Specification Techniques for Web Application Development – A Comparison

By Iris Reinhartz-Berger

1. Specification Techniques for Web Application Development

The development of Web applications can be approached from two different viewpoints: the hypermedia authoring approach and the system and software development approach. The system and software development approach can be further divided into two subgroups: the object-oriented approach and the behavioral-oriented approach. An integrated approach to system development that integrates concepts from both the object-oriented approach and the behavioral-oriented approach is called Object-Process Methodology (OPM) and is discussed in Section 2.

1.1 The Hypermedia Authoring Approach

The techniques that follow the hypermedia authoring approach are either based on the Entity-Relation (E-R) model (the Hypertext Design Model and the Relationship Management Methodology), or on the Object-Oriented (OO) model (the Object-Oriented Hypertext Design Model and the Enhanced Object Relationship Model).

The **Hypertext Design Model (HDM)** [GM93, GPS93] shifts the focus from hypertext data models as a means to capture the structuring primitives of hypertext systems, to hypertext models as a means for capturing the semantics of a hypermedia application domain. It prescribes the definition of an application schema, which specifies classes of information elements in terms of their common presentation characteristics, their internal organization structure, and the types of their mutual interconnections. Web structure is expressed by means of entities, sub-structured into a tree of components. Navigation can be internal to entities (along part-of links), cross-entity (along generalized links), or non-contextual (using access indexes, called collections). HDM is concerned with representational and navigational issues, rather than with behavioral aspects of hypertext and operational features. Furthermore, it provides little information on the procedures for using its representation schemes in the design process. It is therefore not a complete hypermedia design and development methodology.

The **Relationship Management Methodology (RMM)** [ISB95] evolves HDM by embedding its hypermedia design concepts into a structured methodology, splitting the development process into seven distinct steps and giving guidelines for the tasks. The development cycle steps in RMM are E-R design, slice design, navigation design, conversion protocol design, user-interface design, runtime behavior design, and construction & testing.

The **Object-Oriented Hypertext Design Model (OOHDM)** [SRB96, SR98] is a direct descendant of HDM. It differs from HDM in its object-oriented nature, and in that it includes special-purpose modeling primitives for both navigational and interface design. OOHDM comprises four different activities: conceptual design, navigational design, abstract interface design, and implementation. During each activity, except for the implementation, a set of object-oriented models describing particular design concerns are built or enriched from previous iterations.

The **Enhanced Object Relationship Model (EORM)** [LA96] is defined as an iterative process concentrating on the enrichment of the object-oriented model by the representation of relations between objects (links) as objects. The technique is based on three frameworks of reusable libraries: one for class definition, one for composition (link class definition) and one for GUIs.
EORM differs from OOHDM mainly in that OOHDM clearly separates navigational from conceptual design concerns by defining different modeling primitives for each step, while EORM combines them all together. The advantages of EORM over OOHDM are that relations become semantically rich as they are extensible constructs, they can participate in other relations, and they can be part of reusable libraries.

1.2 The Object-Oriented Development Approach

The most common object-oriented methods for system development in general and Web applications in particular are UML and OPEN.

The Unified Modeling Language (UML) [UML99], which is the industry standard object-oriented modeling language, is a general-purpose visual modeling language for specifying, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. UML combines ideas from three previously leading object-oriented methods: OOSE (Object-Oriented Software Engineering) [JA92], OMT (Object Modeling Technique) [RU91], and Booch'93 [BO94]. UML captures information about the static structure

and dynamic behavior of a system. The *structural classification* defines the classes of objects important to a system and to its implementation, as well as the relationships among the classes. The *dynamic behavior* defines the history of objects over time and the communications among them to accomplish goals.
Web applications, like other software-intensive systems, are typically represented by a set of models: a use-case model, an implementation model, a deployment model, a security model, and so forth. An additional model used exclusively by Web information systems is the site map, an abstraction of the Web pages and navigation routes throughout the system. Since UML version 0.91, the developer is able to model Web applications using the built-in extension mechanisms, which are tagged values, stereotypes and constraints, in addition to the above models. The stereotypes UML offers for Web application designers are elaborated in [CO99a, CO99b].

