Designing Adaptive Web Applications

Dolog, Peter

Published in:
SOFSEM 2008: Theory and Practice of Computer Science

DOI (link to publication from Publisher):
10.1007/978-3-540-77566-9_3

Publication date:
2008

Document Version
Publisher's PDF, also known as Version of record

Designing Adaptive Web Applications

Peter Dolog

Aalborg University, Computer Science Department
Selma Lagerlöfs Vej 300, DK-9220 Aalborg-East, Denmark
dolog@cs.aau.dk

Abstract. The unique characteristic of web applications is that they are supposed to be used by much bigger and diverse set of users and stakeholders. An example application area is e-Learning or business to business interaction. In eLearning environment, various users with different background use the eLearning system to study a discipline. In business to business interaction, different requirements and parameters of exchanged business requests might be served by different services from third parties. Such applications require certain intelligence and a slightly different approach to design. Adpative web-based applications aim to leave some of their features at the design stage in the form of variables which are dependent on several criteria. The resolution of the variables is called adaptation and can be seen from two perspectives: adaptation by humans to the changed requirements of stakeholders and dynamic system adaptation to the changed parameters of environments, user or context. Adaptation can be seen as an orthogonal concern or viewpoint in a design process. In this paper I will discuss design abstractions which are employed in current design methods for web applications. I will exemplify the use of the abstractions on eLearning web applications as well as on applications for business to business interaction based on web services.

1 Introduction

Adaptive Web-based applications provide an alternative to the traditional "one-size-fits-all" applications [3]. Such applications try to address diverse requirements of different stakeholders by leaving some of their features at the design stage in the form of variables which are dependent on several criteria. The resolution of the variables is called adaptation and can be seen from two perspectives:

- Adaptation by humans to the changed requirements of stakeholders;
- Dynamic system adaptation to the changed parameters of the environment or context.

User-centered adaptive applications utilize user features to resolve the variability; i.e. to determine appropriate information presentation and navigation sequences for exploring a sufficiently complete set of information. They update a user model in accordance with user interaction and the information which he or she has provided. There are several application domains where such adaptive

web applications have been found useful such as education, eCommerce, and news.

Adaptive eLearning Applications. The UML-Guide [7] is an example of an adaptive web application. It geneates a map of an information space designed for a particular information or learning goal. The map itself is adaptive; some of the links and symbols are annotated according to knowledge about a user which is maintained by the UML-Guide.

One approach to the visualization of the navigation map is depicted in Fig. 1. The navigation map displays a composite hierarchy of information nodes in information space (folder symbols with subfolders and document nodes) and sequencing relations between the nodes (arrow symbol).

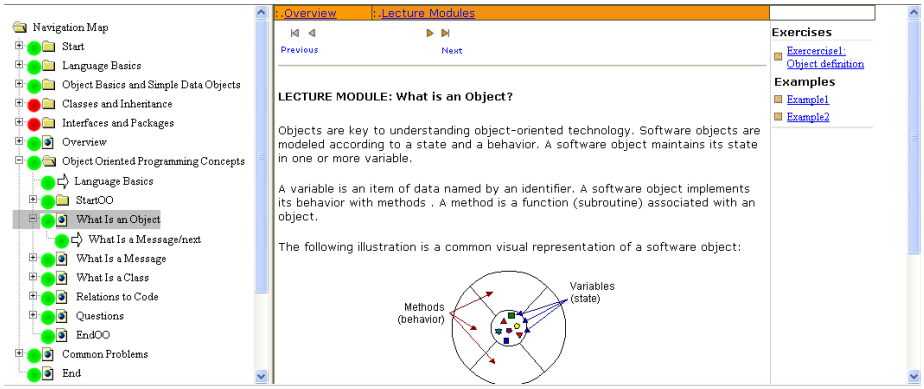

Fig. 1. Visualization of the navigation graph for the Java e-lecture [4]

Besides the adaptation mentioned here, the content served by a web application can be adapted as well. Some fragments of presented content can be hidden, some can be displayed. Composition and placement of the fragments can change according to preferences and user abilities.

