www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 3 March 2013 Page No. 650-658

Adaptive Web Based Application Development

Annan Naidu Paidi
Asst.Professor of CSE
Centurion University, India.
annanpaidi@gmail.com

Abstract:-*The World Wide Web has had an extremely significant impact on access to information and information services. Web-based applications are influencing many domains such as business, commerce, banking and learning. The Web has simplified access to information and information services, enabling a variety of users with different backgrounds, social situations, and so on to participate. The increasing complexity of such applications calls for engineering methods for developing efficient high quality software applications with all the appropriate performance, features and services which are required. Applications and information which are provided on the Web according to the 'one size fits all' approach are not appropriate to such a heterogeneous environment. Adaptive Web-based applications are an alternative to the traditional "one-size fits-all" static approach in the development of applications. Thus a distinctive feature of such an adaptive application is its ability to adapt itself to certain conditions.*

Keywords: *Adaptive web application, Web application,Web engineering, Adaptive* eLearning Applications.

1. Introduction

The purpose of the research being undertaken in adaptive Web based systems is to find answers to questions relating to the heterogeneous needs of many differentWebusers. Applications and information which are provided on the Web according to the 'one size fits all' approaches are not appropriate to such a heterogeneous environment. Several researchers have tried to address personalization issues in their research and have developed a number of adaptive Web-based applications.

Adaptive Web-based applications are an alternative to the traditional "one-size fits-all" static approach in the development of applications and aim to leave some of the features of such applications at the design stage in the form of variables which are dependent on several criteria. Thus a distinctive feature of such an adaptive application is its ability to adapt itself to certain conditions. The most commonly used criteria to determine which features of an application will be used are user cantered criteria. User-Centred adaptive applications utilize user features 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.

Adaptive Web-based applications match conceptual descriptions of adaptive parts of the applications with related descriptions of a user. In some cases, there is an exact match between information description and user features but mostly some similarity measures are applied in addition. Based on the matching results, some further processing is applied, usually at the presentation level. Such processing is classified into several techniques.

The first group of techniques is intended to improve a user's local navigation and orientation in the currently presented page or fragment. For example, such adaptive systems can provide different text or media variants which serve information at different levels of detail to users with varying levels of knowledge or expertise in some field. They can switch between different media types according to differing user preferences or learning styles. They are able to hide or appropriately annotate certain parts of presented information items based on values of user features maintained by a system. These

techniques are called *adaptive presentation techniques*.
The subsequent group of techniques have the purpose of enhancing the user's global orientation in hyperspace. That is to say, they are designed to provide a user with support in exploring required information. This includes techniques such as enabling, disabling, showing, hiding, annotating or removing links when it is appropriate
and applying priorities to them according to different user features. The techniques also deal with generating appropriate information subsequent to the one currently being presented which then provides a further aid to guide a user. These techniques are called *adaptive navigation techniques.*

2. Web Application Development

We have identified several requirements for a methodology which should supportthe adaptive Web-based application engineering . In this section wewill look at current proposals for Web-based application design available in research community and will analyze how they support the requirements. It is widely accepted that models constitute important mechanism in the process of a system development. Models help us understand the system by omitting some details. The choice of *what* to model has a significant effect on understanding a problem and suggesting the solution. Significant influence on the solution has also the matter *when* particular model is created. According to answers on "what" and "when" questions, particular notation is chosen. A notation determines the level of formality of particular model. Notation is also determined by a decision about whether structural or behavioural model is developed. The notation and technique determine how we can express certain features of a system. Regarding to our requirements for the engineering methodology for adaptive Web-based systems, it is important that a notation and technique provide a possibility to express the variable and common features in a domain information which will be provided within a Web application. Also the technique should provide a possibility to model several domains separately as the Web application is usually build on top of several information and/or serving domains. In addition, it should provide notions for expressing variability in connections between the different domains and in navigation in the information space determined by the domains. The questions "when", "how", and "what" provide dimensions for analysis of existing techniques used in development of Web applications:

Phase dimension follows the question "when". It reflects the fact that a modelling technique is applied in particular phase in development of a system (i.e., inception, elaboration, construction, or transition). Not all techniques can be applied in each phase;

Workflow dimension follows the question "when" from the point of view of workflowsin software engineering. It reflects the fact that a modelling technique is applied in particular workflow of development (i.e., requirements, analysis, design, or implementation). Not all techniques can be applied in each workflow;

Notation/Technique dimension emphasizes modelling technique and corresponding notation used. The techniques can be grouped into two aspects: structural and behavioural. Structural models describe structure of a system (e.g., subsystems, packages) and structural relationships between them(e.g.,inheritance,dependency and associations).Behavioural models describe reactions of a system to external or internal events such as collaborations or message passing in the system or the system's reactions to user interactions.