The **Object Process, Environment and Notation (OPEN)** [OPN99] is a third generation, public domain, object-oriented methodology that focuses on the development process specification. It was initially created by a merger of MOSES [HSE94], SOMA [GR95] and Firesmith [FI93] methods, commencing at the end of 1994. In addition to synthesizing the ideas and practices from those methods, OPEN also offers a set of principles for modeling all aspects of software development across the full lifecycle. The development process is described by a contract-driven lifecycle model, which is complemented by a set of techniques and a formal representation using a modeling language such as OML (OPEN Modeling Language) [HS98] or UML. In the OPEN architecture, a modeling process consists of one or more activities, each consisting of one or more tasks. Tasks are carried out by agents (people) using techniques. A two-dimensional matrix links the task (which provides the statement of goals, i.e. the "what") to the techniques (which provide the way the goal can be achieved, i.e. the "how").
Because it concentrates on the development process, OPEN attempts to solve the following deficiencies in the Web application development process [LHS99]: lack of documented process, lack of non-code deliverables, lack of object orientation in the design phase, lack of metrics, and lack of CASE/support tools.

1.3 The Behavioral-Oriented Development Approach

The term 'behavioral-oriented' can be interpreted in several ways. Here it pertains to approaches that specify the conditions or constraints to activate some (perhaps partly specified) functionality at a desired moment. Three of the behavioral-oriented design techniques, which deal also with the development of Web applications and distributed systems, are the superimposition approach, Aspect-Oriented Design and the Event-Condition-Action paradigm.

The **superimposition approach** [KA93] introduces a procedure-like control structure (called a superimposition) that allows convenient expression of the superimposition of one algorithm on another. In this construct, both formal processes and schematic abstractions (called roletypes) are declared, each with formal parameters and a sequential communicating algorithm using those parameters. The declaration captures a distributed algorithm that is separately designed, but is intended to be executed in conjunction to other activities in the same state space. The construct is combined with an existing collection of communicating processes by instantiating the formal processes and associating each process of the collection with one roletype.

Aspect-Oriented Design (AOD) [AOP00, BE98, CBE99] is an abstraction principle intended to help software developers more cleanly separate concerns in their design and source code. An aspect modularizes the features for a particular concern and describes how those features should be integrated (woven) into the system design. In object-oriented methods, an aspect is typically spread across multiple methods in multiple classes. In AOD, an aspect can be separated from the classes to which it applies. The aspect-oriented approach emerged through the development of AspectJ, a programming language based on Java. This was followed by attempts to percolate this concept to earlier stages in the application development lifecycle, i.e., to the design stage. At present, there are two main modeling techniques that use aspects, both based on UML. The first one extends the UML metamodel with another classifier element (like Class, Interface, Component and Node), called **Aspect** [SY99]. The second technique introduces the aspect notion as a new notation – a diamond and an associated rectangle, which consists of four sections: attributes, operations, introduce members and advice methods [KM99].

The **Event-Condition-Action Paradigm (ECA)** [CH93] is used in information systems in general and database systems in particular in order to apply the concept of triggers in specifying the behavior of a system. A trigger is a procedural processing element, stored in the database and executed automatically by the DBMS server under specific conditions. A trigger is composed of three components: Event, Condition and Action. It might also have an "Event Condition then Action else Action" form (also called ECAA principle). In Web applications those principles can be used, for example, for modeling business rules [PE98]. These rules are expressed by an object-oriented meta-language, which is XML-like.

2. The Object-Process Methodology

The Object-Process Methodology (OPM) [DO95, DG96, DO01] is an integrated approach to the study and the development of systems in general and information systems in particular. The basic premise of OPM is that objects and processes are two types of equally important classes of things, that together faithfully describe the structure, function and behavior of systems in a single framework in virtually any domain. OPM unifies the system specification, design and implementation within one frame of reference, using a single diagramming tool – the Object-Process Diagram (OPD) and a corresponding, English-like language – the Object-Process Language (OPL). OPM has been implemented successfully in various industrial applications, including microprocessor manufacturing, metal cutting technology, real-time systems [PD99], and ERP [SDG00], because of its structure modeling preciseness, its behavior modeling flexibility, its scalability and complexity management abilities, and its single framework.

In OPM, system classes, class features, physical devices, and environmental interfaces are all modeled as object classes. An object class can be either internal or environmental, and (orthogonally) either physical or informatical (logical). In addition, an object class can have several states, which represent the possible internal status values of the class. At any point in time, each object is in some state, and object states are transformed through the activation of a process. All these structural features increase the precision of the model, as a process can affect (change the state of) an environmental or a physical device, or an attribute of an object class (rather than the state of the entire class, as in object-oriented methods).

