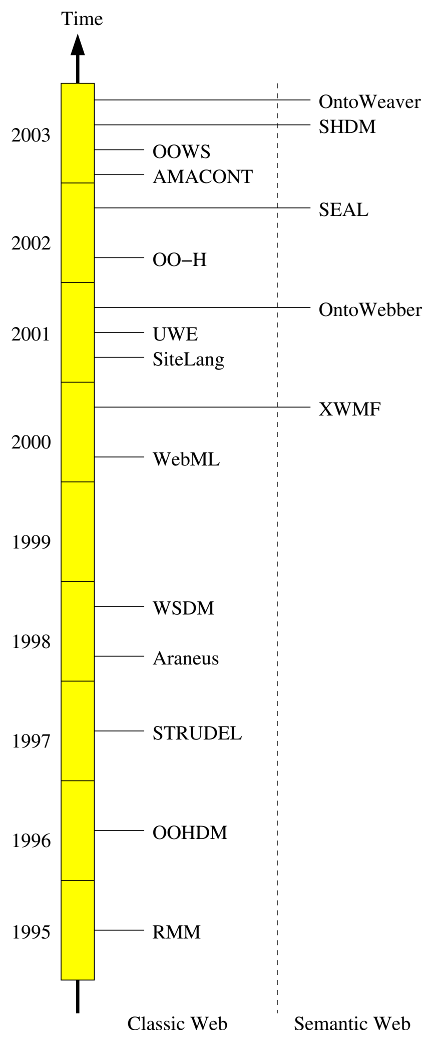

Figure 2.1: WIS methodologies.

Some of the most well-known WIS design methodologies are: RMM [Diaz et al., 1997], OOHDM [Schwabe and Rossi, 1998], STRUDEL [Fernandez et al., 2000], Araneus [Mecca et al., 1998], WSDM [De Troyer and Leune, 1998], WebML [Ceri et al., 2003], Site-Lang [Thalheim and Dusterhoft, 2001], UWE [Koch et al., 2001], OO-H [Gomez and Cachero, 2003], AMACONT [Fiala et al., 2004], and OOWS [Pastor et al., 2003].

There are few design methodologies that exploit the potential of the Semantic Web. Some of the pioneering methodologies for designing SWIS are: XWMF [Klapsing and Neumann, 2000], OntoWebber [Jin et al., 2001], SEAL [Maedche et al., 2002], SHDM [Lima and Schwabe, 2003a], and OntoWeaver [Lei et al., 2003]. Common to all these systems is the use of ontologies (as specifications of conceptualizations) [Gruber, 1993] for describing models. These ontologies are supported by inference layers that use ontology rules (axioms) to deduce new facts based on existing facts.

The main goal of using such an approach is application interoperability. By application interoperability is meant not only the WIS interoperation at data level (the output of one system is the input of another system) but also the ability to reuse existing WIS models in building new WIS. Also the model verification of such systems becomes simpler (compared with the verification of traditional WIS design models) as part of the verification is a direct application of the ontology semantics. The semi-structured aspect of Web data asks for the use of semi-structured representations (as opposed to the structured data present in classical databases) which are supported by some of the Semantic Web languages.

The rest of the chapter is structured as follows. Section 2.2 presents some of the most well-known methodologies for WIS design. Section 2.3 describes some pioneering methodologies for SWIS design. Section 2.4 gives a brief comparison of (S)WIS design methodologies emphasizing for each methodology its strong and weak points. Finally, section 2.5 concludes the chapter suggesting possible evolutions of SWIS design methodologies.

2.2 Methodologies for WIS Design

2.2.1 RMM

Methodology

The Relationship Management Methodology (RMM) [Isakowitz et al., 1995; Diaz et al., 1997] uses a "relationship management" approach for modeling WIS. By "relationship management" it is meant the management of relationships among information objects. RMM is developed from a database perspective by using the popular Entity-Relationship (E-R) diagram. At the core of the methodology there are four different activities: *E-R design*, *application design*, *user interface design*, and *construction/testing*.

The *E-R design* produces an E-R diagram in order to depict the entities and relationships relevant to a particular application domain. Each entity has attributes that describe its data characteristics. The relationships are associative relationships as they depict associations between different entities. These associative relationships have cardinality one-to-one or one-to-many. Similar to database modeling, many-to-many relationships are decomposed into two one-to-many relationships.

The *application design* produces an application model. The application model has two types of navigational elements: slices and access structures. The most significant access structures are indexes, guided tours, and links. A slice groups information into a meaningful presentation unit. The information is given by attributes of E-R entities. Slices can be aggregated in order to form higher level slices. The slices that correspond to pages are called top-level slices. The simple slices contain only one attribute. A slice can be the anchor of a link which has as destination a top-level slice. A slice is owned by an entity from the E-R model and as such can be viewed as an entity extension. Embedded slices need to specify the E-R relationship that associates the embedded slice owner with the embedder slice owner, in case that the two owners are different. For one-to-many relationships the application model designer can choose between indexes and guided tours, as corresponding access structures.

The *user interface design* describes the presentation of each application model element. This involves the layout of slices, anchors, indexes, and guided tours. RMM suggests the use of the low-fidelity (paper and pencil) and high-fidelity prototyping methods. The paper and pencil prototyping gives a high-level overview of how one views the application. The working prototype was built in order to better understand application's behavior. In an interactive manner, based on the observations made available from the running prototype, the application model might be readjusted.

The *construction/testing* phase, as in traditional software engineering projects, builds and tests the WIS based on the previous specifications. Special care needs to be considered for the testing of all navigational paths. More on the details of this phase (the WIS construction) can be found in the next subsection.

Tools

RMCase [Diaz et al., 1995] is an environment to support the development of WIS using RMM. The environment takes in consideration cognitive issues of hypermedia software development. The three cognitive requirements used in RMCase are: feedback loops across methodological phases, manipulation of design objects, and lightweight prototyping. The tool supports: bottom-up, top-down, and middle-out software development styles. RMCase has six contexts (views): *E-R design context*, *application (navigation) context*, *node-link conversion context*, *user interface context*, *hyperbase population*, and *prototyping (simulation) context*. One can recognize four of these contexts as corresponding to the four RMM phases. The designer can easily navigate from one context to another by means of the tool navigation bar.

In the *E-R design context* the user specifies the entities, attributes, and relationships of the application domain. In the *application context* one builds an application model composed of slices, links, indexes, and guided tours. In the *node-link conversion context* one can automatically convert the application model to a node-link web. In the *user interface context* the designer builds HTML templates that are associated to the previously generated nodes. Each attribute or link anchor has an associated "slot" in the HTML template. In the *hyperbase population context* one adds instances into the application database. These instances can be "pumped" into the nodes (data hard-coded in the application) or queries (SQL) can obtain data on-demand from the underlying database. The *prototyping context* enables the application designer to test the capabilities of the future Web application by means of an automatically generated prototype.

2.2.2 OOHDM

Methodology

The Object-Oriented Hypermedia Design Methodology (OOHDM) [Schwabe et al., 1996; Schwabe and Rossi, 1998] uses an object-oriented approach for modeling WIS. The object-oriented approach chosen for OOHDM is motivated by the fact that object orientation is supported by successful standards (e.g., UML) and it offers powerful object view mechanisms. At the core of the methodology there are five different activities: *requirements specification*, *conceptual design*, *navigation design*, *abstract interface design*, and *implementation*.

The *requirements specification* identifies the users of the system and the activities the users would like to perform with the system. A user can play one role or multiple roles in the system. Scenarios, provide narrative descriptions of user activities for a certain role. Related scenarios are aggregated in a use case that informally describes the user interaction with the system without considering the internal aspects of the application. A more formal description of this interaction is provided in a user interaction diagram [Guell et al., 2000]. The user interaction diagram can be used for the derivation of the models specific to the conceptual design and the navigation design of the application.