Web Application Products dimension is concerned with the question "what" to model. Tiers, which are commonly discussed in the Web-based application perspective, are application domain, navigation and presentation. Not all techniques are useful for description of particular product.

3. Requirements for Design Methodology

The adaptation is usually a decision for a particular information item or function based on knowledge about the items being recommended. The knowledge about items usually contains 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 matching to user properties. Different users have different properties or different
property values associated with them. The differences 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 behaviour pattern evolves, recommended items may change. This is ensured by continuous updating and evolution of a user's profile based on his or her behaviour as traced by the application. In this way, the user always receives up to date information items or functions matching the current state of his profile. Following requirements for engineering methodology:

Support for common and variable features—A designer should determine which application parts are common to all users and which are variable (variants suitable for different users). To be able to take decisions about the common and variable features, an appropriate modelling technique should be provided. The technique should support the notion of common and variable features and a notion of feature dependencies when a selection of one feature influences a selection of another one.
Support for several domain models—A Web application usually serves information from several domains. The term "user model" stands for another domain considered especially in the context of adaptive Web-based applications. Also an environment for information delivery and arrangement of the environment sometimes differ. Separation of the features according to the domains to which they belong helps the designer to focus on features which are important for particular domain.
Support for dynamic connectors between several domains — The separation of several domains allows for better decision making about features in a particular domain. When designing 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 and collaborations between domains should be provided. Such a technique will help to 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. The technique should also support the specification of composition of information features into meaningful items of information suitable for presentation.

Support for 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. The constraints can be of different kinds but the focus is especially directed to constraints which evaluate user features.