Unlike in the object-oriented approach, behavior in OPM is not necessarily encapsulated within the object class construct, but one can model a crosscut behavior that bisects the object classes and the system structure. This modeling is done by using stand-alone processes which make the behavior modeling more flexible and can be connected to the involved object classes through one or more of the procedural links. The procedural links can be divided, according to their functionality, into three groups, which are enabling links, transformation links, and event links.

The large number of constructs (objects, processes, structural relations, and procedural links) resulting in a complex system model and the intersections among them is managed by using OPM's build-in scaling mechanisms, which are unfolding/folding and zooming-in/zooming-out. The unfolding/folding mechanism uses structural relations for detailing/hiding the structural parts of a thing. For example, in Figure 1a, the process P1 is unfolded to its parts, processes P1.1 and P1.2, and to its feature, object B1.1. The zooming-in/zooming-out mechanism exposes/ hides the inner details of a thing within its frame. In Figure 1b, the process P1 is zoomed-in, showing the process flow – first P1.1 is executed and creates object B1.1, and then process P1.2 is activated, consuming object B1.1.
This way OPM facilitates focusing on a particular subset of things (objects and/or processes), elaborating on the details of each one by scaling up each thing to any desired level, and managing the complexity of a system model.

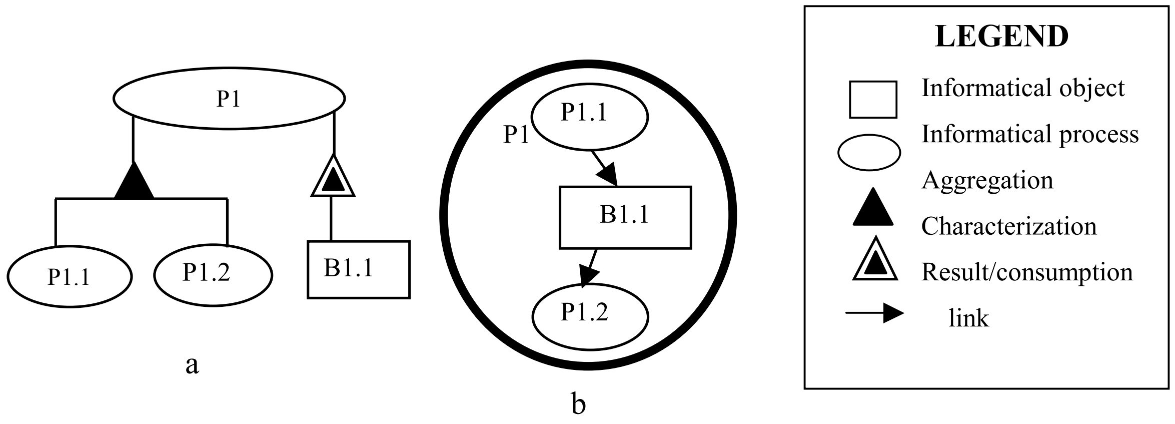

Figure 1. Applying OPM scaling mechanisms to process P1.
(a) Process P1 is unfolded. (b) Process P1 is zoomed-in.

The ability to use a single OPM framework in order to specify all the system aspects, including structure, functionality and dynamics, prevents compatibility and integration problems among the different diagram types. The OPM framework is expressed visually by an OPD-set and textually by an OPL script and provides the ability to grasp the entire system through one visual language (for system developers) and one textual language (for managers and other non-experts in system development tools).

3. Comparison of Web Application Methodologies

In this section I examine the reviewed modeling techniques according to the domain of Web applications. The evaluation criteria include level of structuredness, modularity and reusability, physical architecture representation, user interface modeling, dynamic behavior modeling, and security and privacy management. The results of this comparison are shown in Table 1.

Criterion	Hypermedia Authoring approaches	Object-Oriented approaches	Behavioral-Oriented approaches	Object-Process Methodology
Level of Structuredness	average	good	poor	good
Modularity and Reusability	poor	average	average+	average+
Physical Architecture Representation	poor	average	poor	average
User Interface Modeling	average	average	poor	average
Dynamic Behavior Modeling	poor	average	average	good
Security and Privacy Management	poor	average	poor	average

Table 1. Comparison of Web application modeling techniques by quality of various criteria

3.1 Level of Structuredness

In the *behavioral-oriented approach*, the structure of the application is not specified, so they must rely on combining other approaches (E-R or OO) to complement their modeling.

The *hypermedia authoring techniques* that are based on the E-R model are suited for simple structured data, but not for complex or unstructured data. They do not provide for complexity management facilities, such as hierarchy and aggregation. RMM introduces the "slice" concept for categorizing the entities, but this provision is quite limited.