The adaptation in such applications like the UML-Guide is usually a decision for a particular information item or function based on knowledge about the items being recommended. The knowledge about items usually comprises what particular information items or functions have in common and where they differ. The properties which differ from item to item determine the source for adaptation. The selection of an appropriate item is based on results from matchmaking between user and information properties. The differences between users in terms of properties and their values determine a selection of different information items or functions which best fit to particular user features according to a chosen selection strategy. As the user's behavior pattern evolves, recommended items may change. This is ensured by continuous updating and evolution of a user's profile based on his or her behavior as traced by the application. In this way, the user

always receives up to date information items or functions matching the current state of her profile.

Business to Business Interaction. Supply chain management or financial applications are other examples of application which could benefit from adaptation. In such applications, the adaptivity is considered beyond the user profile and is based on rather different requirements profiles matched with service profiles satisfying a business function. Consider for example a company's monthly payroll processing from [21]. A company has to calculate a salary for each employee. In the next step, the payment of the salary is performed, which comprises several operations. First of all, the salary is transferred from the company's account to the employee's account. Then the company transfers the employee's income tax to the account of the tax authorities. Finally, the company prints the payslip and sends it to the employee. On the employee has only one task which he has to perform each month in this scenario: He transfers the monthly installment for his new car to the car dealer's account.

The company's and the employee's operations are each controlled by a business process, and are implemented using Web services from multiple providers. The business transactions are used in order to guarantee a consistent execution of all required operations. This is depicted in Figure 2. Only the services of transaction T1 are shown.

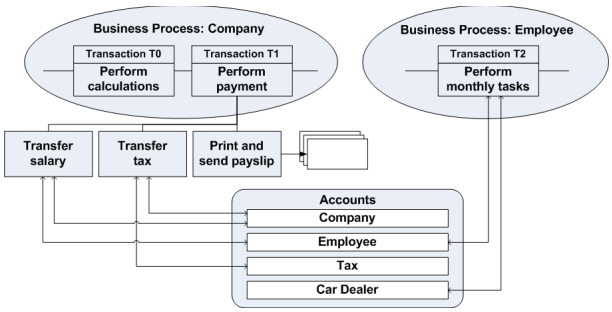

Fig. 2. The motivating scenario

Examples where adaptation/replacement can occur are the following:

1. A service which participates in the transaction fails (for example transfer of the salary fails due to an internal error). Instead of aborting and rolling back to the previous state, a different service can be selected to compensate the failed one. The compensating service is selected based on matchmaking between requested and offered capability. Such a *replacement* is encouraged by the fact that usually multiple services exist that have the same capabilities.
2. A mistake has been made regarding the input data of an operation. In this scenario, it could be that the calculation of the salary is inaccurate, and too

much has been transferred to the employee's account. The flaw is spotted by an administrator, and the system offers a compensating service which will correct the mistake instead of aborting the whole transaction.

In the above examples, the adaptation is considered as a replacement either during the business transaction execution or as a post process compensation execution.

2 Web Application Design

The Web-based application development is usually characterized as an integrated set of activities producing three products of a Web application: application domain, navigation, and presentation and their engineering.

Application Domain Engineering deals with analysis, design, implementation, authoring of concepts which are related to the information content to be made accessible through the Web application, and functions to process, access, and guide through. *Navigation Engineering* deals with activities related to analysis, design, implementation and testing of the modality through which users will navigate through the available information and services. In particular, the navigation engineering is concerned with grouping information fragments and functions into navigation nodes (hypertext nodes, contexts, views) and interconnects them by links. *Presentation Engineering* is concerned with analysis, design, implementation, and testing of appearance of information fragments, functions and their results to a user. The presentation model defines spatial layout and content of information fragments related to the user interface. It also defines presentation classes or objects, spatial relationships between them and content associated with them.

There have been several proposals for Web development methods, describing specific activities for Web application development, like OOHDM [22], WebML [5], UML-based Web Engineering [11], and HERA [13] or reference models like the IMPACT-A method [16]. All of them provide design abstractions for above mentioned activities.