The *conceptual design* produces a conceptual schema using a notation very similar to a UML class diagram. The conceptual schema captures the domain semantics as independently as possible from the different types of users and tasks. Conceptual classes may use different relationships like inheritance, aggregation, and association. In addition to the well-known UML constructs, OOHDM proposes attribute perspectives, i.e., different media types that can characterize a certain attribute (e.g., a building description can have a text and an image associated to it).

The *navigation design* produces a navigation class schema complemented with a context diagram. The original conceptual schema needs to be reorganized such that the application fits the user needs. Different navigation models need to be built for different applications using the same domain data. A navigational class schema has three types of navigational elements: nodes, links, and access structures. Recognizing the need to group class attributes from a presentation perspective, navigation classes (also called nodes) are defined as views on the conceptual schema using an object-oriented query language. The attributes perspectives are mapped to different navigational class attributes. Similar to the class relationships between conceptual classes, links reflect navigational relationships between navigational classes to be explored by users. For each link, some of the navigation class attributes are marked as anchors. Access structures like indexes and guided tours are other possible ways to access the navigation nodes.

In order to structure the navigation space one can define navigational contexts. A *navigational context* is a set of navigational objects. There are five types of navigational contexts: simple class derived (objects of a class that satisfy a property, e.g., buildings with address Rio de Janeiro), class derived group (a set of simple class derived contexts, where the defining property of each context is parameterized, e.g., building by location), simple link derived (all objects related to a given object, e.g., buildings designed by Oscar Niemeyer), link derived group (a set of link derived contexts, where the the source of the link is parameterized, e.g., buildings designed by architect), and enumerated (set of elements explicitly enumerated). Associated to contexts there are access structures like indices. A navigational context contains the specifications of its elements, an entry point, and an associated access structure. If the elements of the context depend on the user browsing behavior, the context is said to be dynamic (e.g., history and shopping basket). "InContext" classes extend certain nodes with attributes when these classes are navigated in a particular context.

The *abstract interface design* produces abstract data views (ADV) and abstract data view charts (ADV-charts). ADVs are abstract in the sense that they represent the interface and the state, and not the implementation. ADVs define the interface appearance of navigational classes and access structures, and other useful interface objects (e.g., menus, buttons, etc.). ADVs capture the statical part of the interface: the perception properties and the interface's events. The dynamical part of the interface is given by ADV-charts, a derivative of the UML state charts that specify the system's reaction to external events.

Among the object-oriented design patterns used in OOHDM we distinguish the observer pattern for the navigational classes and ADVs, and the decorator pattern for the "InContext" classes. From the UML notation we recognize the class diagrams for conceptual schemas and the state diagrams for the ADV-charts. Aggregation and inheritance are used in both conceptual schemas and ADVs.

The *implementation* phase produces a WIS based on the previous OOHDM specifications. The designer has to decide on how to store the conceptual and navigational objects. Also he needs to decide on how to realize the static and dynamic aspects of the interface using HTML and some extensions. More on the details of this phase can be found in the next subsection.

Tools

OOHDM-Web [Schwabe et al., 1999] is an environment to support the development of WIS using OOHDM. It is based on the scripting language Lua [Ierusalimschy et al., 1996] and the CGILua environment [Hester et al., 1997]. Navigational classes and contexts are stored in relational databases. ADVs and ADV-charts are implemented by a combination of HTML templates (ADV structure) and OOHDM-Web library function calls (ADV behavior). OOHDM-Web has three interfaces: the *authoring environment*, the *browsing environment*, and the *maintenance environment*.

In the *authoring environment* the designer specifies the navigation schema generating database definitions. OOHDM assumes that the navigation schema is given (no need to translate it from a conceptual schema) and stores it in a relational database. Each navigational class and link are implemented as tables. For navigational classes each column stores an attribute and each arrow corresponds to an object of that class. All class tables have an attribute "key" defined. The link tables define relations between the corresponding object keys. Attribute perspectives are stored in a separate table. There is also a table to store all context names and their types. Each context type has a corresponding table that stores all context of this type. These tables refer to HTML templates. OOHDM also provides functions to manipulate the state of different contexts.

In the *browsing environment* the designer specifies HTML templates according the corresponding ADV specifications. An HTML template is combined with OOHDM-Web function calls to return dynamically computed data. Examples of OOHDM-Web functions are “Index”, “Attrib”, “PrevLink”, “NextLink”, etc. The different kind of events can be handled by inserting scripting code in the HTML page.

In the *maintenance environment* the designer specifies interfaces to allow the insertion/change of the instance data (nodes and contexts). Also by using summary screens the environment allows one to check the design. For example one can see which are the previously stored navigational classes and contexts.

OOHDM-Web is also supported by a CASE environment [Lyardet and Rossi, 1996] that helps the designer to describe the conceptual, navigational, and interface models of its application using the OOHDM notation. Based on the chosen run-time environment the tool is able to generate appropriate implementations.

OOHDM-Java2 [Jacyntho et al., 2002] introduces a business model as a generalization of the conceptual model and the application’s transactional behavior. In response to a user request the business model is possibly updated based on the application’s business rules (stored in the same business model). The navigational nodes and contexts are created based on data stored in the business model. The requested page template is then populated with the data coming from navigational nodes and contexts. In this implementation OOHDM models are stored in XML and the page templates are defined in JSP. The implementation environment was Java2EE (Java 2 Enterprise Edition), a popular platform for implementing robust distributed multi-tier architectures.

2.2.3 WSDM

Methodology

The Web Site Design Method (WSDM) is an audience-driven design methodology for building WIS [De Troyer and Leune, 1998]. An audience driven philosophy takes as a starting point the explicit modeling of different target users, i.e., audiences and derives from it the system’s navigation structure. WSDM consists out of five phases: *mission statement*, *audience modeling*, *conceptual design*, *implementation design*, and *implementation*. Recently, WSDM has been extended to support localization [De Troyer and Casteleyn, 2004] and adaptive behavior [Casteleyn et al., 2003, 2004].

The *mission statement* expresses the purpose, the subject, and the intended audiences for the WIS. The intention of this phase is to make clear from the very beginning what is the goal of the WIS, what is the subject that is involved in realizing the goal, and that no intended visitors are forgotten.

During *audience modeling*, the mission statement is taken as a starting point to classify the different audiences into a hierarchy, based on their information and functional requirements. Each group of visitors with similar functional and informational requirements form an audience class. Visitors with extra requirements are subclasses, and appear under their parent audience class in the audience class hierarchy. The top of the hierarchy is called the visitor audience class: it groups the most general requirements that all visitors to the website share. Furthermore, for each audience class, their specific characteristics, usability and navigation requirements are specified. These are later taken into account when deciding how to present information to these particular visitors.

The *conceptual design* phase consists out of two important sub-phases: *task and data modeling* and *navigation design*.

During *task and data modeling*, for each requirement, a task model is specified. This task model decomposes each high-level requirement specified during audience modeling into elementary requirements. WSDM uses CTT (Concurrent Task Trees) [Paterno, 2000] to specify the task models, which allows next to standard task decomposition, also the specification of temporal relations between the different subtasks. Then, for each elementary task derived in the task models, a tiny data model, called an object chunk, is specified. Such an object chunk formally describes the information and/or functionality needed to fulfill the elementary requirement it covers. Currently, WSDM uses ORM (Object Role Modeling) [Halpin, 1995] (with some extensions for functionality) to express object chunks, but a shift to Web Ontology Language [Bechhofer et al., 2004] is being done at the time of writing.