The *object-oriented methods and OPM* can model complex structured data by using attributes, aggregation, inheritance, and other structural relations. In addition those approaches can better model unstructured data or semi-structured data, by using the multiplicity property similar to the way XML uses regular expressions. Nevertheless, object-oriented methods, unlike OPM, cannot easily model complex features of object classes and their changes over time.

3.2 Modularity and Reusability

The *object-oriented development methods* apply modularity and reusability through packages and class hierarchy. Yet they do not support crosscut modularity among objects, i.e., concepts that are shared by different object classes. Therefore, adding functionality to the system might affect several different object classes. To tackle this problem, *AOD* suggests aspects as a new unit of software modularity which attempts to provide a better handle on managing crosscut concerns. This kind of modularity can be adopted by *OPM*, but it has not been checked yet.

3.3 Physical Architecture Representation

None of the reviewed techniques regards the dynamic architecture behavior and Web programming concepts such as worms, cookies, Java applets, etc.

The *object-oriented development methods and OPM* are the only ones that are at least partially concerned with the physical architecture aspect. *UML* has a special diagram type, the deployment diagram, for representing these concepts. This type of diagram, whose purpose is to consider the load for building effective applications, models only the structural part of the system architecture, i.e., the system components and their links (connectors). In OPM the structure of the physical architecture is modeled using physical and informatical things. A physical object consists of matter and/or energy, is tangible in the broad sense and can be detected by one or more of our senses, while an informatical object is a piece of information. A physical process is a change that a physical object undergoes and similarly an informatical process is some transformation (or manipulation) of an informatical object.

3.4 User Interface Modeling

The *behavioral-oriented techniques* focus on the functionality of the application and pay little or no attention to the user interface.

In contrast, the *object-oriented development methods and OPM* allow some user interface modeling by using classes for this purpose. Some of these classes are pre-defined in programming languages like JDialog and JMenu in the Java Swing library. Nevertheless, they neglect basic, commonly used navigational structures, such as index guides and paths, and do not connect the user interface and the user actions to the functionality occurring in the system afterwards.

The *hypermedia authoring techniques* introduce only three simple user interfaces: grouping, conditional index and conditional guided tour, which are insufficient to model the large variety of navigation possibilities and user interfaces in Web applications. The hypermedia authoring techniques which are based on the E-R model also neglect the complexity management of the user interface.

3.5 Dynamic Behavior Modeling

Hypermedia authoring techniques model the structure and navigational aspects of the application, but not its functionality.

The *object-oriented development methods* enable modeling application functionality through class services and message passing among the objects, but they are not capable of modeling independent, stand-alone processes, which cannot be affiliated with one specific class, as opposed to *OPM*.

The *behavioral-oriented techniques* model the system functionality separately from the application structure, making it very difficult to model data processing.

Furthermore, none of the techniques provides code migration modeling, such as in Java applets and cookies.

3.6 Security and Privacy Management

Most Web application development techniques do not address at all security and privacy issues during the design phase, leaving it to the implementation stage. *UML and OPM* enable partial authentication and access control by use case diagrams and agent links, respectively. *UML* also supports data integrity by using the Object Constraint Language, OCL [WA99], along with the class diagram.

4. References

[AOP00] The AOP Web site, http://www.parc.xerox.com/csl/projects/aop/

[BE98] U. Becker, D^2AL – A design-based aspect language for distribution control, proceedings of the Europe Conference on Object-Oriented Programming (ECOOP), 1998.

[BO94] G. Booch, Object-Oriented Analysis and Design with Applications, Benjamin/Cummings, 1994.

[CBE99] C. A. Constantinides, A. Bader, and T. Elrad, An Aspect-Oriented Design Framework for Concurrent Systems, proceedings of the Europe Conference on Object-Oriented Programming (ECOOP), 1999, pp. 340-352.

[CH93] S. Chakravarthy, SNOOP: An expressive Event Specification Language for Active Databases, Technical Report UF-CIS-TR-93-007, 1993, pp. 1-25.

[CO99a] J. Conallen, Modeling Web Application Architectures with UML, Communication of the ACM vol. 42 no. 10, 1999.

[CO99b] J. Conallen, Building Web Applications with UML, the Addison-Wesley Object Technology Series, 1999.

[DG96] D. Dori and M. Goodman, From Object-Process Analysis to Object-Process Design, Annals of Software Engineering, vol. 2, 1996, pp. 25-40.