Adaptivity is another concern in web application design which is orthogonal to those mentioned above. Systematic analysis and design of adaptive application features require following requirements to be met [6]:

- *Common and Variable Features* — A method and technique is needed, which will support analysis of *common (nonadaptive)* and *variable (adaptive)* parts of developed applications.
- *Multiple Domains* — A Web application usually serves information from several domains and uses different environments to do so. The adaptation is influenced by parameters from user or client constraints domains. Separation of these concerns, according to domains to which they belong to, helps a designer to focus on features which are important for a particular domain.

- *Dynamic connectors between domains* — The separation between several domains allows for better decision making about features in a particular domain. When designing a particular (instance of) a Web application, the domain features have to be connected (configured) in a certain manner suited to the context of the (instantiated) Web application. Appropriate design technique which supports connectors (compositions) and collaborations between domains is needed. Such a technique helps reason about connections between domains which might encourage their reuse. The connections can be further constrained where the constraints are to be evaluated at run time.
- *Support for adaptive navigation design in connected domains* — the composed information fragments should be further linked to form possible navigation paths. The navigation paths can be further constrained where constraints are to be evaluated at run time.

2.1 Common and Variable Features in Multiple Domains

Adaptation components in adaptive web applications usually recommend one of the several options for links, operations, or content fragments in a content composition. Furthermore, information items can be adaptively configured in the web application based on user profile or abilities of an environment. The options which are planed to be available for adaptation are described using feature models.

Feature Modeling. There have been several proposals for techniques modeling variability and commonality in software systems such as Feature Oriented Domain Analysis (FODA) [15] and or the extended UML structural modeling package with stereotypes for feature modeling (see [6]). Other proposals use different techniques for capturing the variability in software systems like the story boards [2], variation points in assets and components in [1].

Feature model is a set of models which represent configuration aspects of concepts from domains analyzed in web application engineering. Each model in a feature model has one concept and its belonging features. The concept and features are connected to each other by composition relationship. Configuration relations between features and concept are represented as variation points. The concepts and features in feature models are mapped on the concepts and relationships from the conceptual model.

Multiple Domains. The feature models prescribe the parts of the content, environment, and software components, which are stable or common for any user or customer and parts which are variable depending in general on some factors (mostly values of user features or client profiles with requirements for business operations).

The common and variable features can be described separately in:

- *Application Domain* — a domain of information (or content) and user task supporting functions which are served by a web application;

- *Environment Domain* — a domain for representation and organization of information and task supporting functions in a delivery platform;
- *User Domain* — a domain of user features and constrains which are relevant for matchmaking with content, functions and environment to decide for particular option to be offered by a web application.

In application domain, different content can be used to communicate the same information to people with different background. Moreover, the same content can be represented by different media and this content can evolve in time. The content can be also presented in different environment, e.g. as a book, lecture, or an article. Also overall access to the content can be managed through different patterns such as digital library, e-course (virtual university), on-line help, etc.

Each user group may require different information fragment to browse, different composition of presented information (local navigation), and different order and interconnections between information chunks (global navigation). Different navigation styles can also be determined by the target environment where the information is served to a user.

Similarly, different audience may require different appearance and layout of information chunks and different presentation of organization of read information. Target environment can also restrict possibilities to presentation. Thus, it is important to also capture this kind of variability.

Feature Models for Content Intensive Adaptive Web Applications. A feature, as a prominent or distinctive user-visible aspect, quality, or characteristic of a software system [12], in feature models of adaptive web application represents:

- **in an application domain model** — information fragments, which are needed to communicate effectively a concept of a feature model,
- **in an environment/information model** — supporting structural units of a content in particular web-based application,
- **in a user domain model** — qualitative and quantitative features which are needed for decisions about certain adaptation strategy within adaptation process (e.g. a competence acquired within learner performance to decide whether a user is able to grasp particular content item or exercise or metrics of the performance for finer recommendations of next learning steps).