In the next sub-phase, the *navigation design*, the conceptual navigation structure for the website is defined. This is done by constructing a graph of nodes with links between them, and connecting the object chunks to the nodes. A node is thus a conceptual navigation entity, containing information/functionality (i.e., object chunks) that logically belongs together. The basic navigation structure is derived from the audience class hierarchy, subdividing the global site navigation space into different "sub-sites", one for each audience class. Each so-called audience track contains all but only the information/functionality required for that particular audience class. Using the task models, and more particularly the temporal relations specified between elementary tasks, each audience track is further refined.

The *implementation design* phase consists out of three sub-phases: *page design*, *presentation design*, and *data design*.

During *page design*, the designer decides which conceptual navigation nodes from the navigation design will be grouped into one page. To do so, the characteristics of the audience classes are taken into account. Possibly, different site structures can be defined, for example by targeting different browsing devices.

The *presentation design* defines the look-an-feel of the application. This is done again by taking in consideration the different requirements of the audience classes. With this respect for each page a template depicting the page layout (i.e., positioning of nodes and chunks) is defined. In addition it is specified the style (e.g., font type, size, color, etc.) for all pages.

In the *data design*, based on the conceptual schema defined by the object chunks, the schema of the underlying database is specified.

Finally, taking as an input the object chunks, the navigation design, and the implementation design, the actual *implementation* can be generated in the chosen implementation language. Both static and dynamic sites are supported. The transformation is performed fully automatically.

Tools

The Audience Modeler is a CASE tool that supports the audience modeling phase of WSDM. It allows the designer to graphically describe the different audience classes and their requirements. Furthermore, the tool also support the audience class matrix algorithm [Casteleyn and De Troyer, 2001], which automatically constructs the audience class hierarchy based on a simple series of yes/no questions. Naturally, a combined approach, generating the audience class hierarchy and afterward manually adjusting it, is also supported. This tool has been implemented using Java and Wx Windows (now renamed to wxWidgets [Anthemion Software, 2004]) for the graphical parts.

The Chunk Modeler is a CASE tool that supports the data (i.e., information and functional) modeling phase (part of the conceptual design) of WSDM. It allows the designer to graphically model object chunks using ORM. The extensions to ORM needed to model functionality are also supported. This tool has been implemented using C++ and Wx Windows (now renamed to wxWidgets) for the graphical parts.

WSDM has also a prototype of the implementation generation phase. This prototype takes as inputs (in XML format) the object chunks, the navigation design, and the implementation design of a WSDM design, and outputs an actual implementation. The implementation is done using XSLT [Kay, 2005] transformations.

2.2.4 WebML

Methodology

WebML (Web Modeling Language) [Ceri et al., 2000, 2003] is a high-level modeling language for the specification of WIS. It uses conceptual modeling techniques for describing hypertext. WebML concepts have visual representations and are serialized in XML. The WebML design methodology comprises three main phases: *data design*, *hypertext design*, and *implementation*.

The *data design* produces the data schema composed from several sub-schemas: core, access, inter-connection, and personalization schema. In the same way as for RMM the data schema is an E-R diagram. The core sub-schema identifies the main entities in the application domain. The access sub-schema enriches the core sub-schema with access entities. In this phase one can define the categories and localizations of the previously identified entities. In the inter-connection schema the core entities can be connected using relationships. The personalization schema adds to the data schema, entities depicting the user and its preferences. In this phase, for example, the user will be associated to its preferred language.

The *hypertext design* defines the navigational structure of the application. The result of this phase is the hypertext model. The hypertext model is based on the concepts of units and links. WebML uses a hierarchical organization of data in units, pages, areas, and site views. The most primitive element is the unit, an atomic piece of information. Units are grouped into pages. A page provides information to be presented at once to the user in order to achieve a certain communication purpose. Pages that deal with a homogeneous subject are grouped in areas. At the top of the hierarchy there is the site view which can contain areas and pages. A site view defines all the information accessible in a WIS by a certain user.

There are five types of basic units: data units, multidata units, index units, scroller units, and data entry units. Data units display information about one object. Multidata and index units show information about a set of objects. Differently than the multidata unit, the index units allow the selection of one object out of a set. Scroller units support the scrolling (moving backward/forward) through a set of objects. Data entry units enable the user to insert new data into the system. The first four types of units have associated sources and selectors. Sources are used to identify the entity (type) providing the unit data. Selectors define define predicates to select the object(s) (of source type) to be presented by the unit.

Between units/pages one can define links which are directed connections. WebML distinguishes several types of links: navigational, automatic, and transport links. These links can carry information from the source to the destination. The information is stored in the link parameters. The link parameters are used in the unit selectors (selectors with link parameters are called parametric selectors). The navigational links require user intervention. Both automatic and transport links are traversed without user intervention. For automatic links once the source is presented also the associated destination is showed. The transport links do not define navigation and are solely used to transport information. Besides the link parameters, which use a point to point communication, one can define also global parameters. A global parameter is small piece of information extracted during user navigation which will be made available to all units in the system. The global parameters are accessed/modified using the get/set units.

In addition to the previous units, WebML defines operation units used to invoke different operations. These operations can be for example associated to data entry units in order to process information entered by a user. There are some predefined units to model the content management operations like create, delete, and modify units (for entities), and connect and delete units (for relationships). Also one can use operation units in order to call in a synchronous or asynchronous manner Web services. The hypertext model can be organized, if necessary, in a workflow-driven hypertext [Brambilla et al., 2002]. With this respect WebML defines a workflow data model (composed of processes, activities, etc.), workflow-specific operations (like start activity, end activity), and workflow-specific units (like switch unit).

The *implementation* comprises two phases: the data implementation and hypertext implementation. In the data implementation phase the data schema is mapped to data sources using standard database techniques. In the hypertext implementation phase WebML pages are mapped to JSP page templates. Such a JSP page contains a query to retrieve the relevant data and a layout template used to present this data. For contextual links besides the fixed part of the URL one needs to provide also the parameters associated to the link. The operations used in the hypertext model are implemented also as JSP templates. These JSP templates will not present information but will have only a side-effect to implement a particular operation.

Tools

WebRatio [Ceri et al., 2003] is an environment to support the development of WIS using WebML. It is one of the most comprehensive CASE tools for WIS development that we have encountered so far. WebRatio has three interfaces that help to graphically define the application models: *data and hypertext design*, *data mapping*, and *presentation design*.

The *data and hypertext design interface* allows the construction of both the E-R model and the hypertext model of the application. This interface has two work areas: one for the E-R model and one for the hypertext model. The *data mapping interface* assists the designer in connecting entities and relationships to tables. The *presentation design interface* is used to define XSL stylesheets. In order to support style reuse (among units/pages) these stylesheets refer to unit placeholders instead of the actual units.

After defining all the above models one can trigger the *code generator* to produce implementations for different target platforms. In case that HTML is the language of the target platform, the output of the code generator will be a set of JSP page templates. In addition to the JSP template pages, the code generator will produce the operation actions (Java), and a set of XML descriptors. These XML descriptors are used to specify the properties of units, pages, and links. For example a unit descriptor will specify the query, input and output parameters associated with a unit.

2.2.5 SiteLang

Methodology

SiteLang (Site Development Language) [Thalheim and Dusterhoft, 2001; Schewe and Thalheim, 2004] is an abstract language with a strong theoretical foundation for the specification of WIS. The language is based on the integration of three approaches: *extended E-R (Entity-Relationship) modeling* [Thalheim, 2000], *theory of media objects* [Schewe and Thalheim, 2001], and *website storyboarding* [Thalheim and Dusterhoft, 2001]. SiteLang proposes a codesign methodology for the integrated specification of structuring, functionality, distribution, and interactivity for WIS.