3.1 Architecture for adaptive Web-based systems

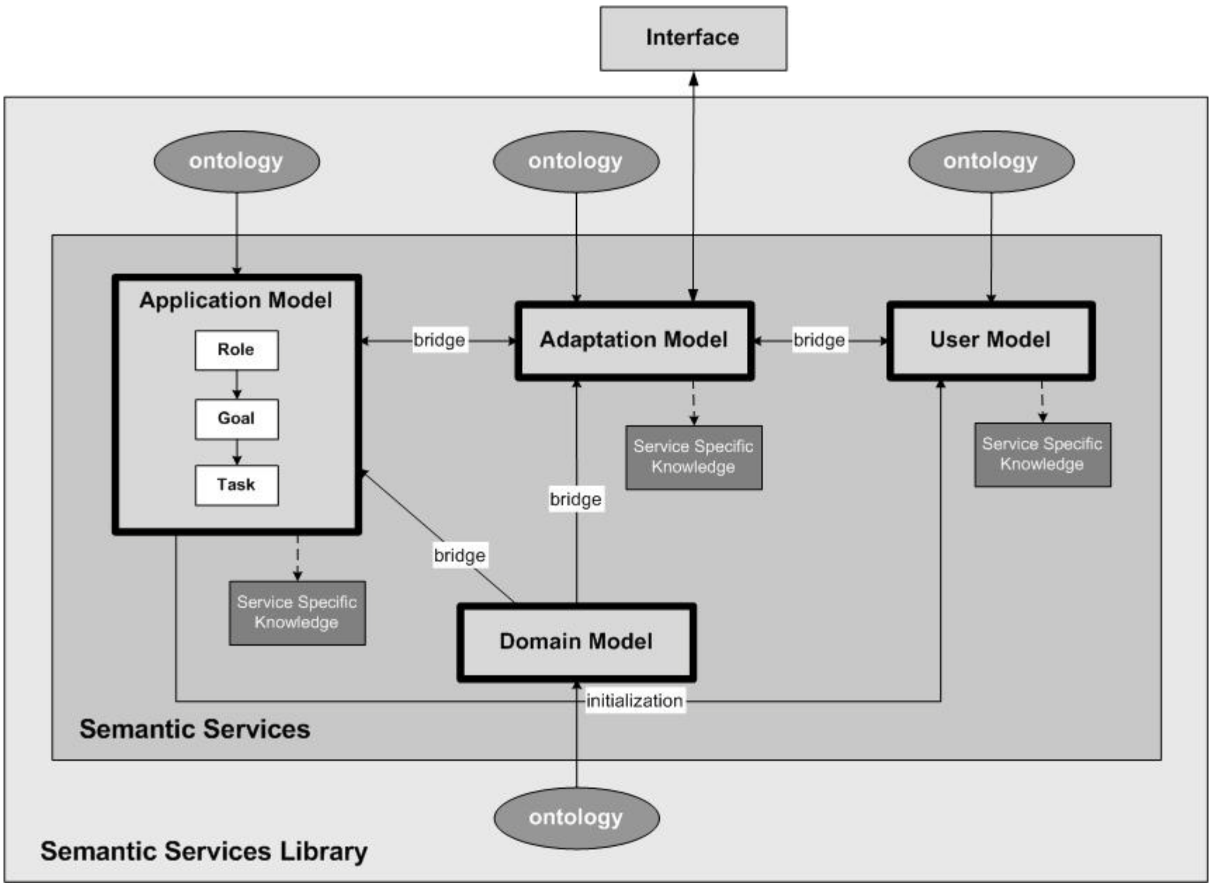


Figure 1. Architecture for adaptive Web-based systems

Figure 1. illustrates our vision of the *modular architecture* for adaptive Web-based systems (Chepegin *et al*. 2003). One of the first characteristic aspects we observe is that the different system components are all equipped with facilities to communicate with the (other) components in terms of service invocations. In this architecture, *bridges* are used in accordance with the UPML framework connector defined by Fensel *et al*. (1999), in order to specify mappings between the different model services within the architecture. *Ontologies* also play an important role to define and unify the system's terminology and properties to describe the knowledge of each system service. Each service can be specified by means of a corresponding ontology, providing common ground for knowledge sharing, exploitation and interoperability among the services. This leads to a highly modularized architecture which offers a high degree of *flexibility*.

In the case of adaptive systems access to the *user model* via a Web service is a good example of this flexibility. It means that designers can design systems that will interact with or react to the user intelligently without knowing anything in advance about that user, but simply using the knowledge collected by other applications and interpreting it in the context of the current application. Crucial for achieving such a flexible architecture is the need for a *standardized* protocol for encoding information about users. As mentioned above, open standards (e.g. XML, RDF, OWL) allow for the specification of ontology to standardize and formalize meaning and to enable reuse and interoperability. Another key aspect is to facilitate mobile user models that follow the user across applications. Results in the field of software agents provide implementation views supporting mobility and autonomous behaviour. A final but important requirement is for the user modelling system to be able to reuse system and knowledge components, and thus benefit from other applications.

A second characteristic seen in Figure 1 is the *separation* of the different components. When transforming these components into services the need for a fourth component, the *application model*, emerges. The main reason for this lies in the fact that in the traditional AHS(adaptive hypermedia systems) approach the adaptation model unites the actual process of *how* to adapt with the decisions *why* to adapt. In most applications, e.g. those realized in AHA(Adaptive hypermedia architecture), the designer's knowledge about why the user is served in a certain way is more or less left implicit: the adaptation model implements the way in which the information is adapted to the user and does so based on the designer's decisions, which are not made explicit. In the situation where we want to share and exchange the different functionalities between systems, it becomes relevant to separate the "how" and "why" in the adaptation. While capturing the designer's intentions about the roles, goals and tasks in the application (related to domain model concepts and user model values), the adaptation model restricts itself to the actual realization of the adaptation to follow the directions given by the application model. In fact this aligns well with the proposal from the IRS-II approach, mentioned above. The application model service contains a generic description of the user tasks in the context of a Role-Goals-Tasks model.

It is clear that this gives the *application model service* a crucial role in the system architecture. It divides the adaptation process into two parts, where the actual technical adaptation is performed by the *adaptation model service*, while the management of the service process is coordinated from the application model service. The entire architecture as displayed in Figure 1 emphasizes the fact that the core knowledge about the application processes and the user activities (tasks, roles and goals) lies in the application model service. In the interaction with the application the user is represented by a particular role (e.g. guest, a system expert, administrator, student). This role defines for him/her a corresponding behavior in terms of goals to achieve. To accomplish the user's goals appropriate tools (applications) are used, which realize one or several corresponding methods. The adaptation model service receives the direct user input and interacts with the application model service in order to define the context for the user input for its most precise adaptation. Further the adaptation model service queries the *domain model service* in order to select the relevant content to be presented to the user. The domain model service is responsible for the explicit storage and description of the domain knowledge in terms of concepts of a domain ontology. Finally, it updates the user model with new values. For instance, when users work with a selected application, every action they perform on the user interface is communicated to the *user model service*, which is responsible for updating the user model with the new values. The user information is stored there and a reasoning engine infers new knowledge from it and makes predictions concerning future user behaviour. In this way the user model service allows for sharability of the user model between applications by following the user (inside and outside the system) in order to collect and further analyze data about the user's activities.

4. Adaptive Web Based Application Development

Adaptively 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.
– Common and Variable Features — A method and technique is needed, which will support analysis of *common (no adaptive)* 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 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.