Mandatory features, *Optional features*, and *variation points* are means to analyse and plan a variability of adaptive web applications in the domains mentioned above. The variation points are means to model the dependencies between features and concepts in feature models and can specify *mutually exclusive (XOR),* mutually required (AND), and *mutually inclusive features (OR).*

Feature Models for Web Services and Business Transactions. The domain engineering activities in Web service environments are realized by different independent service providers. The application engineering activities are realized

by different parties as well, employing service selection mechanisms and matchmaking to fit particular business activities when utilizing Web services from different providers. Some of the variable features of the Web services can be considered at runtime. Therefore, the software product line engineering process can be tailored to the Web service environment with extended compensation capabilities as follows. Service provider tasks are:

- *Service Domain Analysis* — is a domain engineering process where variabilities and commonalities between service variants are designed to support compensations based on failures or based on different constraints and requirements;
- *Service Domain Design and Implementation* — different service features are mapped onto an implementation and an architecture for service provisioning where some of the features need not to be exposed to the public and some of the variabilities may be left to runtime adaptation.

Client/service consumer tasks are:

- *Business Application Analysis and Design* — is an application engineering task which may be performed by a party external to the service provider and involves the definition of requirements for and constraints on the Web service compensations;
- *Retrieving the Abstract Web Services* — is an application engineering task in which a designer looks for and retrieves Web services which are required to perform business to business conversations;
- *Defining Client Side Compensations* — is an application engineering task in which a designer defines a variability for compensations which will be exploited at runtime if more Web services with similar capabilities have been found, or an alternative Web service has been defined by an application developer;
- *Implementing Client Side Compensations and Functionalities* — is an application engineering task in which the additional compensations are implemented at the client side, as well as additional operations for which there was no Web service found are realized by an application developer.

Web service capabilities or client requirements are placed into the *application domain conceptual model* and the *compensation concepts* are placed into the *environment conceptual model*. The *configuration view* on the concepts in the application domain model and the environment model is described by means of feature modelling. Therefore, the *functionality feature model* as well as the *compensation feature model* are created. Subsequently, the functionality and compensation models are merged to describe the offered capabilities by a service provider, or requested functionalities and restrictions regarding compensations by a service consumer.

Similarly, [10] studies product lines in the context of adaptive composite service oriented systems. A pattern based variability has been employed for development of composite service-oriented systems in [14].

2.2 Dynamic Connectors between Domains

The collaboration diagrams provide useful abstraction to link together several application domains, where the content or service capabilities are comming from, with an environment through which they are provided.

A Story Collaboration Model is a set of collaboration diagrams which define dynamic content chunks or interaction spaces with business functions accessible in particular environments constrained by conditions evaluating partial restricting profile state (user profile or client requirements profile). The story collaboration diagrams contain collaborations between roles created as instance roles of features and concepts from an application domain feature model linked to instance roles of features and concepts from an environment feature model.

At runtime, the feature instances collaborate to create a content which can be modeled as active information objects providing a defined interface to access their content and presentation. In web services domain, features from capabilities domain interact with each other to deliver a requested service together with environment features such as the compensations.

Roles are used to model different purposes of a particular feature or concept in an environment component. Roles terminology can form a complex structures. The UML class diagram can be employed to model such a structure. This model can be used similarly to the domain and environment conceptual models.

2.3 Adaptivive Navigation in Connected Domains

State machines provide a useful abstraction for adaptive navigation design in a web application where the navigation is seen as a guidence through a certain path in a hypertext graph.

A *Navigation State* for a user is an information chunk or an interaction space [9] observed by a user at a hypertext node at a given time. In the UML state diagrams, atomic states can be grouped into *superstates*. States usually refer to concepts of an application domain. The superstate may compose substates in alternate or parallel fashions. Concurrency in web presentations can be handled by *Concurrent regions*, *Fork* and *Join* pseudostates, and *SyncState*.

A *Transition in Navigation Trail* is a transition between one navigation state and another. The transition is usually caused by a user interaction *event* or by another event (e.g. time event). When the transition is fired it leads to a production of a new hypertext node for a user — the new navigation state. *Guards* can be used to constrain transitions, entry, and exit actions of states by adaptation rules. Usually, they consist of a predicate over user profile attributes or context information. The transitions and events on states are useful abstractions for assigning sensors observing user evolution. Each transition can have a side effect *action*. Actions can be performed also at entry, exit and as an internal transition side effect of state. The side effect can be, for example, the modification of a user profile, or the choice of presentation styles for a given chunk of information. Actions can also process parameters used in guards of outgoing part of branches.