The *extended E-R model*, called High-Order Entity-Relationship Model (HERM) [Thalheim, 2000] adds to the classical E-R model, behavior specification, i.e., generic operations to support the update/retrieval of data. Also the database types are extended hierarchically such that one can build higher-order types (a type of level k is a set of types of level $k-1$).

Storyboarding is the activity of defining the application story. A WIS can be described as an edge-labeled directed multi-graph in which nodes are scenes and edges are transitions between scenes. Each transition has a label which denotes a certain user action. If the label is absent the transition is based on simple link following. Actions have associated a triggering event, a precondition, and a postcondition.

A scene is a conceptual location at which dialogues (between the user and the WIS) occur. To each scene there are associated: a dialogue step expression, a media object, a set of actors involved in it, a representation, and a context. The dialogue step expression is defined by means of a dialogue step algebra, recently replaced by a story algebra. A media object is an instance of a *media type*, i.e., a view of some database and a defining query. The classical database view is extended to support the notion of a link. With this respect the query is able to create object identifiers that links can reference. Additionally a media objects has associated dynamic operations and adaptivity specifications for a controlled form of information loss. The adaptivity with respect to the device is stored in the representation associated to the media object (e.g., delete images for devices that are not capable to display images). The context specifies the access channel (e.g., high-speed, low-speed), device (e.g., PC, WAP phone), and browsing history of the user.

The storyboarding language is a story algebra [Schewe and Thalheim, 2004]. To each action is associated a scene. Based on actions and scenes one can define inductively processes which are the arguments of the story algebra. Some of the algebra operators are: sequence, skip, parallel, choice, iteration, etc. Processes can have preguards and postguards associated to them. It has been showed that the story algebra is in fact a Many-sorted Kleene Algebra with Tests (MKAT) where the sorts are the scenes (bundles of actions at a certain WIS location) and the tests are the guards associated to processes. In this way a site can be expressed as a MKAT expression. MKATs can be used to formalize the personalization and information needs of the user by means of equations. These equations will result in a simplification of the original story space.

Tools

The Storyboard Editor [Thalheim et al., 2004] is an environment to support the development of WIS using SiteLang. It has several interfaces, each covering a certain WIS design aspect as specified by the SiteLang methodology. This editor is backed by a database that stores the WIS structure and functionality. In addition the tool also supports the automatic generation of the WIS each time the content, structure, and functionality of the system are changed. The WIS specifications expressed in SiteLang are stored as XML documents. The graphical representations built using the Storyboard editor can be automatically serialized in XML.

2.3 Methodologies for SWIS Design

2.3.1 XWMF

Methodology

The eXtensible Web Modeling Framework (XWMF) [Klapsing and Neumann, 2000; Klapsing et al., 2001] is a modeling framework for designing SWIS using RDF. The core of the framework is the *Web Object Composition Model* (WOCM), a formal object based language used to define the structure and content of a Web application.

WOCM is a directed acyclic graph with complexons as nodes and simplexons as leaves. Complexons define the application's structure while simplexons define the application's content. Components are a special kind of complexons representing a physical entity (e.g., a Web page). The representation of an WOCM is done in RDF(S).

By means of RDFS inheritance mechanism one can define different views on the simplexon for different browsing devices (e.g., HTML or WML). Such an implementation (of the view) uses variables to refer to the concrete instances having the same type as the original simplexon. Using RDFS multiple resource classification mechanism, a simplexon instance can be an instance of more than one simplexon class. In this way the same object can have different implementations for different platforms. These implementations share the same object, a property which fosters object reuse. Complexons are defined for a specific view (e.g., HTML or WML) in case that the included simplexons are instances of more than one class.

The RDF extensibility feature allows the integration of different schemas in the same WOCM. For example, in a content management system the data will be annotated with the property "expires" in order to determine if a certain piece of information became obsolete.

Tools

XWMF is supported by a tool suite in order to create, process, and analyze its models. All tools are written in Extended Object Tcl (XOTcl) [Neumann and Nusser, 1993] and in Prolog. As WOCM has an RDF representation a number of tools have been developed to facilitate RDF editing and processing.

The RDF parser was implemented using TclXML parser. RDF Handle provides an interface to query RDF models. Gramtor is a graphical RDF editor able to work with both RDF/XML and RDF triple notation. The WebObjectComposer is able to store WCOMs (in RDF) as XOTcl classes and objects, and to generate a corresponding Web implementation.

The graphical user interfaces were done using the Motif version of Wafe [Neumann and Zdun, 2000], a Tcl interface for XToolkit. For exploring RDFS representations an RDFS parser on top of SWI-Prolog RDF parser [Wielemaker, 2000] was built. In addition to the RDF rule set, one can define new rules to capture the semantics of user-specific predicates.

2.3.2 OntoWebber

Methodology

OntoWebber [Jin et al., 2001] is an ontology-driven design methodology for building SWIS. Here, by ontology, is meant a set of terms (real-world or abstract objects) and their relationships (with semantic significance). The system's architecture is composed of three layers: *integration layer*, *composition layer*, and *generation layer*.

In the *integration layer*, first the syntactic differences between the different data sources are resolved. As a semi-structured data format for data representation was chosen RDF. In the second phase, the semantic differences between the different data sources are resolved. With this respect a reference ontology (domain ontology) is built for a specific domain. The articulation ontology bridges the semantical gap by mapping the concepts and relationships between data sources and the reference ontology.

The *composition layer* uses four ontologies that captures different aspects involved in designing a WIS: the *navigation ontology*, the *content ontology*, the *presentation ontology*, the *personalization ontology*, and the *maintenance ontology*. For a WIS, each ontology will be instantiated with a corresponding site model. All site models are integrated in one graph called the site-view.

The *navigation model* defines the basic elements of the site-view graph: cards, pages, and links. A card is a minimal unit of information. Pages contain one or more cards and correspond to the Web pages. Links connect cards in order to define the WIS navigational structure. Cards are classified as dynamic (depend on the source data change) or static (do not depend on source data change). There are four types of dynamic cards: fact cards (one instance), list cards (index of instances), slide cards (guided-tour of instances), and query cards (input properties).

The *content model* defines the data that will populate the navigation model. For static cards the static elements (types text, image, or anchor) are defined. For dynamic cards an entity from the domain model is specified. Also one needs to specify the entity properties that will be presented for these cards. There are two types of links: foreign (link to an external Web page) and native (link to a internal page). Native links are further classified as: static (no information flow) or dynamic (with information flow). Dynamic links have three properties associated: a query property, which produces the content of the destination card, a binding variables property, which stores values of the source entity, and an initiating property which defines the anchor in the source card. The queries are expressed in the TRIPLE [Sintek and Decker, 2002] RDF query language.

The *presentation model* specifies the look-and-feel of the application. Both cards and pages have associated style elements (font, color, etc.). Card style overrides the style of its embedder (a page). There are three type of layouts: flow layout (one row), grid layout (a table), and frame layout (composed of one static frame and one dynamic frame).

The *personalization model* supports both fine-grained (user) and coarse-grained (group) adaptation. For a specific user/group a site view and user model are defined. The user model contains the capacity property (e.g., name, age, gender, etc), interest property (the navigation, content, and presentation models), and request property (triggers, e.g., site view update, that will be fired if some conditions are fulfilled).

The *maintenance model* focuses mainly about content maintenance (maintenance of the functionality is not considered here). It defines an administrator user and the maintenance rules (triggers) associated with him. The administrator can update the source data and the site view specifications (note that among these models is the personalization model which he can rewrite for some users).