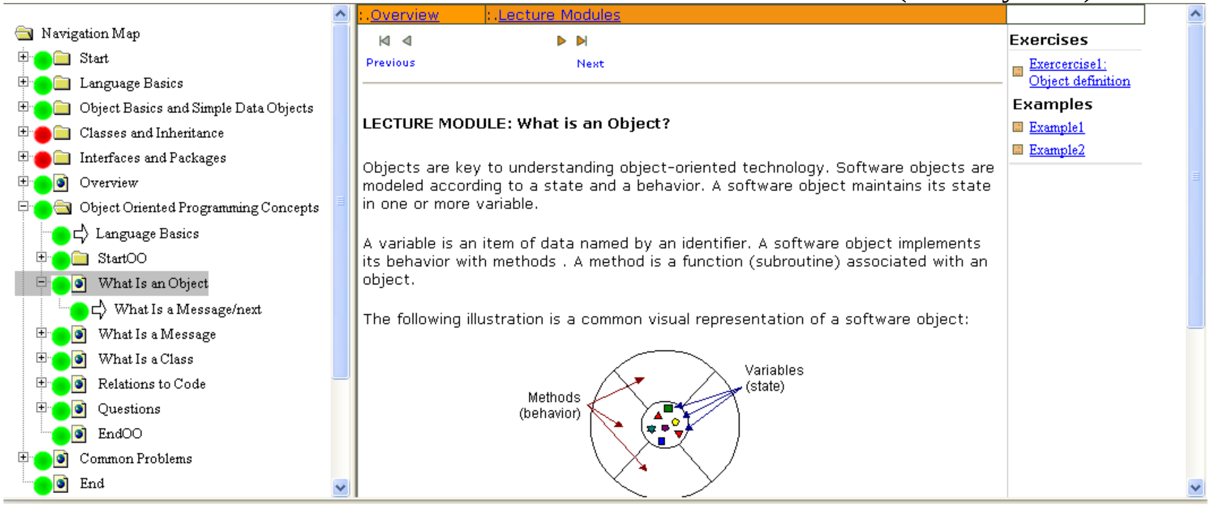
– ***Support for adaptive navigation design in connected domains***—the composed information fragments should be further linked to form possible navigation paths. The navigation paths can be further constrained where constraints are to be evaluated at run time. Adaptive Web-based applications provide an alternative to the traditional “one size-fits-all” applications. 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- centred 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.

4.1 Adaptive eLearning Applications. The UML-Guide is an example of an adaptive web application. It generates a map of an information space designed for a particular information or learning goal. The map itself is adaptive; some ofthe links and symbols are annotated according to knowledge about a user whichis maintained by the UML-Guide. One approach to the visualization of the navigation map is depicted in Fig. 2.The navigation map displays a composite hierarchy of information nodes ininformation space (folder symbols with subfolders and document nodes) and sequencing relations between the nodes (arrow symbol).



LECTURE MODULE: What is an Object?

Objects are key to understanding object-oriented technology. Software objects are modeled according to a state and a behavior. A software object maintains its state in one or more variable.

A variable is an item of data named by an identifier. A software object implements its behavior with methods . A method is a function (subroutine) associated with an object.

The following illustration is a common visual representation of a software object:

Fig. 2. Visualization of the navigation graph for the Java e-lecture

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 match making 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 behaviour pattern evolves, recommended items may change. This is ensured by continuous updating and evolution of a user's profile based on his or her behaviour 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 adaptively is considered beyond the user profile and is based on rather different requirements profiles matched with service profiles satisfying a business function. 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 instalment 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 3. Only the services of transaction T1 are shown