The *variability in the navigation trails* is supported by the alternate (OR) states and by decision symbols which can split transition to several alternative transitions. In this way, the navigation trails can have alternate navigation paths and information chunks constrained by conditions referring to certain user, content, device, or environment features. From user point of view it means that each trail can be adapted by taking into account the user background, level of knowledge, preferences and so on [7].

Similar principles apply to business operations. Those business operations which are dependent on each other may similarly be linked from the user perspective. State machines or transition systems can then be applied in a similar fassion. [18, 20], for example, describe the semi-automatic adaptation of a workflow in case of errors. A change of the workflow process can, for example, consist of a deletion or jump instruction, or the insertion of a whole new process segment. The change can either be done on a running instance, or it can be performed on the scheme which controls the workflow, and which results in a change in all running instances. Refer to [19] for details. Labeled transition systems are also used in context based substitution of web services [17].

3 Further Challanges

As the web evolves new opportunities for innovative applications occur. These opportunities, however, also raise many challenges for design. Here I have mentioned just three categories, which I think are very relevant today.

Rich Internet Applications. Rich internet applications are applications which try to enhance user experience on the web and bring it as close as possible to desktop applications. Such applications become popular especially when multimedia capabilities and programming models behind Adobe/Macromedia products and AJAX were introduced. However, the challenge for web engineering methods is how to deal with asynchronous communication, functionality on server side as well as client side, possibly independent autonomous servers, synchronization and so on. In this setting, computation of links is becoming more complex. Issues such as availability of content, which have not been so crucial in closed, one server environments, are now also becomming important.

Social Web Applications. Open adaptive web applications where dynamic groups of users exist pose other challenges on design. The open question is, for example, how to compose user profiles into group profiles. It is also interesting to study in which context a single individual profile is more useful over the group profile. Furthermore, it is also interesting to study how different activities of different groups and different individuals contribute to an effective personalized access to information and operations. This multilevel interaction of various profiles adds a complexity to web applications, thus influencing their design methods as well.

Composition Models. There are two mainstream approaches to handle business transactions, commits, locking, composition, interaction as well as coordination of adaptive web services and applications followed in web services community. On the one hand, there are plan-based design approaches, for example, based on BPEL, which prescribe composition and interaction between participating services. On the other hand, there are middleware approaches with autonomous protocols focusing on environments where services can join and leave on an ad-hoc manner. It is interesting to study the tradeoffs between them as well as various design techniques either for compositions or for middleware in combination with algorithms, protocols, and computation models for transactions.

References

1. Felix Bachmann, Michael Goedicke, Julio Cesar Sampaio do Prado Leite, Robert L. Nord, Klaus Pohl, Balasubramaniam Ramesh, and Alexander Vilbig. A meta-model for representing variability in product family development. In Frank van der Linden, editor, *Software Product-Family Engineering, 5th International Workshop, PFE 2003*, volume 3014 of *Lecture Notes in Computer Science*, pages 66–80, Siena, Italy, November 2003. Springer.
2. Joachim Bayer, Oliver Flege, Peter Knauber, Roland Laqua, Dirk Muthig, Klaus Schmid, Tanya Widen, and Jean-Marc DeBaud. Pulse: A methodology to develop software product lines. In *SSR*, pages 122–131, 1999.
3. Peter Brusilovsky. Adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 11(1-2):87–100, 2001.
4. Stefano Ceri, Peter Dolog, Maristella Matera, and Wolfgang Nejdl. Adding client-side adaptation to the conceptual design of e-learning web applications. *Journal of Web Engineering*, 4(1):21–37, 2005.
5. Stefano Ceri, Piero Fraternali, and Maristella Matera. Conceptual modeling of data-intensive web applications. *IEEE Internet Computing*, 6(4), August 2002.
6. Peter Dolog. *Engineering Adaptive Web Applications.* Doctoral thesis, Leibniz University of Hannover, Hannover, Germany, March 2006.
7. Peter Dolog and Wolfgang Nejdl. Using UML and XMI for generating adaptive navigation sequences in web-based systems. In Perdita Stevens, Jon Whittle, and Grady Booch, editors, *Proc. of UML 2003 — The Unified Modeling Language. Model Languages and Applications. 6th International Conference*, volume 2863 of *LNCS*, pages 205–219, San Francisco, CA, USA, October 2003. Springer.
8. Peter Dolog and Wolfgang Nejdl. Using UML-based feature models and UML collaboration diagrams to information modelling for web-based applications. In Thomas Baar, Alfred Strohmeier, Ana Moreira, and Stephen J. Mellor, editors, *Proc. of UML 2004 — The Unified Modeling Language. Model Languages and Applications. 7th International Conference*, volume 3273 of *LNCS*, pages 425–439. Springer, October 2004.
9. Peter Dolog and Jan Stage. Designing interaction spaces for rich internet applications with uml. In Piero Fraternali, Luciano Baresi, and Geert-Jan Houben, editors, *ICWE2007: International Conference on Web Engineering*, volume 4607 of *LNCS*, pages 358–363, Como, Italy, July 2007. Springer Verlag.
10. S. Hallstein, E. Stav, A. Solberg, and J. Floch. Using product line techniques to build adaptive systems. In *SPLC'06. 10th Intl. Software Product Line Conference*, 2006.