The *generation layer* has two phases: the constraint verification phase and the site view instantiation phase. In the first phase the constraints imposed to the WIS are checked. As ontologies are defined in RDFS some of the constraints are automatically verified by directly applying the RDFS semantics. One can formulate additional constraints like structural constraints (e.g., every dynamic card has an incoming link), semantic constraints (e.g., the query card should have the same entity associated with it as the destination card), presentation constraints (e.g., suppress images for small devices). All these constraints are expressed as TRIPLE rules. Based on the data sources and site view specifications a site view instantiation is generated in the second phase of this layer.

Tools

OntoWebber is supported by an integrated development environment (IDE) in order to develop WIS [Jin et al., 2002]. The main components of the environment are: Ontology Builder, Site Builder, Site Generator, and Personalization Manager.

The Ontology Builder assists the WIS designer in developing the domain ontology. The Site Builder is used to create the site view (graph) which is exported in three models: navigation, content, and presentation model. Additionally the Site Builder is used to define rules for checking integrity constraints on the site view. The verification of these rules is done in the same tool component. The Site Generator instantiates the site view with data. The Personalization Manager defines model-rewrite rules for the site views in order to present personalized information to its users.

2.3.3 SEAL

Methodology

The Semantic PortAL (SEAL) [Maedche et al., 2002, 2003] is a domain ontology-driven design methodology for building WIS that represent Web portals. By Web portal is meant a WIS which has a large collection of information related to specific topics and often organized in a hierarchical manner. SEAL proposes a number of steps for building a Web portal: *ontology design*, *data integration*, *site design*, and *implementation*.

In the *ontology design* one creates the domain ontology in RDFS and refines it using F-Logic [Kifer et al., 1995] axioms.

The next step, the *data integration*, lifts all the data sources to a common data model, RDF. Several wrappers have been developed, e.g., for HTML, XML, and relational databases. Of course the RDF data present on the Web is ready to be used (no need of wrapping). In addition SEAL can use data coming from an Edutella Peer-2-Peer (P2P) network. Edutella provides in RDF the metadata infrastructure of P2P networks. For the integration SEAL uses the warehouse approach to combine information coming from different sources.

In the *site design* the *navigation model*, *input model*, and *personalization model* are built. The *navigation model* defines the navigational structure over the warehouse. It is generated by combined queries for schema (ontology) and content. First the users are offered a view on the ontology by using different types of hierarchies (e.g., isA, partOf). Second for each shown ontology part the corresponding content is presented. The *input model* is used for knowledge acquisition by defining forms from the ontology. These forms have associated queries that will update the warehouse with user entered data. In the *personalization model* both navigation and input model are tailored for a specific user.

The last step is the *implementation* of the WIS using the above models. For the presentation of different WIS pages specific templates are defined. More on the details of this phase can be found in the next subsection.

Tools

SEAL is supported by a number of tools included in the KArlsruhe ONtology and Semantic Web (KAON) [AIFB, University of Karlsruhe, 2004] tool suite, an ontology management infrastructure.

OntoEdit [Storey et al., 2002] is a tool used for building the domain ontology. The metadata (RDF) available on the Web can be collected by the KAON Syndicator. KAON Reverse is a visual tool to map the logical schema of relational databases to the domain ontology.

The KAON Portal Maker produces a WIS implementation based on the different SEAL models. It uses the model-view-controller design pattern. In this pattern the models are the SEAL specification models, the view is defined by the presentation template, and the default controller provides standard application logic (update data, generate links to the next objects to be presented). The default controller can be replaced with a custom-made controller.

2.3.4 SHDM

Methodology

The Semantic Hypermedia Design Method (SHDM) [Lima and Schwabe, 2003a,b] is an ontology-based SWIS design methodology. It extends the power of expression of OOHDM (see subsection 2.2.2 for a brief OOHDM description) by defining ontologies for each of the OOHDM models. These ontologies are specified in OWL [Bechhofer et al., 2004], a more expressive language than RDFS. In the same way as OOHDM, SHDM identifies four different phases: *conceptual design*, *navigation design*, *abstract interface design*, and *implementation.*

The *conceptual design* builds the conceptual class schema for the application domain. This schema is described in UML extended with a few new characteristics like the ability to specialize relations. The UML diagram is mapped to an OWL model according to some heuristics rules. In addition to the previously defined OWL classes one can define new classes by using boolean expressions specifying necessary and sufficient conditions for class membership. These last type of classes are called inferred classes and are represented graphically by UML stereotypes.

The *navigation design* defines the navigational class schema and the navigational context schema. The main navigational primitives are navigational classes (nodes), navigational contexts, and access structures. In the same way as for the conceptual class schema, one can specialize navigational relations. The mappings between the conceptual schema and navigational class schema are defined using RQL [Karvounarakis et al., 2002]. The navigation context allows the description of sets of navigational objects. The new definition for concepts is more expressive than the one from OOHDM. For example one can create groups of contexts by using the subclassing mechanism. It is the user who will decide which particular specializations he wants to see. Context have associated with them access structures (e.g., indexes). SHDM introduces the powerful concept of facet (i.e., category) access structure that simplifies a lot the description of navigational context schema. This will represent any combination of the classes and their subclasses for navigation with the restriction given by explicitly specified invalid facet combinations.

The *abstract interface design* defines the abstract widget ontology and concrete widget ontology [Moura and Schwabe, 2004; Schwabe et al., 2004]. The abstract widget ontology defines the following widgets: EventActivator (reacts to external events, e.g., link or button), ElementExhibitor (presents some content type, e.g., image), VariableCapturer (receives the value of some variables, e.g., input fields), and any composition of the above. Abstract interface widgets must be mapped on concrete interface widgets in order to appear on the interface (e.g., an EventActivator can be mapped to a Link). Abstract interface widgets must also be mapped to specific navigation elements (e.g., an ElementExhibitor is associated to a certain navigational class attribute).

The *implementation* phase produces a SWIS based on the previous SHDM specifications. In this phase a converter from the UML representation to the OWL representation of the models is needed. More on the details of this phase can be found in the next subsection.

Tools

SHDM is supported by several tools in building a WIS: an SHDM2OWL mapping tool, an ontology editor, the Sesame storage, inference and query environment [Aduna, BV, 2005] and a presentation builder.

The SHDM2OWL mapping tool [Lima and Schwabe, 2003b] is used to convert the SHDM class schema, SHDM navigational class schema, and navigational context schema into OWL representations. The OWL representations are subsequently stored in a Sesame repository. Similar to OOHDM, SHDM defined templates for pages and the styles associated to pages. The OOHDM queries have been replaced by RQL queries.

The ontology editor [Lima and Schwabe, 2003b] allows the designer to directly build one of the SHDM ontologies in OWL or to visualize the ontologies produced by the SHDM2OWL mapping tool. One can also use as an ontology editor the Protege [Noy et al., 2001] environment.

The presentation builder [Schwabe et al., 2004] has an architecture composed of the following components: Request Handler, Template Engine, View Manager, Navigation Manager, Data Manager, Template Engine, and Output Postprocessor. The Request Handler gets the user request specifying the view name with possibly some navigational parameters. The Request Manager communicates with the View Manager, Navigation Manager, and Data Manager (in this order) to get to the right data associated to a user requested view. The page template associated to this view is given by the Template Engine that is responsible for producing the final page. Optionally the Template Engine can call the Output Postprocessor to convert the produced page (e.g., in XML) to a Web browsable format (e.g., HTML).