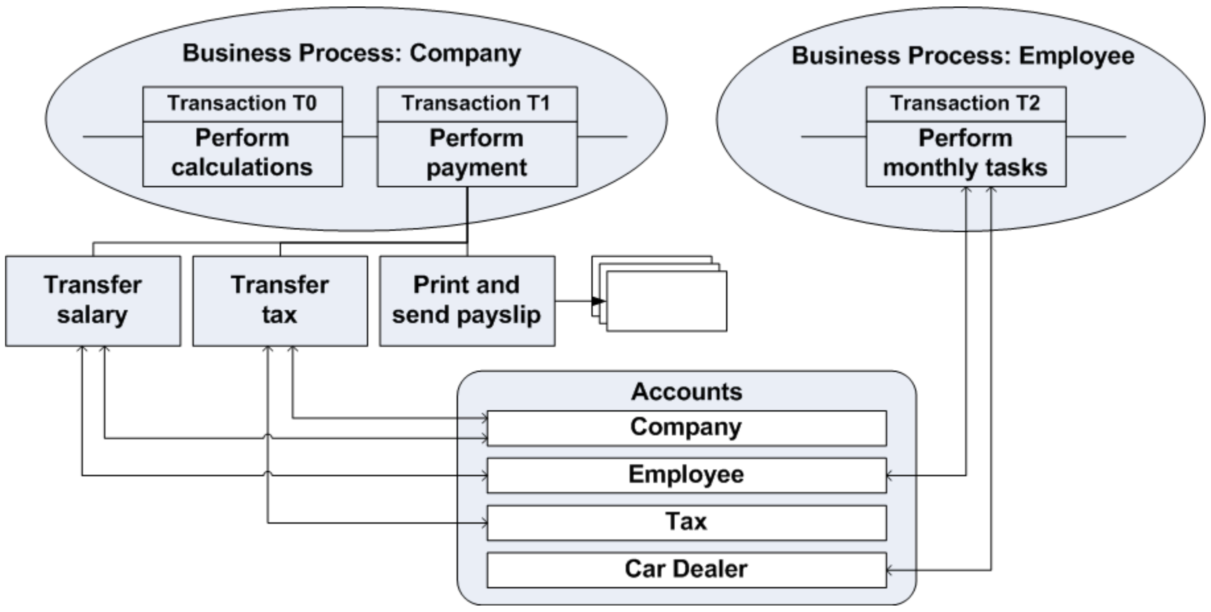


Fig. 3. The motivating scenario

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.

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 , 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)*.

4.2 Adaptive Navigation in Connected Domains

State machines provide a useful abstraction for adaptive navigation design in aweb application where the navigation is seen as a guidance through a certainpath 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 UMLstate diagrams, atomic states can be grouped into *superstates*. States usually refer to concepts of an application domain. The super state may compose substatesin alternate or parallel fashions. Concurrency in web presentations can be handled by *Concurrent regions*, *Fork* and *Join* pseudo states, 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 auser 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[5].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 fashion. [10], 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 [11] for details. Labeled transition systems are also used in context based substitution of web services [12].

5. Further Challenges

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 becoming 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 studying 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 addsa 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.

References

1. Paul De Bra, Natalia Stash, David Smits, Creating Adaptive Web-Based Applications.

2. Irene Garrigós , Jaime Gómez and Cristina Cachero, Modelling Adaptive Web Applications.

3. Guntram Graef and Martin Gaedke, Construction of Adaptive Web-Applications from Reusable Components.

4. Brusilovsky, P.: Adaptive hypermedia. User Modeling and User-Adapted Interaction 11(1-2), 87–100 (2001).

5. Dolog, P., Nejdl, W.: Using UML and XMI for generating adaptive navigation sequences in web-based systems. In: Stevens, P., Whittle, J., Booch, G. (eds.)UML 2003. LNCS, vol. 2863, pp. 205–219. Springer, Heidelberg (2003)

6. Ceri, S., Dolog, P., Matera, M., Nejdl, W.: Adding client-side adaptation to theconceptual design of e- learning web applications. Journal of Web Engineering 4(1),21–37 (2005).

7. Sch¨afer, M., Dolog, P., Nejdl, W.: Engineering compensations in web service environment.In: Fraternali, P., Baresi, L., Houben, G.-J. (eds.) ICWE 2007. LNCS,vol. 4607, pp. 32–46. Springer, Heidelberg (2007).

8. Zolt´an Fiala, Michael Hinz, Klaus Meissner, Construction of Adaptive Web-Applications fromReusable Components.

9. Dolog, P., Stage, J.: Designing interaction spaces for rich internet applicationswith uml. In: Fraternali, P., Baresi, L., Houben, G.-J. (eds.) ICWE2007. LNCS,vol. 4607, pp. 358–363. Springer, Heidelberg (2007).

10. Rinderle, S., Bassil, S., Reichert, M.: A Framework for Semantic Recovery Strategiesin Case of Process Activity Failures. In: Manolopoulos, Y., Filipe, J., Constantopoulos,P., Cordeiro, J. (eds.) ICEIS, pp. 136–143 (2006)

11.Reichert, M., Rinderle, S., Kreher, U., Dadam, P.: Adaptive Process Managementwith ADEPT2. In: ICDE, pp. 1113–1114. IEEE, Los Alamitos (2005)

12. Pathak, J., Basu, S., Honavar, V.: On context-specific substitutability of web services.In: ICWS 2007. IEEE International Conference on Web Services, Salt LakeCity, Utah, USA, pp. 192–199 (July 2007)