11. Rolf Hennicker and Nora Koch. A UML-based methodology for hypermedia design. In S. Stuart A. Evans and B. Selic, editors, *Proc. of UML 2000 Conference*, York, England, October 2000. Springer LNCS 1939.
12. American Heritage. The american heritage dictionary. Houghton Mifflin, Boston, MA, 1998.
13. Geert-Jan Houben, Peter Barna, Flavius Frasincar, and Richard Vdovjak. Hera: Development of semantic web information systems. In Juan Manuel Cueva Lovelle, Bernardo Martin Gonzalez Rodriguez, Luis Joyanes Aguilar, Jose Emilio Labra Gayo, and Maria del Puerto Paule Ruiz, editors, *Proceedings of International Conference on Web Engineering, ICWE 2003*, number 2722 in LNCS, pages 529–538, Oviedo, Spain, July 2003. Springer Verlag.
14. J. Jiang, A. Ruokonen, and T. Systä. Pattern-based variability management in web service development. In *ECOWS'05. Third European Conference on Web Services*, 2005.
15. Kyo C. Kang, Sholom G. Cohen, James A. Hess, and William E. Novak. Feature-oriented domain analysis (foda) feasibility study. Technical Report CMU/SEI-90-TR-21, ESD-90-TR-222, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, 1990.
16. David B. Lowe, Andrew J. Bucknell, and Richard G. Webby. Improving hypermedia development: a reference model-based process assessment method. In *Proceedings of the ACM International Conference on Hypertext and Hypermedia(Hypertext '99)*, pages 139–146, Darmstadt, Germany, February 1999.
17. Jyotishman Pathak, Samik Basu, and Vasant Honavar. On context-specific substitutability of web services. In *2007 IEEE International Conference on Web Services (ICWS 2007)*, pages 192–199, Salt Lake City, Utah, USA, July 2007.
18. M. Reichert and P. Dadam. ADEPTflex: Supporting Dynamic Changes of W orkflow without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.
19. M. Reichert, S. Rinderle, U. Kreher, and P. Dadam. Adaptive Process Management with ADEPT2. In *ICDE*, pages 1113–1114. IEEE, 2005.
20. S. Rinderle, S.Bassil, and M. Reichert. A Framework for Semantic Recovery Strategies in Case of Process Activity Failures. In Yannis Manolopoulos, Joaquim Filipe, Panos Constantopoulos, and José Cordeiro, editors, *ICEIS*, pages 136–143, 2006.
21. Michael Schäfer, Peter Dolog, and Wolfgang Nejdl. Engineering compensations in web service environment. In Piero Fraternali, Luciano Baresi, and Geert-Jan Houben, editors, *ICWE2007: International Conference on Web Engineering*, volume 4607 of *LNCS*, pages 32–46, Como, Italy, July 2007. Springer Verlag.
22. Daniel Schwabe and Gustavo Rossi. An object-oriented approach to web-based application design. *Theory and Practise of Object Systems (TAPOS), Special Issue on the Internet*, 4(4):207–225, October 1998.