2.4 Discussion

Model-driven methodologies have proven to fulfill the practical needs that the WIS designer experiences. For example WebML was successfully used for projects inside Microsoft, Cisco Systems, TXT e-solutions, and Acer Europe, and SiteLang was used for several city information, learning, e-government, and community sites in Germany. Being at their infancy SWIS design methodologies were merely used in research. For example SEAL was used to develop the institute portal of AIFB, University of Karlsruhe and OntoWebber was exploited for realizing the Semantic Web community portal.

In the rest of this section we use several comparison criteria in order to stress the strong and weak points of some of the most representatives WIS and SWIS design methodologies. These comparison criteria are:

- *methodology*: does the methodology provide design steps and guidelines for each step in order to produce models?
- *tools*: is the methodology supported by CASE tools?
- *automation*: is there support to automatically build Web presentations based on previously specified models?
- *data integration*: does the methodology consider the integration of data coming from different heterogeneous sources?
- *personalization*: does the methodology support adaptation mechanisms in order to realize the application personalization?

- *user interaction*: does the methodology support complex forms of user interaction (e.g., by means of forms) with the system?
- *task model*: does the methodology explicitly model the tasks of the user in a separate task model?
- *presentation model*: does the methodology explicitly model the presentation aspects (the look-and-feel aspects, separate from navigation) of the application?
- *verification*: can the methodology models be easily verified for their validity and consistency among each other?
- *reuse*: does the methodology support the design of reusable components?

Most of the (S)WIS design methodologies identify two models: the domain model, which describes the application domain, and the navigation model, which depicts the navigation (linking) aspects through the data.

Table 2.1 shows an overview of the characteristics of some WIS design methodologies with respect to the previously selected criteria.

	RMM	OOHDM	WSDM	WebML	SiteLang
Methodology	Yes	Yes	Yes	Yes	Yes
Tools	Yes	Yes	Partial	Yes	Yes
Automation	Partial	Yes	Partial	Yes	Yes
Data integration	No	No	No	No	No
Task model	No	Partial	Yes	Partial	Yes
Personalization	Partial	Yes	Yes	Yes	Yes
User interaction	No	Partial	No	Yes	Yes
Presentation model	No	Yes	No	No	No
Verification	No	No	No	No	Yes
Reuse	Yes	Yes	Yes	Yes	Partial

Table 2.1: WIS methodologies comparison.

The analyzed WIS design methodologies have a well-structured methodology and good tool support. One of the tools is the code generator that builds in an automatic way a Web presentation based on previously specified models. RMM and WSDM are examples of methodologies that have tools that only partially support this automatic process.

The WIS requirements describing what the user actually wants to do with such a WIS are very often neglected by WIS design methodologies. A notable exception is WSDM that models the audience that the WIS targets. Its characteristic feature is that it proposes a task model associated with a certain audience. The task model is subsequently used to derive the navigation model in which a navigation track corresponds to a certain audience. Similar to the task models are the storyboards used in SiteLang. An alternative approach is proposed by OOHDM which models the interaction between the user and the system in a user interaction diagram. The WSDM task model and the SiteLang storyboard are more expressive than the user interaction diagram as they have complex task operators (like concurrency, choice, iteration, etc.) which are not available in user interaction diagrams.

For all the examined WIS design methodologies the data integration issue was ignored. This is mainly due to the lack of representation languages on the classic Web to express data semantics. It is the Semantic Web with its support for expressing data semantics that fosters application interoperability. Knowing the data semantics one can easily integrate data coming from different sources.

In order to personalize a WIS, the designer identifies user profiles, which stores characteristics of the user and his browsing platform. These user profiles can be aggregated in group profiles which cluster the users with the same requirements. To a user profile or group profile one can attach specific views. The conceptual model is augmented with user-specific entities and relationships that can refer back to the application domain model. In WebML these extensions form the so-called personalization sub-schema. In addition to the above personalization techniques, OOHDM proposes the personalization of the entities (attribute content of an entity) in the conceptual model and of the layouts in the interface model (based on user preferences or selected devices).

Another feature neglected by the examined methodologies is the design support offered for the presentation aspects (the look-and-feel aspects) of WIS. Most of the methodologies refer to templates (for example XSL templates) that describe the styling information of the systems. An exception is OOHDM which has an explicit presentation model. This model does not only depict the static presentation characteristics of the system (e.g., layout) but also the dynamic aspects of the system, i.e., what will be the system reaction to external events.

Modern WIS allow the user to interact with the system in order to let him influence the next page to be generated in the hyperspace. These systems need to make available complex user interaction (e.g., by means of forms). WebML supports the modeling of user input and its processing using data entry units and processing units, respectively. SiteLang defines activities for gathering user input and variables to store the data input by the system. These variables can be used by processing activities that will perform computations based on user input. The results can be made available for display in the scenes associated to the processing activities.

Verification is yet another aspect that was ignored by most of the WIS design methodologies that we analyzed. An exception is SiteLang, a WIS design methodology with strong theoretical foundations. Having strong theoretical grounds SiteLang produces very concise WIS representations by means of formulas. This formulas can be easily verified for well-formedness. Moreover a SiteLang formula can be minimized (optimized) by considering the equations that model certain user characteristics (like preferences or information needs). In this way one can considerably reduce the complexity of a WIS specification.

All examined methodologies provide primitives that can be reused in the model specification. Models can be refined by means of inheritance, enabling thus the reuse of existing specifications. An object-oriented approach like the one used in OOHDM or the extended

E-R Modeling from SiteLang fosters also the reuse of the behavior specifications. In addition, object-oriented approaches benefit from the reuse of design patterns (e.g., observer pattern, decorator pattern). The OOHDM team is also actively involved in defining navigational patterns to support coarse-grained reuse in navigational models. With respect to expressivity and fine-grained reuse in navigation specifications we found that WebML has one of the most extended set of navigational primitives that satisfy most of the WIS designer needs.

Table 2.4 shows an overview of the characteristics of some SWIS design methodologies with respect to the previously selected criteria.

	XWMF	OntoWebber	SEAL	SHDM
Methodology	Partial	Yes	Partial	Yes
Tools	Yes	Yes	Yes	Yes
Automation	Yes	Yes	Yes	Yes
Task model	No	No	No	Partial
Data integration	No	Yes	Yes	No
Personalization	No	Partial	No	Yes
User interaction	No	No	Partial	Partial
Presentation model	No	Partial	No	Yes
Verification	Partial	Yes	Partial	Partial
Reuse	Yes	Yes	Yes	Yes

Table 2.2: SWIS methodologies comparison.

Less mature than the WIS design methodologies, many SWIS design methodologies focus less on the steps needed to build SWIS. These methodologies emphasize how ontologies can be used for their model representations and their support tools. The representation language ranges from RDF (in XWMF) to OWL (in SHDM). Having such a standardized mean to express application semantics greatly improves system's interoperability. SWIS design methodologies (e.g., SHDM) that extend an existing WIS methodology (e.g., OOHDM) benefit from the reuse of the methodological steps and are better structured than the rest of the examined SWIS design methodologies.

Most of the examined SWIS design methodologies have good tool support. Tools for automatic generation of WIS based on previously defined models do also exist. Being at an early development stage most of these methodologies do not provide an integrated development environment like the ones we found for WIS design methodologies (e.g., RMCase, OOHDM-Web, WebRatio).

The SWIS specifications describing what the user tasks in a SWIS are neglected by most of the examined SWIS design methodologies. SHDM is the only SWIS design methodology that addresses this issue by means of the user interaction diagram that it inherits from OOHDM. As SWIS design methodologies will become more mature and their applicability will go beyond research laboratories we hope that more attention will be given to the specification of user tasks.