[DO01] D. Dori, Object-Process Methodology Applied to Modeling Credit Card Transactions, Journal of Database Management, vol. 12, no.1, pp. 2-12, 2001 (to appear), http://iew3.technion.ac.il:8080/Home/Users/dori/JDM-Dori-OPM.pdf.

[DO95] D. Dori, Object-Process Analysis: Maintaining the Balance between System Structure and Behavior, Journal of Logic and Computation, vol. 5, no. 2, 1995, pp. 227-249.

[FI93] D. G. Firesmith, Object-Oriented Requirements, Analysis, and Logical Design: a Software Engineering Approach, John Wiley & Sons, 1993.

[GM93] F. Garzotto and L. Mainetti, HDM2: Extending the E-R approach to hypermedia aoolication desugn, Proceeding of the 12th International Conference on Entity Relationship Approach, 1993, pp. 178-189.

[GPS93] F. Garzotto, P. Paolini, and D. Schwabe, HDM – A Model Based Approach to Hypertext Application Design, ACM Transactions on Information Systems, Vol. 11, No. 1, 1993, pp. 1- 26.

[GR95] I.S. Graham and A. Graham, Migrating to Object Technology, Addison-Wesley Publication Corporation, 1995.

[HS98] B. Henderson-Sellers, OML: proposals to enhance UML, UML'98, Lecture Notes in Computer Science vol. 1618, Springer-Verlag, pp. 349-364.

[HSE94] B. Henderson-Sellers and J.M. Edwards, Booktwo of Object-Oriented Knowledge: The Working Object, Prentice Hall, 1994.

[ISB95] T Isakowitz, E. A. Stohr, and P. Balasubramanian, RMM: A Methodology for Structured Hypermedia Design, Communication of the ACM, Vol. 38, No. 8, 1995, pp. 34-44.

[JA92] I. Jacobson, Object-Oriented Software Engineering, Addison-Wesley, 1992.

[KA93] S. Katz, A Superimposition Control Construct for Distributed Systems, ACM Transactions on Programming Languages and Systems, vol. 15, No. 2, 1993, pp. 337-356.

[KM99] M. Kersten and G. C. Murphy, Atlas: A Case Study in Building a Web-Based Learning Environment using Aspect-Oriented Programming, Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA), 1999.

[LA96] D. Lange, An Object-Oriented Design Approach for Developing Hypermedia Information Systems, Journal of Organizational Computing, 1996.

[LHS99] M. Lin and B. Henderson-Sellers, Adapting the OPEN methodology for Web development, Methodologies for Developing and Managing Emerging Technology Based Information Systems, Proceedings of the Sixth Annual Conference of BCS Information Systems Methodology Specialist Group, Springer-Verlag, 1999, pp. 117-129.

[OPN99] The OPEN Web site, http://www.open.org.au/

[PD99] M. Peleg and D. Dori, Extending the Object-Process Methodology to Handle Real-Time Systems, Journal of Object-Oriented Programming, vol. 11, no. 8, 1999, pp. 53-58.

[PE98] D. Perrault, A Study of Business Rules Concept For Web Application, degree thesis, the faculty of engineering, Politecnico di Milano, 1998, http://www.ing.unico.it/autoWeb/

[RU91] J. Rumbaugh, Object-Oriented Modeling and Design, Prentice Hall, 1991.

[SDG00] P. Sofer, D. Dori and B. Golany, Closing the Gap Between Enterprise Needs and BRP System Capabilities: An Object-Process Based Approach, submitted for publication, 2000.

[SR98] D. Schwabe and G. Rossi, Developing Hypermedia Applications using OOHDM, Workshop on Hypermedia Development Processes, Methods and Models, Hypertext'98, 1998, http://heavenly.nj.nec.com/266278.html

[SRB96] D. Schwabe, G. Rossi, and S. Barbosa, Systematic Hypermedia Application Design with OOHDM, Proceedings of the seventh ACM conference on Hypertext, 1996, pp.116 – 128.

[SY99] J. Suzuke and Y. Yamamoto, Extending UML with Aspects: Aspect Support in the Design Phase, proceedings of the Europe Conference on Object-Oriented Programming (ECOOP), 1999.

[UML99] OMG Unified Modeling Language Specification – version 1.3, 1999, http://www.rational.com/media/uml/resources/documentation/ad99 -06-08-ps.zip.

[WA99] J. B. Warmer, A. G. Kleppe, The Object Constraint Language: Precise Modeling With UML, Addison-Wesley Object Technology Series, 1999.