Data integration is a topic of special interest for SWIS design methodologies. Having the necessary technologies to describe the data semantics facilitates the integration of data coming from different sources. Common to all these methodologies is the wrapping of the data sources in a semantic representation. These semantic representations are mapped to a common (application) data representation. The only methodology that doesn't consider data integration is XWMF, as the authors focus on the presentation aspects of a SWIS.

Most of SWIS design methodologies have defined models and ontologies to describe the model semantics. Seen as an advanced feature few of these methodologies provide model "hot-spots" to support system's personalization. A notable exception is SHDM which reuses the adaptation mechanisms from OOHDM. An interesting approach is provided by OntoWebber which briefly sketches personalization by means of model re-write rules.

There is very little support in the analyzed SWIS design methodologies to model more advanced forms of user interaction with the system than simple link following. SEAL offers a limited form of user interaction by defining forms only for changing the system input data not affecting thus the application's hyperspace. The only form of user interaction that OOHDM supports is the user selection of items from existing data.

As for WIS design methodologies, a feature also neglected in the examined SWIS design methodologies is the design support offered for the presentation aspects (the look-and-feel aspects) of SWIS. OntoWebber briefly sketches similar presentation specification mechanisms.

Differently than WIS design methodologies, SWIS design methodologies support model verification. This is largely due to the direct application of the model semantics as specified in the associated ontologies. Besides the validation feature offered by the use of ontologies, some methodologies (e.g., OntoWebber) offer the possibility to express structural, semantical, and presentational constraint verification.

SWIS design methodologies support reuse by inheritance mechanism not only at concept level but also at property level in their models. This is due to the property-centric view of the Semantic Web languages (e.g., RDF, OWL) that support property specialization. XWMF also shows how the multiple instantiation mechanism can enable the reuse of the same data object for different implementations. SHDM introduces the faceted navigation structure that has a concise representation which reduces the effort of specifying navigation models.

2.5 Conclusions

In this chapter we presented the current situation with respect to (S)WIS design methodologies. While WIS design methodologies did reach maturity, more research has to be done for WIS methodologies that make use of Semantic Web technologies (the so-called SWIS design methodologies). Realizing the benefits of the Semantic Web platform (e.g., interoperability, inference capabilities, increased reuse of the design artifacts, etc.) traditional WIS design methodologies like OOHDM or WSDM are now focusing on designing SWIS. New methodologies like OntoWebber were specifically designed by considering the

Semantic Web peculiarities.

Realizing the importance of a personalized (S)WIS, during the last years a lot of attention was given to the design of the (S)WIS adaptation aspects. Good results were obtained for the static system adaptation (e.g., OOHDM, WebML), i.e., adaptation performed before the user starts browsing the (S)WIS. More work has to be done for the dynamic adaptation of these systems, i.e., adaptation performed during user browsing of the (S)WIS. Research done in the adaptive hypermedia [De Bra et al., 1999] field can prove to be useful for designing adaptive (S)WIS.

As the Semantic Web matures, we hope that the same will happen with SWIS design methodologies. One of the main obstacles in building SWIS is also the absence of standardized query languages and the lack of data transformation languages for the Semantic Web. By defining standards for integration, conceptual, navigational, presentation, and personalization modeling one will greatly contribute for SWIS application interoperability. Also by having the user profile defined in a standard way will enable the reuse of user profiles among SWIS.

An important factor to assure the success of a WIS design methodology is the existence of tool support. A powerful methodology that is not accompanied by adequate tools will make the designer's tasks very difficult to fulfill. Most of the WIS design methodologies have powerful CASE tools. There are very few SWIS design methodologies with a good tool support.

Bibliography

Aduna, BV (2005). openrdf.org ... home of sesame. `http://www.openrdf.org/`.

AIFB, University of Karlsruhe (2004). Karlsruhe ontology and semantic web tool suite. `http://kaon.semanticweb.org/`.

Anthemion Software (2004). wxwidgets. `http://www.wxwidgets.org/`.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., and Stein, L. A. (2004). Owl web ontology language reference. W3C Recommendation 10 February 2004. `http://www.w3.org/TR/owl-ref/`.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*, 284(5):34–43. `http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html`.

Brambilla, M., Ceri, S., Comai, S., Fraternali, P., and Manolescu, I. (2002). Model-driven specification of web services composition and integration with data-intensive web applications. *IEEE Data Engineering Bulletin*, 25(4):53–59.

Bush, V. (1945). As we may think. *The Atlantic Monthly*, 176(1):101–108.

Casteleyn, S. and De Troyer, O. (2001). Structuring web sites using audience class hierarchies. In *Conceptual Modeling for New Information Systems Technologies (ER 2001 Workshops)*, volume 2465, pages 1222–1228. Springer.

Casteleyn, S., De Troyer, O., and Brockmans, S. (2003). Design time support for adaptive behaviour in web sites. In *18th ACM Symposium on Applied Computing (SAC 2004)*, pages 1222–1228. ACM.

Casteleyn, S., Garrigos, I., and De Troyer, O. (2004). Using adaptive techniques to validate and correct an audience driven design of web sites. In *Web Engineering - 4th International Conference (ICWE 2004)*, volume 3140 of *Lecture Notes in Computer Science*, pages 55–59. Springer.

Ceri, S., Fraternali, P., and Bongio, A. (2000). Web modeling language (webml): a modeling language for designing web sites. *Computer Networks, Ninth International World Wide Web Conference*, 33:137–157.

Ceri, S., Fraternali, P., Bongio, A., Brambilla, M., Comai, S., and Matera, M. (2003). *Designing Data-Intensive Web Applications*. Morgan Kaufmann.

De Bra, P., Houben, G. J., and Wu, H. (1999). Aham: A dexter-based reference model for adaptive hypermedia. In *10th ACM conference on Hypertext and Hypermedia (Hypertext'99)*, pages 147–156. ACM.

De Troyer, O. and Casteleyn, S. (2004). Designing localized web sites. In *5th International Conference on Web Information Systems Engineering (WISE 2004)*, volume 3306, pages 547–558. Springer.

De Troyer, O. and Leune, C. (1998). Wsdm: A user-centered design method for web sites. *Computer Networks, Seventh International World Wide Web Conference*, 30:85–94.

Diaz, A., Isakowitz, T., Maiorana, V., and Gilabert, G. (1995). Rmc: A tool to design www applications. In *Fourth International World Wide Web Conference (WWW 4)*.

Diaz, A., Isakowitz, T., Maiorana, V., and Gilabert, G. (1997). Extending the capabilities of rmm: Russian dolls and hypertext. In *30th Hawaii International Conference on System Sciences (HICSS-30)*, volume 6, pages 177–186. IEEE Computer Society.

Fernandez, M. F., Florescu, D., Levy, A. Y., and Suciu, D. (2000). Declarative specification of web sites with strudel. *VLDB Journal*, 9(1):38–55.

Fiala, Z., Frasincar, F., Hinz, M., Houben, G. J., Barna, P., and Meissner, K. (2004). Engineering the presentation layer of adaptable web information systems. In *Web Engineering - 4th International Conference (ICWE 2004)*, volume 3140 of *Lecture Notes in Computer Science*, pages 459–472. Springer.

Gomez, J. and Cachero, C. (2003). *Information Modeling for Internet Applications*, chapter OO-H Method: extending UML to model web interfaces, pages 144–173. Idea Group Publishing.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

Guell, N., Schwabe, D., and Vilain, P. (2000). Modeling interactions and navigation in web applications. In *Conceptual Modeling for E-Business and the Web (ER 2000) Workshops*, volume 1921 of *Lecture Notes in Computer Science*, pages 115–127. Springer.

Halpin, T. (1995). *Model-Based Design and Evaluation of Interactive Applications*. Prentice-Hall.

Hester, A. M., Borges, R., and Ierusalimschy, R. (1997). Xi brazilian software engineering symposium (sbes 1997). In *CGILua: A Multi-paradigmatic Tool for Creating Dynamic WWW Pages*.

Ierusalimschy, R., de Figueiredo, L. H., and Filho, W. C. (1996). Lua-an extensible extension language. *Software: Practice and Experience*, 26(6):635–652.

Isakowitz, T., Stohr, E. A., and Balasubramanian, P. (1995). Rmm: A methodology for structured hypermedia design. *Communications of the ACM*, 38(8):34–44.

Isakowitz, T., Bieber, M., and Vitali, F. (1998). Web information systems. *Communications of the ACM*, 41(1):78–80.

Jacyntho, M. D., Schwabe, D., and Rossi, G. (2002). A software architecture for structuring complex web applications. *Journal of Web Engineering*, 2(1-2):37–60.

Jin, Y., Xu, S., and Decker, S. (2001). Ontowebber: Model-driven ontology-based web site management. In *1st International Semantic Web Working Symposium (SWWS 2001)*, pages 529–547. Stanford University.

Jin, Y., Xu, S., and Decker, S. (2002). Managing web sites with ontowebber. In *8th International Conference on Extending Database Technology (EDBT 2002)*, volume 2287 of *Lecture Notes in Computer Science*, pages 766–768. Springer.

Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., and Scholl, M. (2002). Rql: a declarative query language for rdf. In *Eleventh International World Wide Web Conference (WWW2002)*, pages 592–603. ACM.

Kay, M. (2005). Xsl transformations (xslt) version 2.0. W3C Working Draft 11 February 2005. `http://www.w3.org/TR/xslt20/`.

Kifer, M., Lausen, G., and Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42(1):741–843.

Klapsing, R. and Neumann, G. (2000). Applying the resource description framework to web engineering. In *First International Conference on Electronic Commerce and Web Technologies (EC-Web 2000)*, pages 229–238. Springer.

Klapsing, R., Neumann, G., and Conen, W. (2001). Semantics in web engineering: Applying the resource description framework. *IEEE MultiMedia*, 8(2):62–68.

Koch, N., Kraus, A., and Hennicker, R. (2001). The authoring process of the uml-based web engineering approach. In *First International Workshop on Web-Oriented Software Technology (IWWOST 2001).*

Lei, Y., Motta, E., and Domingue, J. (2003). Design of customized web applications with ontoweaver. In *International Conference on Knowledge Capture (K-CAP 2003)*, pages 54–61. ACM.

Lima, F. and Schwabe, D. (2003a). Application modeling for the semantic web. In *1st Latin American Web Congress (LA-WEB 2003)*, pages 93–102. IEEE Computer Society.

Lima, F. and Schwabe, D. (2003b). Designing personalized web applications. In *Web Engineering, International Conference (ICWE 2003)*, volume 2722 of *Lecture Notes in Computer Science*, pages 417–426. Springer.

Lyardet, F. and Rossi, G. H. (1996). Enhancing productivity in the development of hypermedia applications. In *Workshop on Next Generation CASE Tools (NGCT 1996), CAiSE 1995*.

Maedche, A., Staab, S., Stojanovic, N., Studer, R., and Sure, Y. (2003). Semantic portal: The seal approach. In *Spinning the Semantic Web Bringing the World Wide Web to Its Full Potential [outcome of a Dagstuhl seminar]*, pages 317–359. MIT Press.

Maedche, A., Staab, S., Studer, R., Sure, Y., and Volz, R. (2002). Seal - tying up information integration and web site management by ontologies. *IEEE Data Engineering Bulletin*, 25(1):10–17.

Mecca, G., Atzeni, P., Masci, A., Merialdo, P., and Sindoni, G. (1998). The araneus web-base management system. In *SIGMOD Conference*, pages 544–546.

Moura, S. S. D. and Schwabe, D. (2004). Interface development for hypermedia applications in the semantic web. In *1st Latin American Web Congress (LA-WEB 2004)*, pages 106–113. IEEE Computer Society.

Murugesan, S., Deshpande, Y., Hansen, S., and Ginige, A. (2001). Web engineering: A new discipline for development of web-based systems. In *Web Engineering*, volume 2016 of *Lecture Notes in Computer Science*, pages 3–13. Springer.

Neumann, G. and Nusser, S. (1993). Wafe - an x toolkit based frontend for application programs in various programming languages. In *USENIX Winter*, pages 181–192. USENIX Association.

Neumann, G. and Zdun, U. (2000). Xotcl, an object-oriented scripting language. In *The 7th USENIX Tcl/Tk Conference*, pages 163–174. USENIX Association.

Noy, N. F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R. W., and Musen, M. A. (2001). Creating semantic web contents with protege-2000. *IEEE Intelligent Systems*, 16(2):60–71.

Pastor, O., Fons, J., and Pelechano, V. (2003). Oows: A method to develop web applications from web-oriented conceptual models. In *International Workshop on Web-Oriented Software Technology (IWWOST 2003)*, pages 65–70.

Paterno, F. (2000). *Model-Based Design and Evaluation of Interactive Applications*. Springer.

Schewe, K.-D. and Thalheim, B. (2001). Modeling interaction and media objects. In *Natural Language Processing and Information Systems (NLDB 2000)*, volume 1959 of *Lecture Notes in Computer Science*, pages 313–324. Springer.

Schewe, K.-D. and Thalheim, B. (2004). Reasoning about web information systems using story algebras. In *Advances in Databases and Information Systems (ADBIS 2004)*, volume 3255 of *Lecture Notes in Computer Science*, pages 54–66. Springer.

Schwabe, D., Rossi, G., and Barbosa, S. D. J. (1996). Systematic hypermedia application design with oohdm. In *The Seventh ACM Conference on Hypertext (Hypertext 1996)*, pages 116–128. ACM.

Schwabe, D., de Almeida Pontes, R., and Moura, I. (1999). Oohdm-web: an environment for implementation of hypermedia applications in the www. *ACM SIGWEB Newsletter*, 8(2):18–34.

Schwabe, D. and Rossi, G. (1998). An object oriented approach to web-based application design. *Theory and Practice of Object Systems*, 4(4):207–225.

Schwabe, D., Szundy, G., Moura, S. S. D., and Lima, F. (2004). Design and implementation of semantic web applications. In *WWW Workshop on Application Design, Development and Implementation Issues in the Semantic Web (WE-SW 2004)*, volume 105 of *CEUR Workshop Proceedings*, pages 275–284.

Sintek, M. and Decker, S. (2002). Triple - an rdf query, inference, and transformation language. In *First International Semantic Web Conference (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science*, pages 364–378. Springer.

Storey, M.-A. D., Noy, N. F., Musen, M. A., Best, C., Fergerson, R. W., and Ernst, N. (2002). Ontoedit: Multifaceted inferencing for ontology engineering. In *International Conference on Intelligent User Interfaces (IUI 2002)*, pages 239–239. ACM.

Thalheim, B. (2000). *Entity-Relationship Modeling*. Springer.

Thalheim, B. and Dusterhoft, A. (2001). Sitelang: Conceptual modeling of internet sites. In *Conceptual Modeling (ER 2001)*, volume 2224 of *Lecture Notes in Computer Science*, pages 179–192. Springer.

Thalheim, B., Schewe, K.-D., Romalis, I., Raak, T., and Fiedler, G. (2004). Website modeling and website generation. In *Web Engineering - 4th International Conference (ICWE 2004)*, volume 3140 of *Lecture Notes in Computer Science*, pages 577–578. Springer.

Wielemaker, J. (2000). Swi-prolog rdf parser. `http://www.swi-prolog.org/packages/rdf2pl.html`.